O Jogo do Layout Perfeito

Sorry,
I’m fixed

Flex

Flex

Unidimensional (linha ou coluna)

Flex

Unidimensional (linha ou coluna)

Distribuição dinâmica

Alinhamento de elementos dentro de um container

Flex

Unidimensional (linha ou coluna)

Melhor para distribuição dinâmica

Alinhamento de elementos dentro de um container

Estrutura

O eixo default é o horizontal (row) , mas também temos o eixo vertical (column). Conseguimos definir O eixo utilizando a propriedade flex-direction.

O espaço entre os filhos também pode ser ajustado através da propriedade gap.

Alinhamento

justify-content - Alinha os itens no eixo principal

flex-start → items no início.

center → items no centro.

space-between → espaço igual entre os items.

space-around → espaço ao redor dos items.

flex-end → items no fim.

Alinhamento

align-items - Alinha os itens no eixo transversal

flex-start → items no início.

center → items no centro.

flex-end → items no fim.

stretch →

Alinhamento

Align-self - Alinha individualmente um item no container

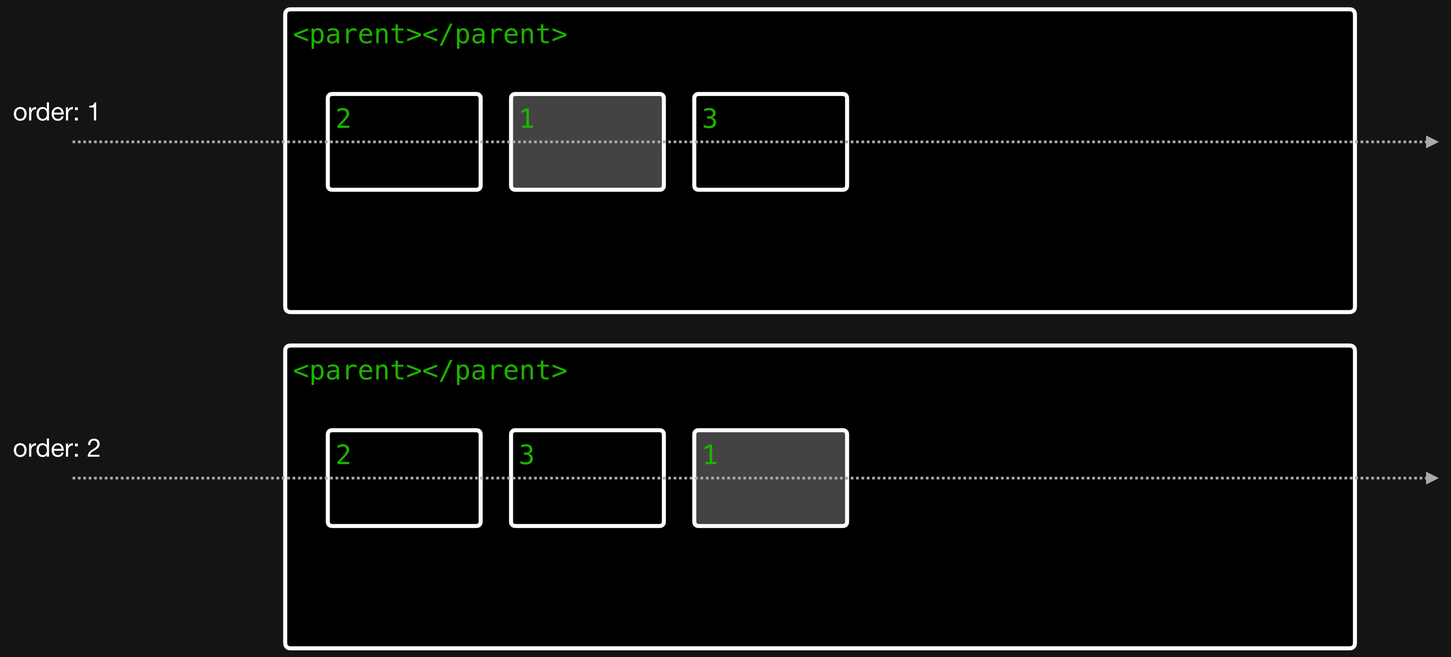
flex-start → items no início.

center → items no centro.

flex-end → items no fim.

Alinhamento

Order - Define a ordem dos items independentemente da ordem do html



Alinhamento

flex-wrap - permite que os items quebrem para uma nova linha senão houver espaço no eixo principal do container

wrap → Permite ajuste

no-wrap → restringe o ajuste

Grid

Grid

Bidimensional (linhas e colunas)

Melhor para layouts fixos

Construção de layouts completos

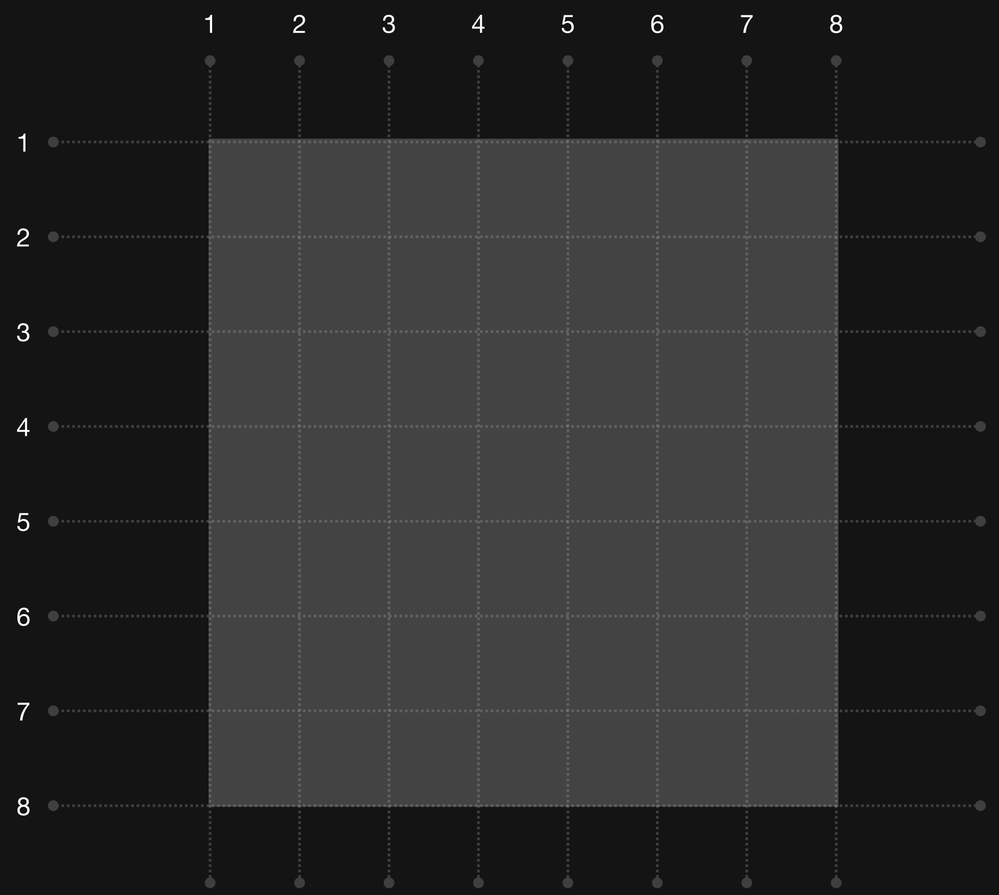
Construção

A construção da grid começa no pai com a aplicação da propriedade display: grid.

