Bom dia
Descodificadas!

Grid

Hoje em dia usamos CSS para estilizar layouts muito diferentes do que imaginávamos na época que a web foi criada. De tabelas para divs, de floats para Flexbox, as ferramentas disponíveis para criação de layouts na web evoluíram muito desde os anos 90.

Agora, CSS Grid já é suportado em todos os navegadores mais atuais e podemos dizer com segurança que esse é o método de layout mais poderoso que já foi adicionado ao CSS.

O que é grid?

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.

Dentro de um grid, respeitando-se a configuração criada pelas suas linhas, pode-se inserir elementos da marcação.

O grid é constituído por 12 colunas.

O que é grid?

Em palavras muito simples Grid System (sistema de grade, em português) é usado para dividir uma página em pequenos e espaçados segmentos.

Dimensões fixas ou flexíveis?

Podemos criar grids com dimensões fixas – por exemplo: definindo dimensões em pixels.

Ou criar grids com dimensões flexíveis definindo-as com uso de porcentagem ou da nova unidade CSS fr criada para esse propósito.

Posicionamento de itens

Podemos posicionar com precisão itens de uma página usando o número que define uma linha do grid, usando nomes ou ainda fazendo referência a uma determinada região do grid.

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

```
.container {
  display: grid;
  grid-template-columns: [sidebar]
```

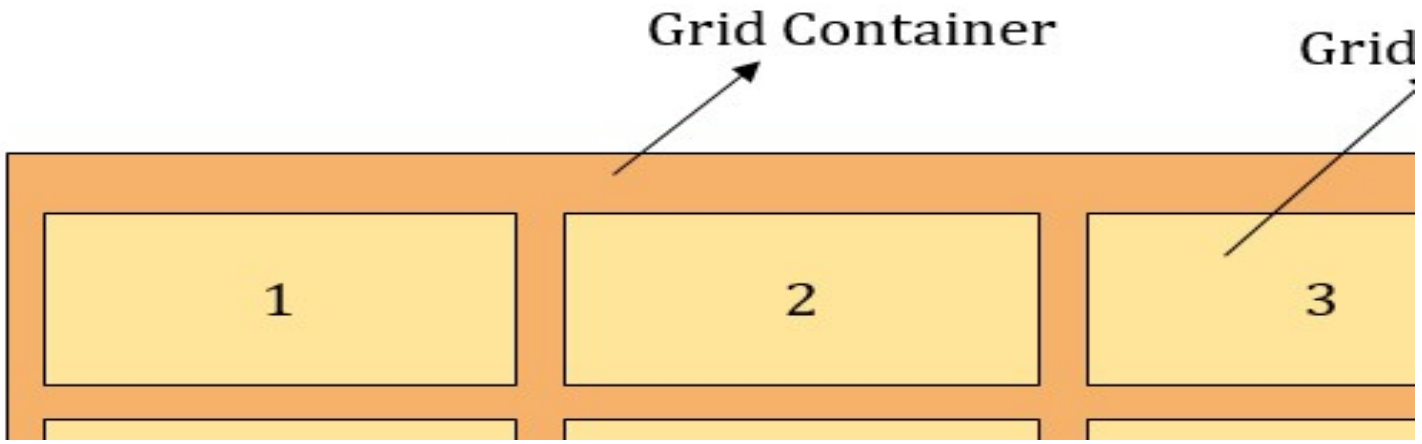
Posicionamento de itens

CSS Grid Layout é uma poderosa especificação que se for combinada com outras partes do CSS, tal como flexbox, possibilita a criação de layouts que até então eram impossíveis de serem criados com CSS.

Grid - Estrutura

Grid container é o elemento que envolve sua estrutura.

Em um contêiner grid, todos os seus filhos diretos se tornam itens grid. Com um contêiner grid em mãos podemos começar a criar layouts usando colunas e linhas.

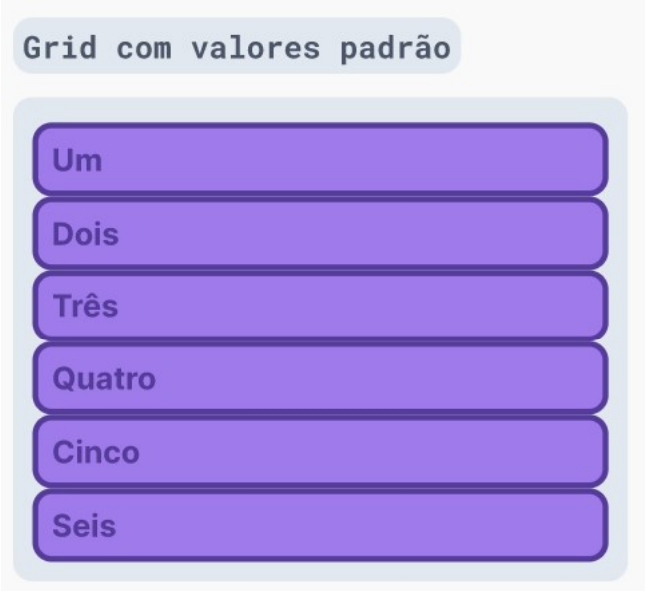


Container

display: grid;

```
<div class="container">
  <div>Um</div>
  <div>Dois</div>
  <div>Três</div>
  <div>Quatro</div>
  <div>Cinco</div>
  <div>Seis</div>
</div>
```

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



Cada um dos seis elementos que estão dentro do nosso contêiner são itens grid. Por padrão, temos apenas uma coluna, acompanhada de linhas criadas automaticamente para cada elemento.

grid-template-columns

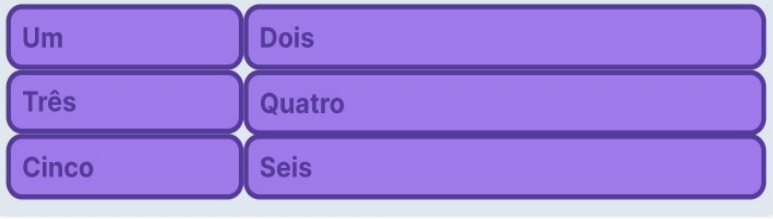
Usado para criar colunas, recebe os valores de tamanho de cada coluna separados por espaço.

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

```
.container {
  display: grid;
  width: 500px;
  grid-template-columns: 150px 1fr;
}
```

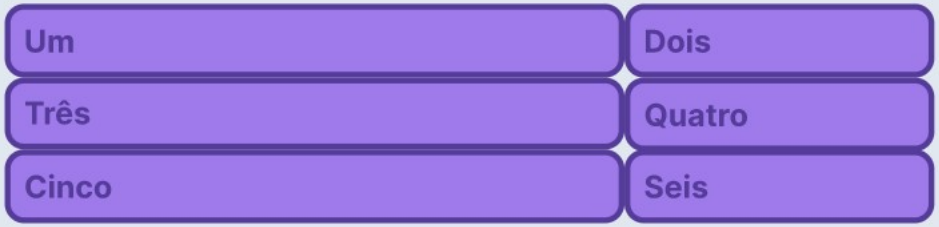
`grid-template-columns: 200px 200px;`

`grid-template-columns: 150px 1fr;`




```
.container {
  display: grid;
  width: 500px;
  grid-template-columns: 2fr 1fr;
}
```

