

O melhor guia para CSS Grid

Você pode [me seguir no Twitter](#), onde publico meus tutoriais do Medium.



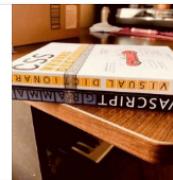
Professor de JavaScript [Segue](#)
25 de abril · 11 min de leitura ★



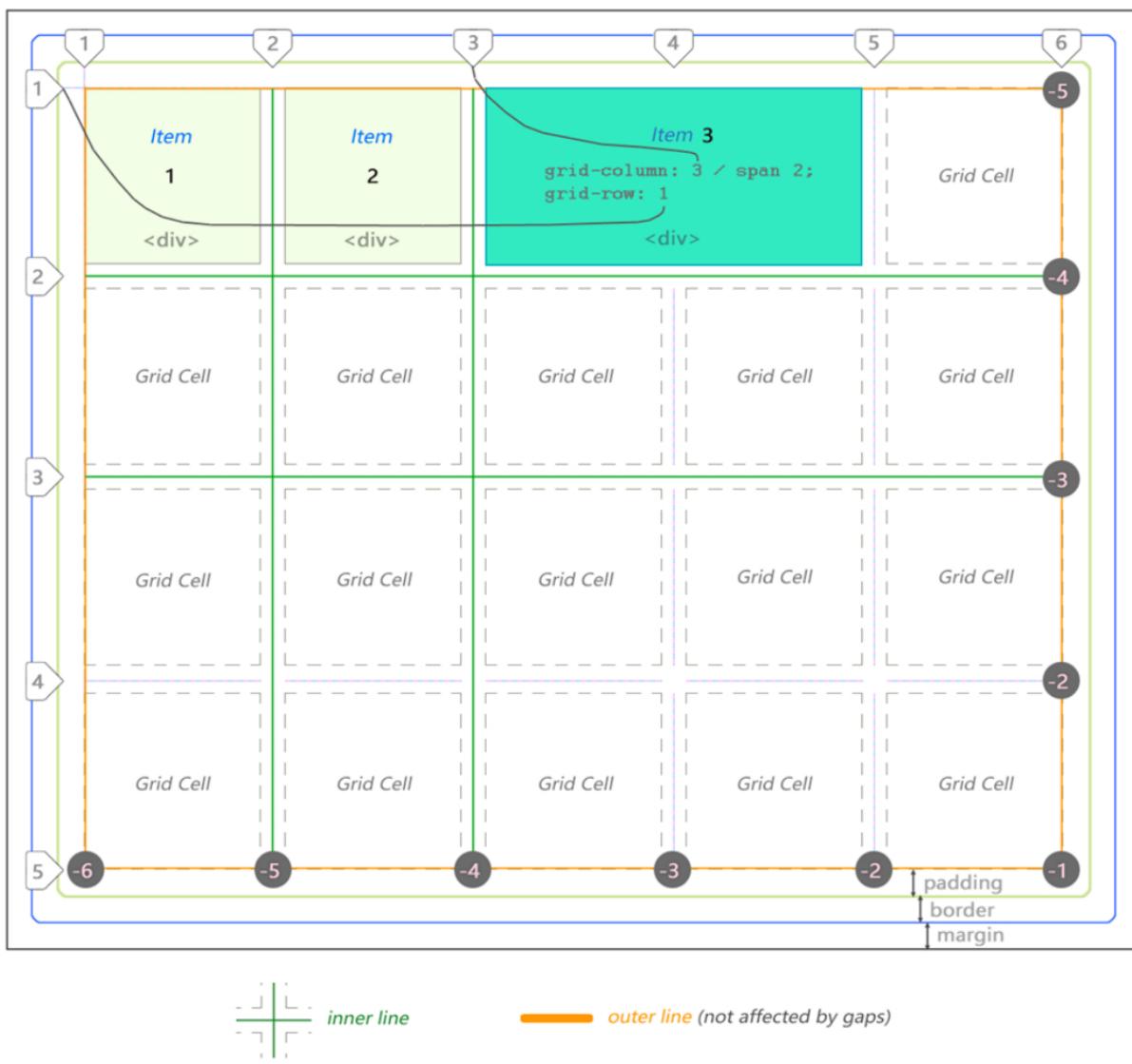
Meus livros de codificação

O CSS Visual Dictionary contém diagramas visuais para todas as propriedades CSS de uso comum. A gramática JavaScript contém...

[medium.com](#)



Você provavelmente já conhece o **modelo de caixa** CSS para elementos regulares. Vamos começar com uma representação similar de "*visão geral*" da CSS Grid:



O CSS Grid Anatomy é composto pelo **contêiner primário**, que é apenas o elemento `<div>` padrão que possui margem, borda e preenchimento. Para criar um contêiner de grade CSS pai com qualquer elemento, adicione `display: grid` a ele. Os itens da grade são filhos aninhados dentro do contêiner pai. Eles geralmente são definidos como uma lista de elementos que podem representar um **cabeçalho**, **barra lateral**, **rodapé** ou elementos de layout de site semelhantes, dependendo do design do aplicativo.

Nesse caso, existem 3 *itens* <div>. O terceiro é esticado por 2 células.

As *linhas de aviso* também podem ser contadas para trás usando o *sistema de coordenadas negativo*.

A grade acima tem 5 por 4 *células* na dimensão. É definido da seguinte forma:

```
div # grid {           /* Este é o nosso contêiner pai da
  grade CSS */
  display: grid;
  colunas de modelo de grade : 100px 100px 100px 100px 100px; /* 5
  colunas */
  linhas de modelo de grade : 100px 100px 100px 100px;
  /* 4 linhas */
}
```

O número de *linhas* e *colunas* é assumido *implicitamente* pelo número de valores definidos.

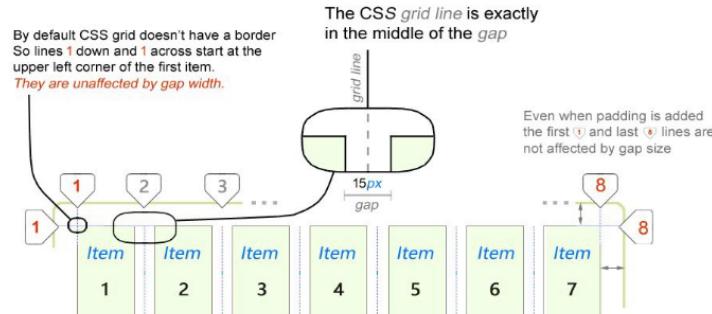
Entre cada *célula*, há uma *linha* e uma *lacuna* opcional.

