



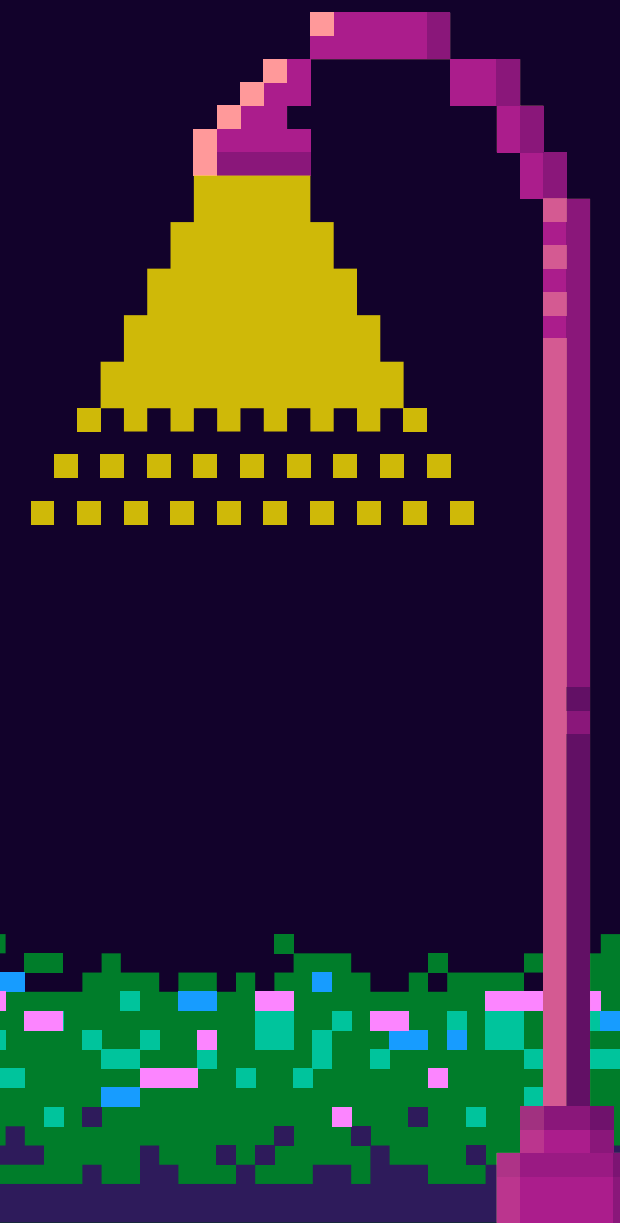
HELLO
WORLD



O QUE É GRID?

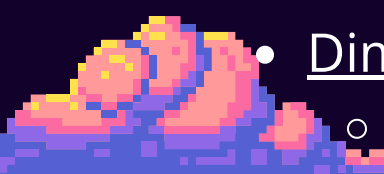
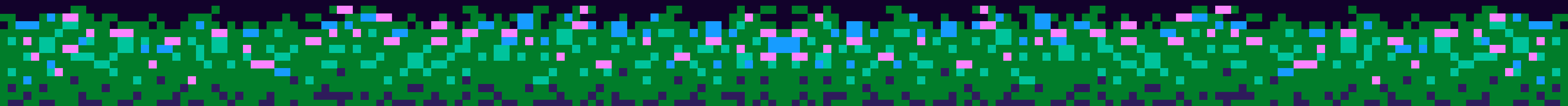
Grid é uma malha formada pela interseção de um conjunto de linhas horizontais e um conjunto de linhas verticais – um dos conjuntos define colunas e outro linhas. Dentro de um grid, respeitando-se a configuração criada pelas suas linhas, pode-se inserir elementos da marcação.

1	2	3
4	5	6
7	8	9





FUNCIONALIDADES DO GRID:

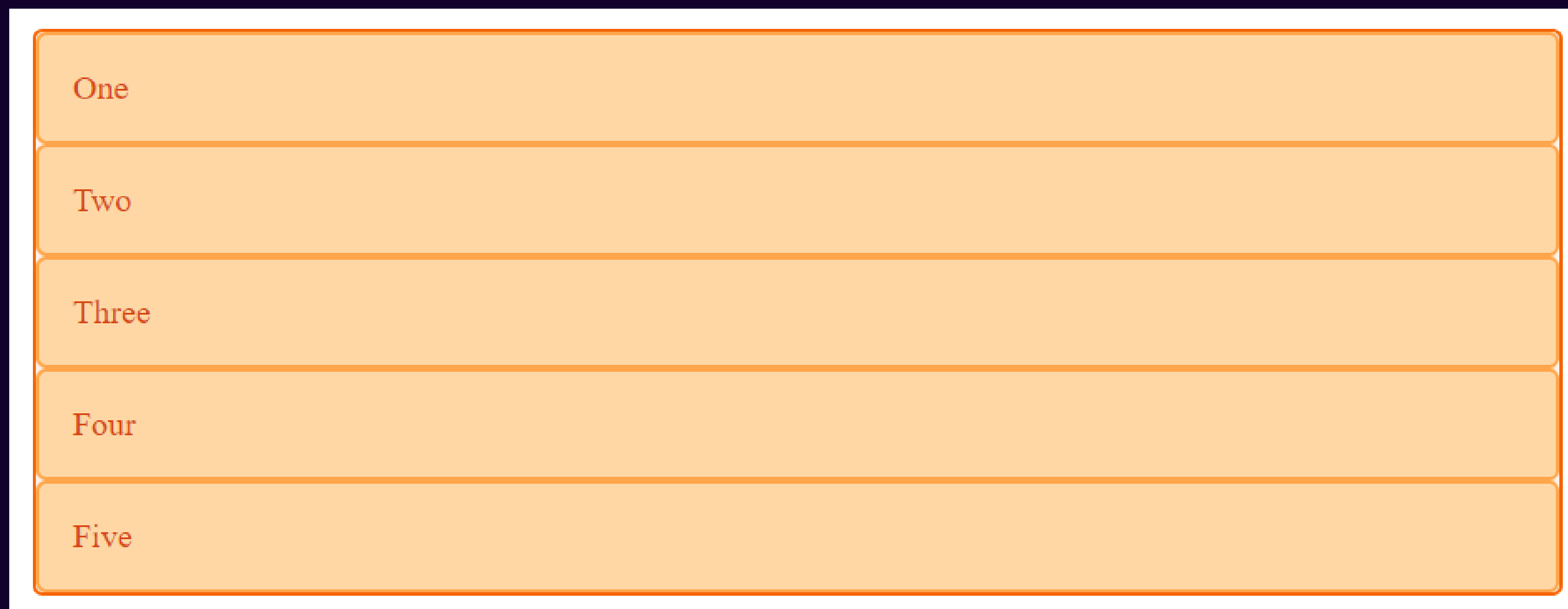
- 
- Dimensões fixas ou flexíveis
 - Você pode criar grids com dimensões fixas – por exemplo: definindo dimensões em pixels. Ou criar grids com dimensões flexíveis definindo-as com uso de porcentagem ou da nova unidade CSS fr criada para esse propósito.
 - Posicionamento de itens
 - Você pode posicionar 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. Existe ainda um algoritmo de controle do posicionamento de itens da página que não possuem uma posição capaz de ser explicitamente definida no grid.
 - Criação de grids adicionais
 - Além da possibilidade de se criar um grid inicial para o layout a Especificação prevê também a possibilidade de se adicionar linhas e colunas para layoutar conteúdos adicionados fora do grid inicial. Funcionalidades tal como adicionar "tantas colunas quanto necessárias em um grid container existente" são previstas nas Especificações.
 - Alinhamento
 - Estão previstas funcionalidades de alinhamento que possibilitam controlar o alinhamento dos itens de uma página posicionados no grid e também o alinhamento do próprio grid como um todo.
 - Controle sobre conteúdos sobrepostos
 - Em uma célula do grid podem ser posicionados mais de um item da página e também é possível que se defina sobreposição parcial de áreas. Esse controle de layers é feito com uso de z-index.
- 

