Esta página foi traduzida do inglês pela comunidade. Saiba mais e junte-se à comunidade MDN Web Docs.

Conceitos básicos de Grid Layout

CSS Grid Layout introduz um sistema bi-dimensional de grid (literalmente "grades") para CSS. Grids podem ser usados para o design de layouts de grandes seções de uma webpage, assim como de pequenos elementos de interface. Esse artigo apresenta o CSS Grid Layout e a terminologia que é parte da especificação CSS Grid Layout Level 1. As funcionalidades demonstradas neste resumo serão posteriormente explicadas em maiores detalhes nas demais seções desse guia.

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. As CSS grid layout preveem as seguintes funcionalidades:

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 algorítmo de controle do

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` (inglês).

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. Tudo começa com a criação de um grid dentro de um **grid container**.

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*.

No exemplo mostrado a seguir um elemento div container ao qual foi atribuída a classe wrapper contém cinco elementos filhos.

```
HTML                                                    Play
```

```
  <div>Four</div>
  <div>Five</div>
</div>
```

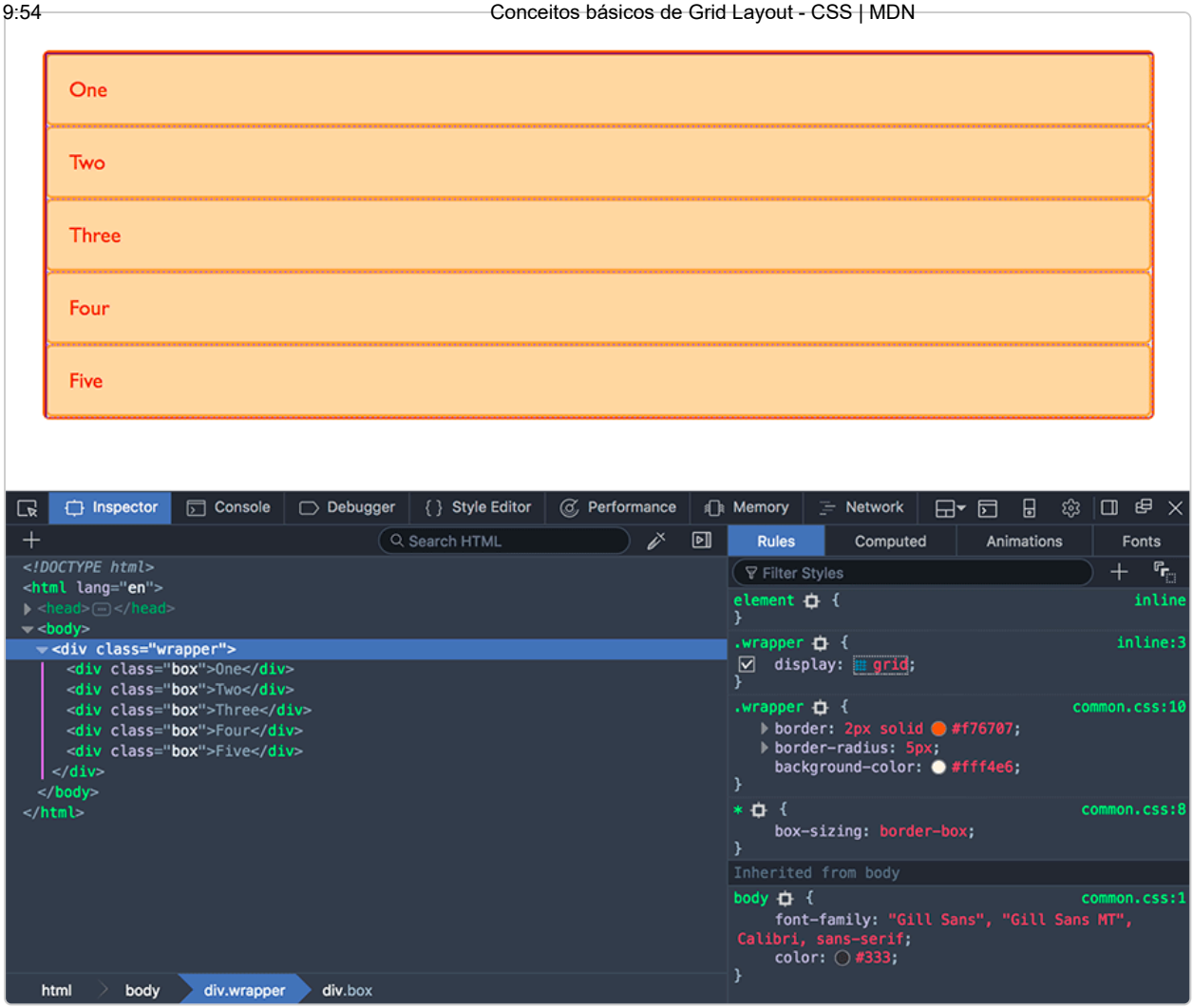
Façamos de `.wrapper` um grid container.

