# Extra - CSS Grid

quinta-feira, 7 de setembro de 2023 12:22

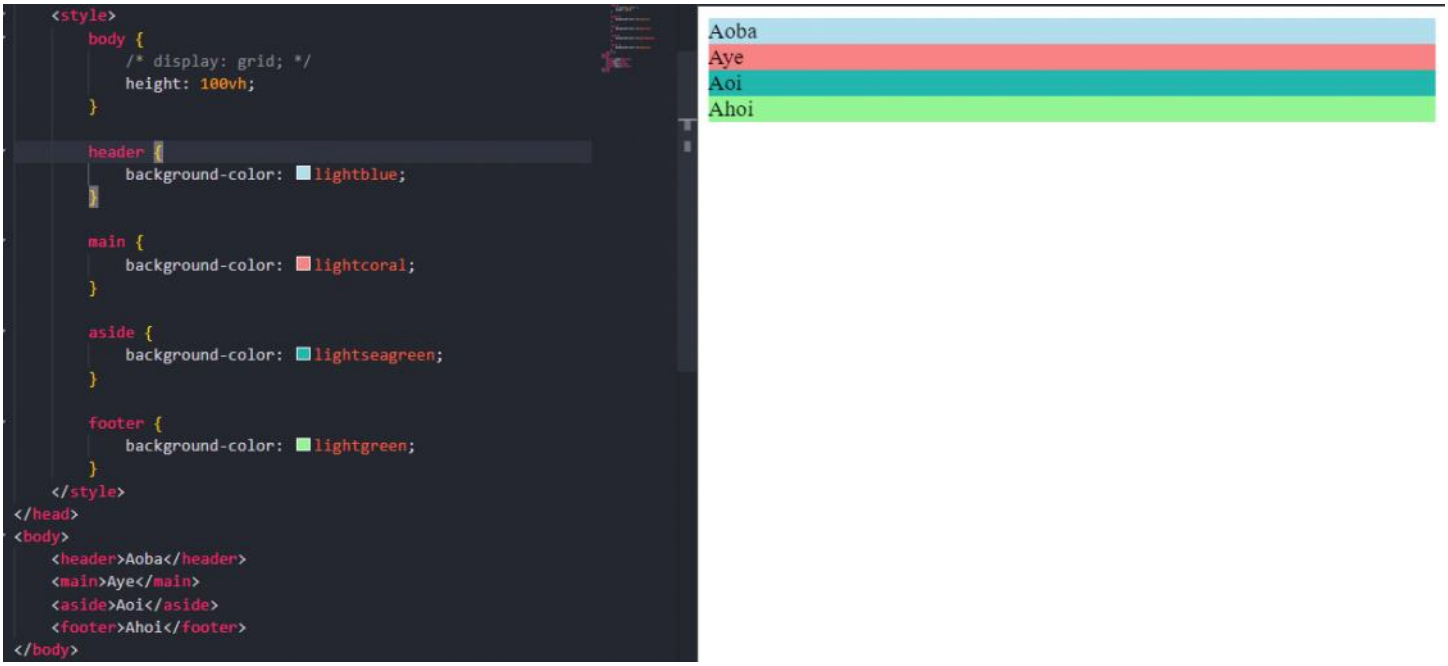
## Grid, não grind

A organização em forma de grade é uma forma comum de organizar os elementos de um site. O figma, por exemplo, permite que adicionemos uma para elaborar um protótipo. A diferença do grid, em relação ao flex, inclusive, é que ele permite organizar os elementos levando em consideração linhas e colunas.