GRID CONTAINER

Cria-se um grid container com as declarações CSS `display: grid` ou `display: inline-grid` para um elemento da marcação. Assim declarando, todos os elementos filhos diretos daquele container se transformam em grid items.

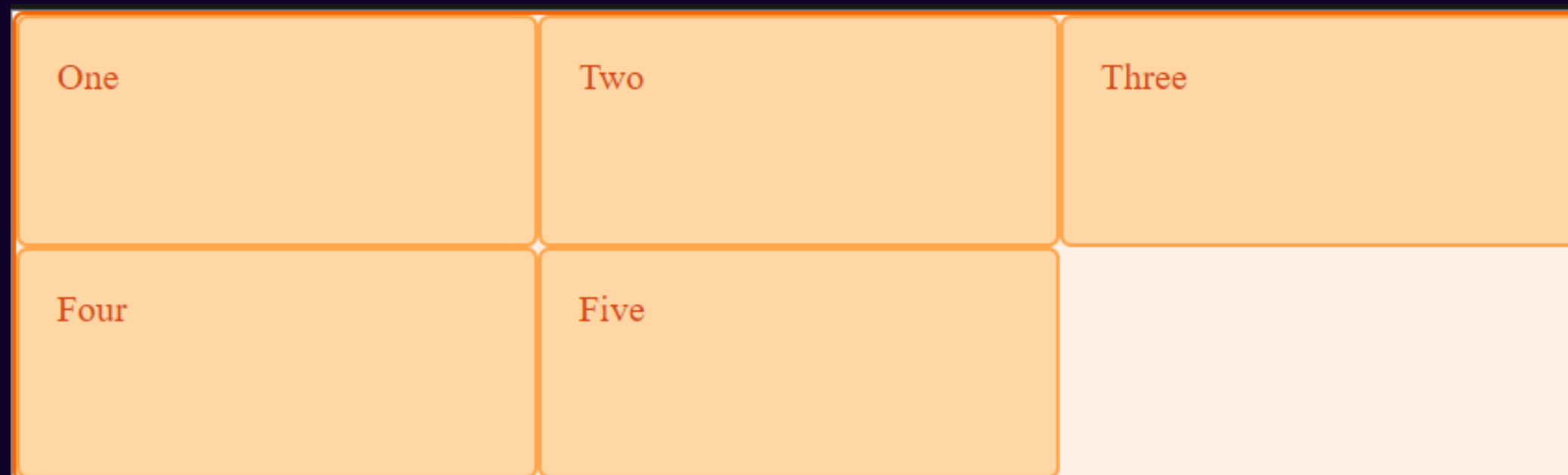
```
<div class="wrapper">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
  <div>Four</div>  
  <div>Five</div>  
</div>
```

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



GRID TRACKS

Nós definimos linhas e colunas no nosso grid com as propriedades `grid-template-columns` e `grid-template-rows`. Isso define o grid tracks. Um grid track é o espaço entre duas linhas em um grid. Na imagem abaixo você pode ver um track highlighter – o track na primeira linha do nosso grid.



```
<div class="wrapper">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
  <div>Four</div>  
  <div>Five</div>  
</div>
```

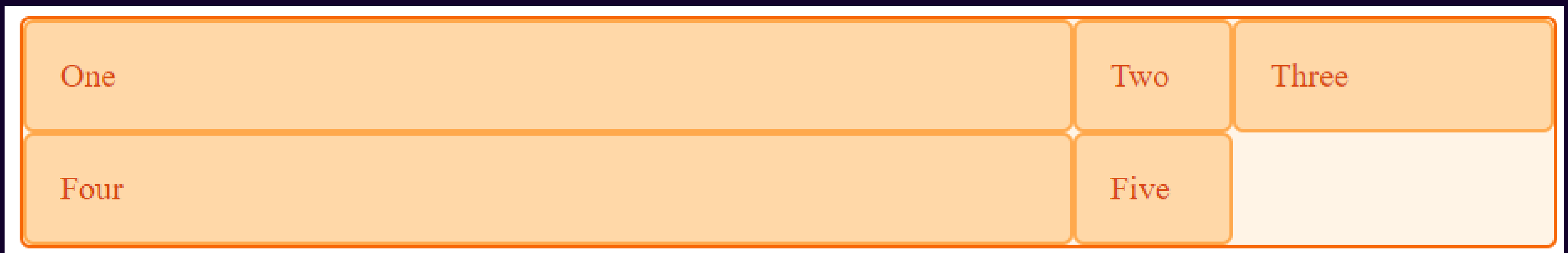
```
.wrapper {  
  display: grid;  
  grid-template-columns: 200px 200px 200px;  
}
```

UNIDADE FA

Nós definimos linhas e colunas no nosso grid com as propriedades `grid-template-columns` e `grid-template-rows`. Isso define o grid tracks. Um grid track é o espaço entre duas linhas em um grid. Na imagem abaixo você pode ver um track highlighter – o track na primeira linha do nosso grid.

```
<div class="wrapper">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
  <div>Four</div>  
  <div>Five</div>  
</div>
```

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



UNIDADE FR

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

No próximo exemplo, criamos um container .wrapper com uma coluna de 2fr e duas colunas de 1fr. Portanto, o espaço disponível será dividido em quatro partes. Duas partes serão para a primeira coluna e uma parte para cada um das próximas duas colunas.

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

Nesse exemplo final, nós misturamos unidades de medida fixa com as de fração. A primeira coluna tem 500px, que será fixa. O espaço disponível restante será dividido em três partes, sendo uma parte para a segunda coluna e mais duas partes para a terceira coluna.