CSS Play

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

Play

Todos os elementos filhos diretos agora são grid items. Observando a renderização em um navegador web você não notará nenhuma diferença em relação a renderização já conhecida de um elemento container com seus cinco elementos filhos, pois no exemplo foi criado um grid constituído de uma única coluna para acomodar os elementos filhos. A partir desse ponto você pode achar mais útil trabalhar no *Firefox Developer Edition*, que possui o Grid Inspector disponível como parte das Ferramentas do desenvolvedor. Se você ver este

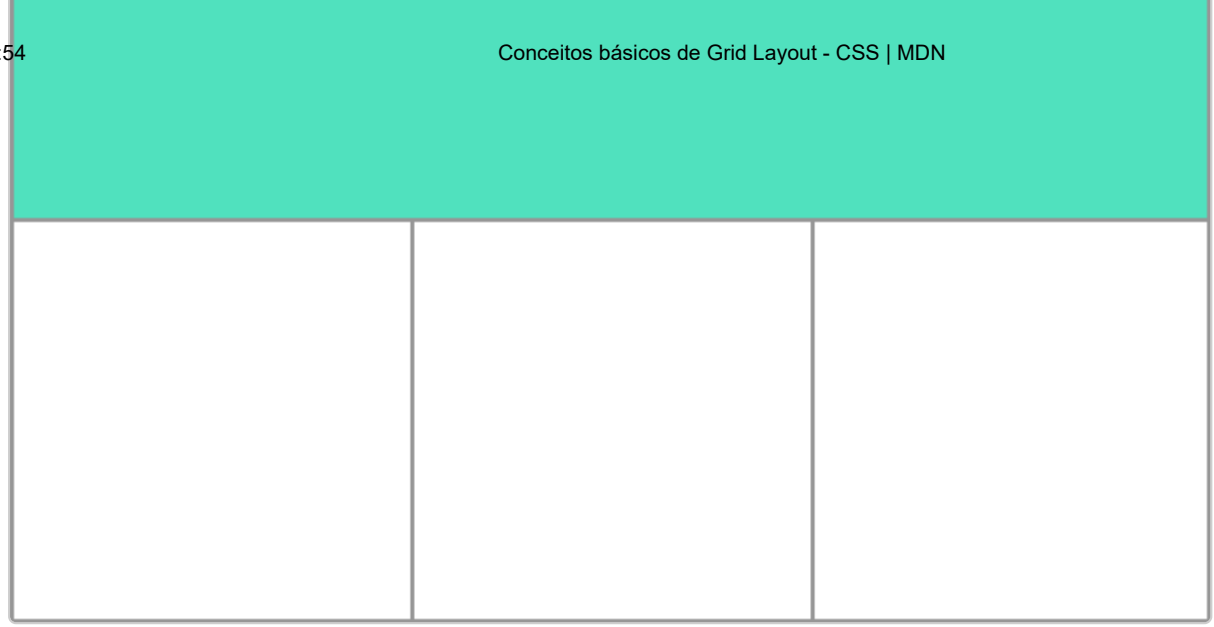


À medida que você aprende e depois trabalha com *CSS Grid Layout*, esta ferramenta lhe dará uma idéia melhor do que está acontecendo com suas *Grids* visualmente.

Se nós queremos começar a fazer isso parecer mais com os *Grids layouts* , nós precisamos adicionar *collumn tracks*.

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.



Posso adicionar ao exemplo anterior a propriedade `grid-template-columns`, depois definir o tamanho da column tracks.

Agora criei um grid com três colunas fixas de 200px. Os itens filhos serão dispostos nessa grade, um em cada célula da grade.

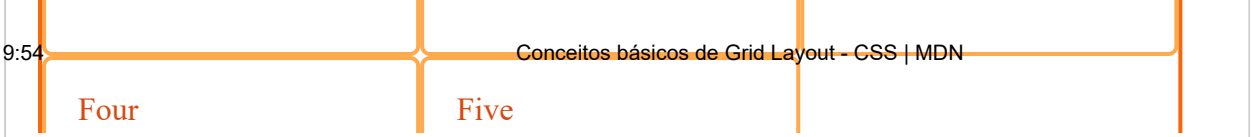
HTML Play

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

CSS Play

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

Play



A unidade `fr`

Nas propriedades de grid podem ser utilizadas quaisquer unidades de medida. Para nos ajudar a criar layouts flexíveis utilizando o grid, foi criada uma unidade nova. A unidade `fr` representa uma fração do espaço disponível no container do grid. A próxima definição de grid cria três espaços com tamanhos iguais que aumentam e diminuem de acordo com o espaço disponível.

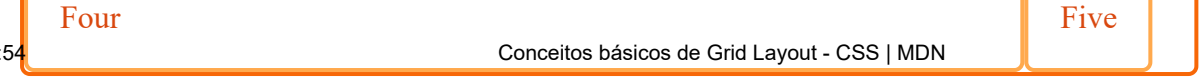
HTML Play

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

CSS Play

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

Play



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.

CSS Play

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

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.

CSS Play

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

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:

CSS Play

Também pode ser escrita como:

CSS Play

```css
.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.

CSS Play

```css
.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.

CSS Play

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

O grid implícito e explícito

Ao criar nosso grid de exempo 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

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 basedo 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` (inglês) e `grid-auto-columns` (inglês).

No exemplo abaixo usamos `grid-auto-rows` para garantir que as trilhas criadas na grid implícita tem 200 pixels de altura.