Linhas e colunas entre as *linhas* são chamadas de *trilhas* da grade.

Sempre há [*célula* + 1] *linhas* por dimensão.

Portanto, 5 colunas terão 6 *linhas*, enquanto 4 linhas terão 5 *linhas*.

No exemplo a seguir, existem 7 *colunas* e apenas 1 *linha*:



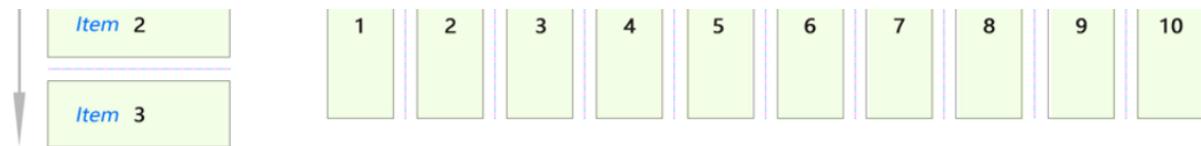
Algumas das primeiras coisas que você notará sobre o comportamento da grade CSS.

A primeira coisa *importante* que você notará sobre a grade CSS é que *as linhas externas* não são afetadas pelo tamanho da lacuna. Somente *linhas internas*. Vamos aprofundar isso um pouco mais adiante neste tutorial, quando examinarmos unidades fracionárias (*fr*).

A grade CSS é *bidirecional*. Seus itens podem fluir *horizontalmente* (*coluna*) ou *verticalmente* (*linha*). Defina o valor com a propriedade *grid-auto-flow*.

Funciona como o Flex:





Usando `grid-auto-flow: row` ou `grid-auto-flow: column` para determinar a direção do fluxo dos itens da grade.

Pense na grade desta maneira abstrata:

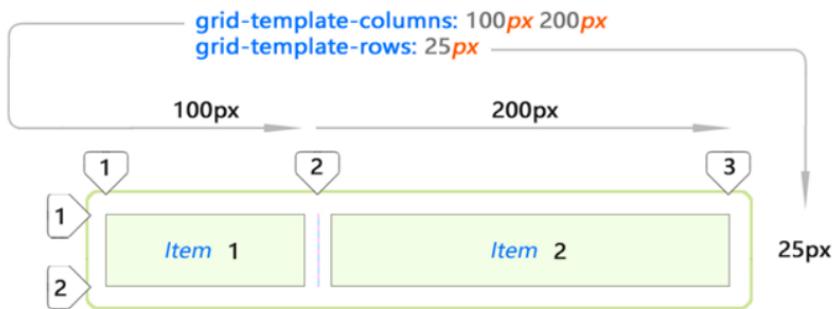


`grid-template-columns: 100px 100px 100px 100px 100px`
`grid-template-rows: 100px 100px`

Ok - então tivemos a ideia básica de como funciona.

A parte criativa surge quando você se depara com o problema de manipular os canais de itens para criar um *layout de aplicativo sensato*. CSS Grid oferece várias propriedades para fazer exatamente isso. Vamos dar uma olhada neles na próxima seção deste tutorial em apenas um momento.

Vamos consolidar nosso conhecimento até agora, analisando estes exemplos:



Eu usei apenas dois elementos `<div>` para os itens. Portanto, a grade acima.

Posicionamento implícito e explícito do conteúdo

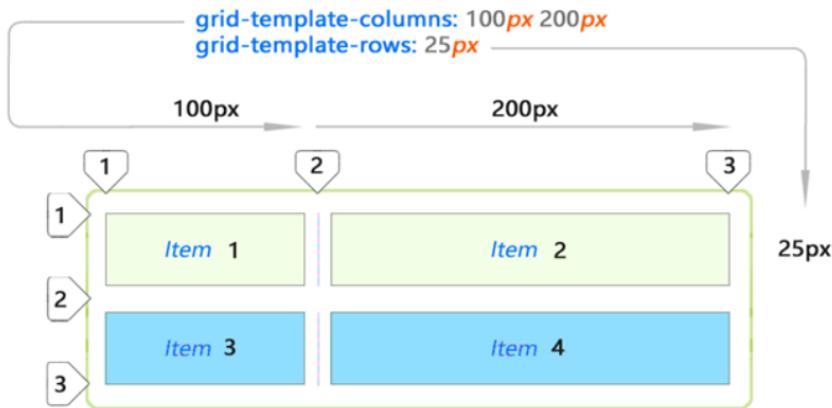
Mas o que acontece se *adicionarmos mais um item* à lista?



Adicionar o item 3 ao mesmo layout *o* estenderá **automaticamente** (item azul.)

Esse novo espaçamento é criado **automaticamente**, *copiando* valores da primeira linha.

Vamos adicionar o *item 4*, devemos?

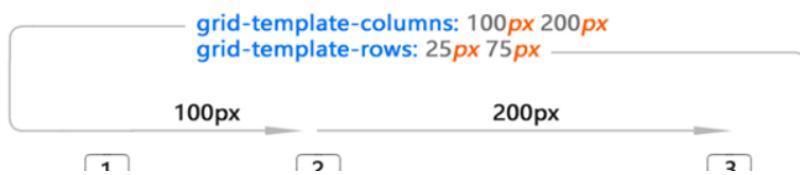


E, novamente, nossa grade de CSS decidiu estender o *Item 4* pelo espaço restante na segunda linha. Isso ocorre porque **as linhas de modelo de grade** especificaram espaço suficiente apenas para *1 linha*. O resto é automático.

O posicionamento dos itens azuis não é especificado *explicitamente* por você. Esse é *um* posicionamento **implícito** (*automático*). Eles meio que caem nesse espaço.

