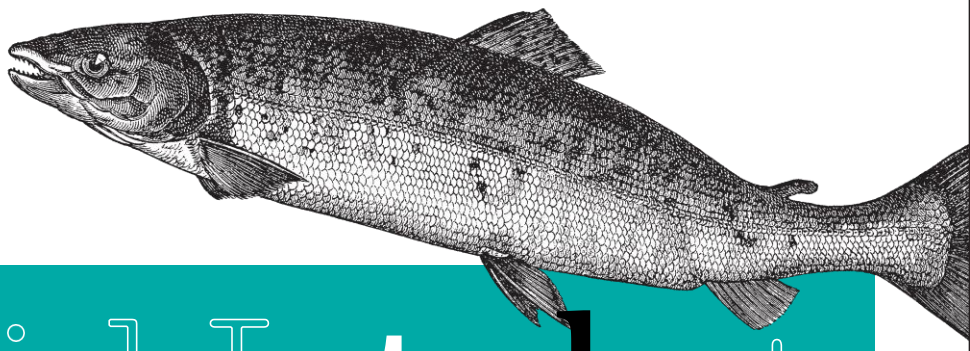
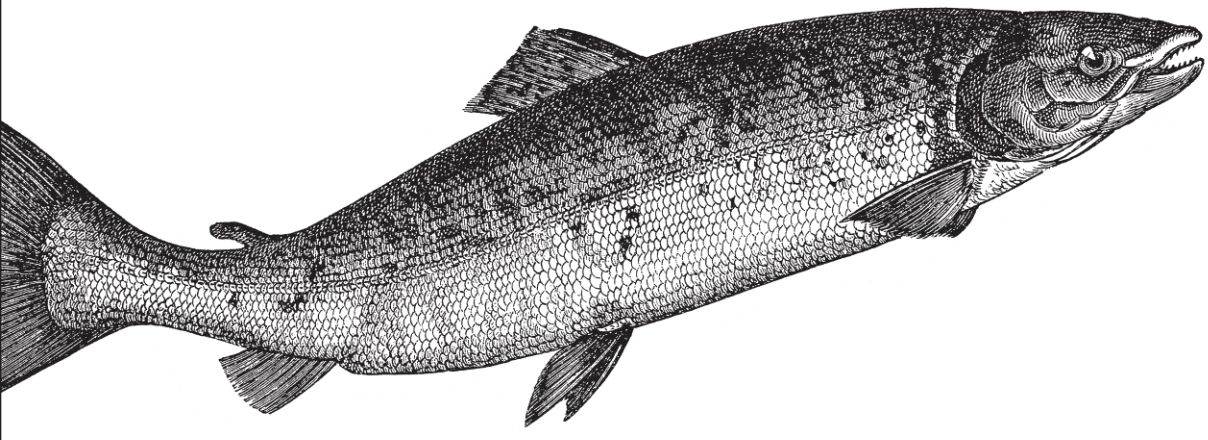
O'REILLY®



Layout de grade em CSS

INTERFACE



Érico A. Meyer

O'REILLY®

Grid Layout in CSS

CSS has had a layout-shaped hole at its center since the beginning. Designers have bent features such as float and clear to help fill that hole, but nothing has quite done the job. Now that's about to change. With this concise guide, you'll learn how to use CSS grid layout, a generalized system that lets you lay out pieces of your design independent of their document source order and with full awareness of the overall design.

Short and deep, this book is an excerpt from the upcoming fourth edition of *CSS: The Definitive Guide*. When you purchase either the print or the ebook edition of *Grid Layout in CSS*, you'll receive a discount on the entire *Definitive Guide* once it's released. Why wait? Learn how to make your web pages come alive today.

- Explore the differences between grid boxes and block containers
- Create block-level grids, inline grids, and even nest grids inside grids
- Learn best practices for attaching elements to your layout, using explicitly defined grid lines or grid area
- Understand how the implicit grid automatically adjusts for oversized elements
- Create gutters between grid elements, and align and justify individual items

Eric A. Meyer is an author, speaker, blogger, sometime teacher, and cofounder of An Event Apart. He's a two-decade veteran of the Web and web standards, a past member of the W3C's Cascading Style Sheets Working Group, and the author of O'Reilly's *CSS: The Definitive Guide*

CSS/WEB DEVELOPMENT

US $7.99 **CAN $9.99**

ISBN: 978-1-491-93021-2

Twitter: @oreillymedia
facebook.com/oreilly

Layout de grade em CSS

Layout de interface *para a Web*

Érico A. Meyer

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

Layout de grade em CSS

por Eric A. Meyer

Publicado por O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Os livros da O'Reilly podem ser comprados para uso educacional, comercial ou promocional de vendas. Edições online também estão disponíveis para a maioria dos títulos (*http://safaribooksonline.com*). Para maisinformações, contacte o nosso departamento de vendas corporativas/ institucionais : 800-998-9938 ou *corporate@oreilly.com.*

Edição: Meg Foley
Editor de Produção: Colleen Lobner
Redator: Molly Ives Brower
Revisor: Sharon Wilkey
Designer de Interiores: David Futato
Designer de Capa: Karen Montgomery **Ilustrador:** Rebecca Demarest

Maio 2016: Primeira Edição

Histórico de Revisões da Primeira Edição
2016-04-14: Primeiro lançamento

Consulte *http://oreilly.com/catalog/errata.csp?isbn=9781491930212* para obter detalhes da versão.

978-1-491-93021-2

[LSI]

Índice

Prefácio

Convenções Usadas neste Livro

As seguintes convenções tipográficas são usadas neste livro:

Itálico
: Indica novos termos, URLs, endereços de e-mail, nomes de arquivos e extensões de arquivo.

Largura constante
: Usado para listagens de programas, bem como dentro de parágrafos para se referir a ele- mentos de programas, como nomes de variáveis ou funções, bancos de dados, tipos de dados, variáveis de ambiente, instruções e palavras-chave.

Largura constante em negrito
: Mostra comandos ou outro texto que deve ser digitado literalmente pelo usuário.

Largura constante itálico
: Mostra o texto que deve ser substituído por valores fornecidos pelo usuário ou por valores dissuadem o contexto.

Este elemento significa uma nota geral.

Esse elemento indica um aviso ou advertência.

Manuais Online do Safari®

O Safari Books Online é uma biblioteca digital sob demanda que oferece conteúdo especializado em forma de livro e vídeo dos principais autores mundiais em tecnologia e negócios.