Depois, é necessário definir a estrutura de columns e rows que precisamos para a planta do layout.

grid-template-columns: 1fr 1fr 1fr 1fr 1fr 1fr 1fr 1fr;
Grid-template-rows: 1fr 1fr 1fr 1fr 1fr 1fr 1fr 1fr;

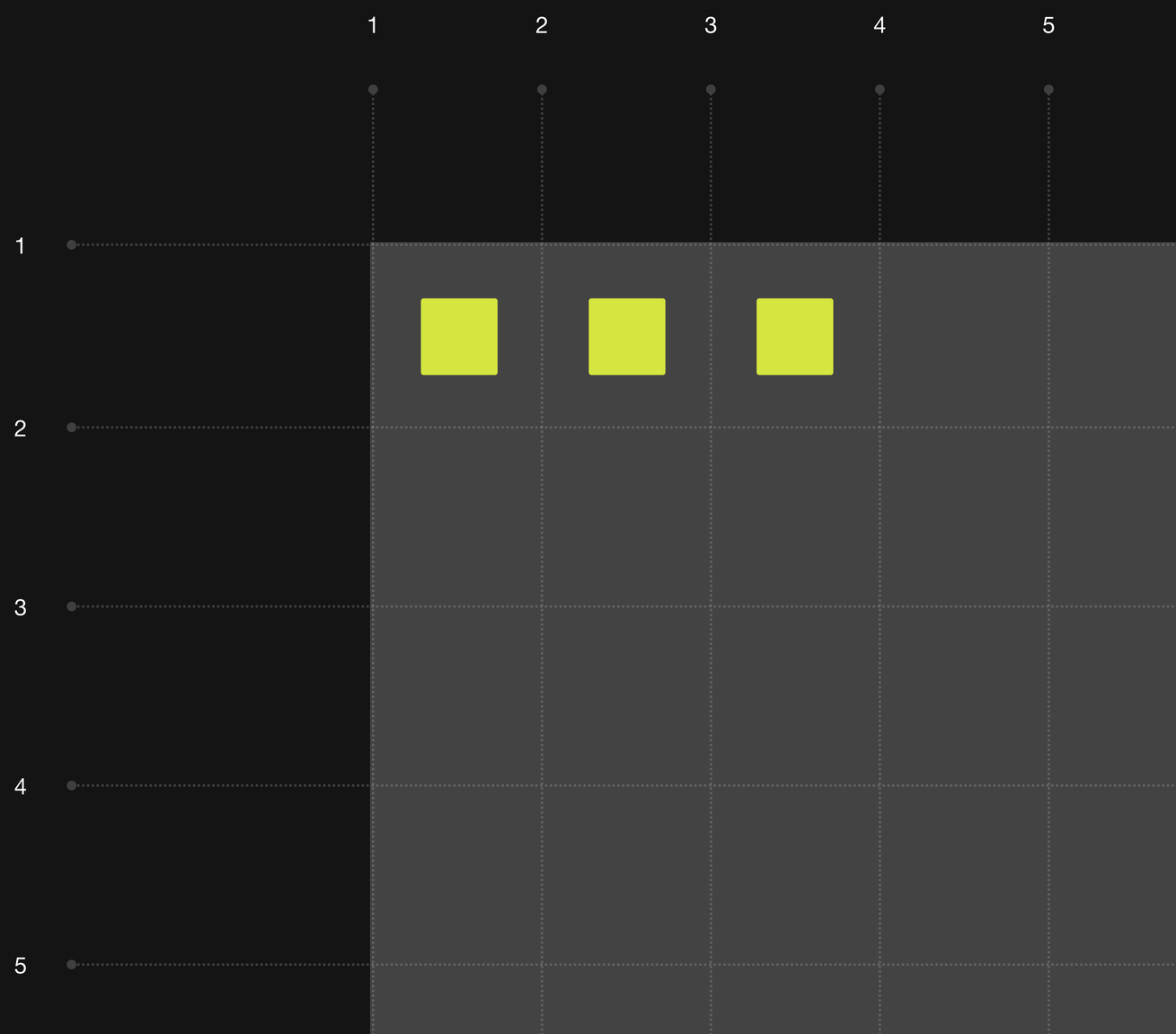
<fr> representa uma unidade fracional, ou seja, uma parte igual do container.



Relação Parent > child

À diferença do flex, que cria um seguimento fluído entre os siblings, o grid cria espaços onde os siblings habitam.