Posicionamento explícito do conteúdo

Isso é exatamente o que você esperaria das células da grade se você definir valores personalizados para todos os itens da lista:

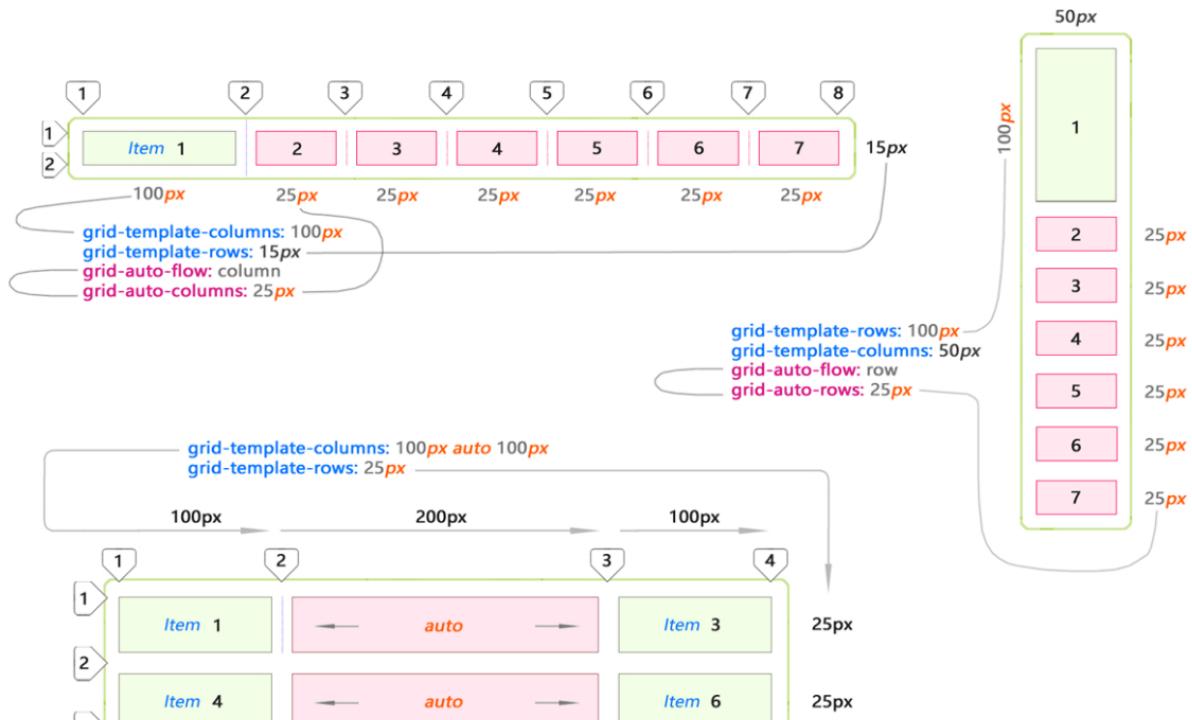




Basicamente, você pode obter controle sobre o espaço em todas as linhas consecutivas adicionando mais valores à propriedade `grid-template-lines`. Observe que os itens não estão mais *implícitos* aqui. Você os definiu para serem exatos. (25px 75px)

Espaçamento automático

A grade CSS oferece algumas propriedades para expandir automaticamente suas células por uma quantidade variável / desconhecida de espaço. Aqui estão os principais exemplos de casos de *fluxo automático de colunas e linhas*:

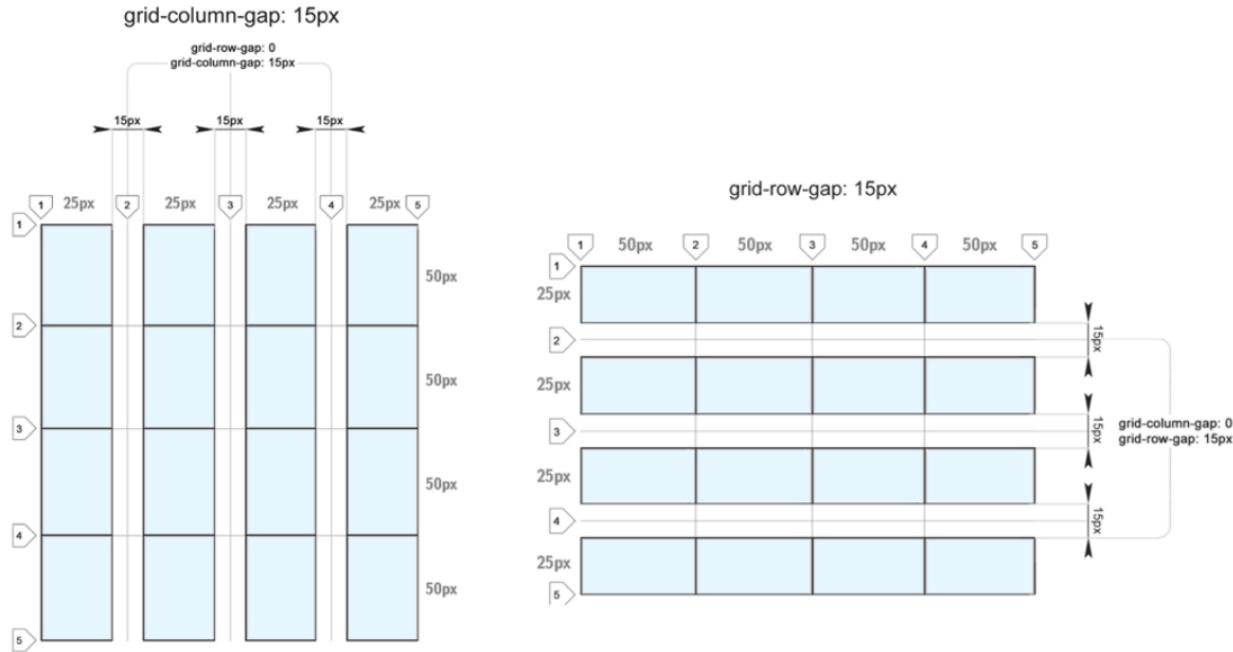


O exemplo inferior demonstra o uso da palavra-chave `automática`. Isso significa apenas que a célula se estenderá para preencher, no entanto, resta muito espaço no contêiner pai depois que ele já foi preenchido por itens colocados *explicitamente*.

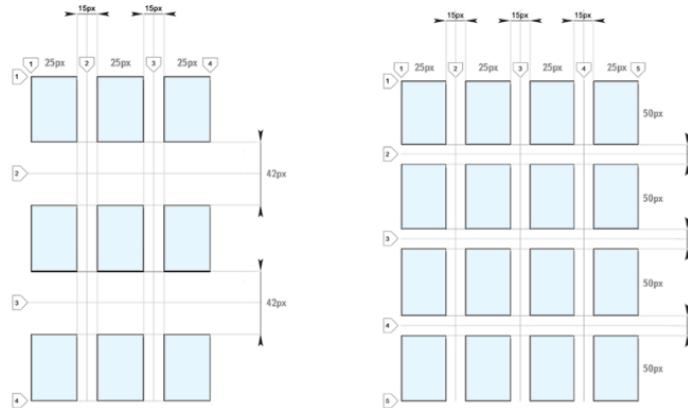
Lacunas na grade do CSS

Falando sobre grade CSS, você não pode deixar de falar sobre lacunas. Lacunas são os espaços horizontais e verticais entre as células da grade.