Profissionais de tecnologia, desenvolvedores de software, web designers, uma empresa e profissionais criativos usam os Manuais Online do Safari como seu principal recurso para pesquisa, resolução de problemas, aprendizado e treinamento de certificação.

Os Manuais Online do Safari oferecem uma variedade de planos e preços para empresas, governos, educação e indivíduos.

Os membros têm acesso a milhares de books, vídeos de treinamento e manuscritos de pré-publicação em um banco de dados totalmente pesquisável de editores como O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press , John Wiley & Sons, Syngress, Morgan Kauf- mann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology e centenas de outros. Para mais informações sobre os Manuais Online do Safari, visite-nos online.

Como entrar em contato conosco

Por favor, envie comentários e perguntas sobre este livro para a editora:

O'Reilly Media, Inc.
1005 Gravenstein Highway
Sebastopol Norte, CA 95472
800-998-9938 (nos Estados Unidos ou Canadá) 707-829-0515 (internacional ou local)
707-829-0104 (fax)

Temos uma página web para este livro, onde listamos errata, exemplos e qualquer informação adicional. Você pode acessar esta página em *http://bit.ly/grid-layout-in-css*.

Para comentar ou fazer perguntas técnicas sobre este livro, envie um e-mail para o *s@oreilly.com de reservas*.

Para obter mais informações sobre nossos livros, cursos, conferências e notícias, consulte nosso site na *http://www.oreilly.com*.

Encontre-nos no Facebook: *http://facebook.com/oreilly*

Siga-nos no Twitter: *http://twitter.com/oreillymedia*

Veja-nos no YouTube: *http://www.youtube.com/oreillymedia*

Layout de grade

Desde que o CSS existe – o que é, acredite ou não, há duas décadas – ele tem um buraco em forma de layout em seu centro. Nós dobramos outros recursos para os propósitos do layout, mais notavelmente flutuante e claro, e geralmente cortamos nosso caminho em torno desse buraco. O layout da caixa Flex- ajudou a preenchê-lo, mas o flexbox é realmente destinado apenas a casos de uso específicos , como barras de navegação (barras de navegação).

O layout de grade, por outro lado, é um sistema *de layout generalizado*. Com sua ênfase em linhas e colunas, a princípio pode parecer um retorno ao layout da tabela – e de certa forma isso não está muito longe – mas há muito, muito mais no layout da grade do que no layout da tabela. O Grid permite que partes do design sejam dispostas independentemente da ordem de origem do documento e até sobreponham partes do layout, se esse for o seu desejo. Existem métodos poderosamente flexíveis para definir padrões repetitivos de linhas de grade, anexar elementos a essas linhas de grade e muito mais. Você pode aninhar grades dentro de grades ou, para esse assunto, anexar tabelas ou contêineres flexbox a uma grade. E muito , muito mais.

Em suma, o layout da grade é o sistema de layout pelo qual esperamos há muito tempo. Há muito a aprender, e talvez ainda mais a desaprender, à medida que deixamos para trás os hacks inteligentes e os problemas que nos levaram nos últimos 20 anos.

Antes de começarmos, uma palavra de cautela: enquanto escrevo isso em abril de 2016, o layout da grade ainda está em um pouco de fluxo. Demoramos a aumentar a publicidade duas vezes devido a mudanças na especificação e nos navegadores de suporte, e pode haver ainda mais por vir. É por isso que o livro só está disponível eletronicamente neste momento. Portanto, tenha em mente que, embora os recursos e capacidades gerais discutidos neste livro quase certamente permaneçam, alguns detalhes ou mesmo adereços podem mudar ou ser descartados. Se você tem o hábito de verificar se há atualizações para seus ebooks, esse título pode justificar uma verificação mais frequente. Se você não tem esse hábito, agora é um excelente momento para começar.

Criando um contêiner de grade

A primeira etapa para criar uma grade é definir um *contêiner de grade*. Isso é muito parecido com um bloco de con- taining no posicionamento, ou um contêiner flexível no layout de caixa flexível: um con- tainer de grade é um elemento que define um *contexto* de *formatação* de grade para seu conteúdo.

Neste nível muito básico, o layout da grade é realmente bastante reminiscente do flexbox. Para o exame, os elementos filho de um contentor de grelha tornam-se *itens de grelha*, tal como os elementos filhos de um contentor flexível tornam-se elementos flexíveis. Os filhos desses elementos filho *não* se tornam elementos de grade – embora qualquer item de grade possa ser feito um con-tainer de grade e, portanto, ter seus elementos filho se tornando itens de grade para a grade aninhada. É possível aninhar grades dentro de grades, até que sejam grades até o fim. (O layout de grade também tem um conceito separado de *subgrades* que é distinto dos contêineres de grade de aninhamento, mas chegaremos a isso mais tarde.)

Existem dois tipos de grades: grades *regulares* e grades *embutidas*. Eles são criados com valores especiais para a propriedade de exibição: grade e inline-grid. O primeiro gera uma caixa de nível de bloco e o segundo uma caixa de nível embutido. A diferença é ilustrada na Figura 1.

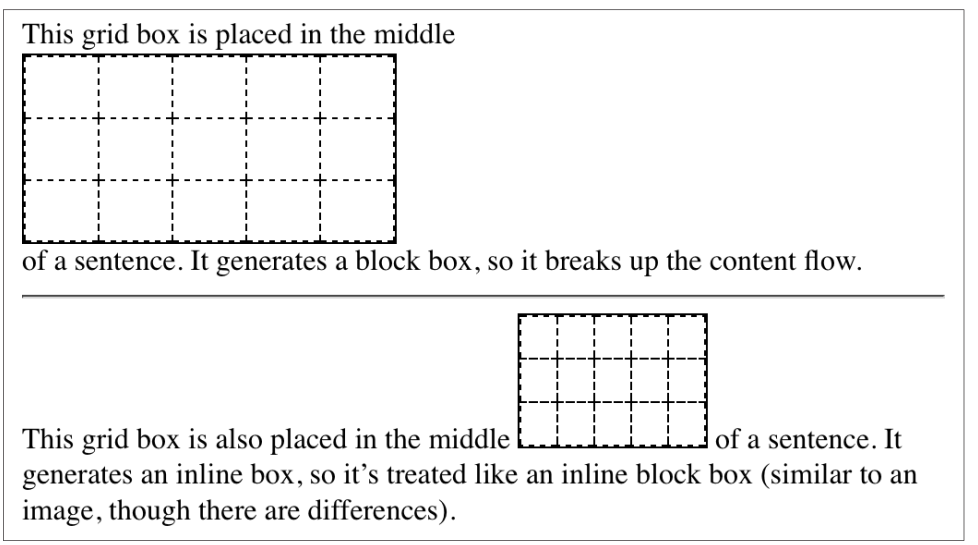


Figura 1. Grades e grades embutidas

Estes são muito semelhantes aos valores de bloco e bloco embutido para exibição. A maioria das grades que você cria provavelmente será de nível de bloco, embora, é claro, a capacidade de criar grades embutidas esteja sempre lá.