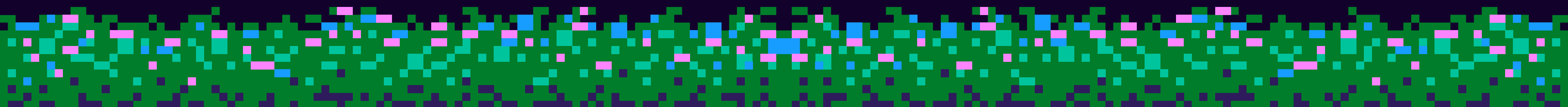
TRACK LISTINGS COM A NOTAÇÃO REPEAT()

Grids grandes, com muitas tracks podem usar a notação repeat() para repetir todas ou uma seção de track listing. Por exemplo a definição de grid a seguir:

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

Também pode ser escrita como:

```
.wrapper {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
}
```



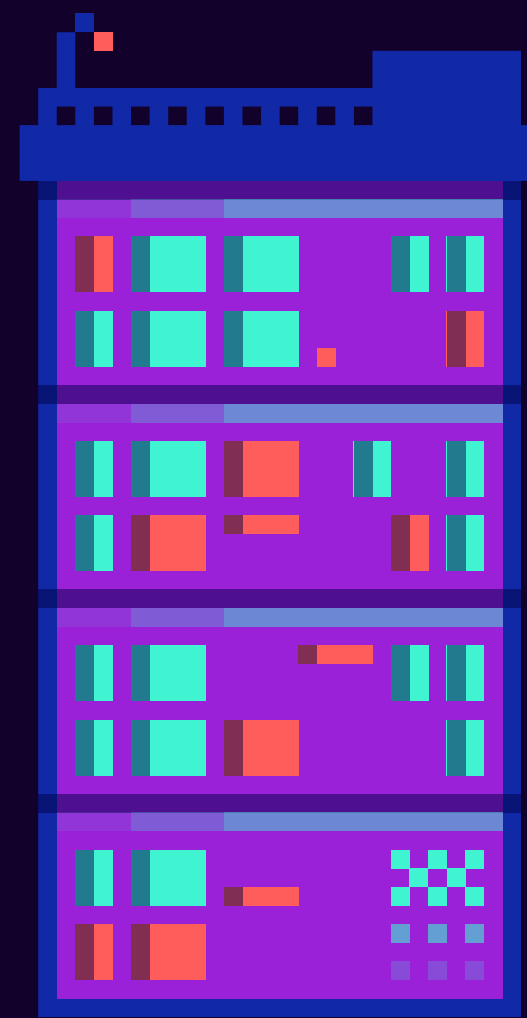
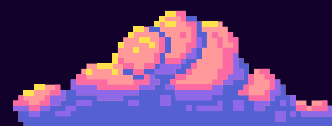
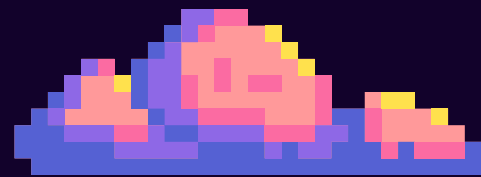


A notação `repeat ()` pode ser usada apenas para uma parte da track listing. No próximo exemplo estou criando um grid com uma coluna inicial de 20-pixels, repetindo uma sessão de 6 colunas de 1fr, e por fim uma coluna de 20-pixels.

```
.wrapper {  
  display: grid;  
  grid-template-columns: 20px repeat(6, 1fr) 20px;  
}
```

A notação `Repeat` tem como parâmetro um track listing, sendo assim você pode usá-lo para criar um padrão de repetição de tracks. Neste exemplo meu grid terá 10 tracks, uma trilha de 1fr seguida por uma trilha de 2fr , repetida cinco vezes.

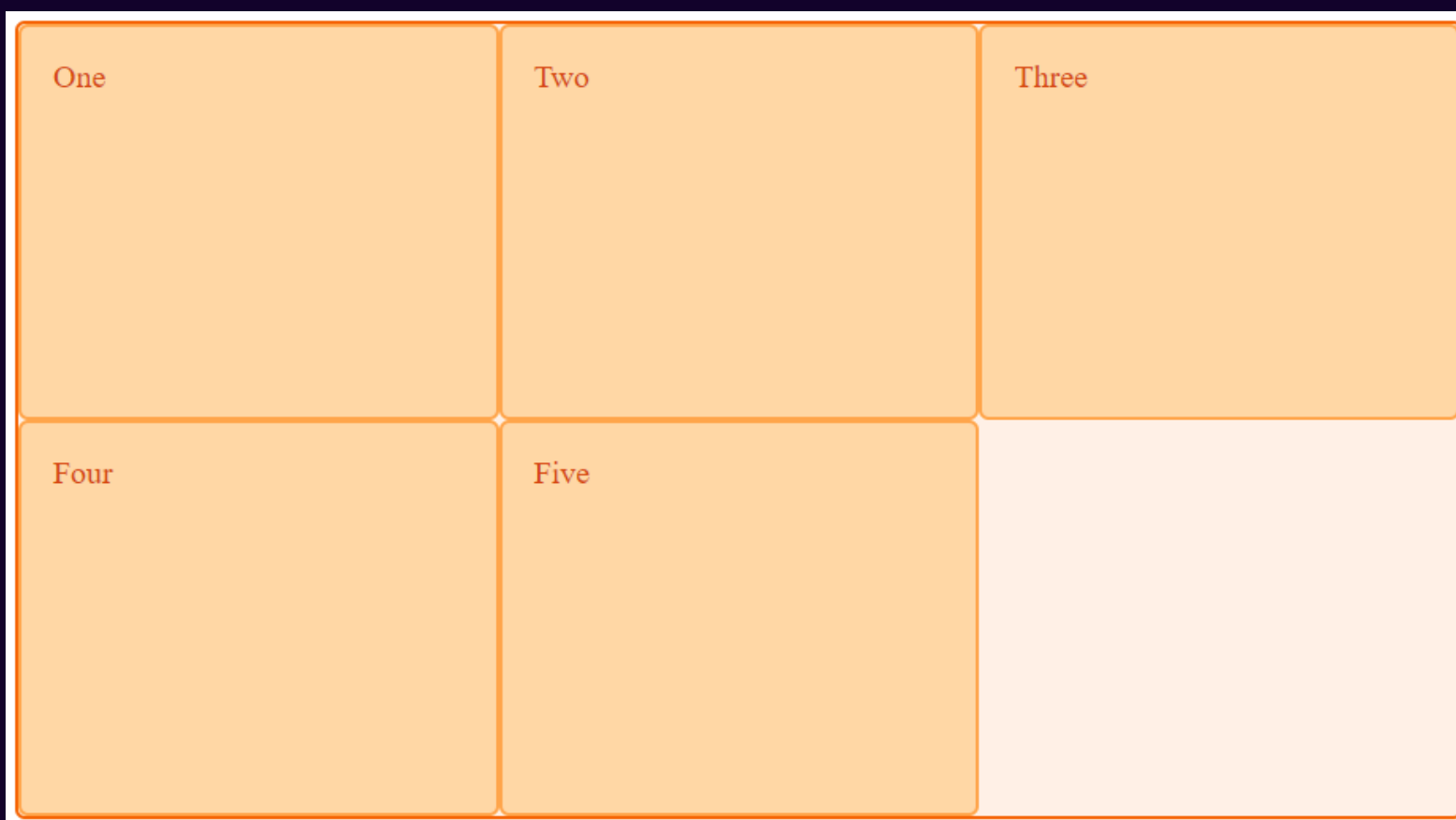
```
.wrapper {  
  display: grid;  
  grid-template-columns: repeat(5, 1fr 2fr);  
}
```