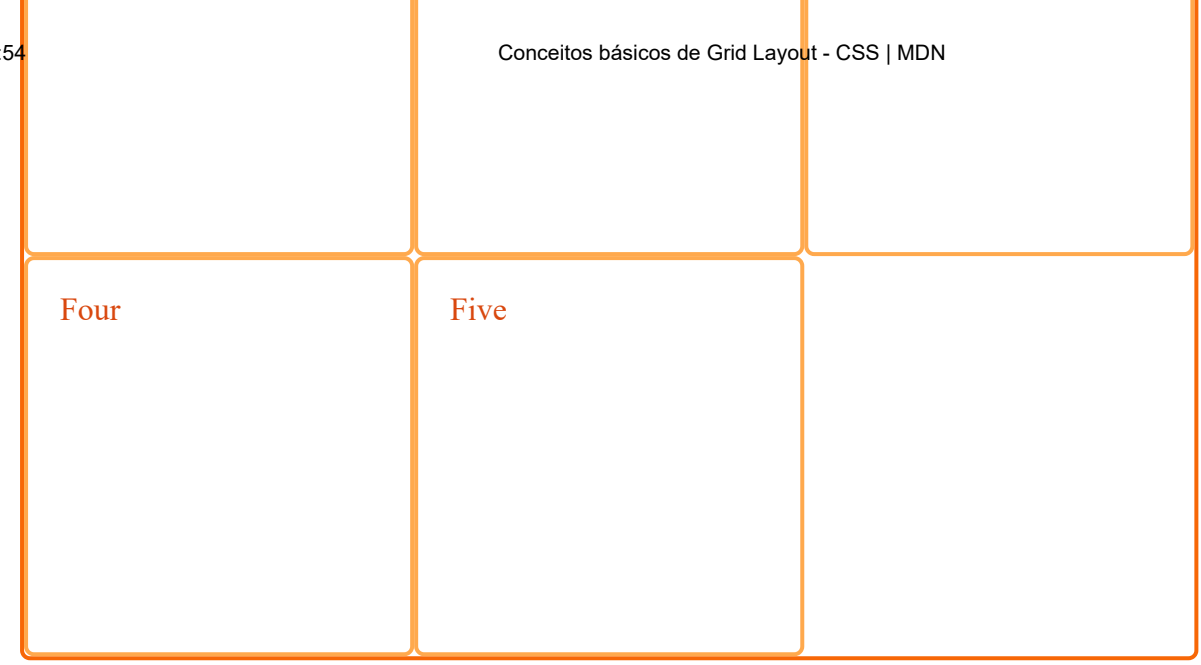
HTML Play

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

CSS Play

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

Play



Track sizing and `minmax()`

Quando criando um grid explícito ou definindo o tamanho para as colunas e linha serem criadas automáticamente podemos querer dar um tamanho minimo para os tracks, mas assegure que eles expandam para o tamanho necessário do conteúdo adicionado. Por exemplo eu gostaria que as minhas linhas nunca colapsem para um tamanho menor que 100 pixels, mas se o meu conteúdo aumentar até 300 pixels de altura eu gostaria que a linha aumentasse para essa mesma altura.

Grid tem uma solução para isso com a função `minmax()` (inglês). No próximo exemplo estou usando `minmax()` no valor de `grid-auto-rows` (inglês). Automaticamente as linhas criadas terão uma altura mínima de 100 pixels, e a máxima de `auto`. Usar `auto` significa que o tamanho vai olhar para o conteúdo e, assim, esticar para que a linha tenha o espaço necessário.

CSS Play

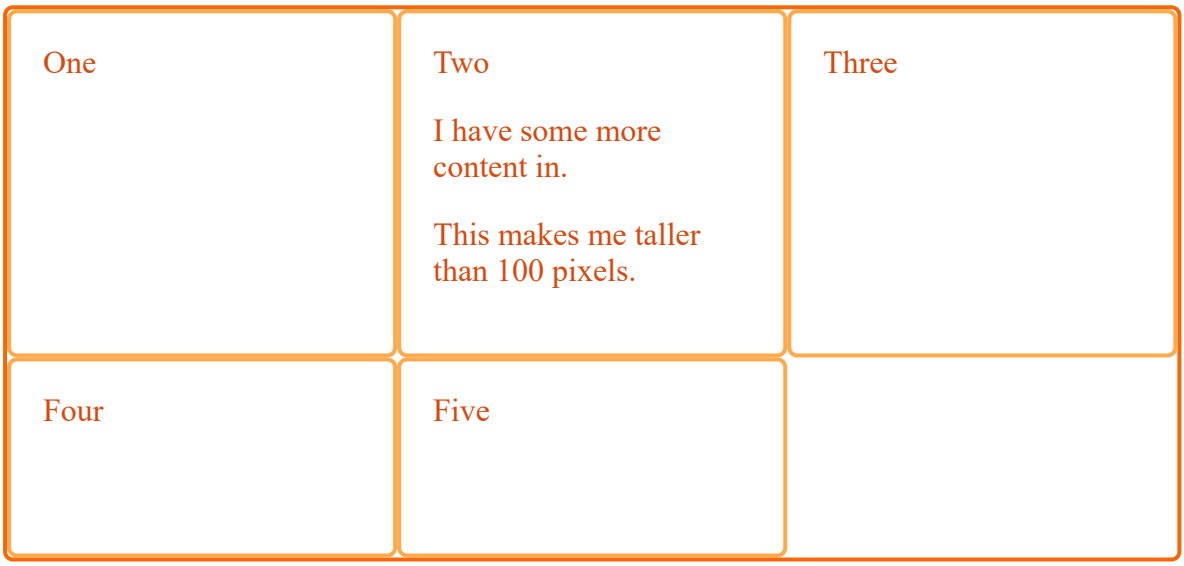
```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
```