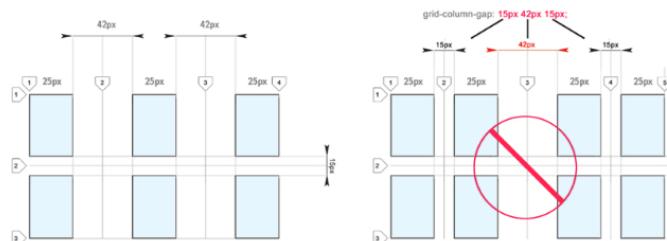
Os intervalos são controlados usando as propriedades de **intervalo de coluna da grade** e **intervalo de linha da grade** :



Você pode usar intervalos diferentes nas duas dimensões. talvez isso possa ser útil para criar **galerias de vídeos ou imagens** :



As lacunas nas dimensões (*colunas e linhas*) podem diferir em tamanho. Mas o tamanho da lacuna é especificado uma vez para todas as lacunas na grade em uma determinada dimensão. Como você pode ver aqui, não são permitidas lacunas de *tamanhos variados na mesma dimensão* :

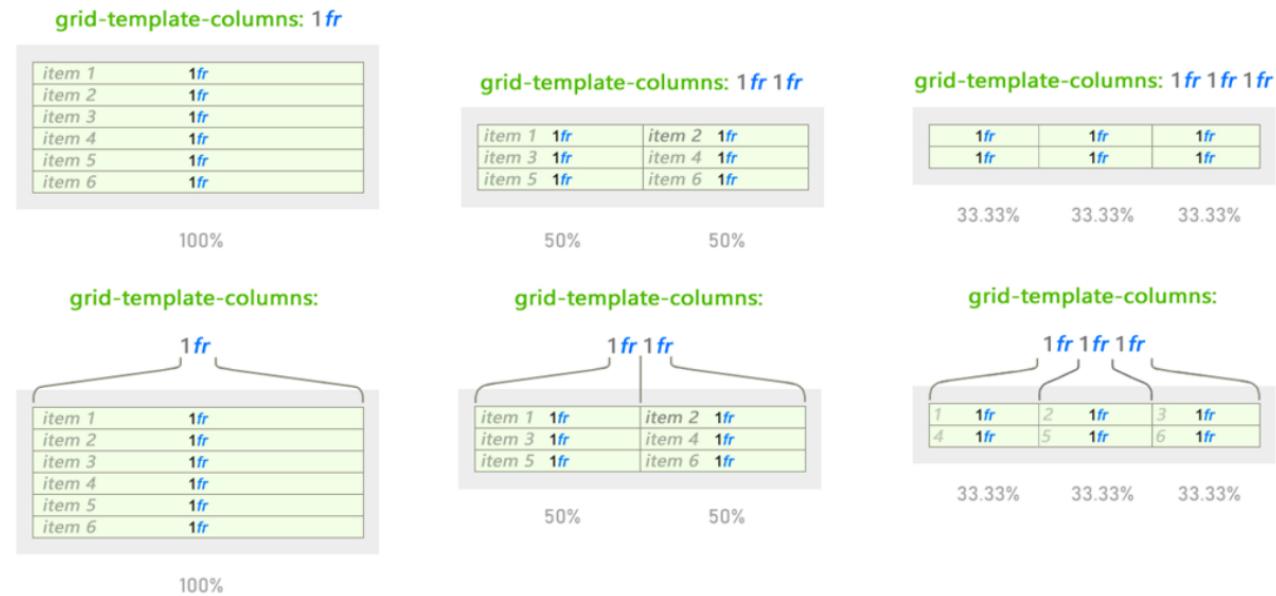


Eu realmente gostaria que possíveis diferenças de tamanho fossem possíveis. Eu posso ver como isso pode realmente ser útil. Alguns sugerem o uso de faixas vazias para obter um efeito *semelhante*.

Unidades FR (unidades fracionárias)

As unidades fracionais (*fr*) são exclusivas da grade CSS.

Uma unidade *fracionária* é alocada em relação a todos os outros elementos no pai:



O comportamento muda, mas *1fr* permanece o mesmo, independentemente quando valores diferentes são usados. As unidades fracionárias funcionam de maneira *semelhante aos valores%*, mas são mais fáceis e *intuitivas* para dividir o espaço com:



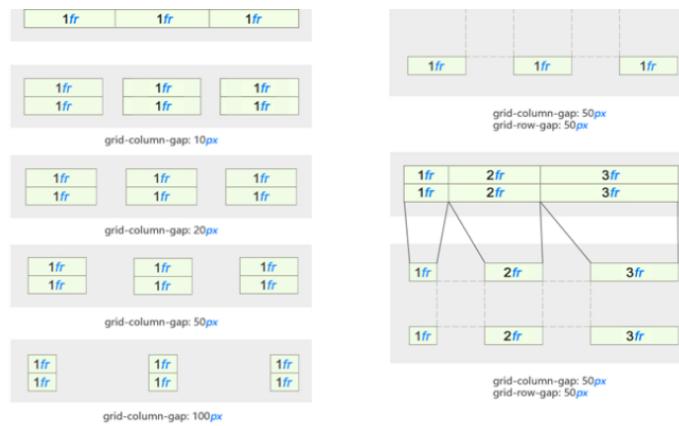
O comportamento das unidades fracionais (unidade *fr*) é alterado com base em todos os valores fornecidos em qualquer dimensão.

Neste exemplo, apenas o comportamento no sentido da *coluna* é mostrado por uma questão de simplicidade. Mas funciona da mesma forma para as linhas também. Basta usar a propriedade **grid-template-lines**.

Unidades fracionárias e sua relação com lacunas

O espaço definido usando unidades fracionárias muda com base nas lacunas. O mesmo *1fr* dentro do mesmo pai diminuirá para um tamanho menor quando as lacunas forem adicionadas:



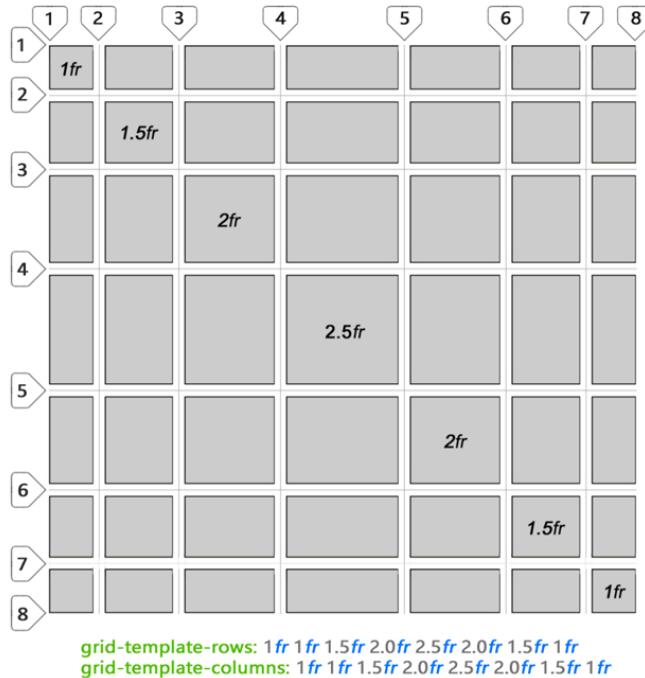


Aqui, adicionamos espaços nas células especificadas usando unidades fr .