O GRID IMPLÍCITO E EXPLÍCITO

Ao criar nosso grid de exemplo definimos nossa coluna de trilhas com a propriedade `grid-template-columns`, mas adicionalmente deixamos o grid criar as linhas necessárias para qualquer outro conteúdo. Estas linhas são criadas no grid implícito. O grid explícito é constituído por linhas e colunas que você define com `grid-template-columns` e `grid-template-rows`. Se você posicionar algo fora do grid definido, ou caso devido à quantidade de conteúdo mais trilhas de grid sejam necessárias, o grid então cria linhas e colunas no grid implícito. Estas trilhas serão seu tamanho definido automaticamente por padrão, resultando em seu tamanho ser baseado no conteúdo que elas contém.

Você também pode definir o tamanho do conjunto para trilhas criadas na grid implícita com as propriedades `grid-auto-rows` e `grid-auto-columns`.



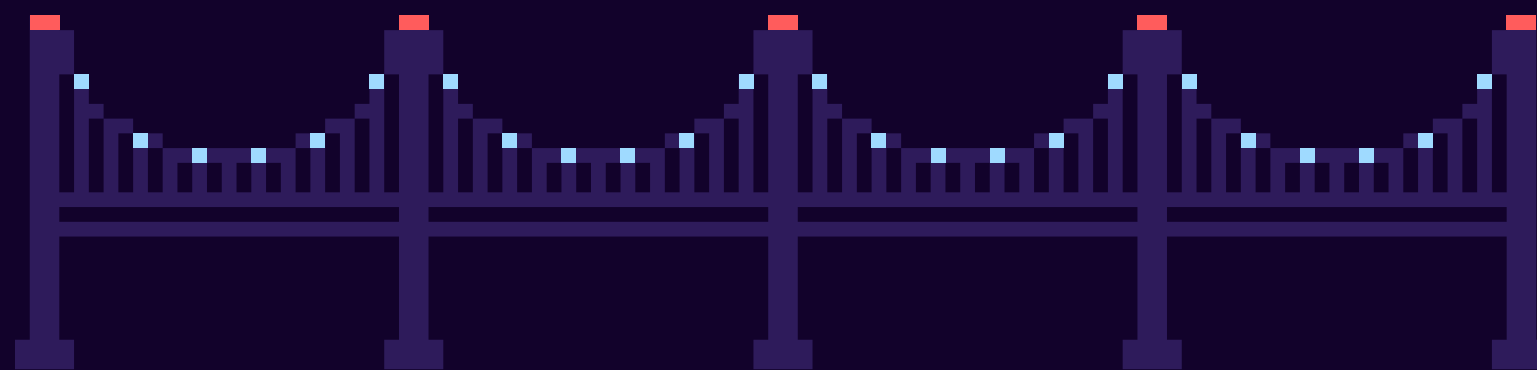
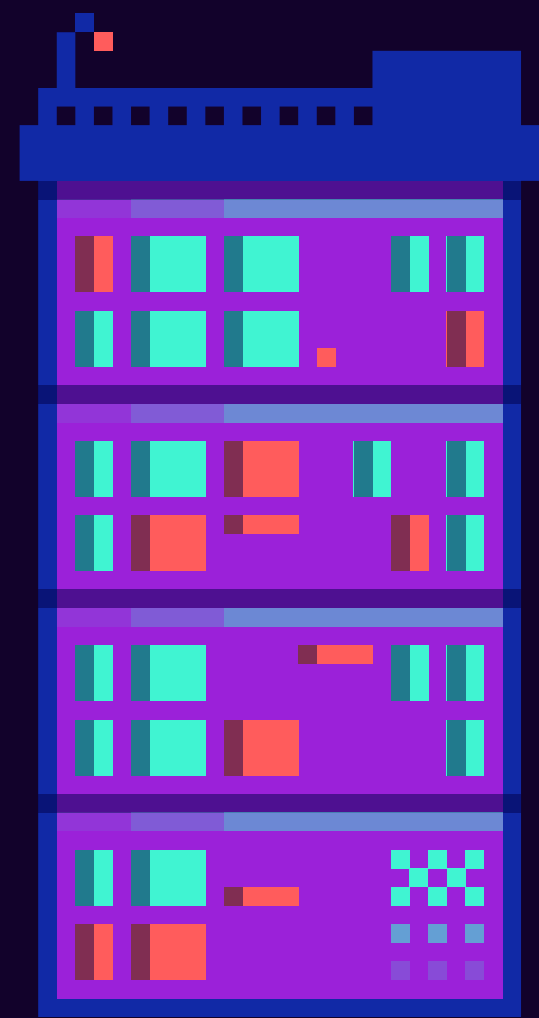
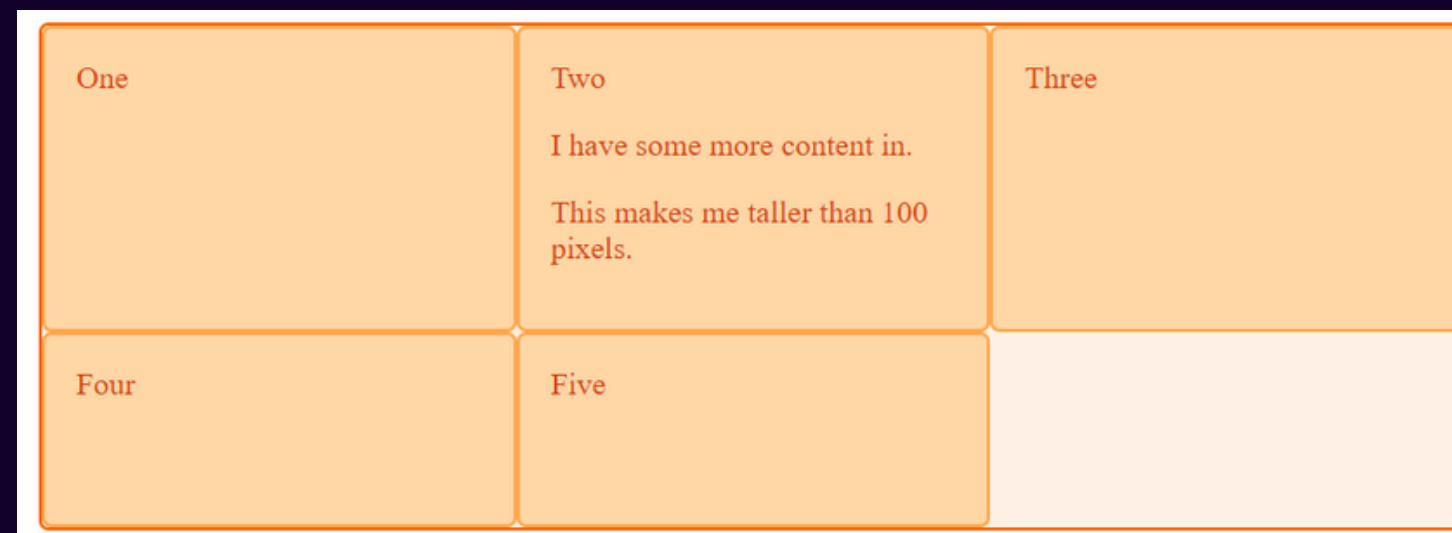
```
.wrapper {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  grid-auto-rows: 200px;  
}
```

```
<div class="wrapper">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
  <div>Four</div>  
  <div>Five</div>  
</div>
```