HTML Play

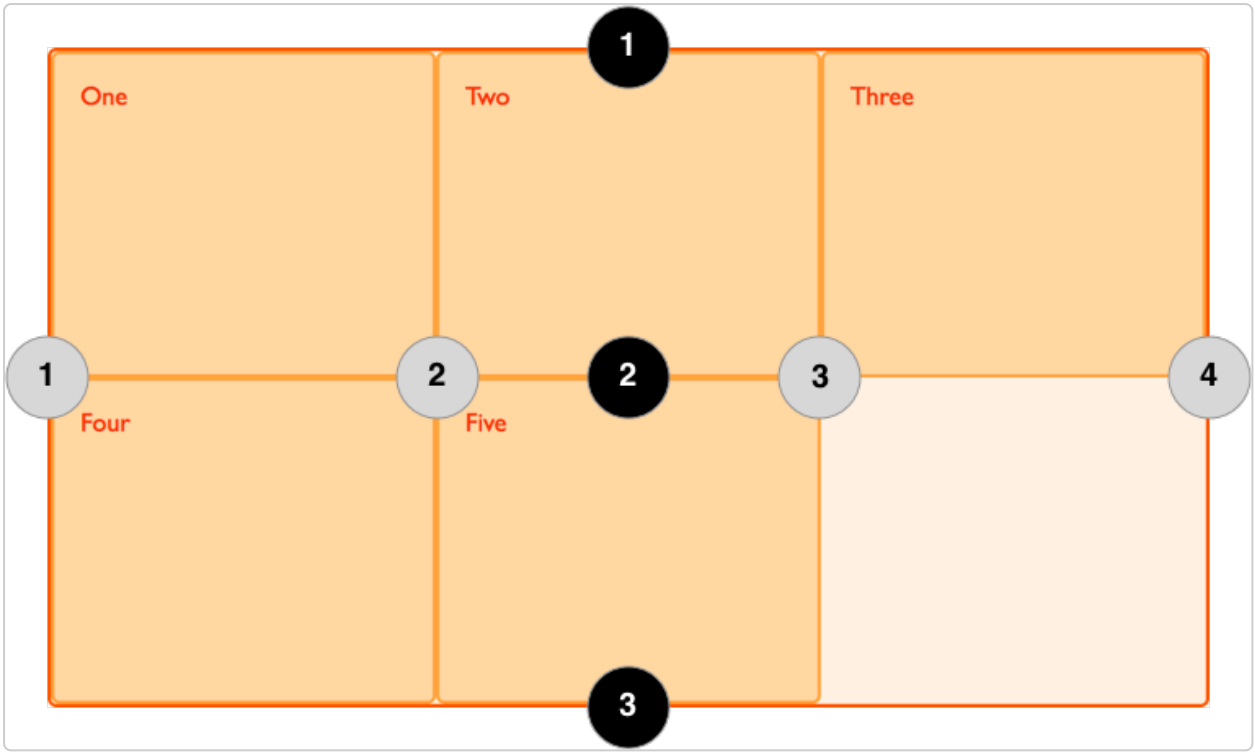
```
<div class="wrapper">
  <div>One</div>
  <div>
    Two
    <p>I have some more content in.</p>

    <p>This makes me taller than 100 pixels.</p>
  </div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

Play



itens. Em nossa grid de 3 colunas por duas linhas, temos quatro linhas de colunas.



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

Exploraremos posicionamento de itens baseado em linhas em um artigo posterior. O exemplo a seguir mostra como fazer isso de forma simples. Quando posicionar um item, focamos a linha - ao invés do rastro.

Nesse exemplo eu estou posicionando os primeiros dois itens do nosso grid de três colunas, usando as propriedades `grid-column-start` (inglês), `grid-column-end` (inglês), `grid-row-start` (inglês) e `grid-row-end` (inglês). Da esquerda para a direita, o primeiro item é posicionado contra a linha da coluna 1, e vai até a linha da coluna

O segundo item inicia na linha da coluna 1, e expande uma linha. Esse é o padrão, então não preciso especificar a linha final. Também vai da linha 3 a linha 5. Os outros itens serão posicionados de acordo com espaço disponível no grid.

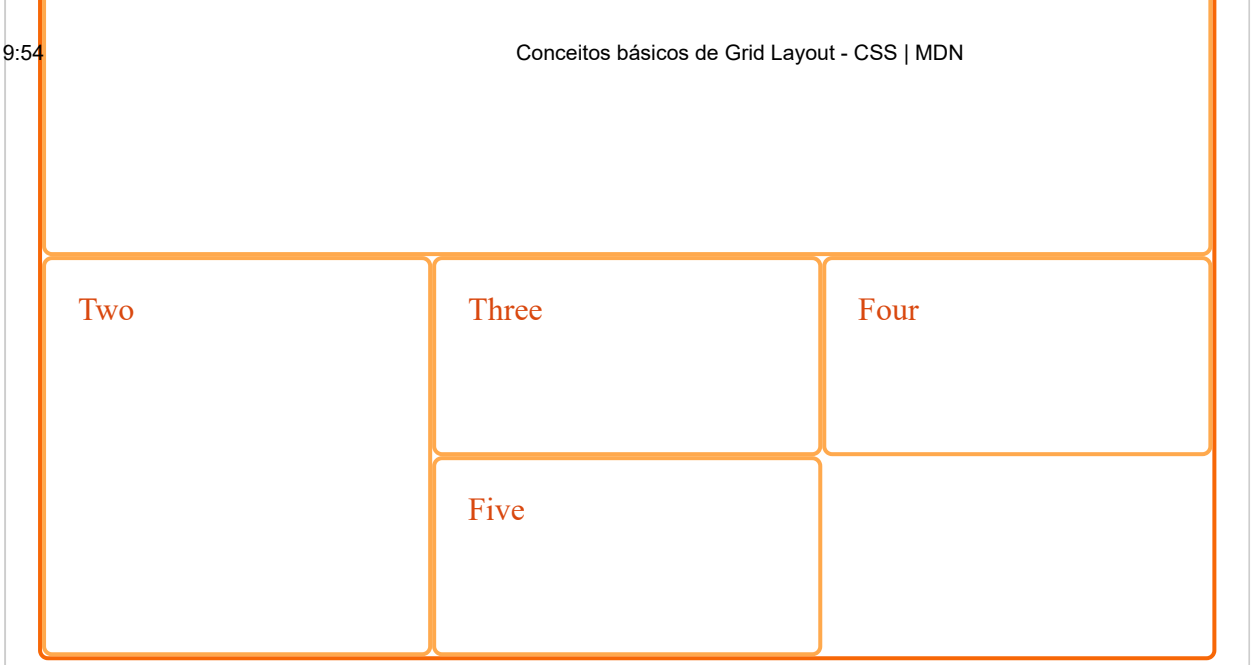
HTML Play