Há uma relação maior entre Parent > child do que entre siblings



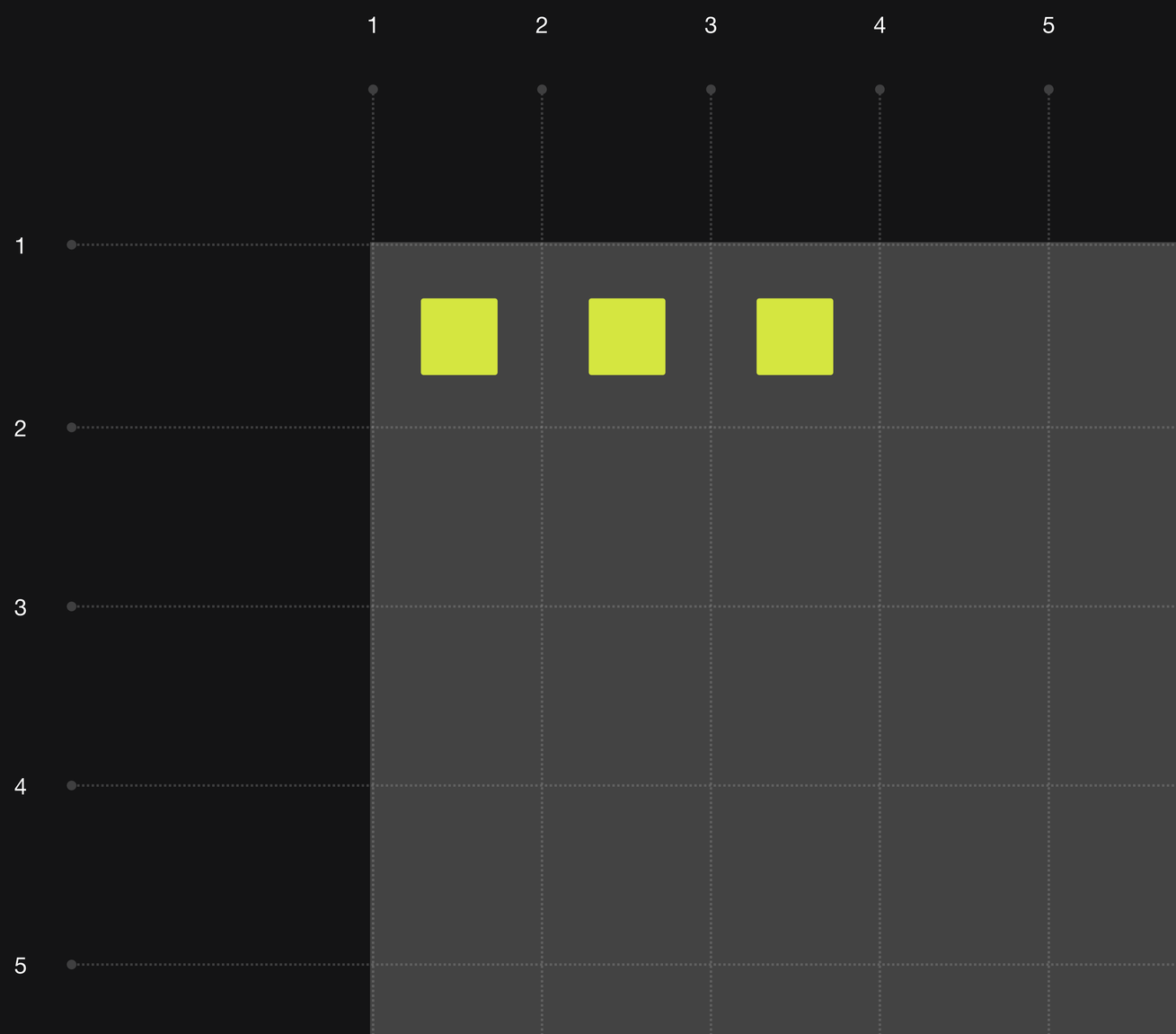
Relação Parent > child

À diferença do flex, que cria um seguimento fluído entre os siblings, o grid cria espaços onde os siblings habitam.

Há uma relação maior entre Parent > child do que entre siblings.

A propriedade justify-items permite alinhar os siblings dentro do seus perímetros e não em relação uns aos outros.

justify-items: start | end | center | stretch



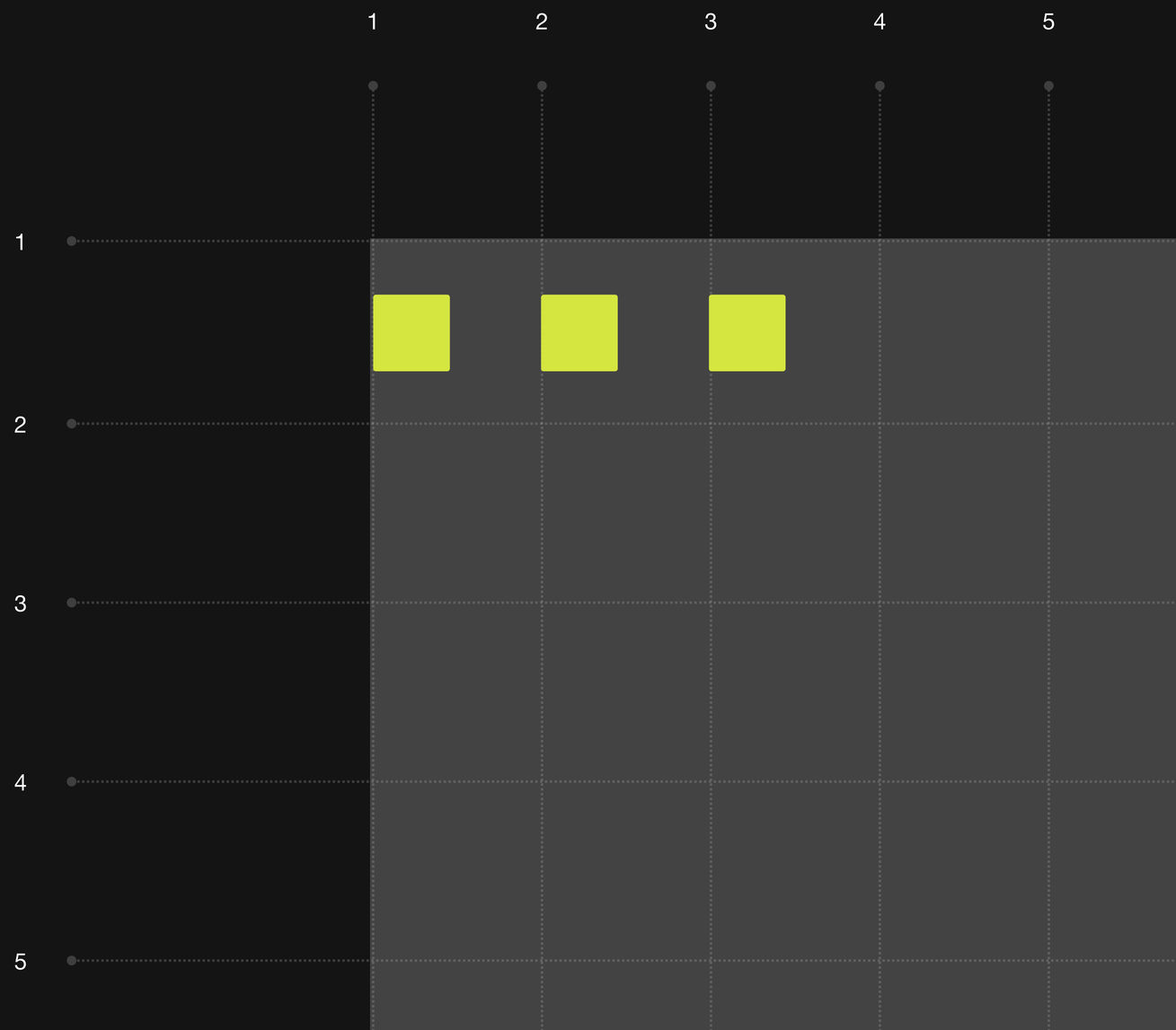
Relação Parent > child

À diferença do flex, que cria um seguimento fluído entre os siblings, o grid cria espaços onde os siblings habitam.

Há uma relação maior entre Parent > child do que entre siblings.