Embora display: grid crie uma grade em nível de bloco, a especificação tem o cuidado de declarar explicitamente que "contêineres de grade não são contêineres de bloco". O que isto significa é que embora a caixa de grade participe do layout da mesma forma que um contêiner de blocos, há várias diferenças entre elas.

Em primeiro lugar, os elementos flutuantes não se intrometem no contêiner de

grade. O que isso significa na prática é que uma grade não "deslizará embaixo" de um elemento flutuante, como um contêiner de bloco fará. Consulte a Figura 2 para uma demonstração da diferença.

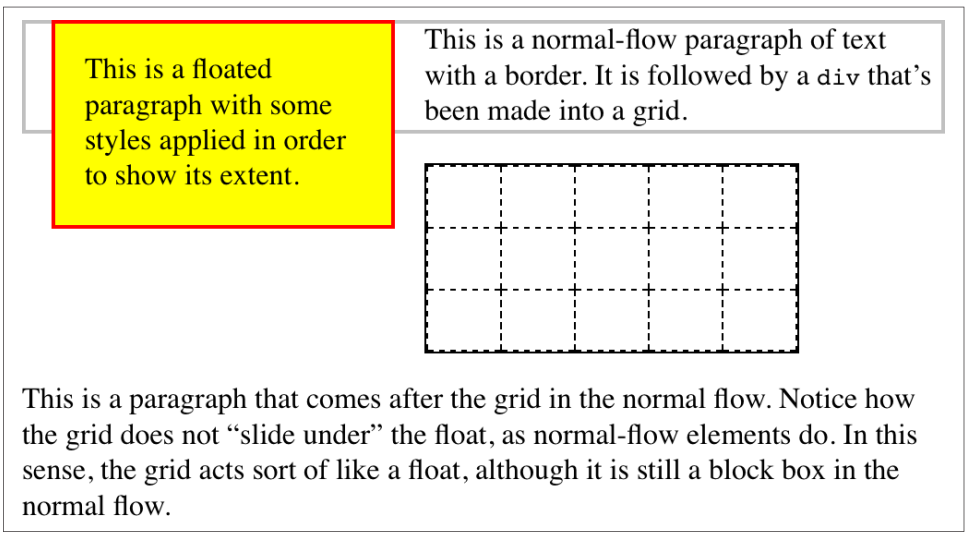


Figura 2. Os flutuadores interagem de forma diferente com blocos e grades

Além disso, as margens de um contêiner de grade não colapsam com as margens de seus descendentes. Novamente, isso é distinto das caixas de bloco, cujas margens (por padrão) colapsam com descendentes. Por exemplo, o primeiro item de lista em uma lista ordenada pode ter uma margem superior , mas essa margem será recolhida com a margem superior do elemento de lista. A margem superior de um item de grade *nunca* entrará em colapso com a margem superior de seu contêiner de grade. A Figura 3 ilustra a diferença.

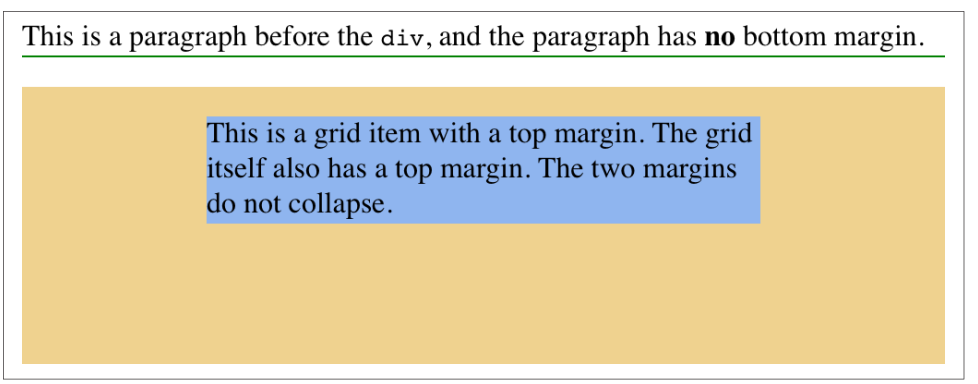


Figura 3. Colapso da margem e a falta dela

Existem algumas propriedades e recursos CSS que não se aplicam a contêineres de grade e itens de grade; especificamente:

- Todas as propriedades da coluna (por exemplo, contagem de colunas, colunas, etc.) são ignorados quando aplicados a um contêiner de grade.
- Os pseudoelementos ::first-line e ::first-letter não se aplicam a contêineres de grade e são ignorados.
- flutu e claro são efetivamente ignorados para itens de grade (embora ar não contenham grade-
  calculado da propriedade de exibição para filhos de um contêiner de grade, porque o valor de exibição dos itens de grade é resolvido *antes* de serem transformados em itens de grade.
- O alinhamento propriedade | não tem efeito sobre itens de grade, vertical | embora possa afetar
  o conteúdo dentro fazer item de grau. (Existem outras maneiras |

Por fim, se o valor de exibição declarado de um contêiner de grade for inline-grid *e* o elemento estiver flutuado ou absolutamente posicionado, o valor calculado de exibição se tornará grade (descartando assim a grade inline).

Depois de definir um contêiner de grade, a próxima etapa é configurar a grade dentro. Antes de explorarmos como isso funciona, no entanto, é necessário cobrir alguma terminologia.

Terminologia básica da grade

Já falamos sobre contêineres de grade e itens de grade, mas vamos defini-los com um pouco mais de detalhes. Como foi dito antes, um contêiner de grade é uma caixa que estabelece um *contexto de formatação* de grade; isto é, uma área na qual uma grade é criada e os elementos são dispostos *de* acordo com as regras do layout da grade em vez do layout do bloco. Você pode pensar nisso da maneira como um elemento definido para exibição: tabela cria um contexto de formatação de tabela dentro dela. Dada a natureza de grade das tabelas, essa comparação é bastante adequada, embora não se certifique de não supor que as grades são apenas tabelas de outra forma. As grades são muito mais poderosas do que as mesas já foram.