TRACK SIZING AND MINMAX()

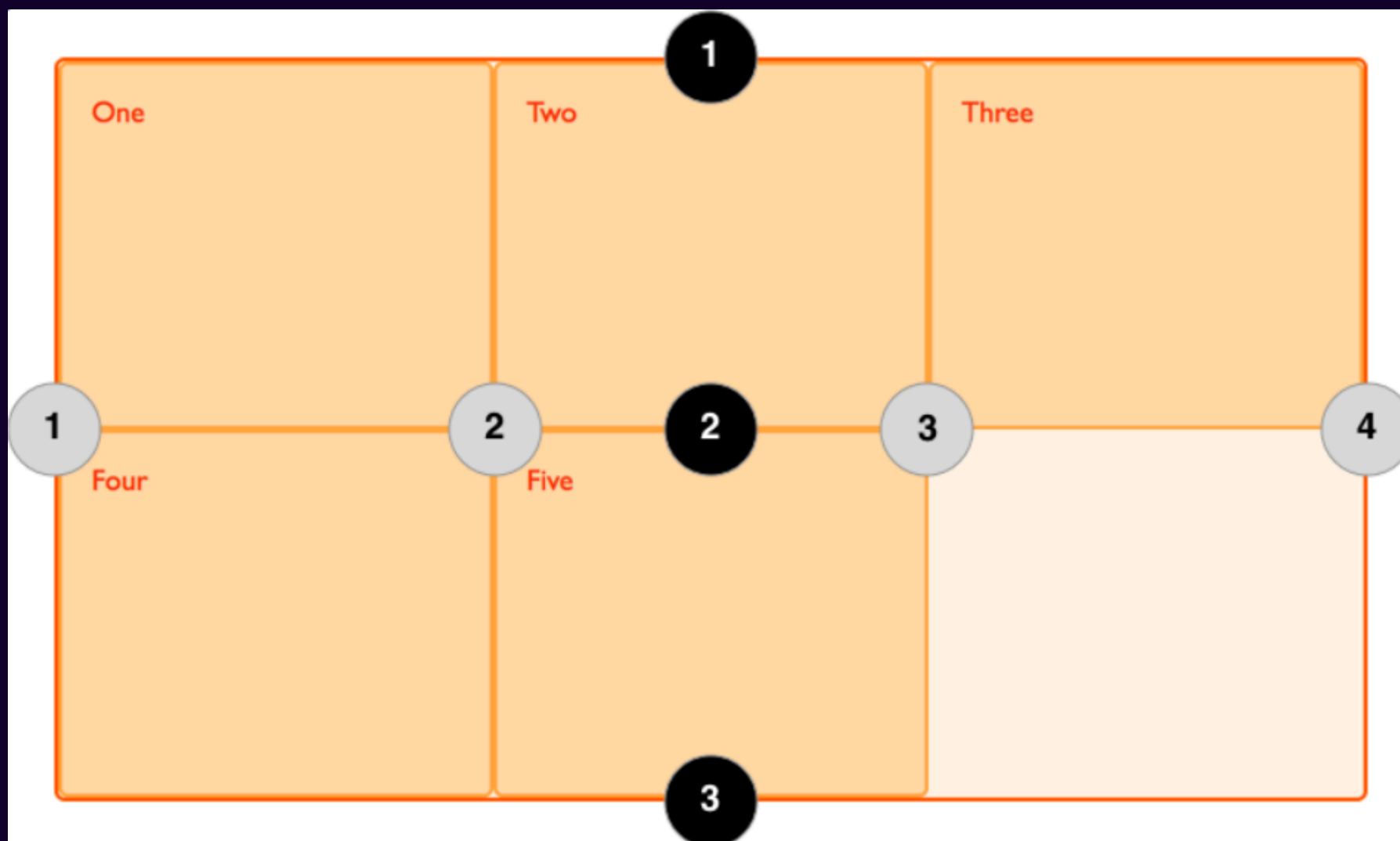
Quando criando um grid explícito ou definindo o tamanho para as colunas e linha serem criadas automaticamente podemos querer dar um tamanho minimo para os tracks, mas assegure que eles expandam para o tamanho necessário do conteúdo adicionado.

```
.wrapper {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  grid-auto-rows: minmax(100px, auto);  
}
```



LINHAS DE GRID

Devemos observar que quando definimos um grid definimos as trilhas do grid, não as linhas. O grid então nos devolve linhas numeradas para usarmos ao posicionar itens. Em nossa grid de 3 colunas por duas linhas, temos quatro linhas de colunas.



OBS.:

Linhas são numeradas de acordo a forma de escrita do documento. Em uma linguagem da esquerda para a direita, a linha 1 está à esquerda do grid. Em uma linguagem da direita para a esquerda, ela está no lado direito do grid. Linhas também podem ser nomeadas, e veremos como fazer isso em um guia posterior nessa série.