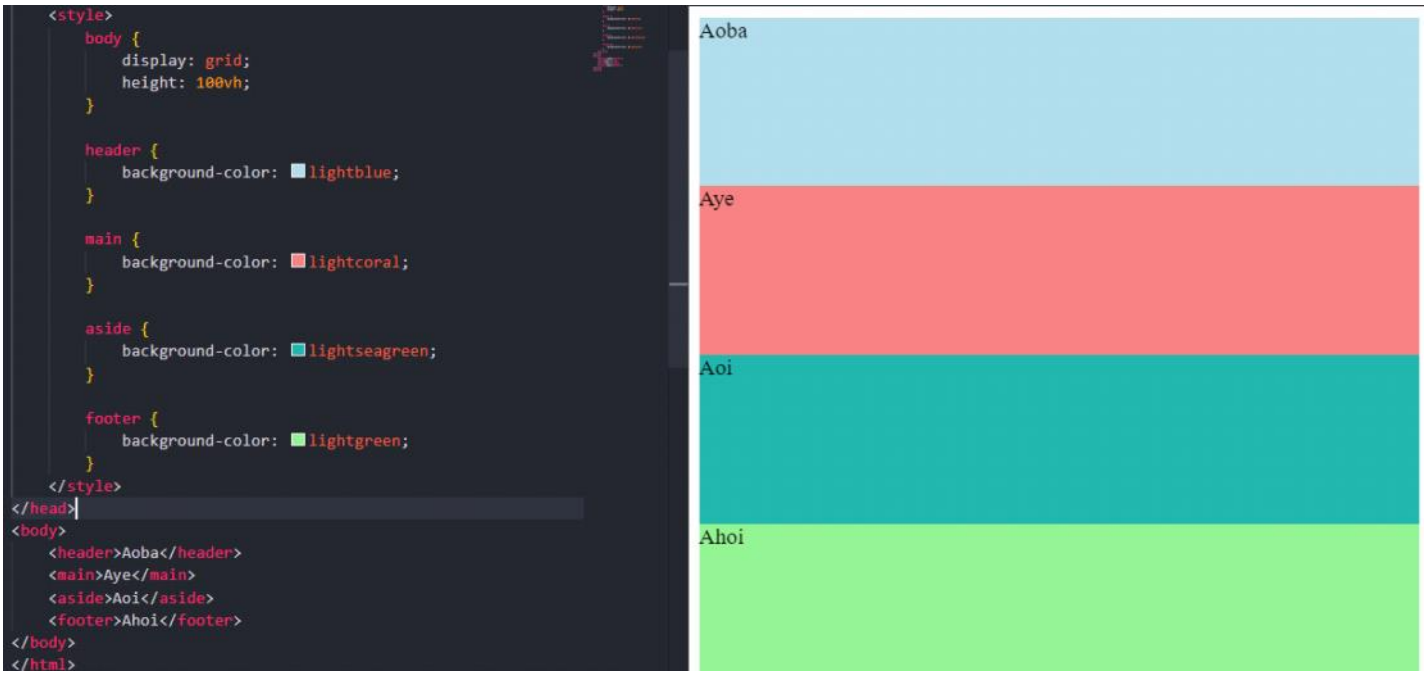
## Display: grid

Quando aplico um "***display: grid***" em uma div que contém outros elementos, todos eles mudam seu comportamento para block, independentemente do comportamento anterior. Esta div pode ser chamada de *container* e os seus elementos internos de *items*.

Antes do grid:

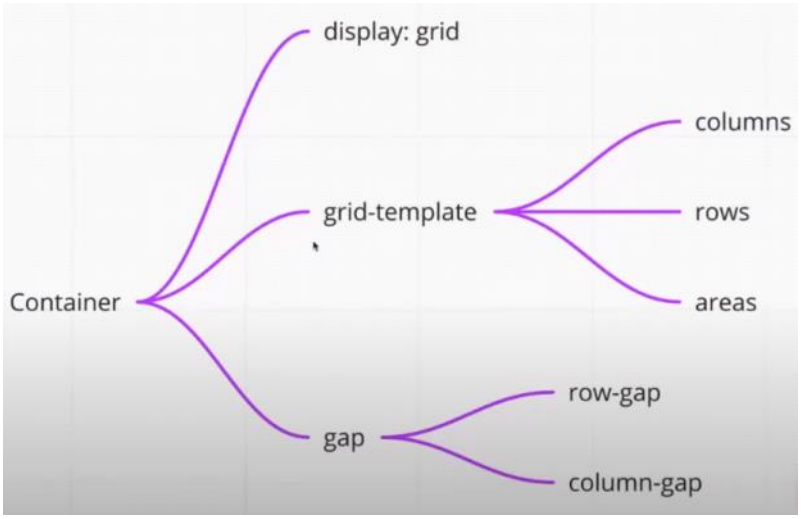



Depois do grid:



## Desbloqueando novas configurações com display grid

Assim como ocorria com o display flex, eu posso obter novos modos de edição com o grid.