Um *item* de grade é uma coisa que participa do layout de grade dentro de um contexto de formatação de grade. Isso geralmente é um elemento filho de um contêiner de grade, mas também pode ser os bits anônimos (ou seja, não contidos em um elemento) de texto que fazem parte do conteúdo de um elemento. Considere o seguinte, que tem o resultado mostrado na Figura 4:

```
#warning {display: grade;
    antecedentes: #FCC; preenchimento:
    0.5em; grid-template-rows: 1fr;
```

```
    grid-template-columns: repeat(7, 1 fr);}
<p id="warning"><img src="warning.svg"><strong>Nota:</strong> Este elemento é um
    <em> grade</em> com vários itens <em>grade</em> dentro dele.</p>
```

Note: This element is a *grid container* with several *grid items* inside it.

Figura 4. Itens de grade

Observe como cada elemento, *e* cada bit de texto entre eles, tornou-se um item de grade. A imagem é um item de grade, tanto quanto os elementos e o texto são executados — sete itens de grade ao todo. Cada um deles participará do layout de grade, embora as execuções de texto anônimo sejam muito mais difíceis (ou impossíveis) de afetar com as várias propriedades de grade que discutiremos.

Se você está se perguntando sobre linhas de modelo de grade e colunas de modelo de grade, abordaremos elas na próxima seção.

No decorrer do uso dessas propriedades, você criará ou fará referência a vários componentes principais do layout da grade. Estes estão resumidos na Figura 5.

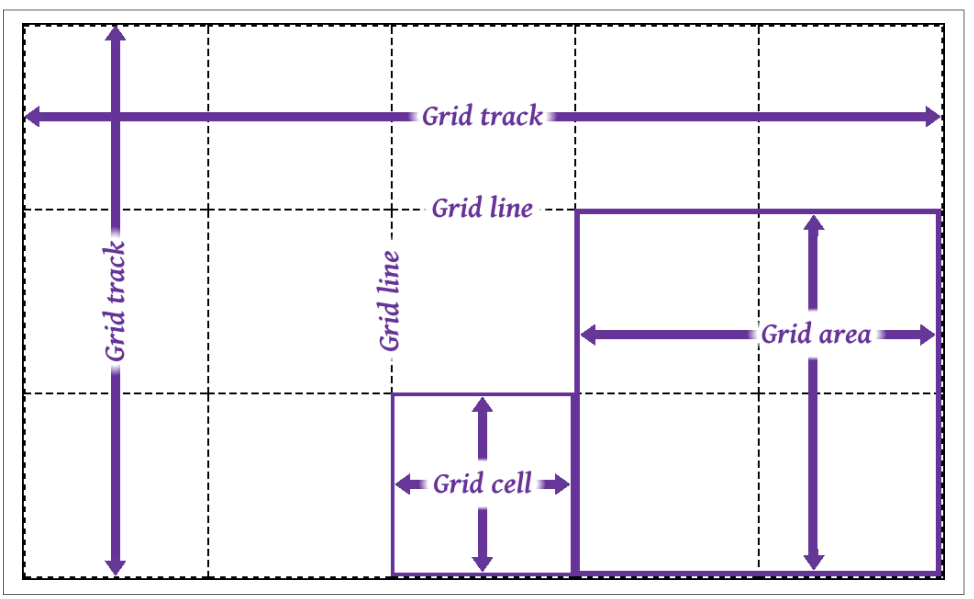


Figura 5. Componentes da grade

A unidade mais fundamental é a *linha de grade*. Ao definir o posicionamento de uma ou mais linhas de grade , você cria implicitamente o restante dos componentes da grade:

1. Uma *trilha* de grade é uma corrida contínua entre duas linhas de grade adjacentes — em outras palavras, uma coluna de grade ou uma *linha* de grade . Ele vai de uma borda do contêiner de grade para a outra. O tamanho de uma faixa de grade depende do posicionamento das linhas de grade que a definem. Estes são análogos às colunas e linhas da tabela. Mais genericamente, estes podem ser referidos como faixas de eixo de bloco e eixo em linha, onde (em lan- guages ocidentais) faixas de coluna estão no eixo do bloco e *faixas* de linha estão no eixo em linha.

1. Uma célula de grade é qualquer espaço delimitado por quatro linhas de grade, sem linhas de grade passando por ela, análogas a uma célula de tabela. Esta é a menor unidade de área no layout da grade. As células de grade não podem ser endereçadas diretamente com propriedades de grade CSS; ou seja, nenhuma propriedade permite que você diga que um item de grade deve ser associado a uma determinada célula. (Mas veja o próximo ponto para mais detalhes.)

1. Uma área de grade é qualquer área retangular delimitada por quatro linhas de grade e composta de uma ou mais células de grade. Uma área pode ser tão pequena quanto uma única célula, ou tão grande quanto todas as células na grade. As áreas de grade *são* diretamente endereçáveis pelas propriedades de grade CSS, que permitem definir as áreas e, em seguida, associar itens de grade a elas.

Uma coisa importante a notar é que essas faixas de grade, células e áreas são inteiramente estruturadas de linhas de grade e, mais importante, não precisam corresponder a itens de grade . Não há exigência de que todas as áreas da grade sejam preenchidas com um item; é perfeitamente possível que algumas ou mesmo a maioria das células de uma grade estejam vazias de qualquer conteúdo. Também é possível que os itens de grade se sobreponham uns aos outros, definindo áreas de grade sobrepostas ou usando referências de linha de grade que criam situações de sobreposição.

Outra coisa a ter em mente é que você pode definir quantas ou quantas linhas de grade desejar. Você poderia literalmente definir apenas um conjunto de linhas de grade verticais, criando assim um monte de colunas e apenas uma linha. Ou você poderia ir para o outro lado, criando um monte de faixas de linha e nenhuma faixa de coluna (embora houvesse uma, estendendo-se de um lado do contêiner de grade para o outro).

O outro lado disso é se você criar uma condição em que um item de grade não possa ser colocado dentro das faixas de coluna e linha que você definir, ou se você colocar explicitamente um item de grade for a dessas faixas, novas linhas de grade e faixas serão adicionadas automaticamente à grade para acomodar.

Colocando linhas de grade

Acontece que a colocação de linhas de grade pode ficar bastante complexa. Isso não é tanto porque o conceito é difícil; existem tantas maneiras diferentes de fazê-lo, e cada uma usa sua própria sintaxe sutilmente diferente.