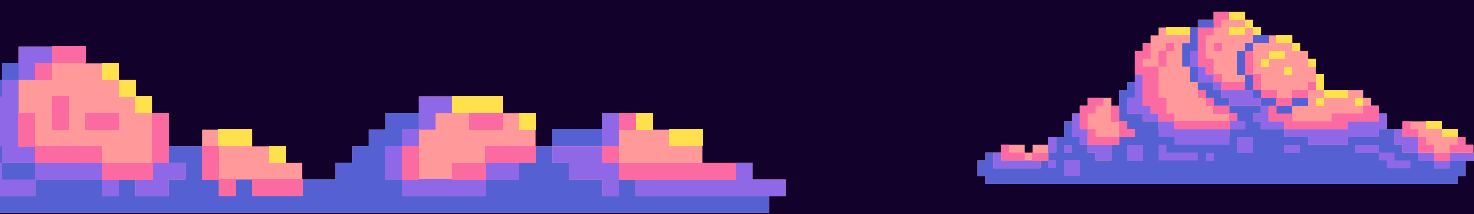
POSICIONANDO ITENS CONTRA LINHAS

Um operador é um símbolo matemático que produz um resultado baseado em dois valores (ou variáveis). Na tabela a seguir, você pode ver alguns dos operadores mais simples, juntamente com alguns exemplos para experimentar no console JavaScript.

```
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three</div>
  <div class="box4">Four</div>
  <div class="box5">Five</div>
</div>
```

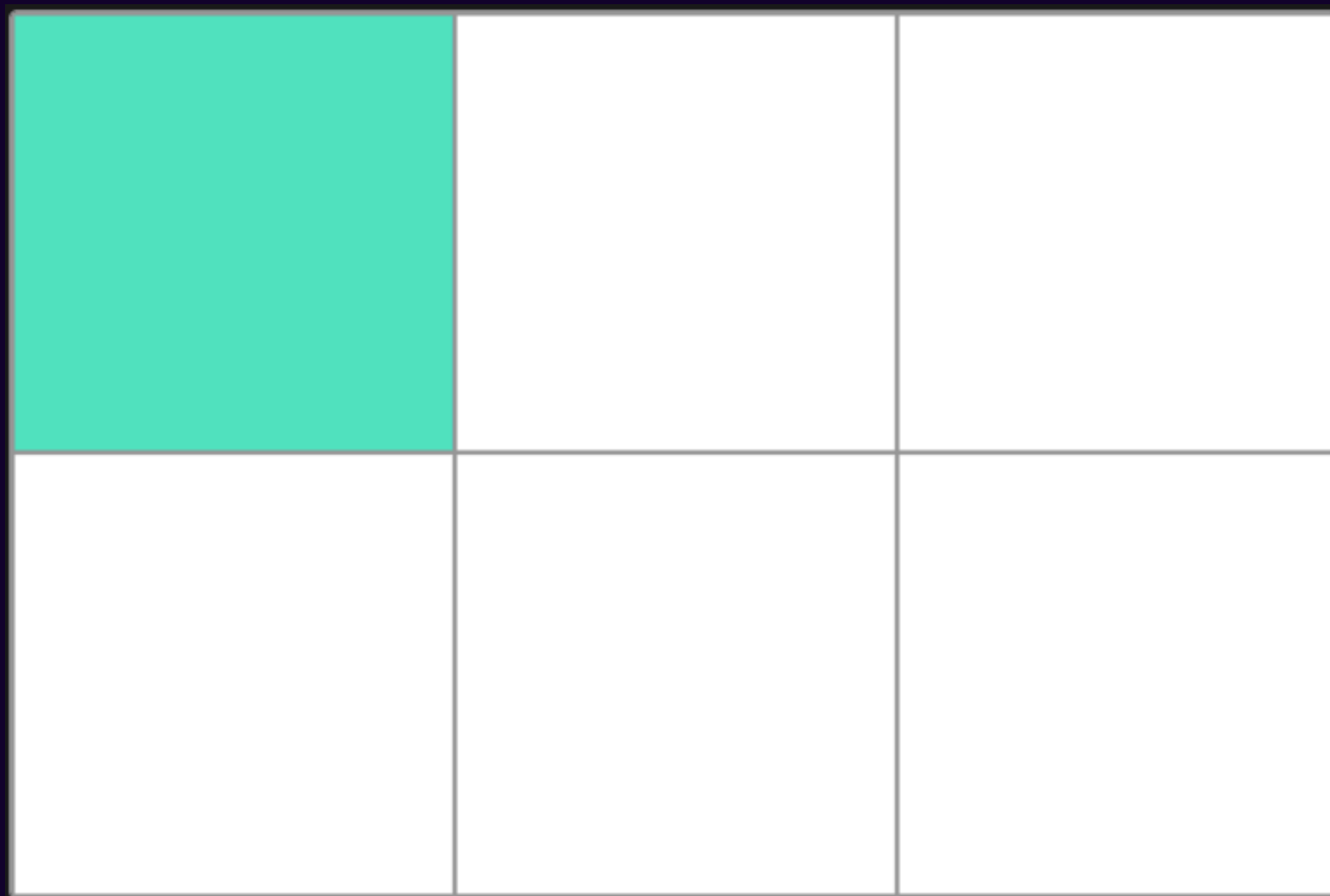
```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
}
.box2 {
  grid-column-start: 1;
  grid-row-start: 3;
  grid-row-end: 5;
}
```