`grid-template-columns: 2fr 1fr;`



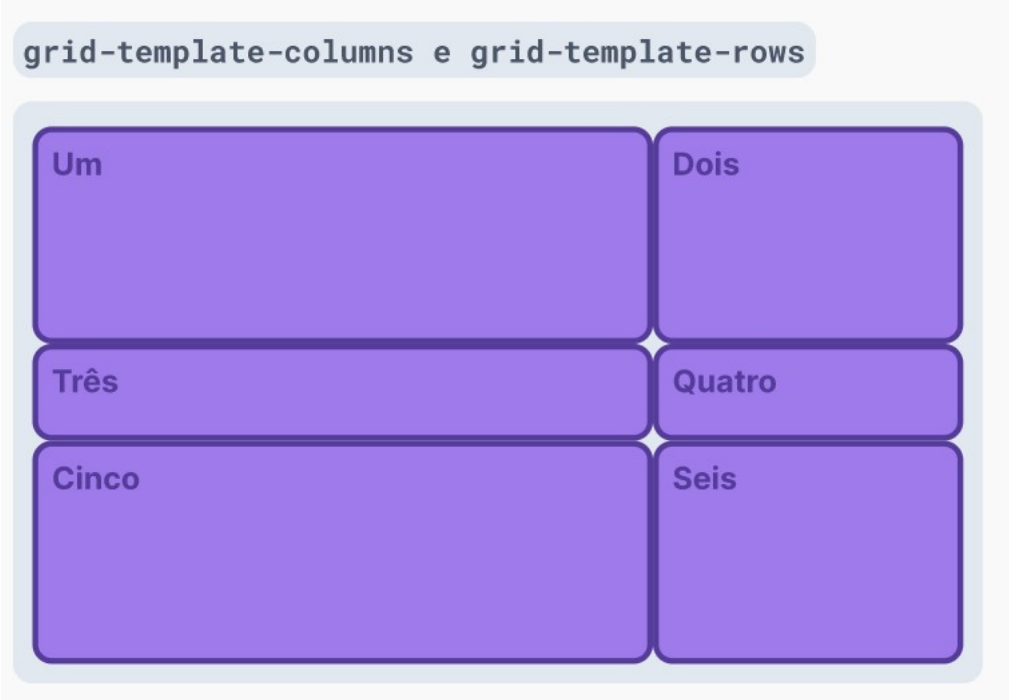
grid-template-rows

Usado para definir linhas, usando valores separados por espaço.

```html
<div class="container">
  <div>Um</div>
  <div>Dois</div>
  <div>Três</div>
</div>
```

```css
.container {
  display: grid;
  height: 300px;
  grid-template-rows: 3fr 1fr 3fr;
}
```

```html
<div class="container">
  <div>Um</div>
  <div>Dois</div>
  <div>Três</div>
  <div>Quatro</div>
  <div>Cinco</div>
  <div>Seis</div>
</div>
```



Com apenas três declarações CSS, conseguimos criar um layout altamente flexível e bastante difícil de replicar com qualquer outra técnica CSS.

É na combinação de colunas, linhas e tamanhos flexíveis que CSS Grid se destaca de tudo o que já existiu e põe na mão das pessoas que desenvolvem um poder imenso para criar layouts.

Adicionando espaço entre itens grid com gap

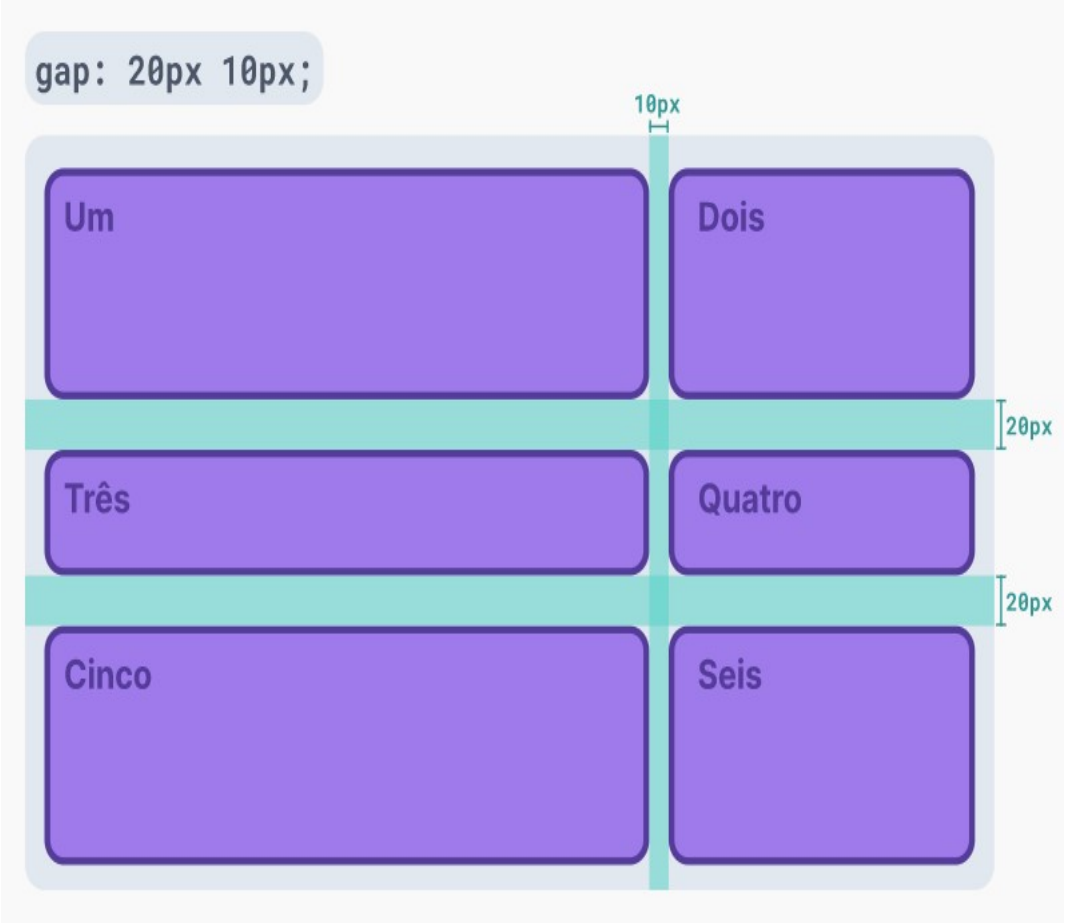
Mas... e se quiséssemos adicionar um espaço entre todas essas linhas e colunas? Normalmente conhecido como “gutter”, esse espaço pode ser declarado usando a propriedade gap.

gap

É uma forma abreviada de declarar os valores de row-gap (que define o espaço entre linhas) e column-gap (que define o espaço entre colunas). Para declarar valores diferentes podemos adicionar um segundo valor de tamanho (gap: 20px 10px) ou então escrever row-gap: 20px e column-gap: 10px separadamente.

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

```
  gap: 20px 10px;
  /* ou então */
  row-gap: 20px;
  column-gap: 10px;
}
```



Direção do fluxo de um contêiner

grid-auto-flow

Por padrão, cada elemento em um contêiner grid automaticamente cria um nova linha. Então um contêiner grid com três elementos dentro dele teria uma coluna e três linhas.

Para mudar este comportamento e criar colunas, basta usar grid-auto-flow: column.

```
<div class="container">
  <div>Um</div>
  <div>Dois</div>
  <div>Três</div>
</div>
```

```
.container {
  display: grid;
  grid-auto-flow: column;
}
```

Posição de um elemento no contêiner grid

Além de definir o tamanho e quantidade de linhas e colunas em um contêiner grid, nós podemos definir a posição de cada elemento separadamente.