Começaremos examinando duas propriedades intimamente relacionadas.

linhas de modelo de grade, grid-template-

Valores	nenhum | < > de faixas de | <lista de faixas automáticas> | sub-grade
Valor inicial:	nenh
Aplica-se	Contêineres de
Herdada:	Nã
Percentagens:	Consulte o tamanho | embutido | (geralmente largura) do contêiner de grade para colunas de modelo de grade | e o tamanho | fazer bloco | (geralmente altura) fazer contêiner de grade para
Valor calculado:	Conforme declarado, com comprimentos tornados

Com essas propriedades, você pode definir as linhas de grade em seu *modelo* de grade geral ou o que a especificação CSS chama de *grade explícita*. Tudo depende dessas linhas de grade; não consegue colocá-los corretamente, e todo o layout pode muito facilmente desmoronar.

Quando você está começando com o layout de grade CSS, provavelmente é uma boa ideia esboçar onde as linhas de grade precisam estar no papel primeiro ou em algum análogo digital próximo. Ter uma referência visual de onde as linhas devem estar e como elas devem se comportar tornará a escrita de sua grade CSS muito mais fácil.

Há muitas maneiras de especificar o posicionamento de suas linhas de grade, portanto, antes de começarmos a aprender esses padrões, há algumas coisas básicas a serem estabelecidas.

Primeiro, as linhas de grade sempre podem ser referidas por número e também podem ser nomeadas pelo autor. Pegue a grade mostrada na Figura 6, por exemplo. A partir do seu CSS, você pode usar qualquer um dos números para se referir a uma

linha de grade, ou você pode usar os nomes definidos, ou você pode misturá-los juntos. Assim, você poderia dizer que um item de grade se estende da linha de coluna 3 à linha steve, e da claraboia da linha à linha 2.

Observe que uma linha de grade pode ter mais de um nome. Você pode usar qualquer um deles para se referir a uma determinada linha de grade, embora não possa combiná-los da mesma forma que vários nomes de classe. Você pode pensar que isso significa que é uma boa ideia evitar repetir nomes de linhas de grade, mas isso nem sempre é o caso, como veremos em breve.

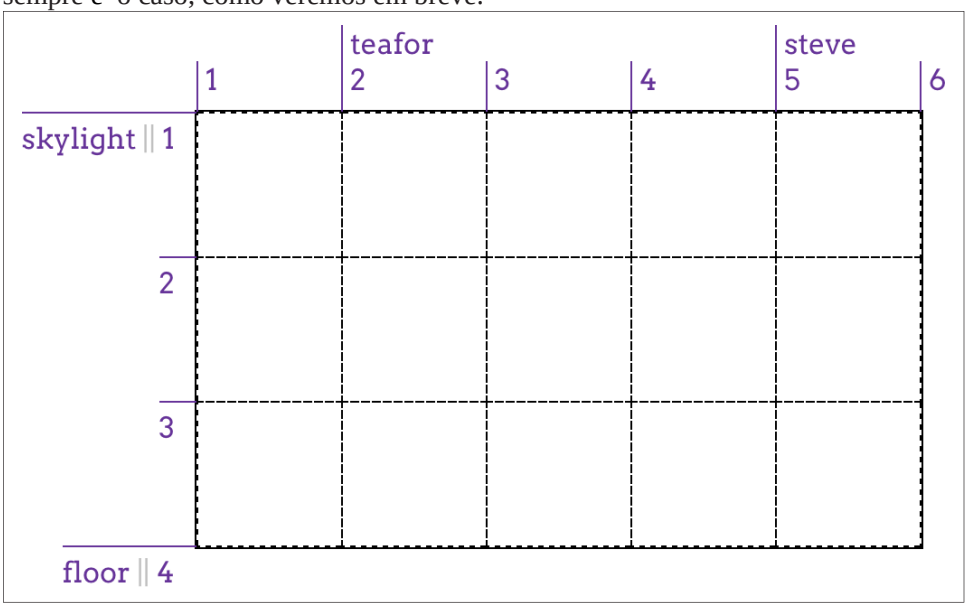


Figura 6. Números e nomes de linhas de grade

Usei nomes de linha de grade intencionalmente bobos na Figura 6 para ilustrar que você pode escolher qualquer nome que quiser e também para evitar a implicação de que existem nomes "padrão". Se você tivesse visto o início da primeira linha, talvez tivesse assumido que a primeira linha é sempre chamada assim. Não. Se você quiser esticar um elemento do início ao fim, precisará definir esses nomes você mesmo. Felizmente, isso é simples de fazer.

Como eu disse, muitos padrões de valor podem ser usados para definir o modelo de grade. Começaremos com os mais simples e trabalharemos em direção aos mais complexos.

Faixas de grade de largura fixa

Nosso primeiro passo é criar uma grade cujas faixas de grade sejam de largura fixa. Não precisamos de um comprimento fixo como pixels ou ems; as porcentagens também contam como largura fixa aqui. Neste contexto, entende-se por "largura fixa" que as linhas de grelha são colocadas de modo a que a distância entre elas não se

altere devido a alterações de conteúdo.

Então, como exemplo, isso conta como uma definição de três colunas de grade de largura fixa:

```
#grid {display: grade;
    grid-template-columns: 200px 50% 100px;}
```

Isso colocará uma linha de 200 pixels a partir do início do contêiner de grade (por padrão, o lado esquerdo); uma segunda linha de grade com metade da largura do contêiner de grade longe do primeiro; e uma terceira linha a 100 pixels de distância da segunda. Isso é ilustrado na Figura 7.

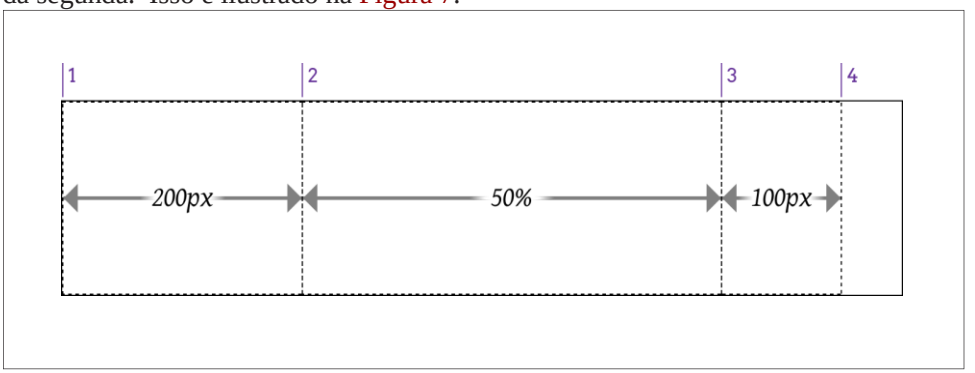


Figura 7. Posicionamento da linha de grade

Embora seja verdade que a segunda coluna pode mudar de tamanho se o tamanho do contêiner de grade for alterado, ela *não* será alterada com base no conteúdo dos itens de grade. Por maior ou menor que seja o conteúdo colocado nessa segunda coluna, a largura da coluna sempre será metade da largura do contêiner de grade.