Como você pode ver, isso fornece um conjunto muito bom de propriedades para espaçar o conteúdo basicamente da maneira que você desejar, sem se preocupar com os valores de pixel.

Essas novas dinâmicas tornam o design com *pixel perfeito* como *algo do passado* . Vamos agora pensar no design do layout usando a abordagem intuitiva!

Finalmente, para ter uma idéia melhor do uso *de* unidades fracionárias *não inteiras* , aqui está uma grade divertida que criei. Você também pode especificá-los usando *números de ponto flutuante* :



Posicionamento de conteúdo

Acabamos de dissecar a anatomia da grade CSS. Espero que você tenha uma idéia melhor de como a grade CSS *estrutura* o conteúdo. Mas agora precisamos ser criativos e colocar alguns itens dentro dela. Como isso é feito pode modificar o comportamento padrão da grade CSS. Vamos explorar como isso acontece nesta seção.

Para organizar seus itens entre *células* ou *áreas de modelo* na grade, você se referirá a eles por *linhas* entre *células* . Não <table> - como spans.

A grade CSS permite o uso de **extensões** para determinar a *largura* e a *altura* da área de conteúdo (*no espaço da célula*), assim como as tabelas. Vamos explorar isso daqui a pouco. Mas você ainda pode e provavelmente deve especificar a célula inicial usando **números de linhas** ou **linhas nomeadas** (*mais sobre isso daqui a pouco.*) Isso depende da sua preferência.

No que diz respeito ao posicionamento do conteúdo em várias células, a coisa mais óbvia e tentadora é **a abrangência das células**.

Abrangência do conteúdo da célula

Você pode estender um item por várias células.

Importante: A extensão muda a localização dos itens ao redor.

Extensão usando coluna de grade e linha de grade

Usando **grade da coluna** e **grade-fila** propriedades no elemento item em si:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

1	2	3	4	5
6	7			8
9	10			10
11	12	13	14	15
16	17	18	19	20

Os **itens azuis** mudaram de local depois de fazer o **Item 7** se estender por várias células. E **itens laranja** foram colididos em uma fileira.

Há também outra maneira de fazer a mesma coisa...

Abrangendo usando grid-column-start...

... **grid-column-end**, **grade-row-start** e **grade-row-end** você pode especificar partida real e pontos terminando através da qual você deseja conteúdo da célula span.

Eu removi os itens além dos 15 (**laranja**) porque não precisamos mais deles:

1	2	3	4	5
---	---	---	---	---

1	2	3	4	5
---	---	---	---	---

6	7	8		
9		10		
11	12	13	14	15

```
grid-column-start: 2;
grid-column-end: 5;
grid-row-start: 2;
grid-row-end: 4;
```

6	7	8		
9		10		
11	12	13	14	15

```
grid-column-start: 5;
grid-column-end: 2;
grid-row-start: 4;
grid-row-end: 2;
```

Digite essas propriedades diretamente no item que você deseja que elas sejam afetadas.

Alongamento conteúdo em *colunas* e *linhas* funciona em ambas as direções.

conteúdo mínimo e conteúdo máximo

Os valores *min-content* e *max-content* são fornecidos às propriedades de *colunas-modelo-grade* ou *linhas-modelo-grade*, como qualquer outro valor relacionado ao tamanho (por exemplo , *px* , *1fr* etc.)

grid-template-columns: 1fr 1fr 1fr
hello mid right
<div>hello</div> <div>mid</div> <div>right</div>

Vamos dar uma olhada neste espécime. É o nosso ponto de partida. Mudaremos um pouco as coisas para ver como os valores *mínimo* / *máximo* afetam as células.

Vamos ver que tipo de resultados será produzido se mudar uma das colunas para *min-content* e *max-contúdo* :

```
grid-template-columns:
min-content 1fr 1fr
```

hello	mid	right
-------	-----	-------

```
grid-template-columns:
max-content 1fr 1fr
```

hello	mid	right
-------	-----	-------

Com o texto de *uma palavra* , não há diferença entre os resultados observados, se usamos *conteúdo mínimo* ou *máximo* . Aqui está porque *olá* é uma única palavra. Seus valores *mínimo* e *máximo* são exatamente os mesmos.

Mas as coisas ficam interessantes com textos mais complexos. O exemplo que se segue vai demonstrar a ideia básica por detrás *min-teor* e *max-teor* :

```
grid-template-columns:
min-content 1fr 1fr
```

Hello there stranger	mid	right
1fr	1fr	1fr

```
grid-template-columns:
max-content 1fr 1fr
```

Hello there stranger	mid	right
1fr	1fr	1fr

Aqui o **conteúdo mínimo** usava a palavra mais longa na frase (*estranho*) como *largura da base*.

Ao usar o **conteúdo máximo**, toda a cadeia de texto com espaços preencheu o espaço.

Mas o que acontece se aplicarmos **conteúdo mínimo** ou **máximo** a *todas as células*?