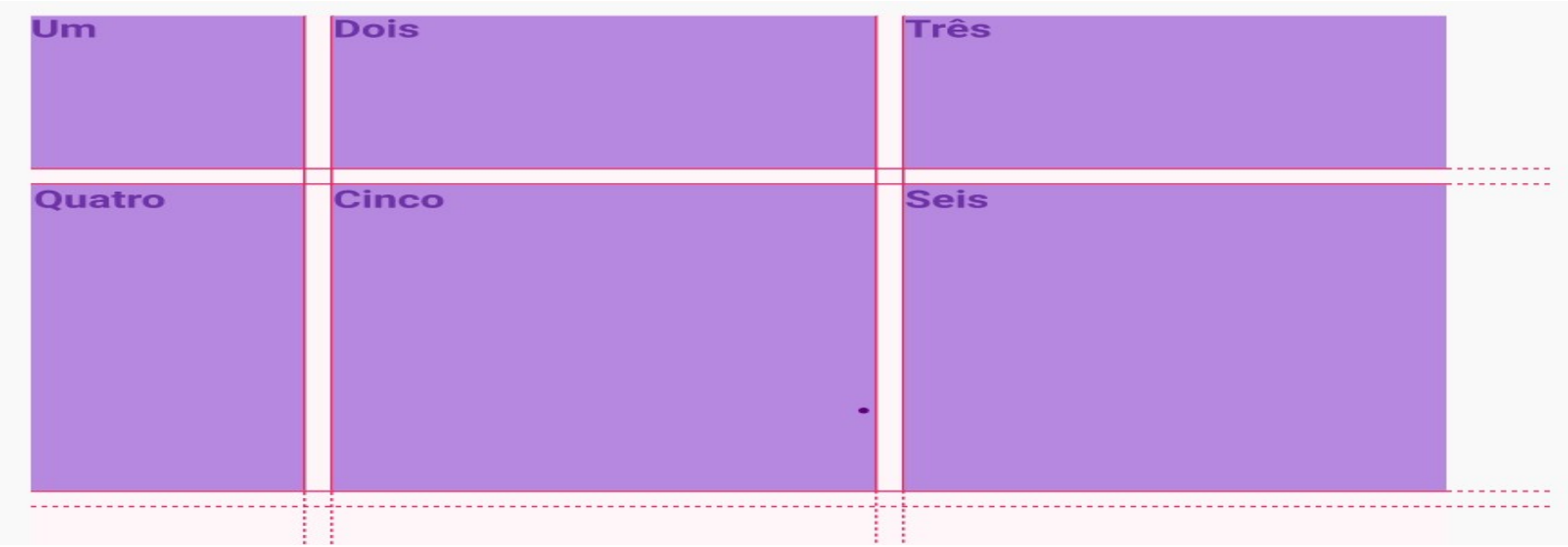
Usando grid-column e grid-row, podemos definir em qual coluna ou linha um item começa e termina.

Exemplo

- Criando algumas linhas e colunas:

```html
<div class="container">
  <div class="item item-1">Um</div>
  <div class="item item-2">Dois</div>
  <div class="item item-3">Três</div>
  <div class="item item-4">Quatro</div>
  <div class="item item-5">Cinco</div>
  <div class="item item-6">Seis</div>
</div>
```

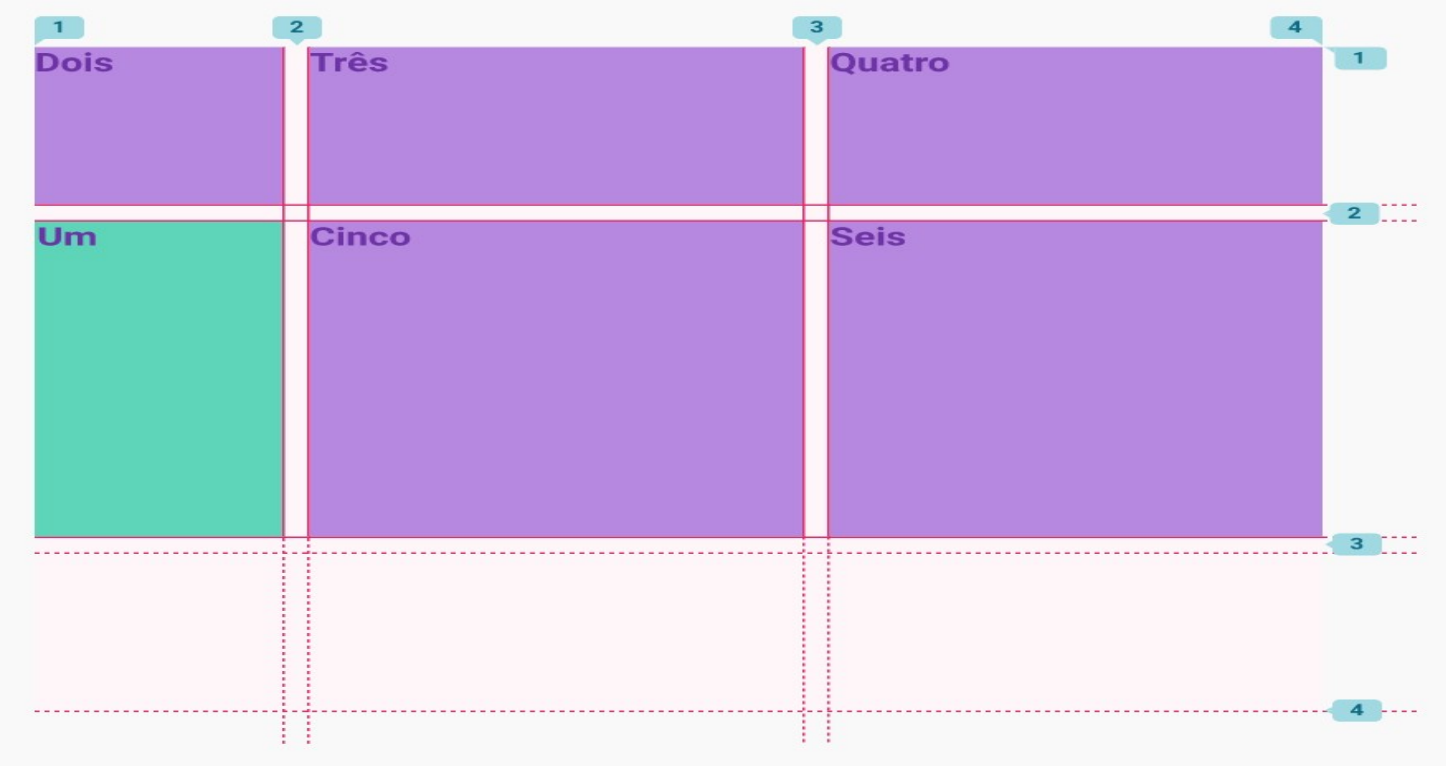
```css
.container {
  display: grid;
  grid-template-rows: 100px 200px 100px;
  grid-template-columns: 100px 200px 200px;

  background: #fff6fa;
  width: max-content; /* impede que o contêiner seja maior
  gap: 10px;
}

.item {
  background: #b688e0;
  color: #6e33a6;
}
```



Exemplo

- Usando grid-row, podemos mudar a posição de um elemento como o .item-1 para qualquer linha.