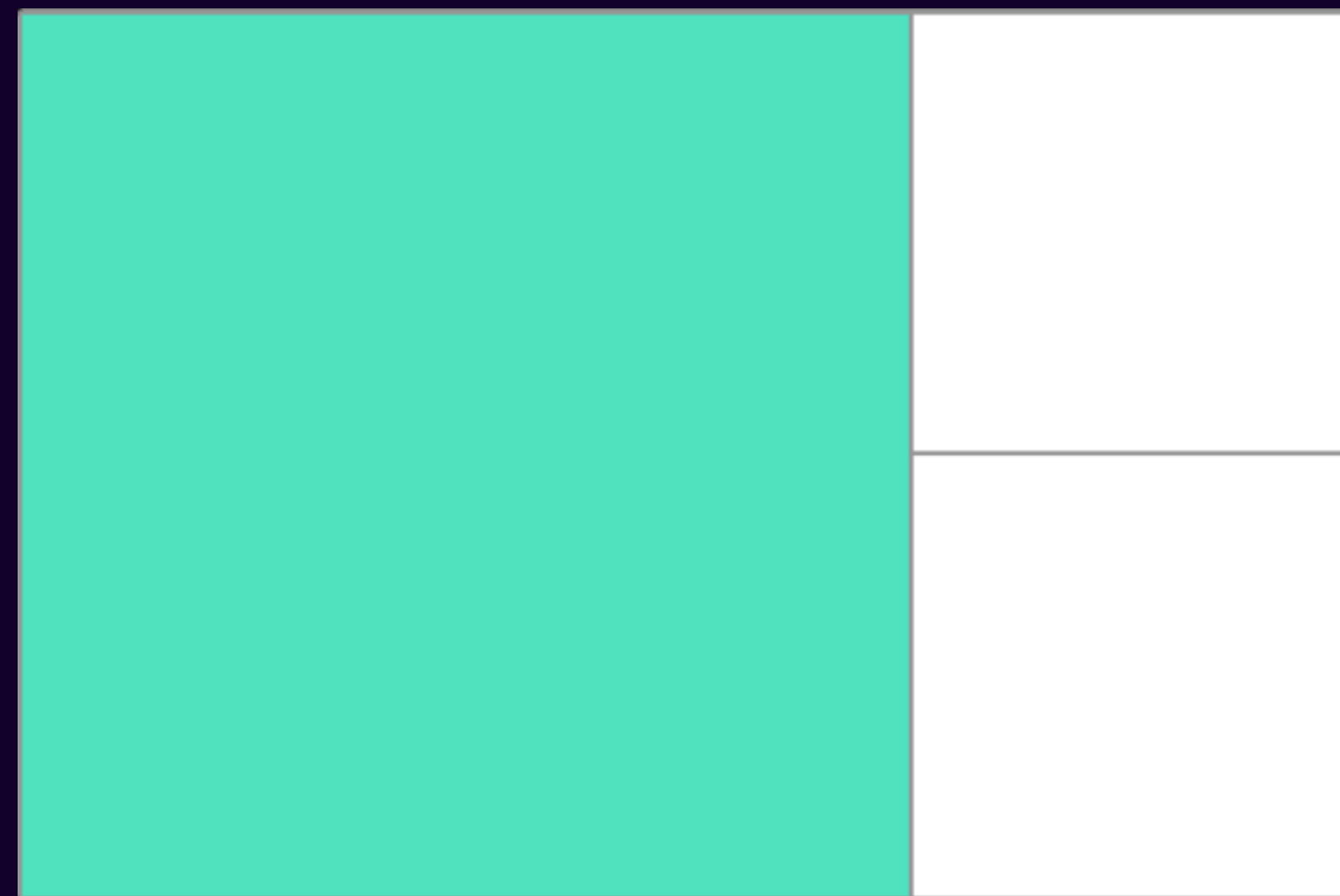
CÉLULAS DO GRID

Uma célula de grid é a menor unidade em um grid. Conceitualmente é como se fosse uma célula de tabela. Como vimos em nossos exemplos anteriores, uma vez que o grid é definido como o pai os itens filhos serão organizados cada um em uma célula do grid definido.



ÁREAS DO GRID

Itens podem se espalhar por uma ou mais células ambas entre linhas ou colunas, e isto cria uma área de grid. Áreas de grid devem ser retangulares – não é possível criar uma área em L por exemplo. A área destacada se espalha por duas trilhas de linhas e duas trilhas de colunas.



GUTTERS

Gutters ou alleys (espaçamentos ou separadores) entre células do grid podem ser criadas usando a propriedade `grid-column-gap` e `grid-row-gap` ou de forma resumida `grid-gap`.



```
.wrapper {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  grid-column-gap: 10px;  
  grid-row-gap: 1em;  
}
```

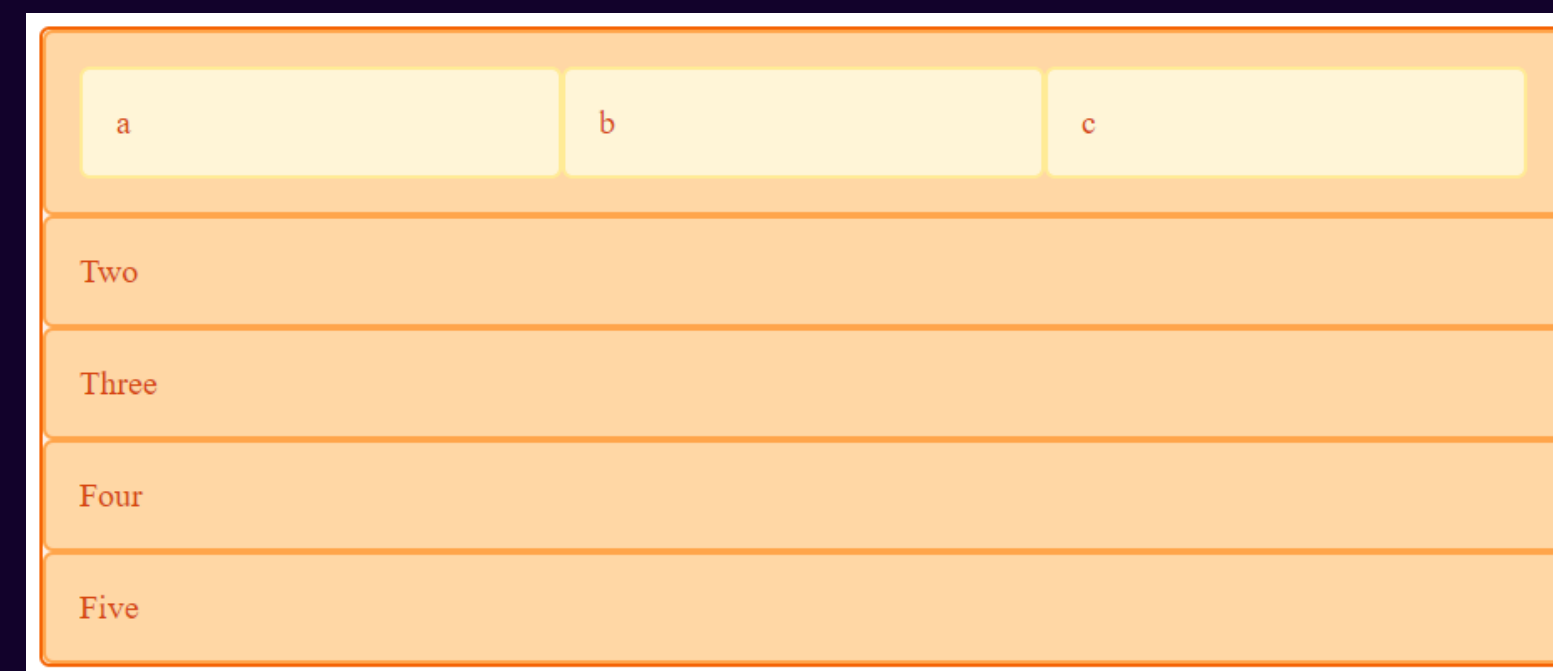
```
<div class="wrapper">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
  <div>Four</div>  
  <div>Five</div>  
</div>
```


ANINHANDO GRIDS

Um item de grid pode se tornar um container de grid. No exemplo a seguir, temos o grid de três colunas criadas anteriormente, com nossos dois itens posicionados. Neste caso o primeiro item possui dois sub itens. Como estes itens não são filhos diretos do grid eles não participam no layout do grid e dessa forma são exibidos no fluxo normal do documento.

```
<div class="wrapper">
  <div class="box box1">
    <div class="nested">a</div>
    <div class="nested">b</div>
    <div class="nested">c</div>
  </div>
  <div class="box box2">Two</div>
  <div class="box box3">Three</div>
  <div class="box box4">Four</div>
  <div class="box box5">Five</div>
</div>
```

```
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```



Nesse caso o grid aninhado não possui relação com o pai. Como é possível perceber no exemplo ele não herdou o grid-gap do pai e as linhas no grid aninhado não estão alinhadas com as linhas do grid pai.



SUBGRID

No nível das especificações do grid tem uma feature chamada subgrid que nos permitiria criar grids aninhados que usa aquilo que foi definido no grid pai.

Nota: Subgrids ainda não foram implementados em nenhum browser, e as especificações são sujeitas a mudanças.

Na especificação atual, no exemplo acima editaríamos o grid aninhado usando `display: subgrid` ao invés de `display: grid`, e remover o que havia sido definido. O grid aninhado vai usar as propriedades definidas no pai para dispor os itens.