Com o *grid-template* eu posso determinar o espaço que as minhas colunas, linhas ou áreas terão. Vamos começar com o *grid-template-columns*. Nele, eu posso indicar o comprimento em cada uma das colunas usando as medidas que preferir. Um detalhe interessante é que se eu inserir a medida "1fr", determino que a coluna receberá a "fração restante" do container para ocupar. Vejamos:

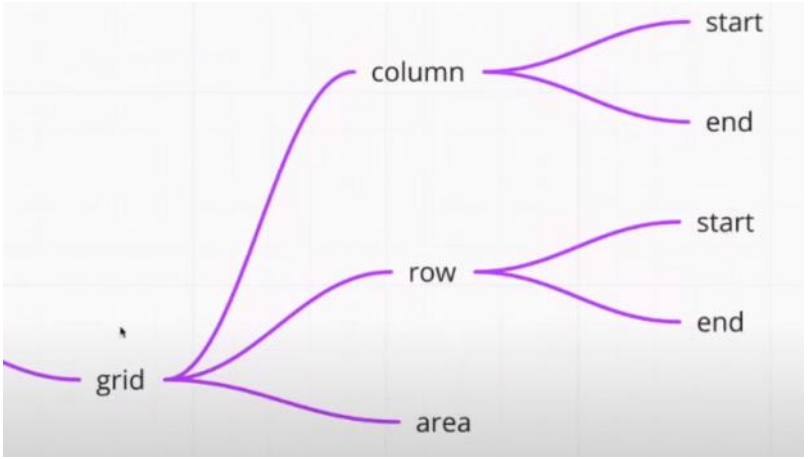

Para o grid-template-rows, eu posso configurar da mesma forma (determinei 3 linhas):

```
grid-template-rows: 20% 1fr 10%;
```

Além disto, tenho o gap, que pode ser aplicado em linhas, colunas ou os dois.

## E os nossos items?

Para eles, tenho as seguintes opções:



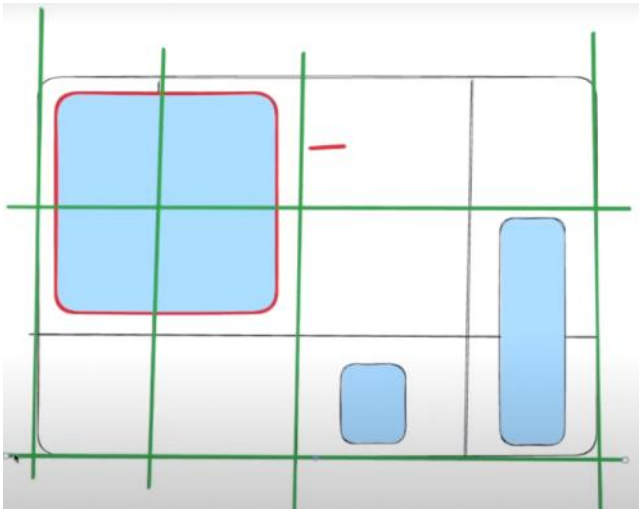

- grid-column-start: <nº da linha>;
- Grid-column-end: <nº da linha>;

- Grid-row-start: <nº linha>;
- Grid-row-end: <nº da linha>;

Mas, cuidado! As linhas/colunas que vamos identificar não são as linhas/colunas que "vemos" no layout. Na verdade, são as linhas/colunas da grade. Geralmente, chamados de linha/coluna o espaço de uma tabela, mas, neste caso, devemos considerar literalmente as linhas que delimitam o espaço da grade.

Assim, se nossa coluna tem dois espaços, ela tem 3 linhas. E se possui 3 espaços que marcam as linhas, possui 4 linhas que delimitam este espaço.

Numeramos as linhas/colunas com números de 1 a n. Se não entendeu ainda, essas linhas/colunas são as marcadas em verde:



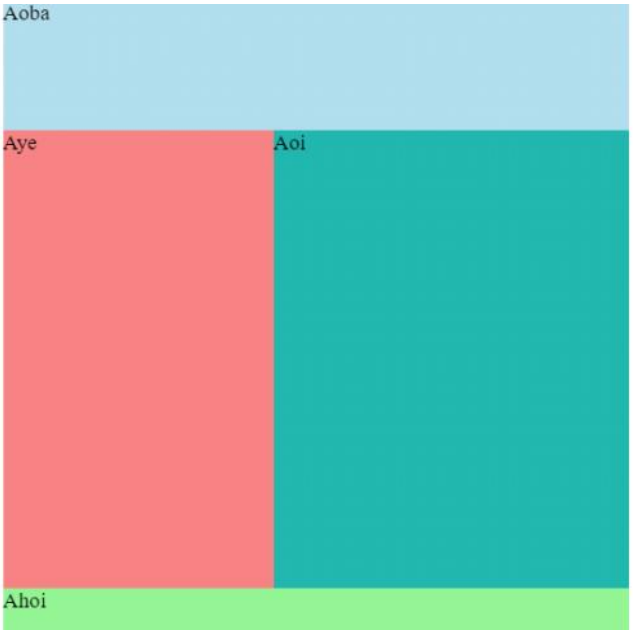
Assim, vamos ao código:

```
grid-column-start: 1;
grid-column-end: 3;
```