A propriedade justify-items permite alinhar os siblings dentro do seus perímetros e não em relação uns aos outros.

justify-items: start | end | center | stretch



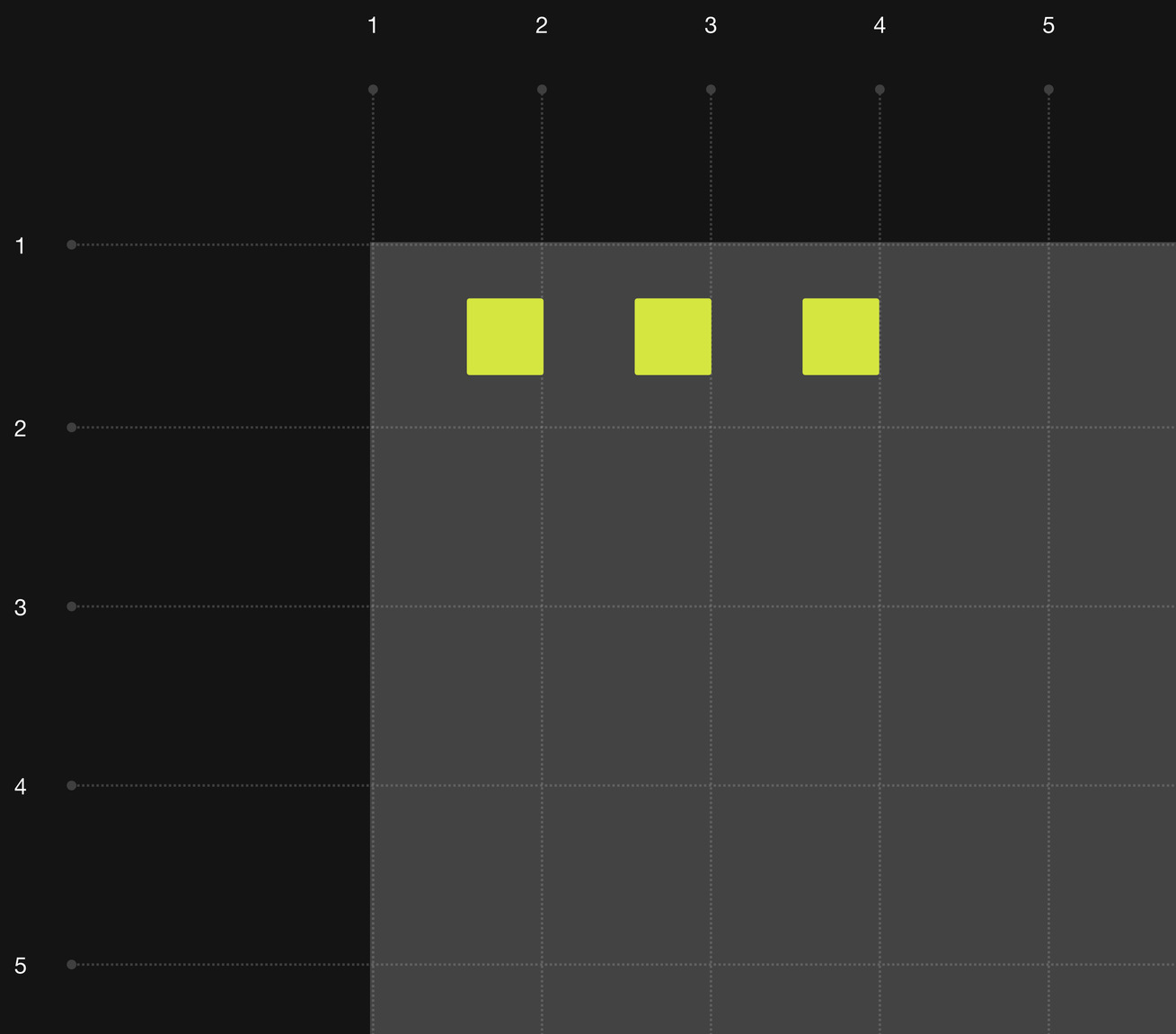
Relação Parent > child

À diferença do flex, que cria um seguimento fluído entre os siblings, o grid cria espaços onde os siblings habitam.

Há uma relação maior entre Parent > child do que entre siblings.

A propriedade justify-items permite alinhar os siblings dentro do seus perímetros e não em relação uns aos outros.