```
.box1 {  
  grid-column-start: 1;  
  grid-column-end: 4;  
  grid-row-start: 1;  
  grid-row-end: 3;  
  display: subgrid;  
}
```

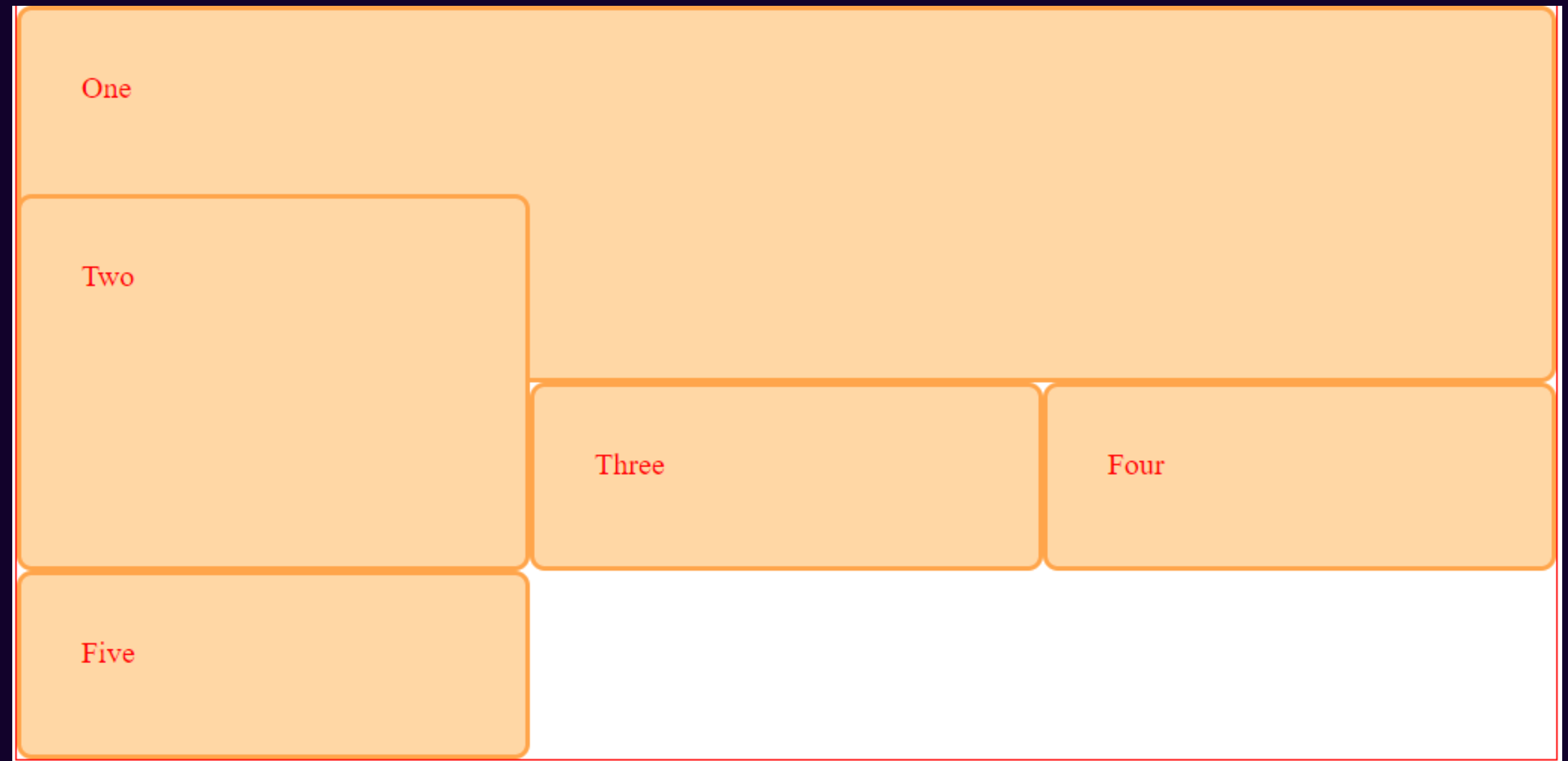
SOBREPONDO ITENS COM Z-INDEX

Itens de uma mesma grade podem ocupar uma mesma célula. Se retornarmos ao nosso exemplo com itens dispostos pela linha, podemos fazer com que dois se sobreponham.

O container box2 está sobrepondo box1, é renderizado acima pois vem depois na ordem.

```
<div class="wrapper">
  <div class="box box1">One</div>
  <div class="box box2">Two</div>
  <div class="box box3">Three</div>
  <div class="box box4">Four</div>
  <div class="box box5">Five</div>
</div>
```

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
}
.box2 {
  grid-column-start: 1;
  grid-row-start: 2;
  grid-row-end: 4;
}
```



MANIPULANDO A ORDEM

Podemos controlar a ordem na qual os itens irão empilhar-se usando a propriedade z-index. Se box2 recebe um z-index menor que box1 será mostrado abaixo de box1.


```
.wrapper {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  grid-auto-rows: 100px;  
}  
.box1 {  
  grid-column-start: 1;  
  grid-column-end: 4;  
  grid-row-start: 1;  
  grid-row-end: 3;  
  z-index: 2;  
}  
.box2 {  
  grid-column-start: 1;  
  grid-row-start: 2;  
  grid-row-end: 4;  
  z-index: 1;  
}
```





EODRA

CODAR



DESAFIO

Join our community

30-day, hassle-free money back guarantee

Gain access to our full library of tutorials along with expert code reviews.
Perfect for any developers who are serious about honing their skills.

Monthly Subscription

\$29 per month

Full access for less than \$1 a day

Sign Up

Why Us

- Tutorials by industry experts
- Peer & expert code review
- Coding exercises
- Access to our GitHub repos
- Community forum
- Flashcard decks
- New videos every week

MEUS CONTATOS:



27 99500-7495



<https://beacons.ai/prismatech>



producaoprismatech@gmail.com



Avenida Jerônimo Monteiro 145, Vitória





THANK
YOU