E existe um shorthand pra isso: separamos os dois números identificadores com uma barra.

```
footer {
    background-color: lightgreen;
    grid-column: 1/3;
}
```

```
.container {
    display: grid;
    grid-template-columns: 200px 1fr;
    grid-template-rows: 20% 1fr 10%;
}

header {
    background-color: lightblue;
    grid-column-start: 1;
    grid-column-end: 3;
}

main {
    background-color: lightcoral;
}

aside {
    background-color: lightseagreen;
    grid-column: 2/3;
}

footer {
    background-color: lightgreen;
    grid-column: 1/3;
}
style>
>
class="container">
eader>Aoba</header>
```




E se eu usar algo como grid-line: 1/3 no header? O CSS cria novas linhas, mesmo que não fosse essa a intenção.

## Grid-template-areas

E se eu quiser trocar a posição dos elementos? Posso usar o grid-template-areas para indicar quais áreas quero ocupar com cada elemento. Para isso, preciso escrever o nome das áreas de acordo com a posição real do grid.

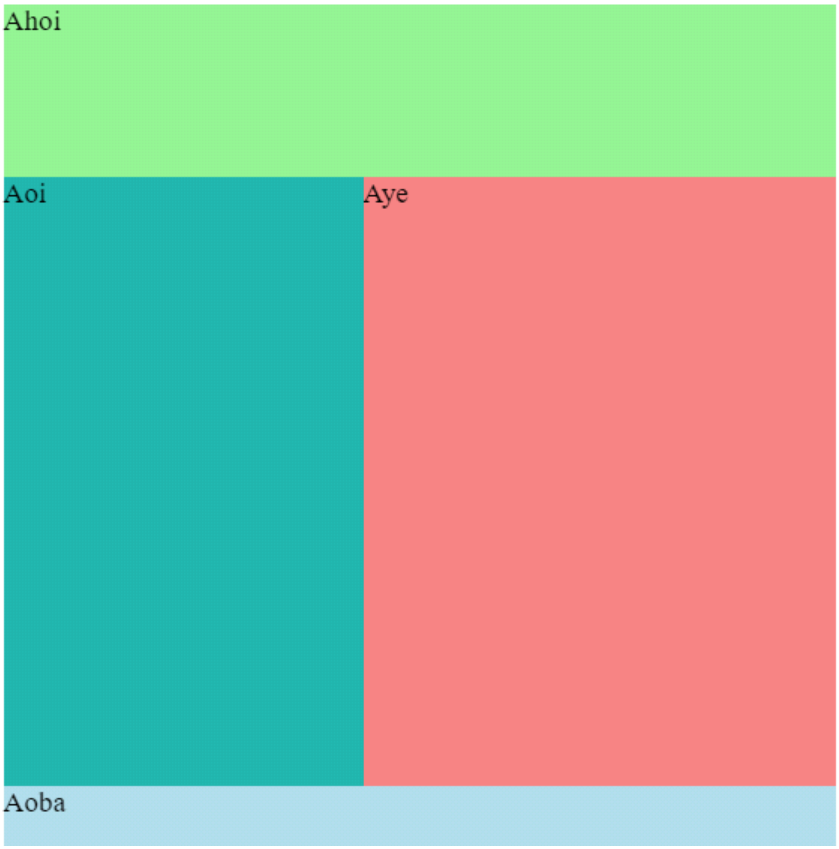
```
grid-template-columns: 200px 1fr;
grid-template-rows: 20% 1fr 10%;

grid-template-areas:
'header header'
'main aside'
'footer footer'
;
```

Importante: o desenho dos nomes tem que seguir o mesmo esquema definido por nós no grid-template-columns/lines.