justify-items: start | end | center | stretch



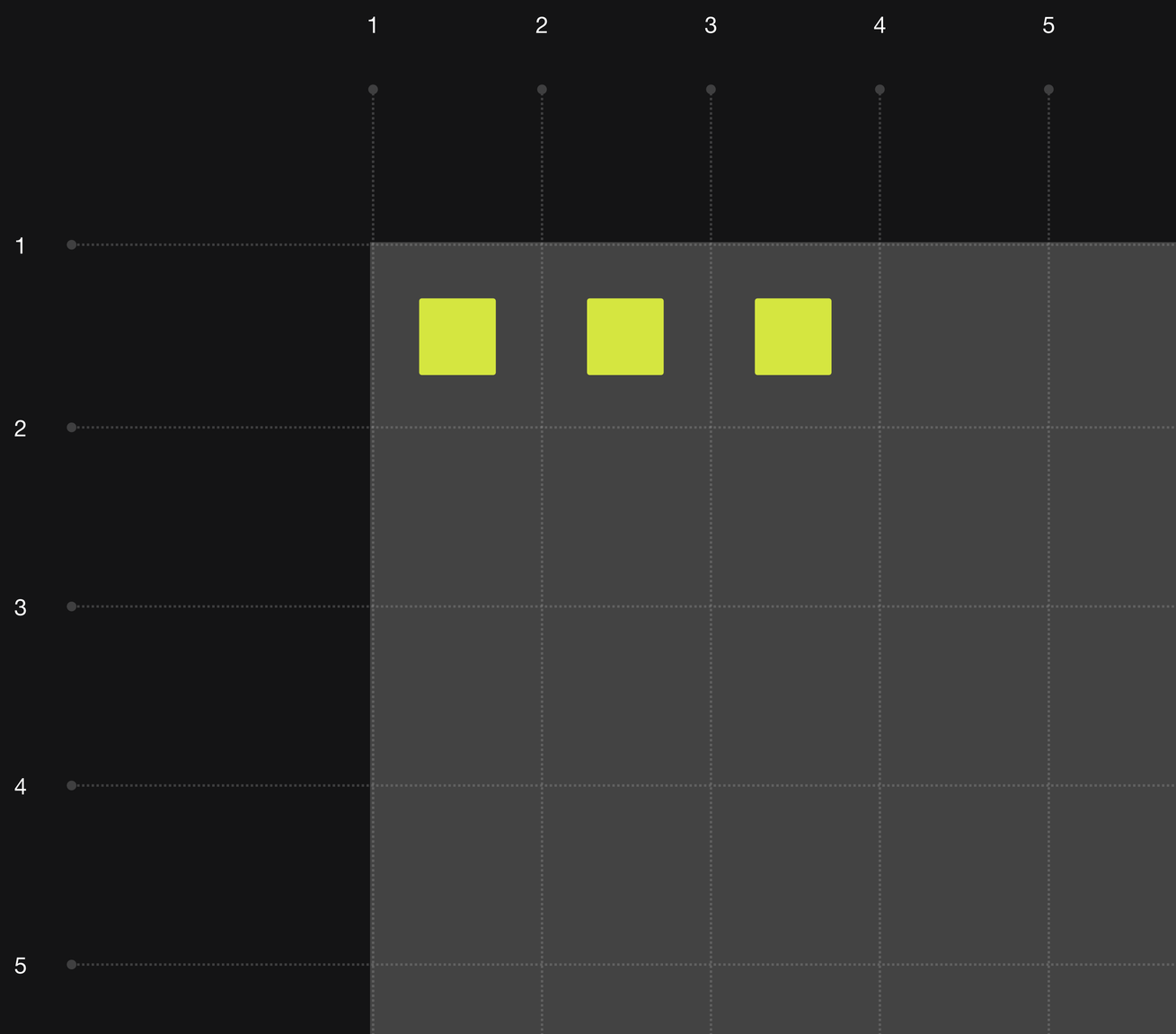
Relação Parent > child

À diferença do flex, que cria um seguimento fluído entre os siblings, o grid cria espaços onde os siblings habitam.

Há uma relação maior entre Parent > child do que entre siblings.

A propriedade justify-items permite alinhar os siblings dentro do seus perímetros e não em relação uns aos outros.

justify-items: start | end | center | stretch



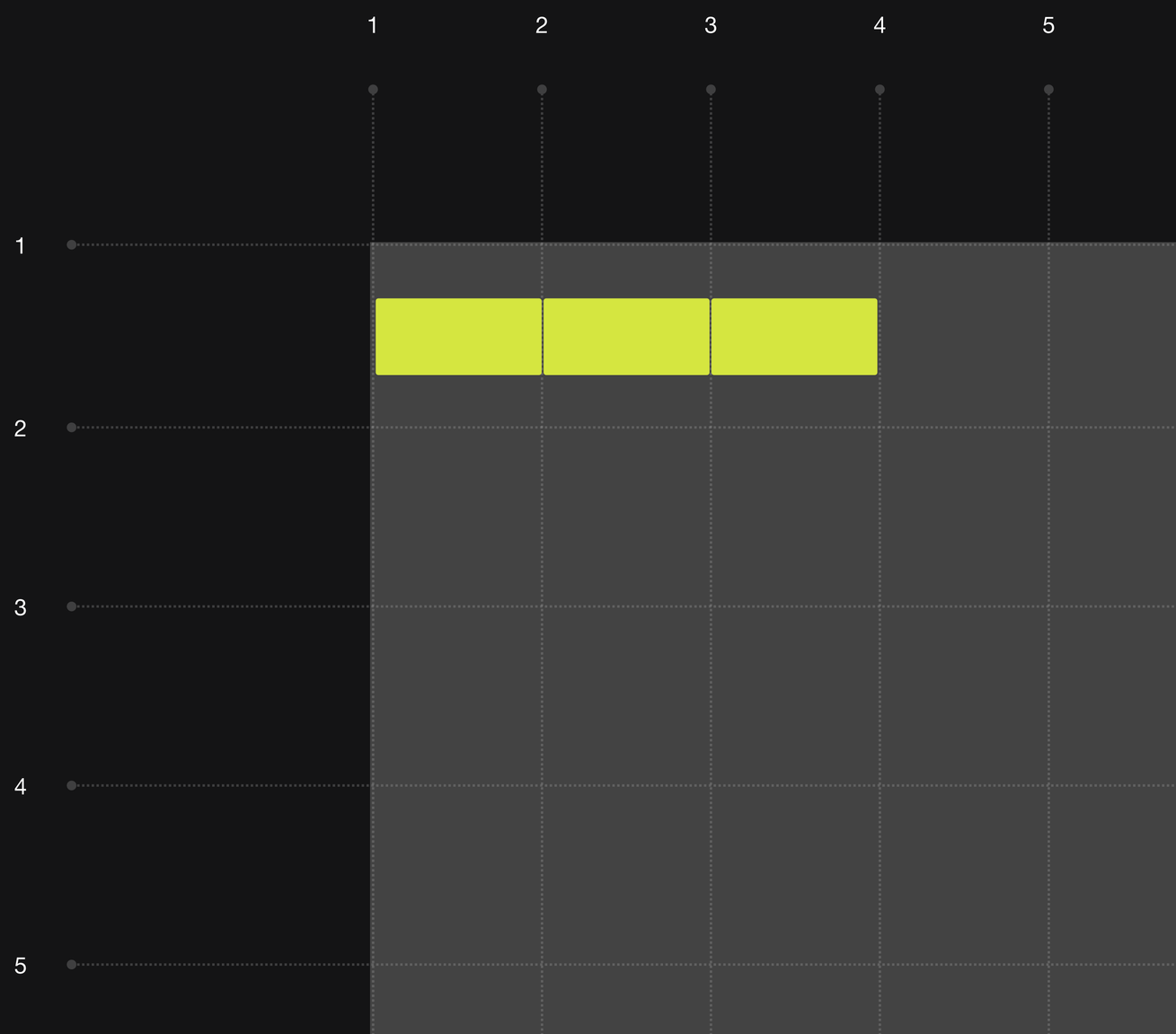
Relação Parent > child

À diferença do flex, que cria um seguimento fluído entre os siblings, o grid cria espaços onde os siblings habitam.

Há uma relação maior entre Parent > child do que entre siblings.

A propriedade justify-items permite alinhar os siblings dentro do seus perímetros e não em relação uns aos outros.

justify-items: start | end | center | stretch



Relação Parent > child

À diferença do flex, que cria um seguimento fluído entre os siblings, o grid cria espaços onde os siblings habitam.

Há uma relação maior entre Parent > child do que entre siblings.

A propriedade justify-items permite alinhar os siblings dentro do seus perímetros e não em relação uns aos outros.