Também é verdade que a última linha de grade não atinge a borda direita do contêiner de grade. Tudo bem, não precisa. Se você quiser – e provavelmente o fará – veremos várias maneiras de lidar com isso daqui a pouco.

Isso tudo é adorável, é claro, mas e se você quiser nomear suas linhas de grade? Basta colocar qualquer nome de linha de grade desejado, e quantos desejar, no local apropriado no valor, cercado por colchetes. Realmente é tão simples quanto isso! Vamos adicionar alguns nomes ao nosso exemplo anterior, com o resultado mostrado na Figura 8:

```
#grid {display: grade;
    grid-template-columns:
        [iniciar col-a] 200px [col-b] 50% [col-c] 100px [stop end last];
```

}

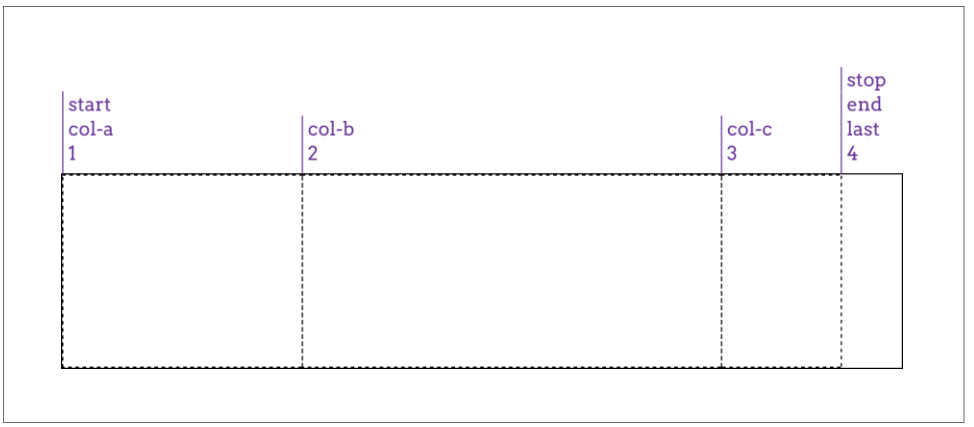


Figura 8. Posicionamento do nome da grade

O que é bom é que adicionar os nomes deixa claro que cada valor está realmente especificando a largura de uma faixa de grade , o que significa que sempre há uma linha de grade para cada lado de um valor de largura. Assim, para as três larguras que temos, existem na verdade quatro linhas de grade criadas.

As linhas de grade de linha são colocadas exatamente da mesma maneira que as colunas, como mostra a Figura 9.

```
#grid {display: grade;
    grid-template-columns:
        [iniciar col-a] 200px [col-b] 50% [col-c] 100px [stop end last]; linhas
    de modelo de grade:
        [iniciar masthead] 3em [conteúdo] 80% [rodapé] 2em [fim de parada];
    }
```

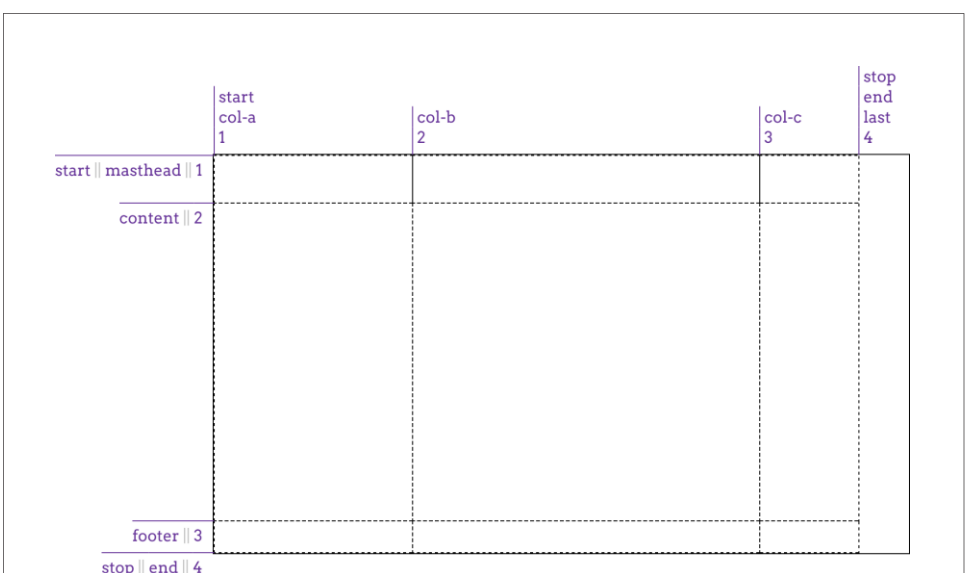


Figura 9. Criando uma grade

Há algumas coisas a salientar aqui. Primeiro, há linhas de coluna e linha com os nomes início e fim. Tudo bem. Grades e colunas não compartilham o mesmo namespace, portanto, você pode reutilizar nomes como esses nos dois contextos.

O segundo é o valor percentual da faixa de linha de conteúdo. Isso é calculado com relação à altura do contêiner de grade; assim, um contêiner de 500 pixels de altura produziria uma linha de conteúdo com 400 pixels de altura. Isso obviamente requer que você saiba de antemão o quão alto o contêiner de grade será, o que nem sempre será o caso.