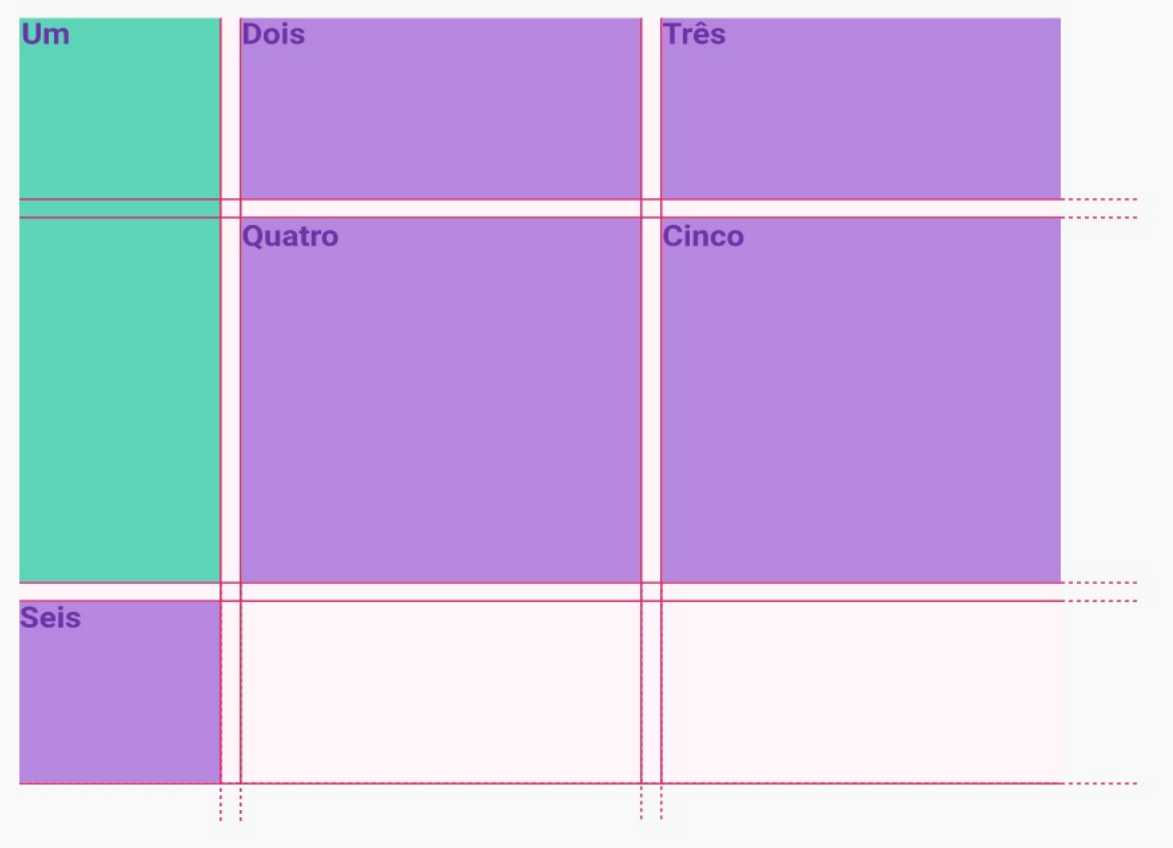
```
.item-1 {
  grid-row: 2;
  background: #5ed4b9; /* apenas para destacar o item-1 */
}
```



Exemplo

- Além de receber o número da linha ou coluna, as propriedades grid-row e grid-column também podem receber valores como 1 / 3 ou 2 / 4. Estes valores definem o número da linha/coluna em que o elemento começa e termina.

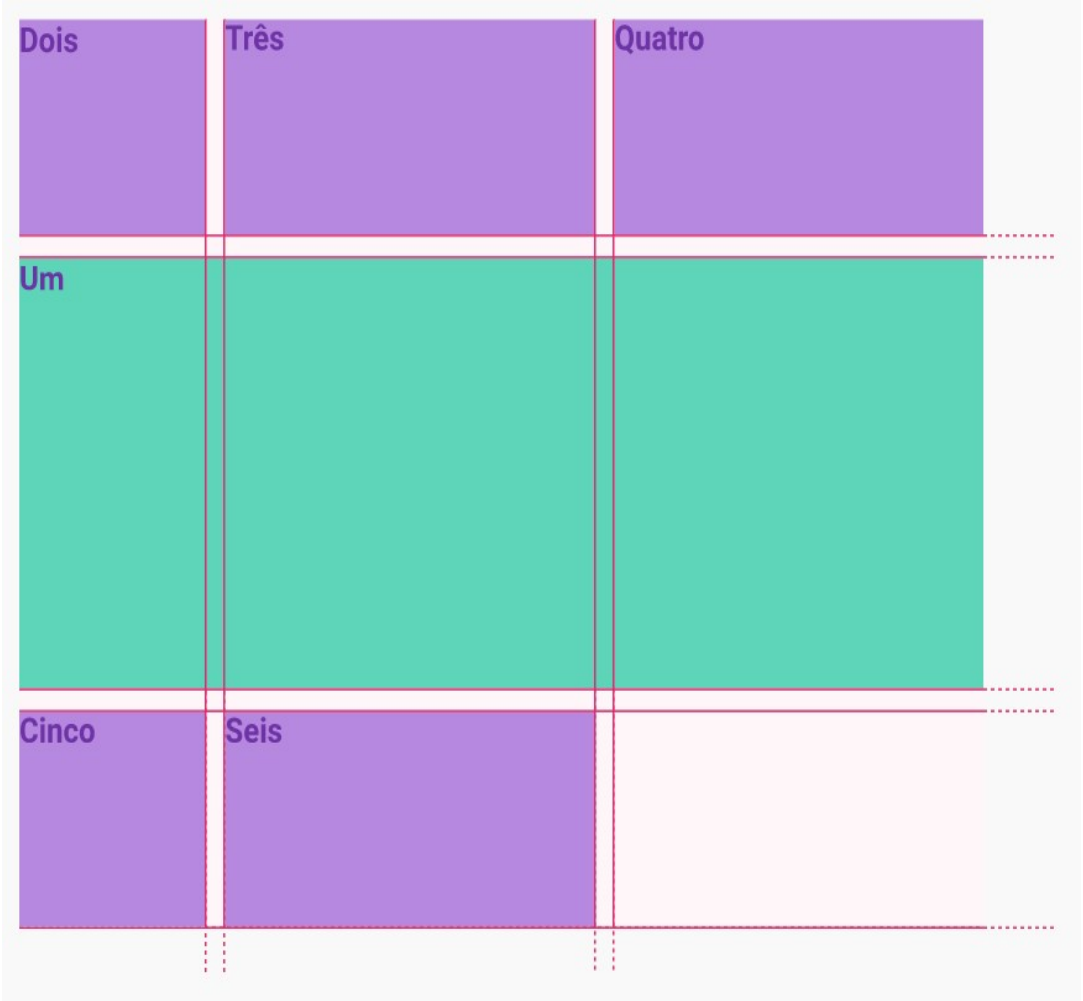
```
.item-1 {
  grid-row: 1 / 3;
  background: #5ed4b9; /* apenas para destacar o item-1 */
}
```



Exemplo

- Podemos usar grid-row e grid-column ao mesmo tempo para posicionar elementos com mais precisão.

```
.item-1 {
  grid-row: 2;
  grid-column: 1 / 4;
  background: #5ed4b9; /* apenas para destacar o item-1 */
}
```