Para indicar o nome da área em cada elemento, uso grid-area: <nome da área sem aspas>;

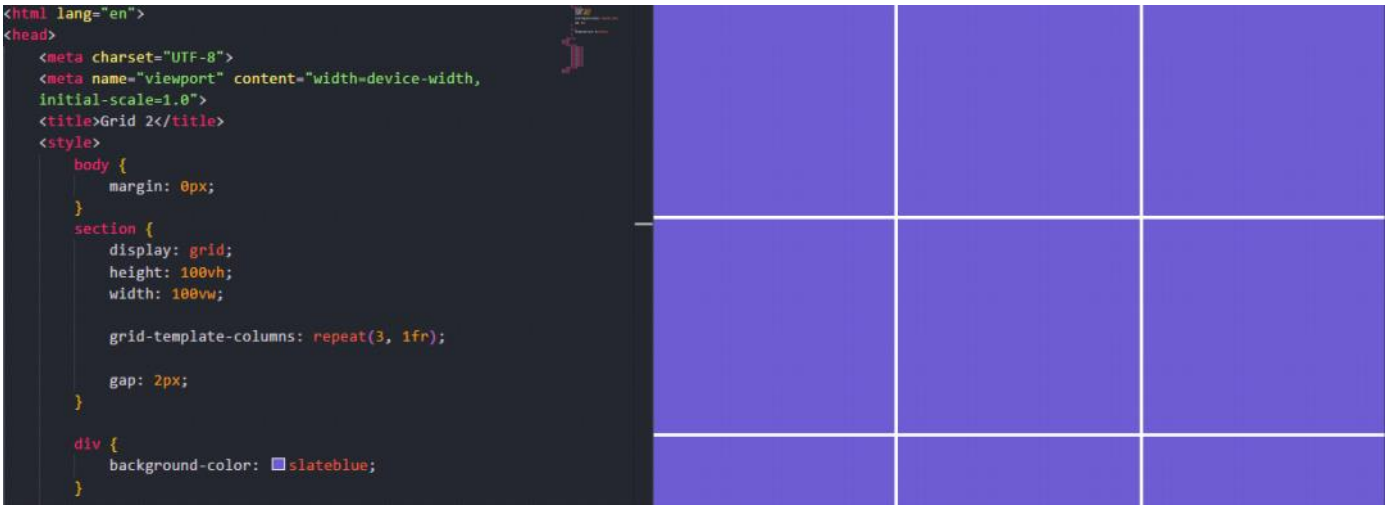
```
header {
    background-color: lightblue;
    grid-column-start: 1;
    grid-column-end: 3;

    grid-area: footer;
}

main {
    background-color: lightcoral;
    grid-area: aside;
}

aside {
    background-color: lightseagreen;
    grid-column: 2/3;

    grid-area: main;
}

footer {
    background-color: lightgreen;
    grid-column: 1/3;

    grid-area: header;
}
```

O resultado:




## Alinhamento em grid

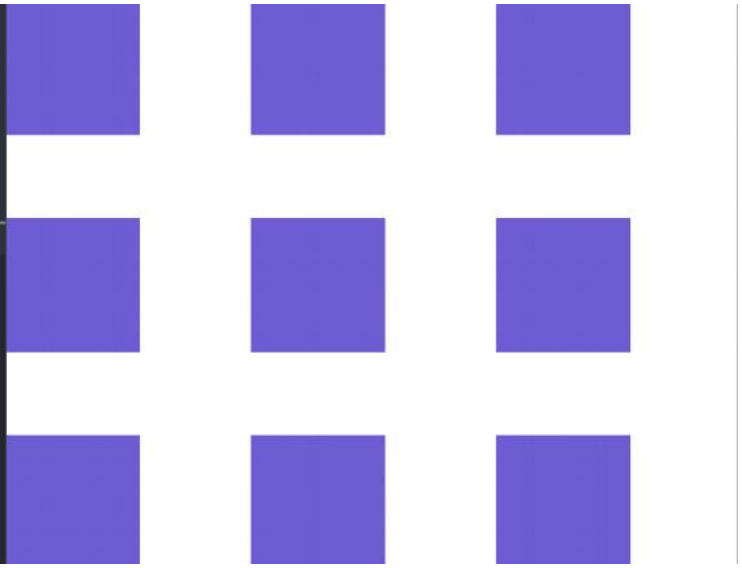
Vamos criar uma nova grade, com 9  dentro de uma <section>.  Uma forma de criar as colunas desta grade é usando a função repeat(). Nela, digo que quero 3 colunas em frações iguais (ou fracionadas).



```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>Grid 2</title>
    <style>
        body {
            margin: 0px;
        }
        section {
            display: grid;
            height: 100vh;
            width: 100vw;

            grid-template-columns: repeat(3, 1fr);

            gap: 2px;
        }

        div {
            background-color: slateblue;
        }
```

No entanto, se a minha div for menor que o espaço dedicado a ela pelas colunas, ela vai automaticamente para a posição inicial da coluna e da linha (start).