Você pode pensar que poderíamos apenas dizer 100% e fazer com que ele preenchesse o espaço, mas isso não funciona, como ilustra a Figura 10: a faixa da linha de conteúdo será tão alta quanto o próprio con-tainer da grade, empurrando assim a linha do rodapé. rastrear para for a do recipiente completamente.

```
#grid {display: grade;
    grid-template-columns:
        [iniciar col-a] 200px [col-b] 50% [col-c] 100px [stop end last]; linhas
    de modelo de grade:
        [iniciar masthead] 3em [conteúdo] 100% [rodapé] 2em [fim de parada];
    }
```

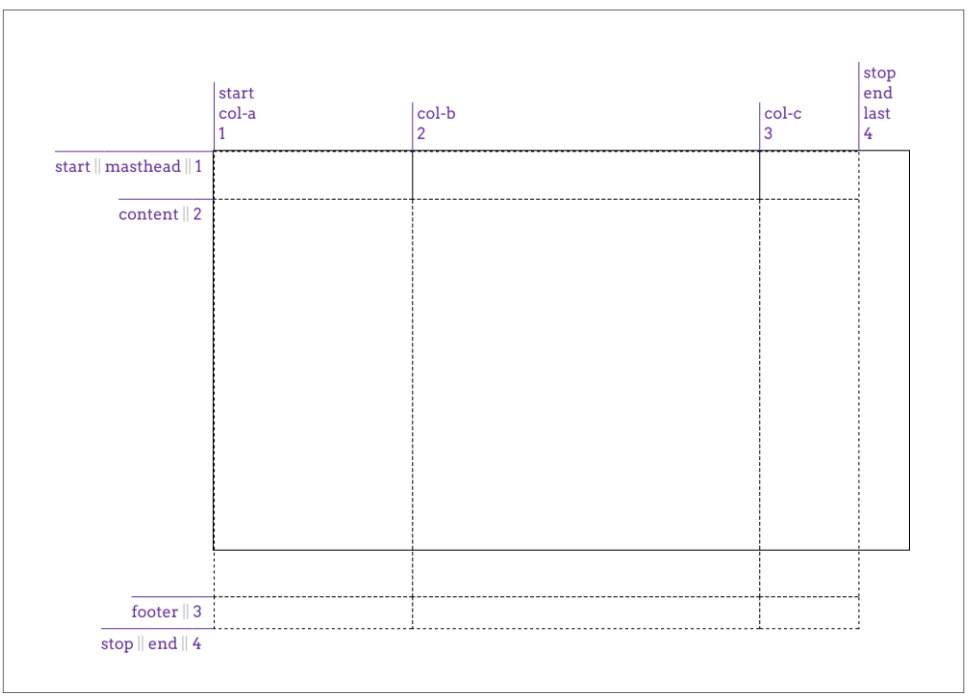


Figura 10. Excedendo o contêiner de grade

Uma maneira (não necessariamente a melhor maneira) de lidar com esse cenário é

minmax o valor da linha, dizendo ao navegador que você deseja que a linha não seja menor que uma quantidade e não mais alta que outra, deixando o navegador preencher o valor exato. Isso é feito com o padrão minmax(a,b), onde a é o tamanho mínimo e b é o tamanho máximo:

```
#grid {display: grade;
    grid-template-columns:
        [iniciar col-a] 200px [col-b] 50% [col-c] 100px [stop end last]; linhas
    de modelo de grade:
        [iniciar masthead] 3em [conteúdo] minmax(3em,100%) [rodapé] 2em [fim de parada];
    }
```

O que dissemos lá é para fazer com que a linha de conteúdo nunca tenha menos de 3 ems de altura e nunca seja mais alta do que o próprio contêiner de grade. Isso permite que o navegador aumente o tamanho até que seja alto o suficiente para caber o espaço que sobrou das trilhas do mastro e do rodapé, e nada mais. Claro, ele também permite que o navegador o torne mais curto do que isso, pois desde que não seja menor que 3em, então este não é um resultado garantido. A Figura 11 mostra um possível resultado dessa abordagem.

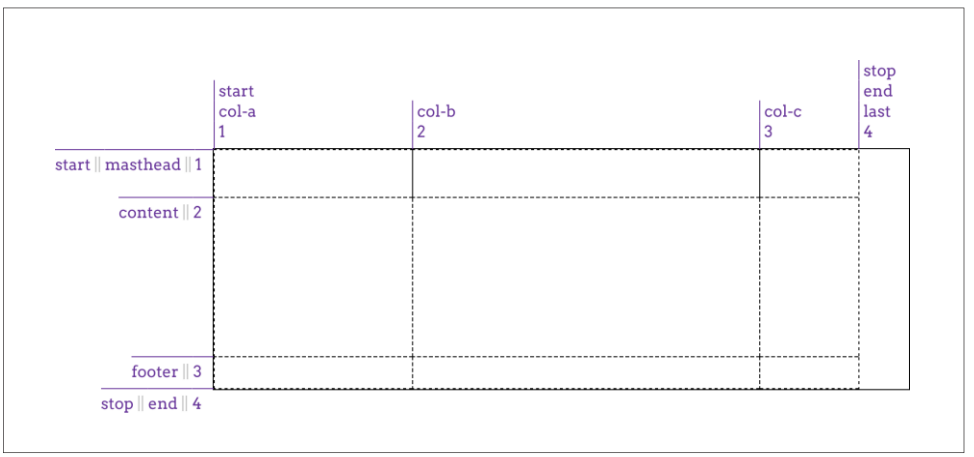


Figura 11. Adaptação ao contentor de grelha

Da mesma forma, com as mesmas ressalvas, minmax() poderia ter sido usado para ajudar a coluna col-b a preencher o espaço em todo o contêiner de grade. A coisa a lembrar com minmax() é que, se o máximo for menor que o *min*, o valor *máximo* será descartado e o valor *mínimo* será usado como um comprimento de faixa de largura fixa. Assim, minmax(100px, 2em) resolveria para 100px para qualquer valor de tamanho de fonte menor que 50px.

Se a imprecisão do comportamento do minmax() o incomoda, existem alternativas para esse cenário. Também poderíamos ter usado o padrão de valor calc() para chegar à altura (ou largura) de uma faixa. Por exemplo:

```
linhas de modelo de grade:
    [iniciar masthead] 3em [conteúdo] calc(100%-5em) [rodapé] 2em [fim de parada];
```

Isso produziria uma linha de conteúdo exatamente tão alta quanto o contêiner de grade menos a soma das alturas do mastro e do rodapé, como vimos na figura anterior.

Isso funciona até onde vai, mas é uma solução um pouco frágil, já que quaisquer alterações na altura do mastro ou do rodapé também exigirão um ajuste do cálculo. Também se torna muito mais difícil (ou impossível) se você quiser que mais de uma coluna flexione dessa maneira. Acontece que existem maneiras muito mais robustas de lidar com esse tipo de situação, como veremos em breve.

Faixas de grade flexíveis

Até agora, todas as nossas faixas de grade têm sido *inflexíveis* – seu tamanho determinado por uma medida de comprimento ou as dimensões do contêiner de grade, mas não afetadas por quaisquer outras considerações. Faixas de grade *flexíveis*, por outro lado, podem ser baseadas na quantidade de espaço no grid container não consumido por faixas inflexíveis, ou alternativamente, pode ser baseado no conteúdo real de toda a faixa.