```
<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>
```

CSS Play

```
.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;
}
```

Play



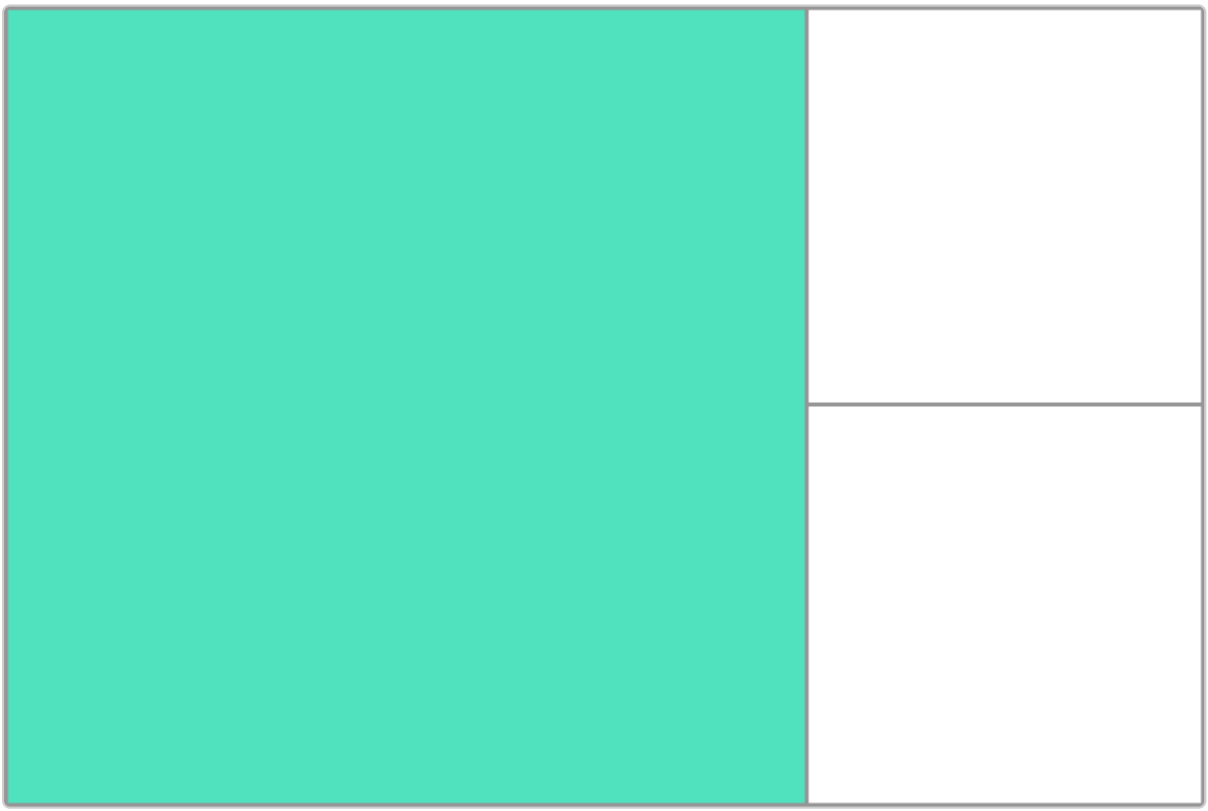
Não esqueça que o Grid Inspector pode ser usado no Firefox Developer Tools para ver como os itens estão posicionados no grid.

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. Na imagem abaixo, temos destacado a primeira célula do grid.

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.



criadas usando a propriedade `grid-column-gap` e `grid-row-gap` ou de forma resumida `grid-gap` (inglês). No exemplo abaixo, criamos um espaço vazio de 10 pixels entre colunas e um espaço vazio de `1em` entre linhas.

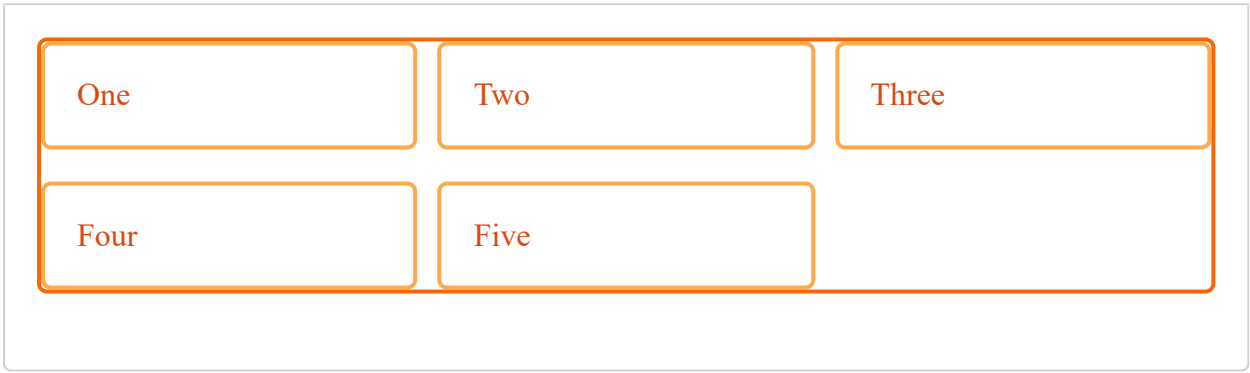
CSS Play

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

HTML Play

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