`grid-template-columns:
min-content min-content min-content`

`or also: repeat(3, min-content)`

Hello	mid	right
there		
stranger		

`grid-template-columns:
max-content max-content max-content`

`or also: repeat(3, max-content)`

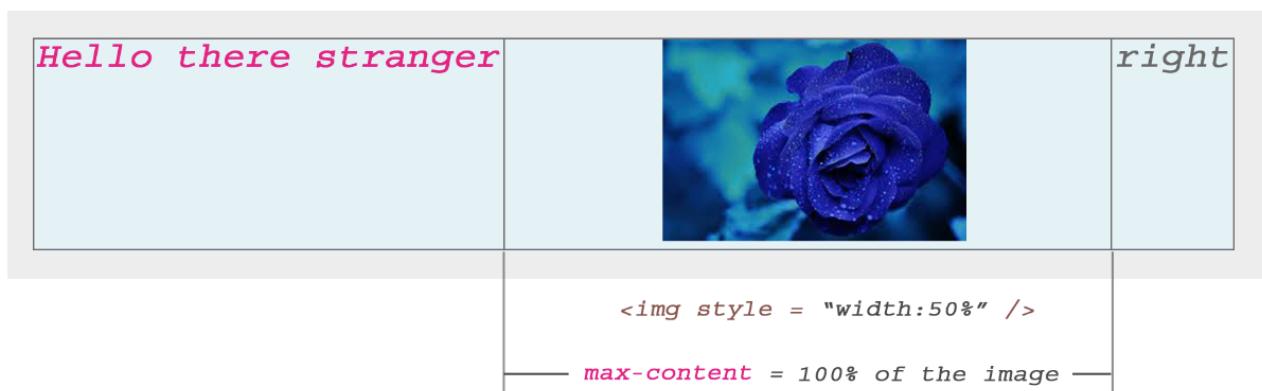
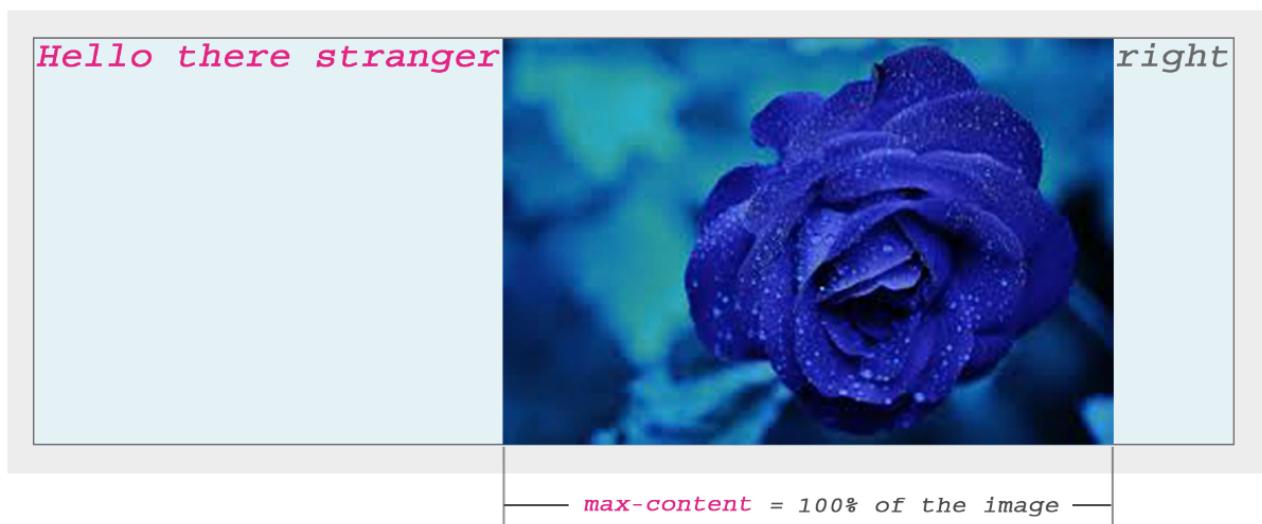
Hello	there	stranger	mid	right

Percebi que, por padrão, o texto era centralizado sempre que eu usava **conteúdo mínimo**, embora o **alinhamento de texto: o centro** não estivesse definido no item.

Imagens e conteúdo máximo

Coloquei a imagem desta *rosa azul* na cela.

E, como esperado, a grade foi expandida para alocar espaço suficiente:



Quando defino explicitamente a largura da imagem para 50%, *apenas para ver o que acontece*, a CSS Grid *ainda mantinha a largura da célula em 100% da imagem*, mas exibia a imagem com 50% de largura (*conforme o esperado*) e *a centralizava automaticamente horizontalmente* na célula.

Tanto o texto quanto as imagens (*ou qualquer conteúdo*) serão automaticamente centralizados nas células do CSS Grid por padrão.

Posicionamento de Conteúdo

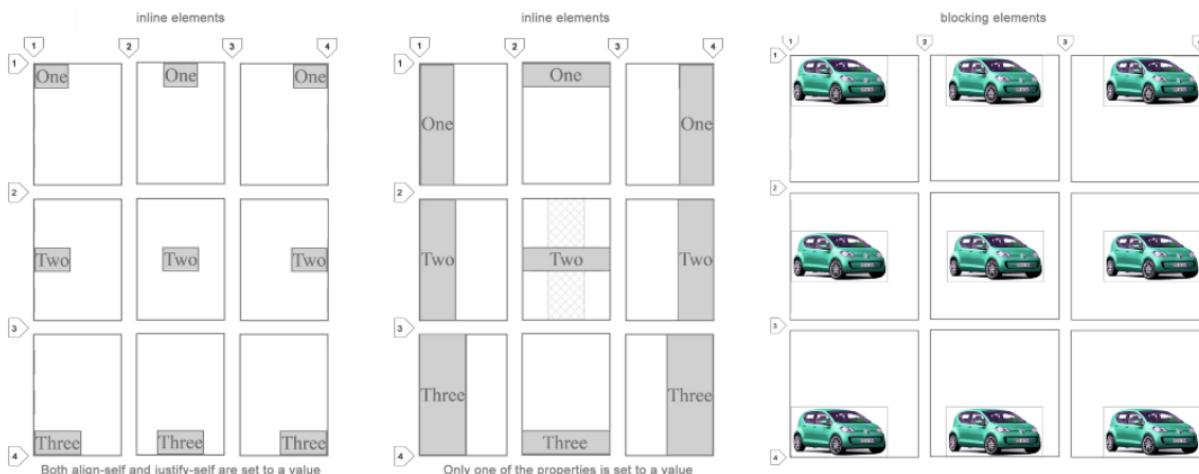
Até este ponto, falamos sobre a estrutura do Grid em geral.

Na próxima seção, veremos como obter flutuações "multidirecionais" *dentro das células*. Não *usaremos* a propriedade *float* aqui, é claro.

Flutuador multidirecional de 360 °

Eu não acho que a especificação CSS Grid chama assim. Mas, de fato, é possível criar exatamente isso ... um *comportamento flutuante de 360 graus*.

Isso funciona nos elementos *inline* e de *bloqueio*! E acho que esse é o meu recurso favorito de todo o conjunto de habilidades do CSS Grid.