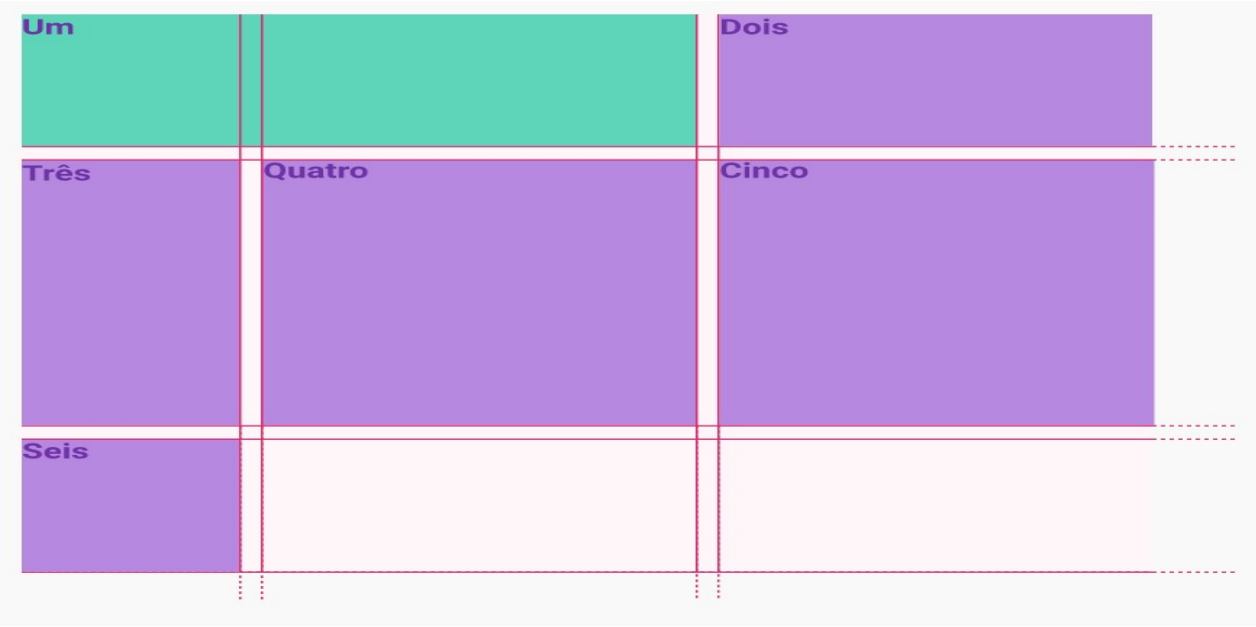
Exemplo

- span em grid-column e grid-row

Além de valores absolutos, grid-column e grid-row também podem receber valores de span, que significa distância em inglês.

Um uso comum para span é definir que um elemento do contêiner grid deve ocupar uma certa quantidade de colunas:

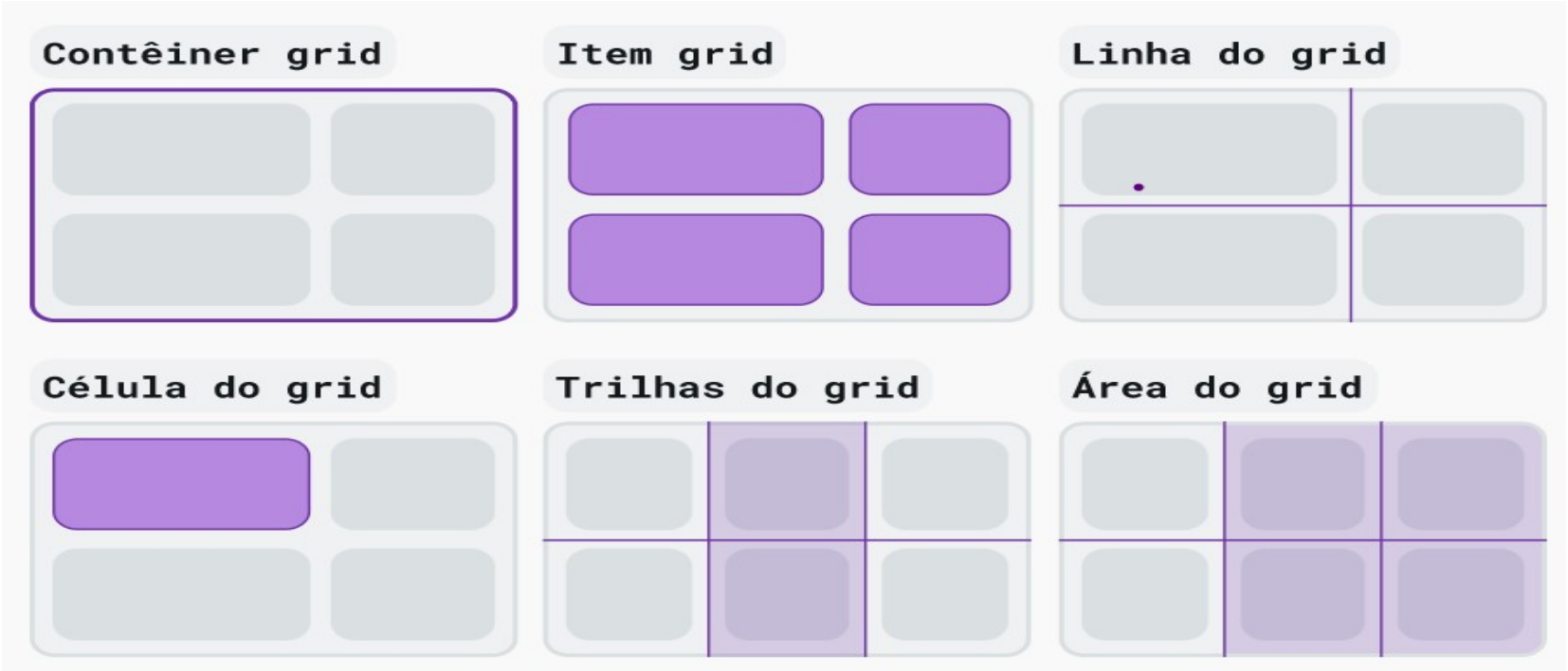
```
.item-1 {
  grid-column: span 2;
  background: #5ed4b9;
}
```



Alinhando Elementos

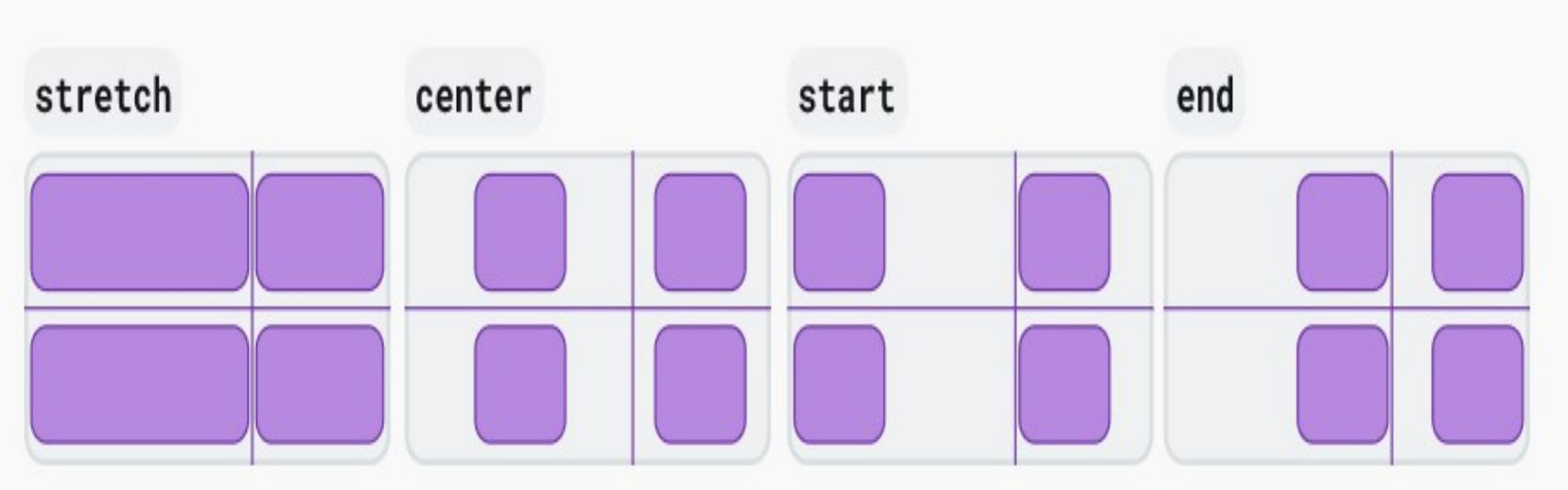
Temos diversas formas de alinhar elementos em um contêiner grid.