Play



Qualquer espaço utilizado entre espaços vazios ou *gaps* deverá ser considerado antes do espaço ser designado para as trilhas de tamanho flexível `fr` , e *gaps* agem para propósitos de tamanho como uma trilha de grid normal, entretanto você não pode colocar coisa alguma em um *gap* Em termos de posicionamento baseado em linhas, o *gap* age como uma linha preenchida (*fat line*).

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.

HTML Play

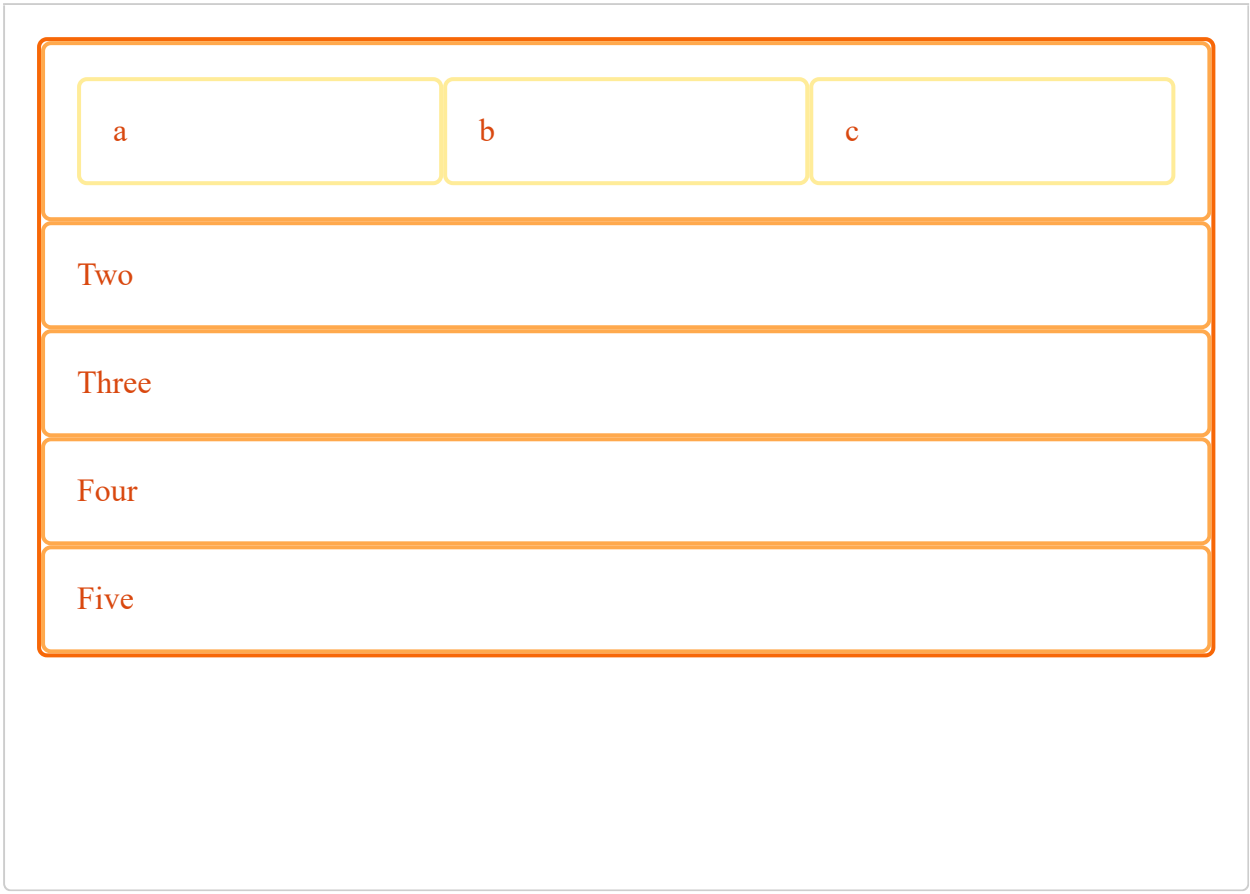
```
<div class="wrapper">
  <div class="box box1">
    <div class="nested">a</div>
    <div class="nested">b</div>
    <div class="nested">c</div>
  </div>
  <div class="box box2">Two</div>
  <div class="box box3">Three</div>
  <div class="box box4">Four</div>
  <div class="box box5">Five</div>
</div>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