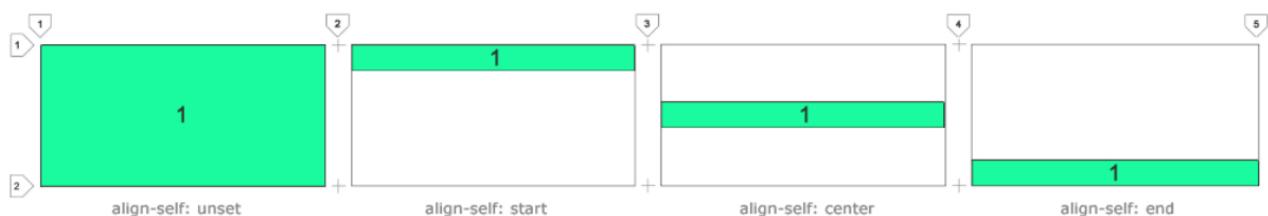


Todos os 9 combinações são possíveis usando **alinhamento auto** e **justificam-
auto** propriedades.

Eles são explicados abaixo.

Alinhar Self (Alinhar Self)

Essa propriedade ajuda a posicionar o conteúdo *verticalmente*.



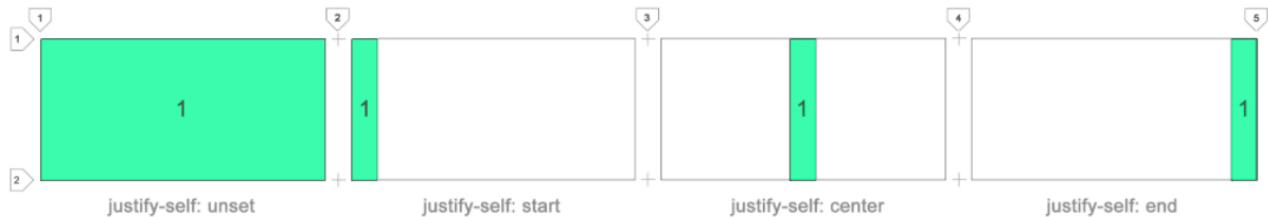
Use **align-self: comece** a alinhar o conteúdo com a borda superior da célula.

Use **align-self: center** para alinhar o conteúdo ao meio vertical.

Use **align-self: end** para alinhar o conteúdo com a parte inferior da célula.

Justificar-se (justificar-se)

Essa propriedade ajuda a posicionar o conteúdo **horizontalmente**.



Use **justify-self: comece** a alinhar o conteúdo com a borda esquerda da célula.

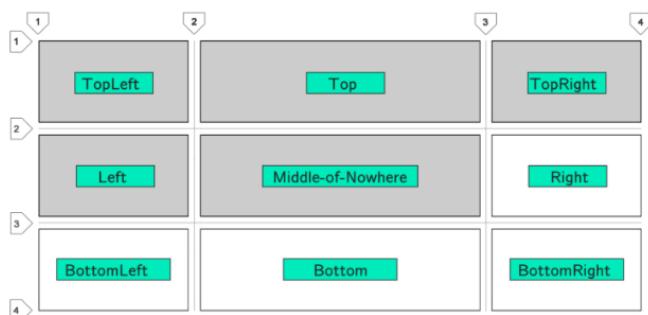
Use **justify-self: center** para alinhar o conteúdo ao meio horizontal.

Use **justify-self: end** para alinhar o conteúdo à borda direita da célula.

Você pode usar qualquer uma das 9 combinações **justificar-auto** x **alinear-auto** para alinhar qualquer coisa em qualquer lugar, também conhecido como *flutuador multidirecional*.

Áreas de modelo

As áreas de modelo são definidas usando a propriedade **grid-template-areas**.



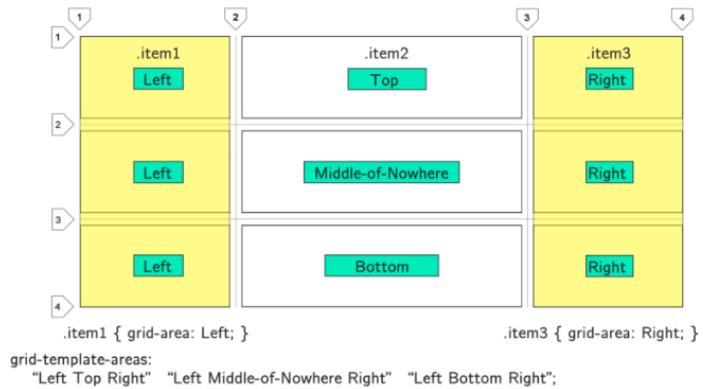
`grid-template-areas:`
"TopLeft Top TopRight" "Left Middle-of-Nowhere Right" "BottomLeft Bottom BottomRight";

Observe que as áreas de modelo para cada *linha* são colocadas entre **aspas duplas**.

Cada *coluna* é separada pelo espaço .

Neste exemplo, eu simplesmente expliquei como nomear ares. Para tirar proveito real das áreas do modelo, você precisa **categorizar blocos retangulares de células** com o mesmo nome.

Blocos Tetris não são permitidos . Você só pode usar *retângulos*:



Aqui, a **esquerda** é uma área que mede 3 células para baixo. CSS Grid trata automaticamente como um único bloco. O mesmo vale para a **direita**. Neste exemplo simples, criei duas colunas. Mas você entendeu. Bloqueie áreas maiores, nomeando-as.

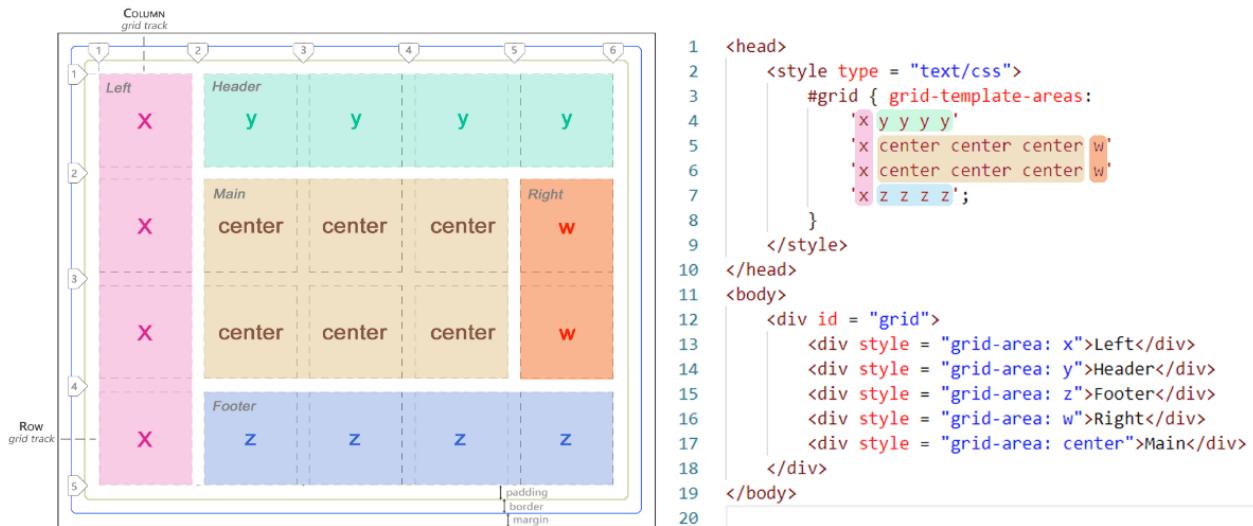
Para colocar um item nessa área, basta adicionar a **área da grade**: `TemplateName`. Nesse caso, é a **área da grade: esquerda ou a área da grade: direita**.