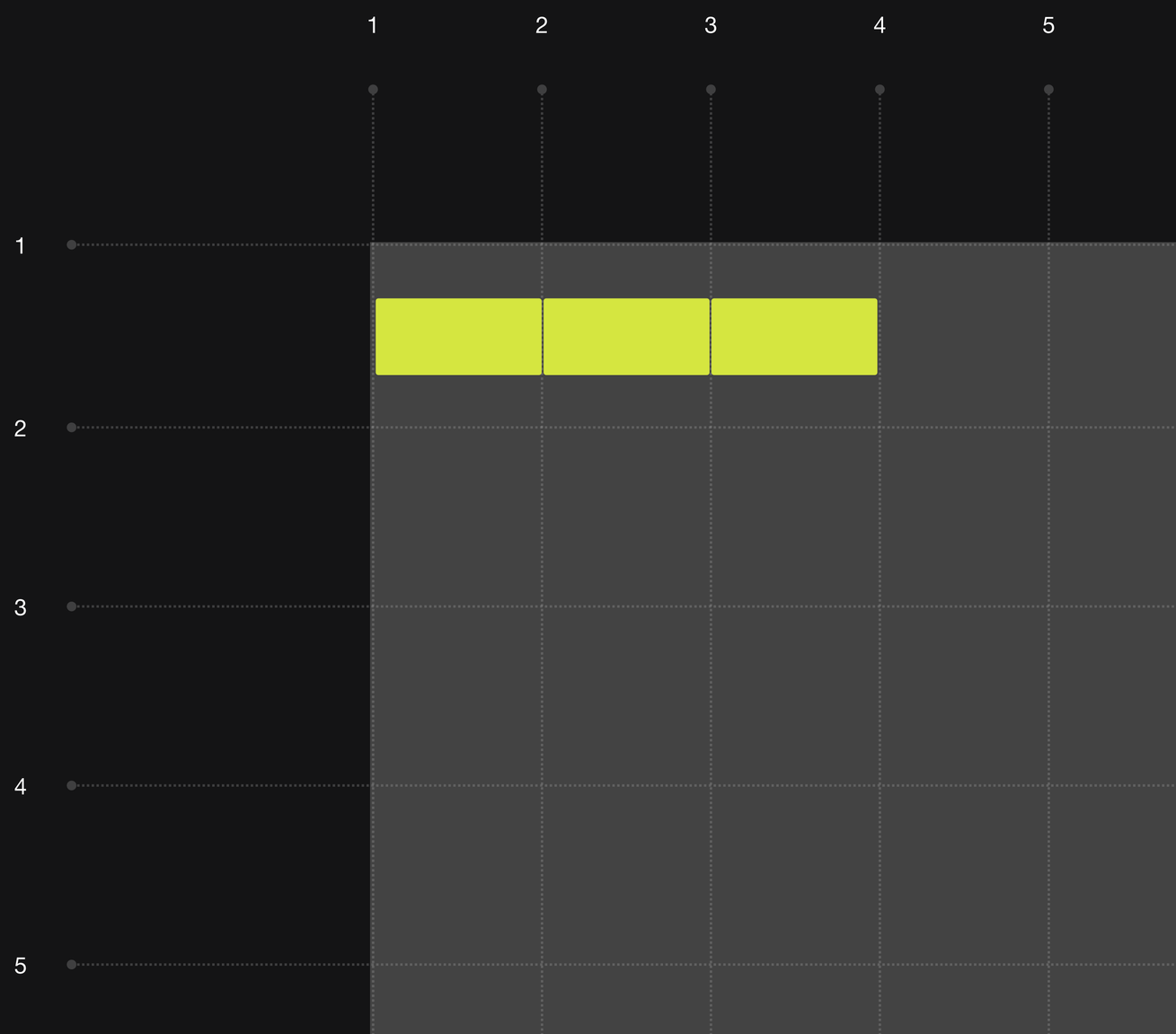
justify-items: start | end | center | stretch

Todos estes comportamentos podem também ser definidos pelo próprio child sand o **justify-self**.

E o eixo transversal por também ser manipulado da mesma maneira com a propriedade **align-items**.

align-items: start | end | center | stretch

A propriedade **place-items**, funciona como um conjunto de **align-items** e **justify-items**



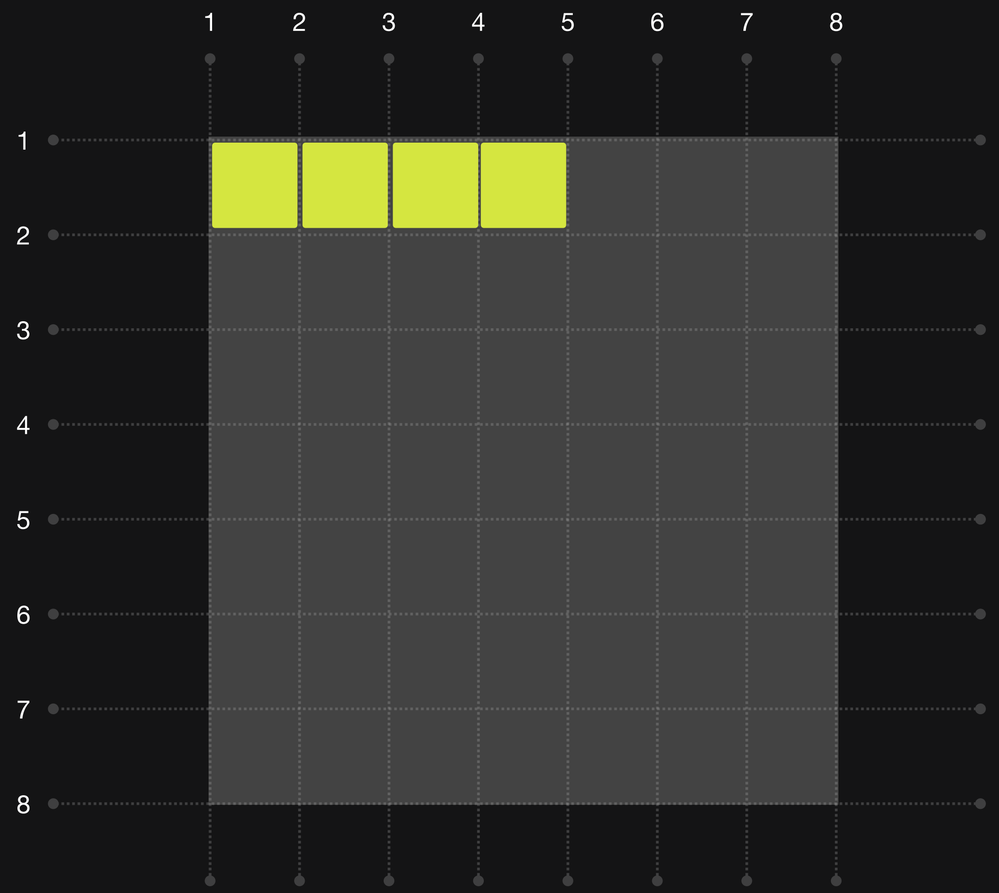
Construção

Podemos usar a função repeat para melhorar a leitura do nosso código.

grid-template-columns: repeat(8, 1fr);
Grid-template-rows: repeat(8, 1fr);

repeat(n, <track-size>). // e.g. 1fr | 10px | 1rem | 10%

Para cada linha, é atribuída um número para nos ajudar geograficamente a localizar os limites de cada novo espaço.