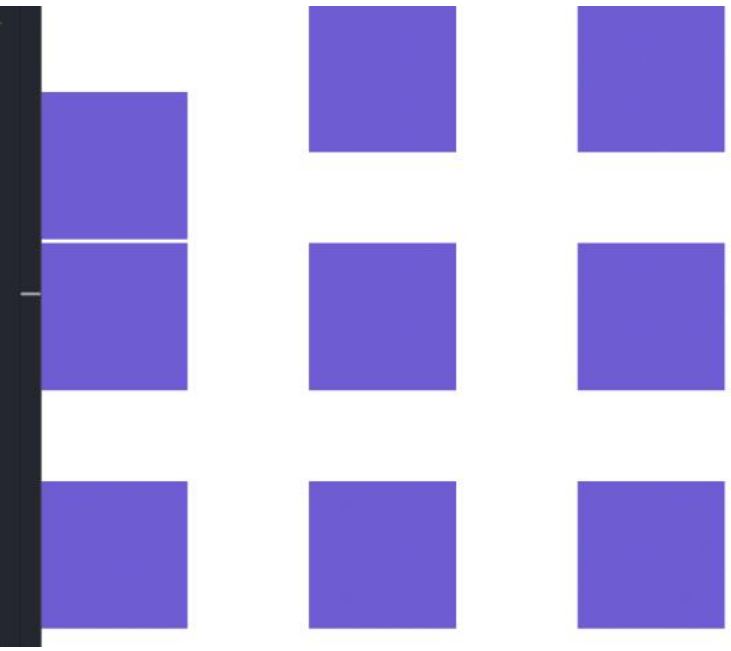
```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>Grid 2</title>
    <style>
        body {
            margin: 0px;
        }
        section {
            display: grid;
            height: 100vh;
            width: 100vw;

            grid-template-columns: repeat(3, 1fr);

            gap: 2px;
        }

        div {
            background-color: slateblue;

            width: 100px;
            height: 100px;
        }
```



Para organizar as divs dentro de cada uma das colunas delas, poso usar o align. Veja que align é sempre utilizado para alinhar elementos verticalmente e o justify, horizontalmente.

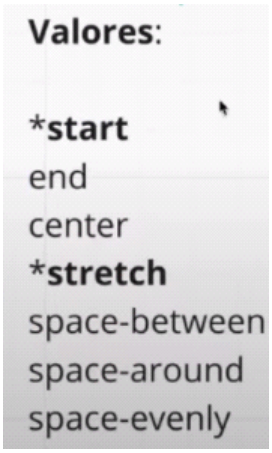
Selecionando a primeira div, vamos alinhá-la no final do eixo vertical.

```
        div:nth-child(1) {
            align-self: end;
        }
    </style>
ead>
dy>
    <section>
        <div></div>
        <div></div>
        <div></div>
        <div></div>
        <div></div>
        <div></div>
        <div></div>
        <div></div>
        <div></div>
    </section>
ody>
tml>
```



Posso usar também somente o "place-self: center" para posicionar todas as divs na posição central (verticalmente e horizontalmente). Outra curiosidade é o "align-self: strech" para esticar o elemento para ocupar todo o espaço reservado a ele (é o padrão do grid).

Fora isto, posso usar também o justify-content e o align-content para escolher a posição do grid como um todo (e até delimitar o espaço, usando space-between ou space-around).



E vamos ver uma forma de centralizar divs


```
<style>
    body {
        height: 100vh;

        /* Centralizando a div */
        display: grid;
        place-items: center;
    }

    div#app {
        outline: 1px solid red;

        height: 40vh;
        width: 50vw;

        /* Centralizando o conteúdo da div */
        display: grid;
        place-content: center;
    }
</style>
<div id="app">
    <h1>Texto</h1>
</div>
```