Se definimos `box1` como `display: grid` podemos dar uma definição de tracks e ela também se tornará uma grid. Os itens então são dispostos nesse novo grid.

CSS Play

```
  grid-row-end: 3;
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

Play



Nesse caso o grid aninhado não possui relação com o pai. Como é possível perceber no exemplo ele não herdou o `grid-gap` (inglês) 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.

Na especificação atual, no exemplo acima estaríamos 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.

Note que o grid aninhado está nas duas dimensões — linhas e colunas. Não há garantia de que o grid implícito funcione com subgrid. Por isso é necessário que haja certeza de que o grid pai tem linhas e colunas suficientes para todos os sub itens.

CSS Play

```css
.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.

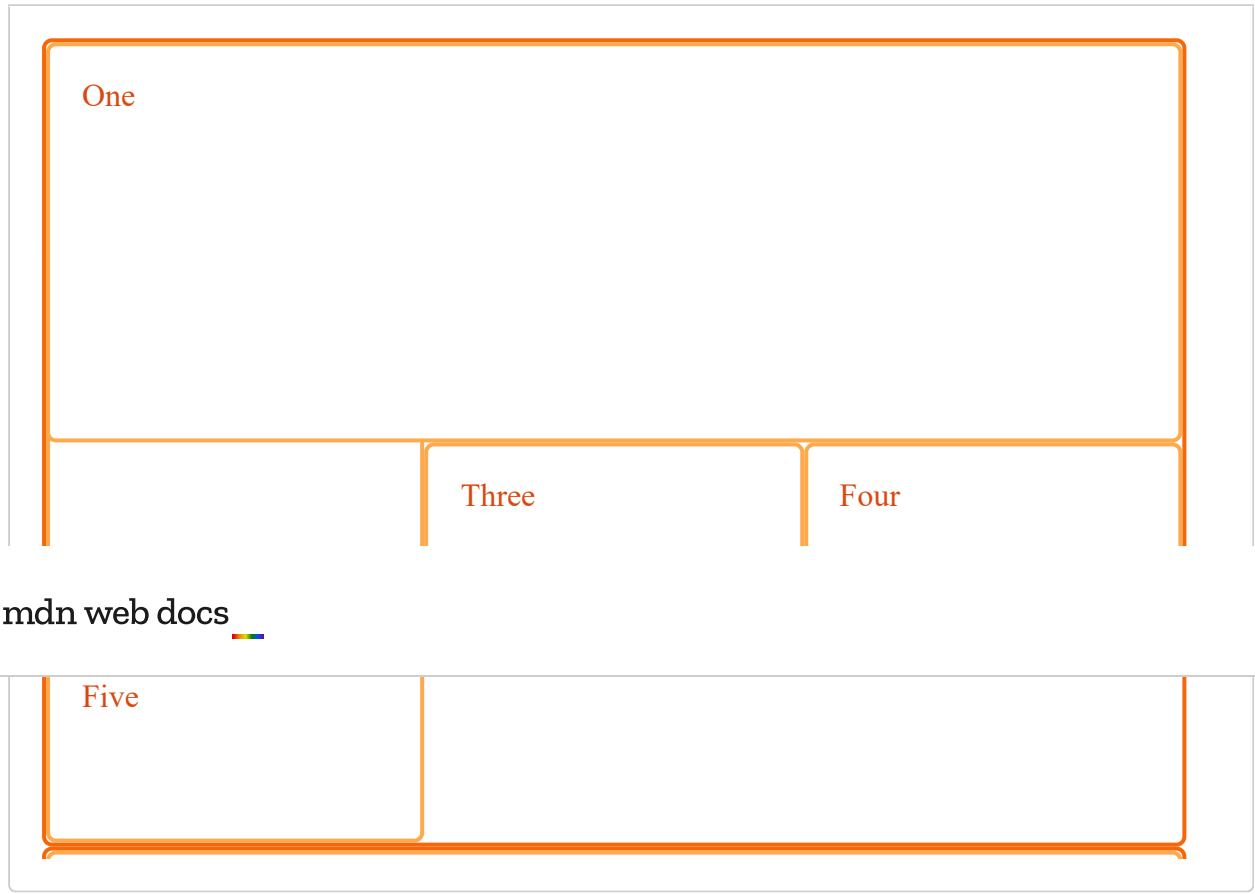
HTML Play

```html
<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>
```