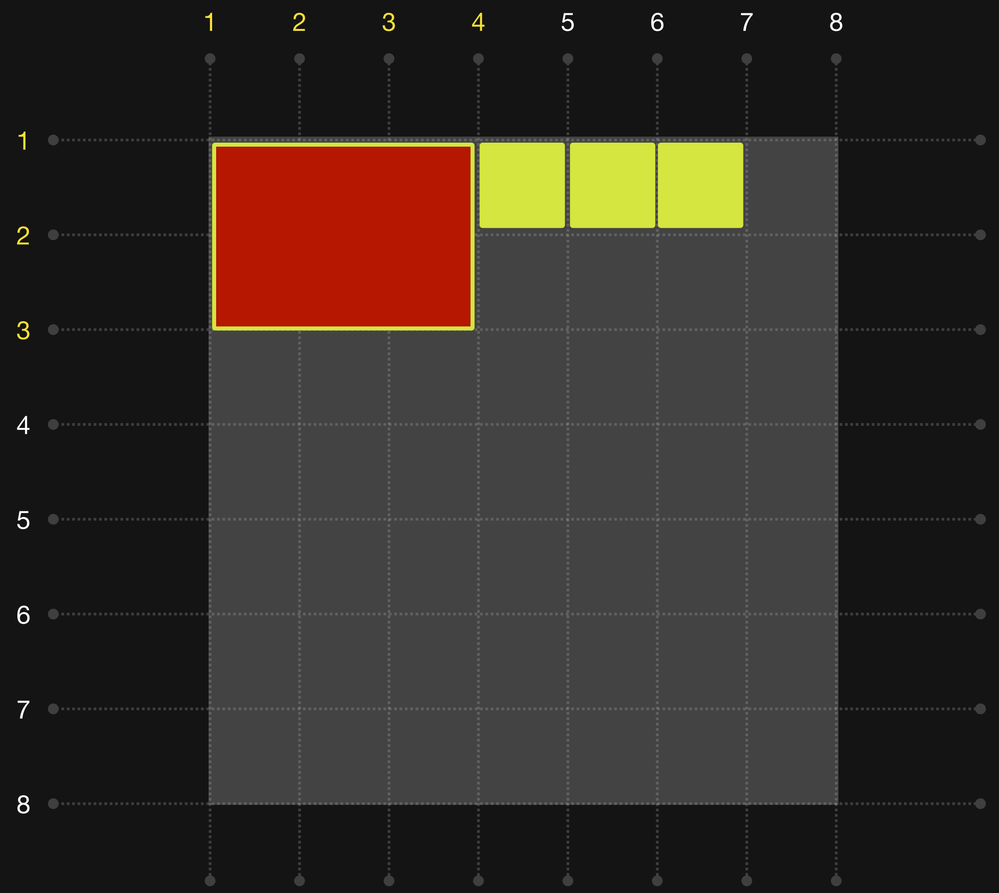


Alinhamento

A definição geografica de um elemento depende do início e o fim da grelha construída no container.

O elemento vermelho define o seu lugar através das propriedades grid-column e grid-row.

grid-column: 1 / 4 - começa no 1 e acaba no início do 4.
grid-row: 1 / 3 - começa no 1 e acaba no início do 3



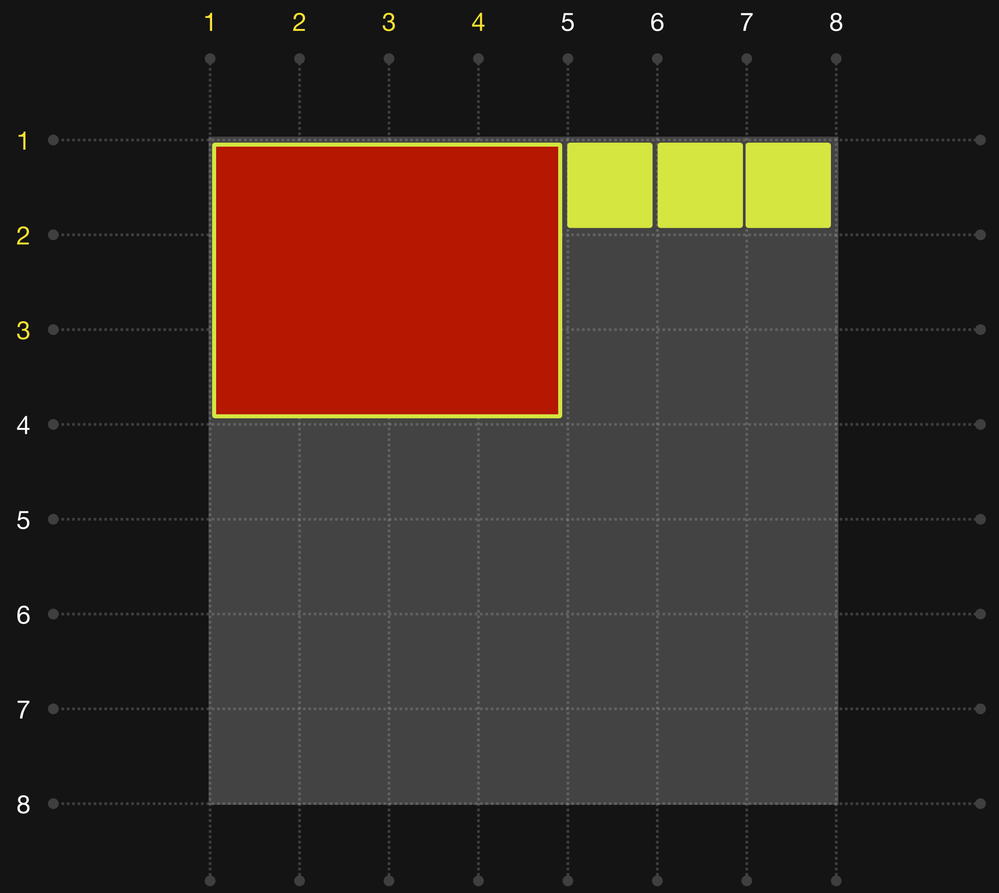
Alinhamento

A definição geografica de um elemento depende do início e o fim da grelha construída no container.

O elemento vermelho define o seu lugar através das propriedades grid-column e grid-row.

grid-column: 1 / span 4 - começa no 1 e acaba no fim 4.
grid-row: 1 / span 3 - começa no 1 e acaba no fim 3

Utilizar a palavra span , define que o limite poderá albergar também o fim da etapa.