Conhecendo cada parte do grid:



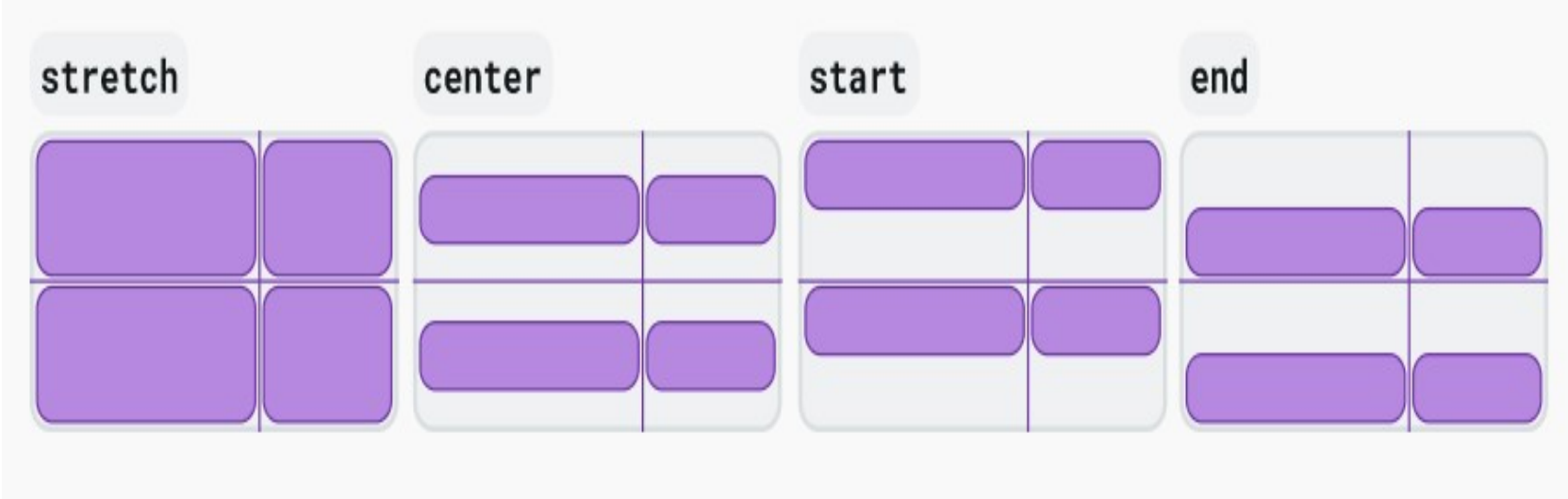
justify-items

A propriedade justify-items alinha todos os itens grid ao longo do eixo inline.



align-items

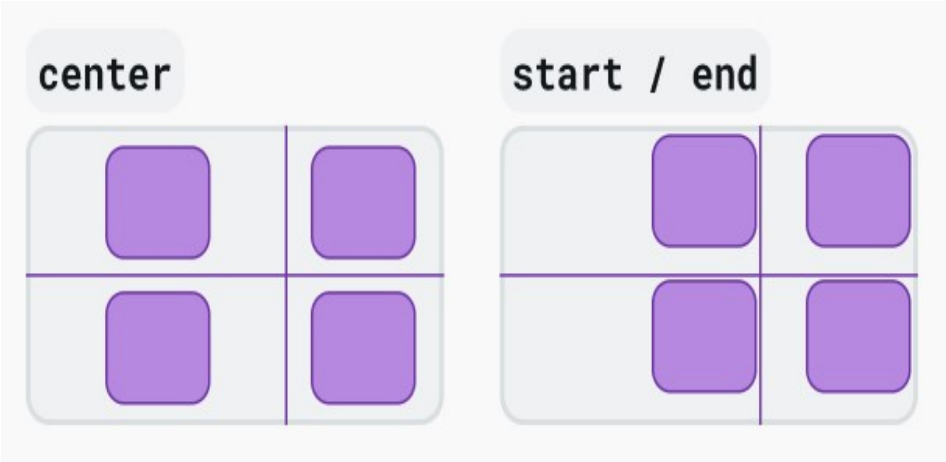
Parecido com a justify-items, a propriedade align-items alinha todos os itens grid que são menores do que o espaço disponível em suas células ao longo do eixo block.



place-items

A propriedade place-items nos ajuda a definir align-items e justify-items ao mesmo tempo. Podemos usar apenas um valor como start ou passar valores do formato <align-items> / <justify-items>.

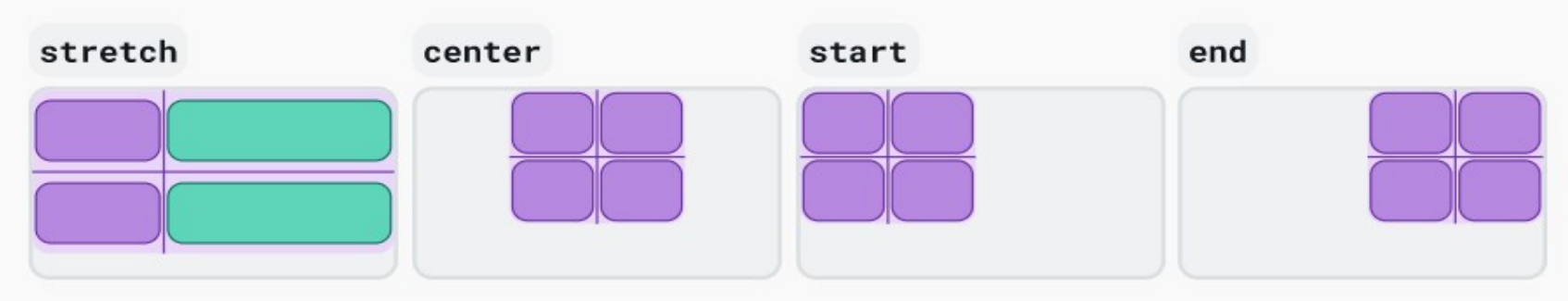
```css
.grid-1 {
  display: grid;
  place-items: center;

  /* é como escrever: */
  align-items: center;
  justify-items: center;
}

.grid-2 {
  display: grid;
  place-items: start / end;

  /* é como escrever: */
  align-items: start;
  justify-items: end;
}
```