CSS Play

```
}
.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;
}
```

Play



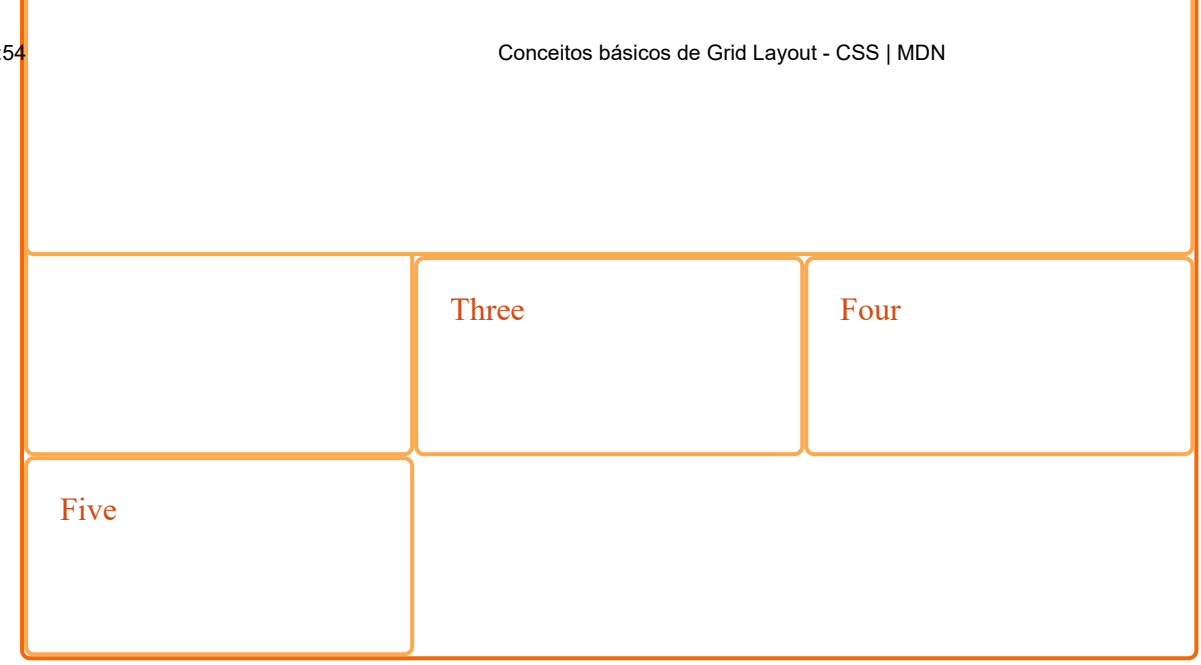
O container `box2` está sobrepondo `box1`, é renderizado acima pois vem depois na ordem.

Manipulando a ordem

CSS Play

```css
.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;
}
```

Play



Próximos Passos

Nesse artigo abordamos um pouco da especificação do Grid Layout. Pratique com os exemplos propostos, depois disso passe para a próxima parte onde estudaremos mais a fundo o CSS Grid Layout.

1. **CSS**
2. **CSS Reference**
3. CSS Grid Layout
4. **Guias**
    i. **Basics concepts of grid layout**
    ii. Relationship to other layout methods
    iii. Line-based placement (inglês)
    iv. Grid template areas (inglês)
    v. Layout using named grid lines (inglês)
    vi. Auto-placement in grid layout (inglês)
    vii. Box alignment in grid layout (inglês)

xi. Realizing common layouts using grids (inglês)

5. **Propriedades**
   i. grid
   ii. grid-area (inglês)
   iii. grid-auto-columns (inglês)
   iv. grid-auto-flow
   v. grid-auto-rows (inglês)
   vi. grid-column (inglês)
   vii. grid-column-end (inglês)
   viii. grid-column-gap (inglês)
   ix. grid-column-start (inglês)
   x. grid-gap (inglês)
   xi. grid-row (inglês)
   xii. grid-row-end (inglês)
   xiii. grid-row-gap (inglês)
   xiv. grid-row-start (inglês)
   xv. grid-template (inglês)
   xvi. grid-template-areas (inglês)
   xvii. grid-template-colunms
   xviii. grid-template-rows
6. **Glossário**
   i. Grid lines (inglês)
   ii. Grid tracks (inglês)
   iii. Grid cell (inglês)
   iv. Grid areas
   v. Gutters (inglês)

Help improve MDN

Was this page helpful to you?

Yes No

Learn how to contribute.

This page was last modified on 3 de ago. de 2023 by MDN contributors.