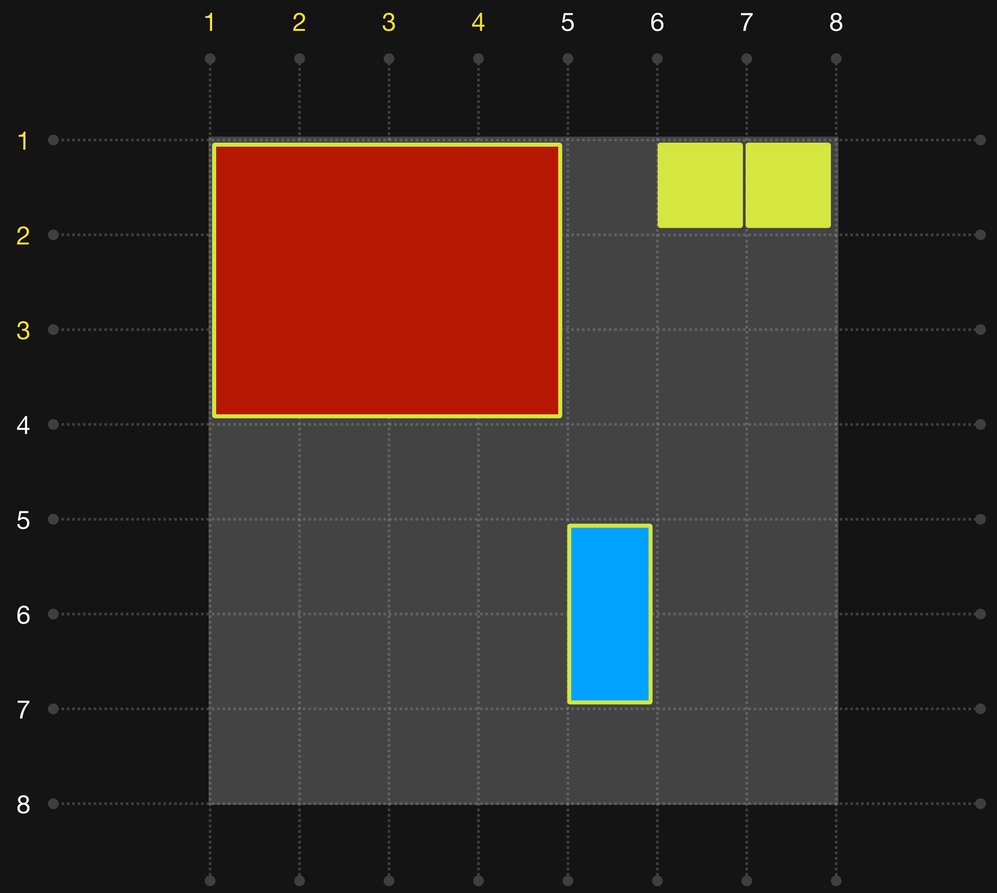


Alinhamento

A definição geográfica de um elemento depende do início e o fim da grelha construída no container.

O elemento azul poderá ocupar geograficamente coordenadas através dos elemento do pai, não dependendo do espaço , ou lugar, ocupado pelos irmãos.

grid-column: 5 / 5 - começa no 1 e acaba no fim 4.
grid-row: 5 / span 6 - começa no 1 e acaba no fim 3

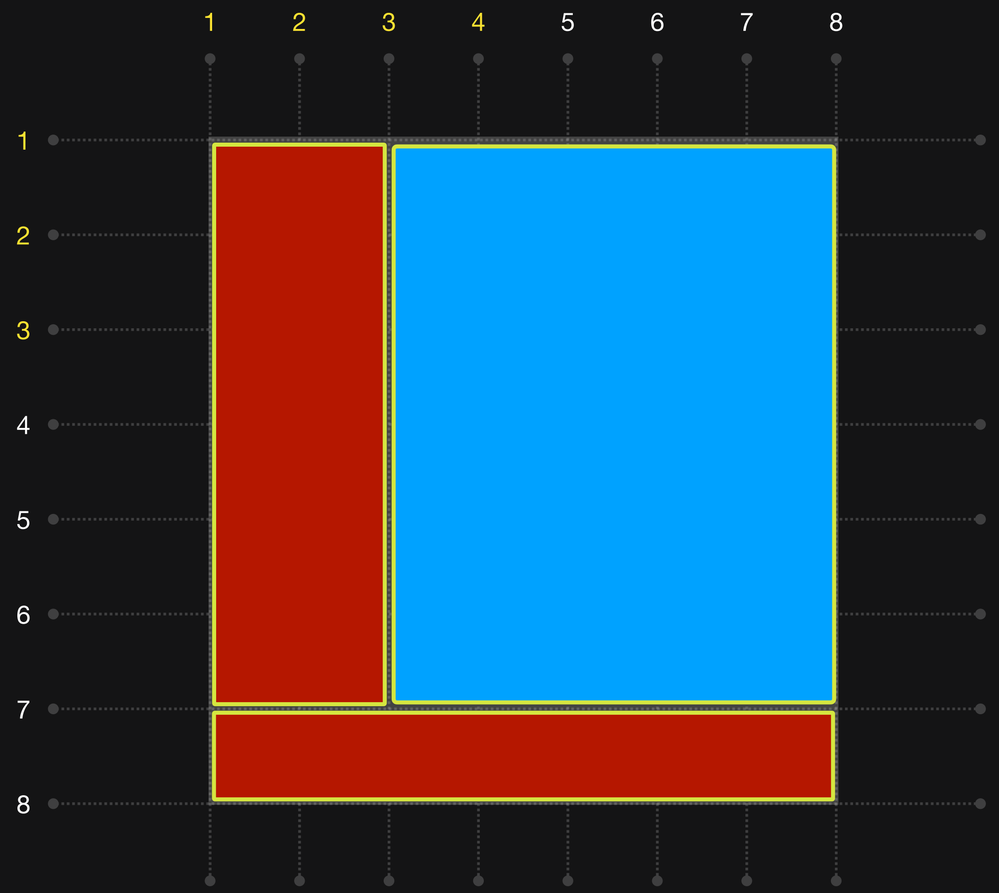


Templates

Podemos definir um template de grid para definir o conteúdo e a sua distribuição.

grid-template-areas é a propriedade que nos ajuda a definir onde colocar os childrens do parent.

Em definição, a grid-template é da responsabilidade do parent e os childrens só precisam de uma nomenclatura que os identifique através do grid-area



Templates

```
.container {
  display: grid;
  grid-template-columns: repeat(8, 1fr);
  grid-template-rows: repeat(8, 1fr);
  grid-template-areas:
    "sidebar main main main"
    "sidebar main main main"
    "footer footer footer footer";
}

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

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

.item-green {
  grid-area: footer;
}
```