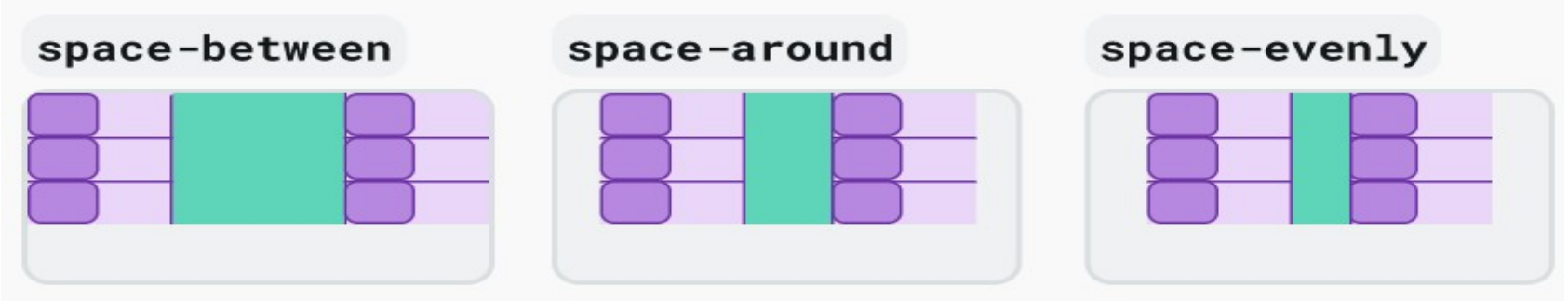
Alinhamento de Conteúdo

justify-content

A propriedade justify-content alinha todo o conteúdo de um contêiner grid ao longo do eixo inline, o que é bem útil quando o conteúdo não preenche todo o espaço dentro do contêiner.

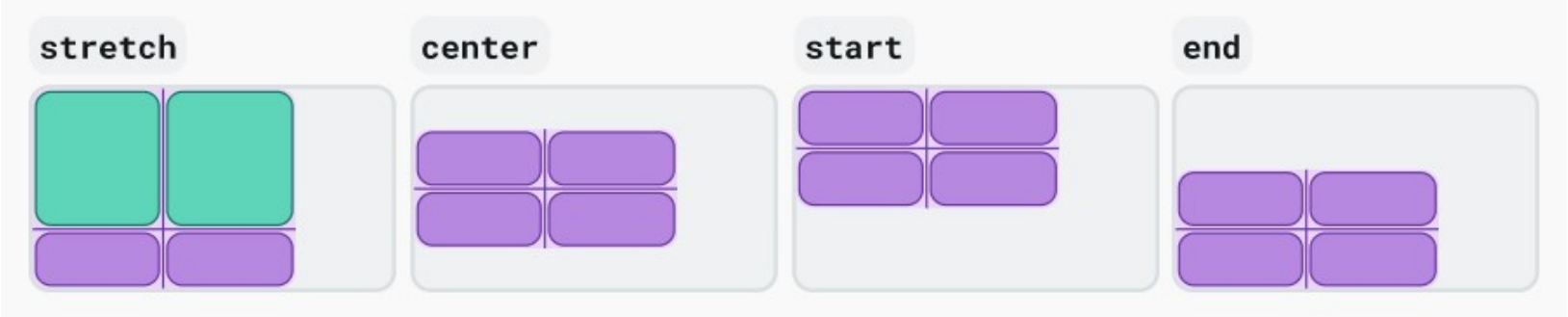


Para espaçar o conteúdo, também temos:



align-content

A propriedade align-content modifica o alinhamento do conteúdo de um contêiner grid quando ele é menor do que o espaço disponível no eixo block.

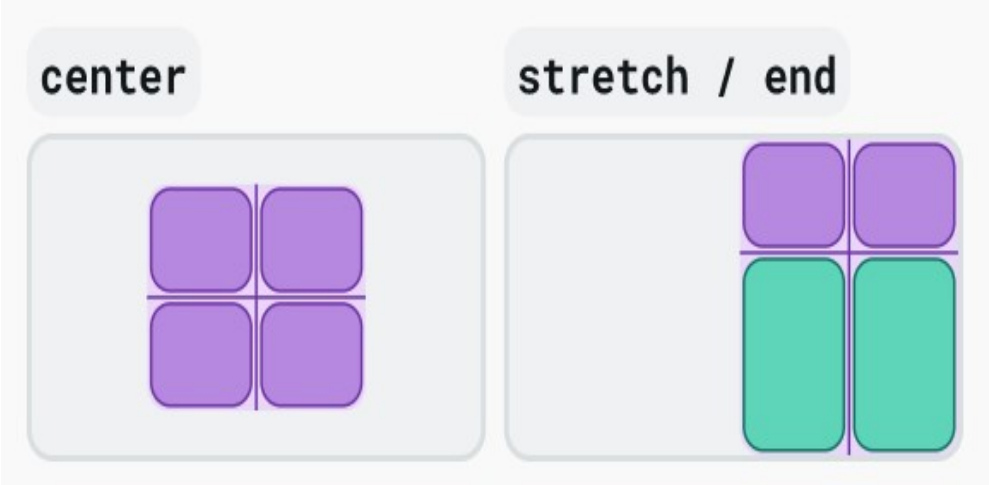


Para espaçar o conteúdo no eixo block, também temos:

place-content

A propriedade place-content é um atalho para definir align-content e justify-content ao mesmo tempo. Você pode usar apenas um valor como start ou passar valores do formato <align-content> / <justify-content>.

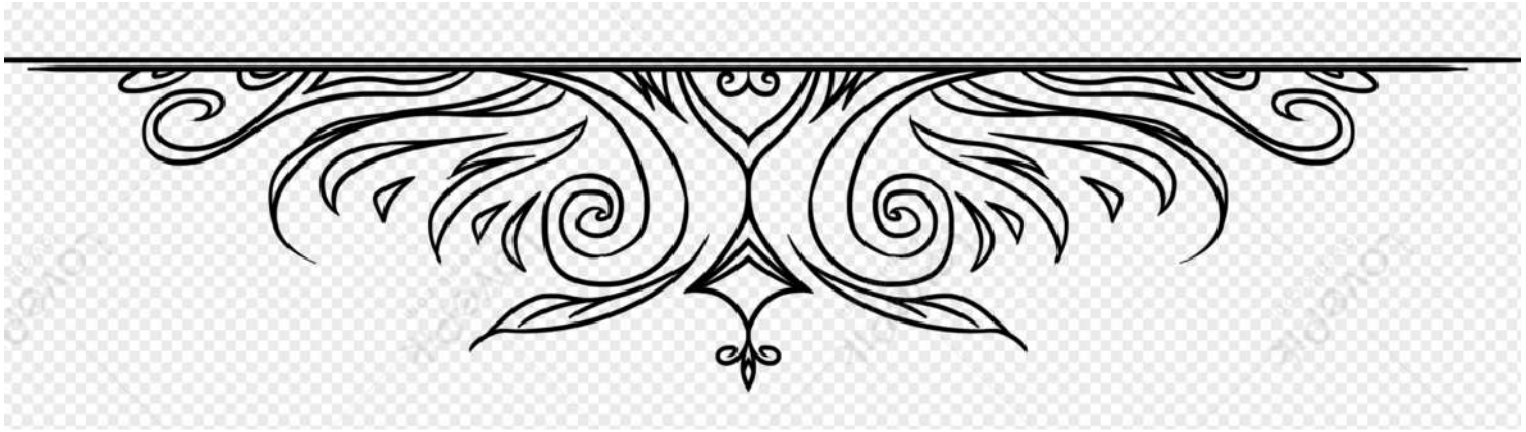
```
.grid-1 {
  display: grid;
  place-content: center;

  /* é como escrever: */
  align-content: center;
  justify-content: center;
}

.grid-2 {
  display: grid;
  place-content: stretch / end;

  /* é como escrever: */
  align-content: stretch;
  justify-content: end;
}
```