Os nomes de área do modelo não podem usar espaços . Eu usei traços aqui.

Exemplo prático de áreas de modelo de grade CSS

Agora entendemos como bloquear áreas retangulares. Vamos dar uma olhada em um cenário potencialmente real. Aqui vou demonstrar um layout muito básico.

Bloqueiei um *layout de site muito simples* com duas **barras laterais**, áreas de **cabeçalho e rodapé**. A área principal fica no **centro**, ocupando **3 x 2** células:



Cada área codificada por cores é denominada área de modelo. Você só precisa de 1 elemento para o seu conteúdo.

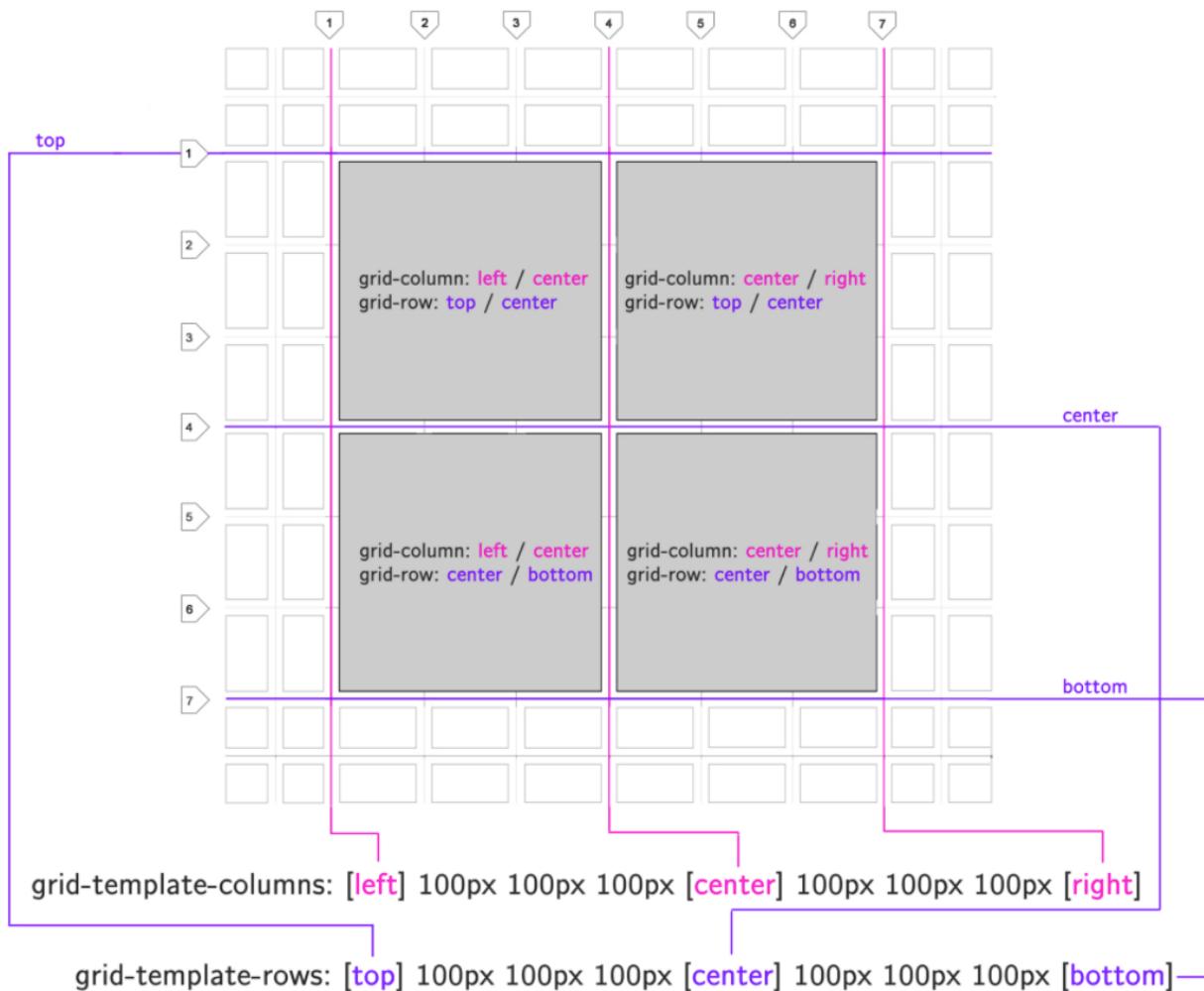
Só precisamos de 5 itens aqui. Adicione mais e eles serão empurrados para fora da área da grade principal para células implícitas.

Apenas certifique-se de manter sempre suas áreas *quadradas* ou *retangulares*.

Linhas de nomeação

Em vez de sempre se referir às linhas pelo número, também é possível nomeá-las. Dessa forma, eles serão fáceis de lembrar para esticar itens em várias células. Os números podem ficar entediantes!

Abaixo está uma representação de como ela se parece:



Você também pode nomear linhas (lacunas) com a propriedade `grid-template-column` ou `grid-template- lines` !

Use colchetes para nomear suas linhas. Em seguida, use esses nomes ao especificar o comprimento que seus itens precisam abranger usando / barra.

Em conclusão

CSS Grid é um assunto abrangente. Portanto, este *não* é um tutorial completo da Grade CSS sobre como criar layouts reais de CSS.

Simplesmente usei um exemplo para cada parte separada como ponto de partida para alguém novo na grade.

Esperamos que as informações aqui tenham um interesse perspicaz e inspirado na criação de sites usando CSS Grid.