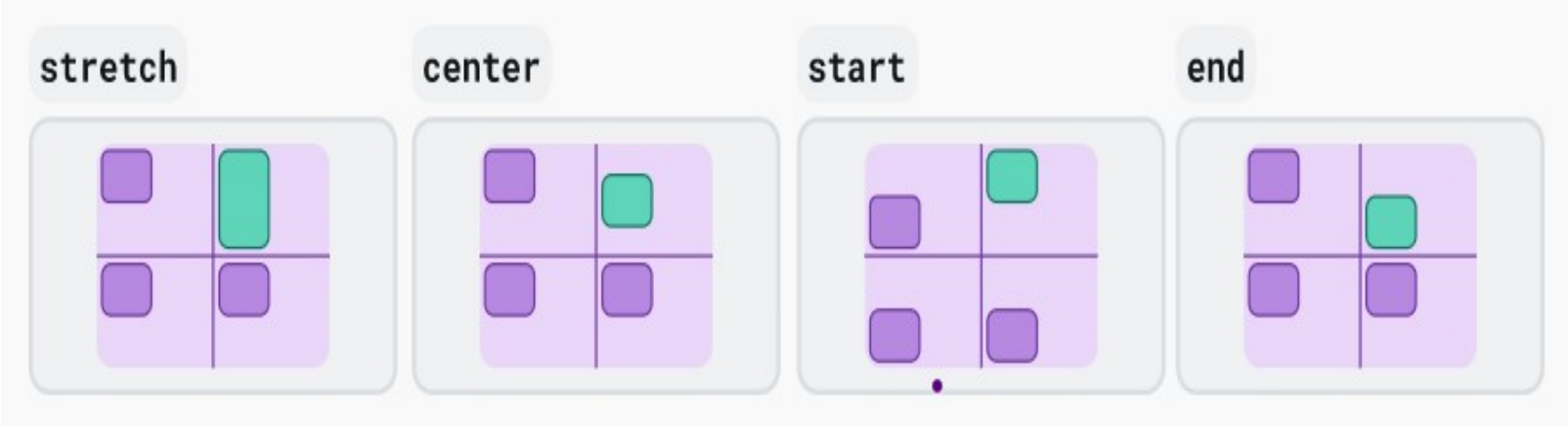
Alinhamento de um item no grid com justify-self e align-self

Quando queremos mudar o alinhamento apenas de um item grid, podemos usar justify-self, align-self e place-self.

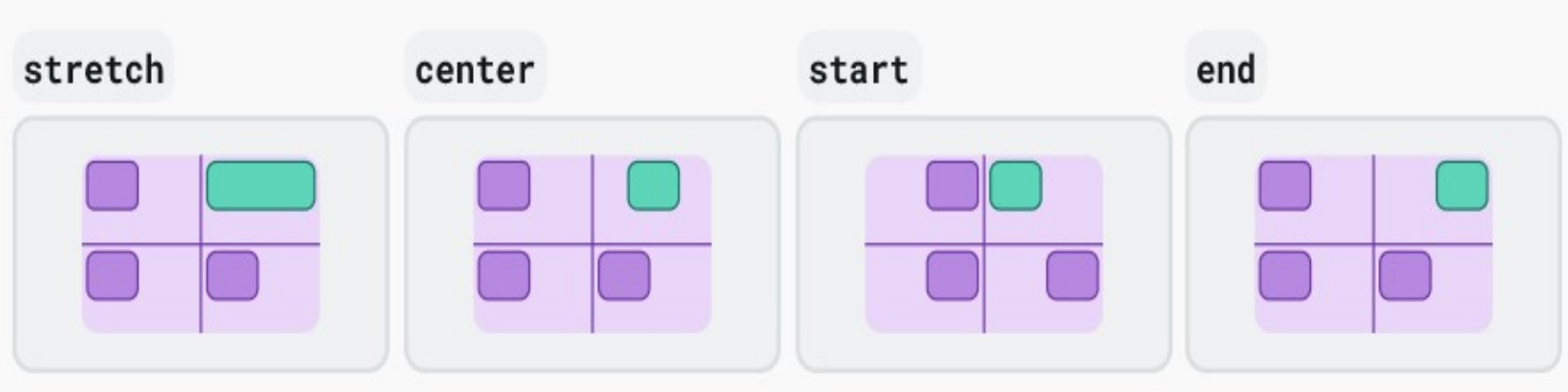
Essas propriedades são aplicadas no item grid.



align-self



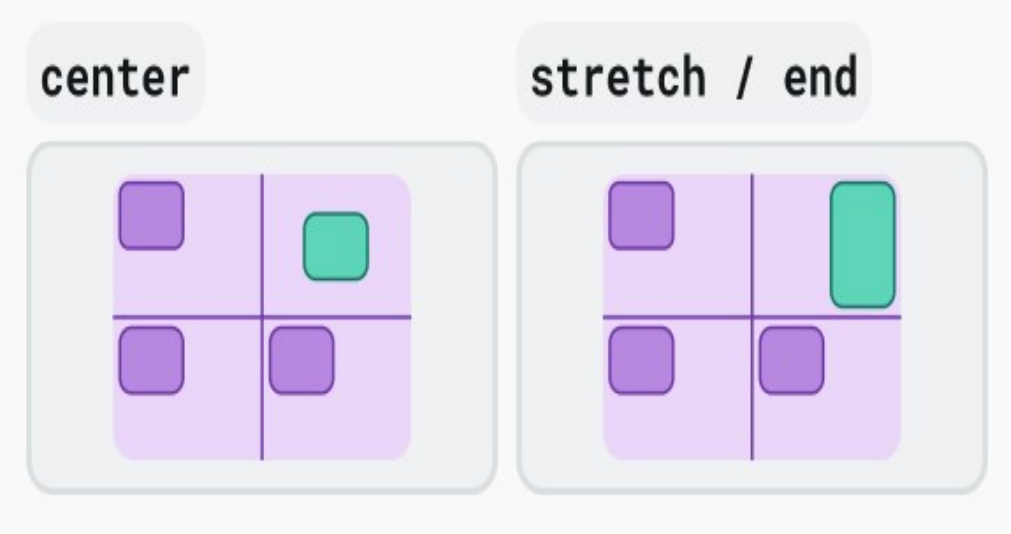
justify-self



place-self

A propriedade place-self é um atalho para definir align-self e justify-self ao mesmo tempo. Você pode usar apenas um valor como start ou passar valores do formato <align-self> / <justify-self>.

```
.grid {
  display: grid;
}

.item-1 {
  place-self: center;

  /* é como escrever: */
  align-self: center;
  justify-self: center;
}

.item-2 {
  place-self: stretch / end;

  /* é como escrever: */
  align-self: stretch;
  justify-self: end;
}
```



Saiba mais:

https://triangulo.dev/posts/usando-grid-template-areas-do-css-grid/

https://triangulo.dev/posts/criando-layouts-responsivos-css-grid/

https://css-tricks.com/snippets/css/complete-guide-grid/

https://developer.mozilla.org/pt-BR/docs/Web/CSS/CSS_Grid_Layout/Basic_Concepts_of_Grid_Layout

https://www.w3schools.com/css/css_grid.asp

https://youtu.be/HN1UjzRSdBk

https://youtu.be/x-4z_u8LcGc

Você encontrará todas as orientações e conteúdos respectivos a cada semana na nossa plataforma!

Bons estudos!!

descodificadas

descodificadas

Sigam nossas redes sociais!!!