Unidades fracionárias

Se você quiser dividir qualquer espaço disponível por alguma fração e distribuir as frações para várias colunas, a unidade fr está aqui para você.

No caso mais simples, você pode dividir todo o contêiner por frações iguais. Por exemplo, se você quiser quatro colunas, você pode dizer

```
grid-template-columns: 1 fr 1 fr 1 fr 1 fr;
```

Neste caso muito limitado, isso equivale a dizer

```
grid-template-columns: 25% 25% 25% 25%;
```

O resultado de qualquer maneira é mostrado na Figura 12.

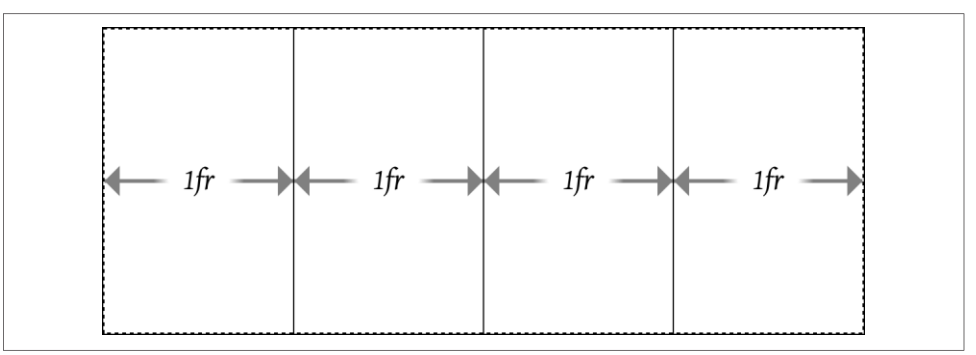


Figura 12. Dividindo o contêiner em quatro colunas

Agora suponha que queremos adicionar uma quinta coluna e redistribuir o tamanho da coluna para que todos ainda sejam iguais. Com porcentagens, teríamos que reescrever

todo o valor para cinco instâncias de 20%. Com fr, porém, podemos simplesmente adicionar mais 1fr ao valor e ter tudo feito para nós automaticamente:

```
grid-template-columns: 1 fr 1 fr 1 fr 1 fr 1 fr;
```

A maneira como as unidades fr funcionam é que todos os valores fr são somados, com o espaço disponível dividido por esse total. Em seguida, cada faixa recebe o número dessas frações indicadas por seu número.

O que isso significou para o primeiro dos exemplos anteriores é que, quando havia quatro valores fr, seus números eram somados para obter um total de quatro. O espaço disponível foi, portanto, dividido por quatro, e cada coluna recebeu um desses quartos. Quando adicionamos um quinto 1fr, o espaço foi dividido por cinco, e cada coluna recebeu um desses quintos.

Você não é obrigado a usar sempre 1 com suas unidades fr! Suponha que você queira dividir um espaço de tal forma que haja três colunas, com a coluna do meio duas vezes mais larga que as outras duas. Isso ficaria assim:

```
grid-template-columns: 1 fr 2 fr 1 fr;
```

Novamente, estes são somados e, em seguida, 1 é dividido por esse total, de modo que o fr base neste caso é 0,25. A primeira e a terceira faixas são, portanto, 25% da largura do contêiner, enquanto a coluna do meio é metade da largura do contêiner, porque é 2fr, que é duas vezes 0,25, ou 0,5.

Você também não está limitado a inteiros. Um cartão de receita para torta de maçã pode ser disposto usando estas colunas:

```
grid-template-columns: 1 fr 3.14159 fr 1fr;
```

Vou deixar a matemática sobre isso como um exercício para o leitor. (Sorte sua! Basta lembrar para começar com 1 + 3,14159 + 1, e você terá uma boa vantagem.)

Esta é uma maneira conveniente de fatiar um contêiner, obviamente, mas há mais aqui do que apenas substituir porcentagens por algo mais intuitivo. As unidades fracionárias realmente se destacam quando há algumas colunas fixas e algum espaço flexível. Considere, por exemplo, o seguinte, que é ilustrado na Figura 13:

```
grid-template-columns: 15em 1fr 10%;
```

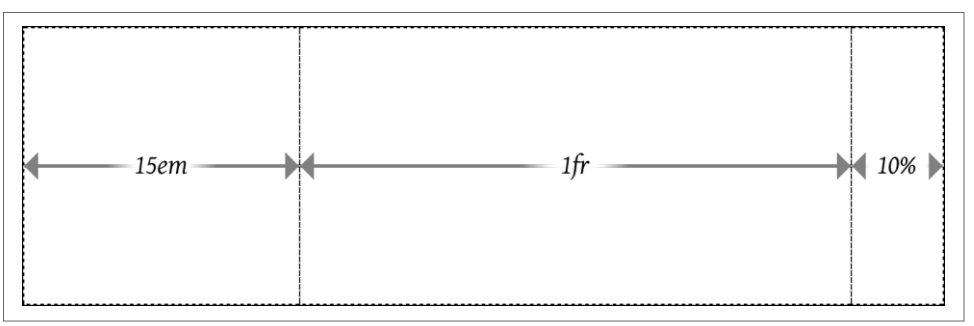


Figura 13. Dando à coluna central o que estiver disponível

O que aconteceu lá é que o navegador atribuiu a primeira e a terceira faixas às suas larguras inflexíveis e, em seguida, deu o que restou no contêiner de grade para a faixa central. Isso significa que, para um contêiner de grade de 1.000 pixels de largura cujo tamanho da fonte é o padrão usual do navegador de 16px, a primeira coluna terá 240 pixels de largura e a terceira terá 100 pixels de largura . Isso totaliza 340 pixels, deixando 660 pixels não atribuídos a faixas inflexíveis. As unidades fracionárias totalizam um, então 660 é dividido por um, produzindo 660 pixels, todos os quais são dados à única faixa de 1fr. Se a largura do contêiner de grade for aumentada para 1.400 pixels, a terceira coluna terá 140 pixels de largura e a coluna central 1.020 pixels de largura.

Assim , temos uma mistura de colunas fixas e flexíveis. Podemos manter isso, dividindo qualquer espaço flexível em quantas frações quisermos. Considere o seguinte:

```
largura: 100em; grid-template-columns: 15em 4. 5 fr 3fr 10%;
```

Nesse caso, as colunas serão dimensionadas conforme mostrado na Figura 14.

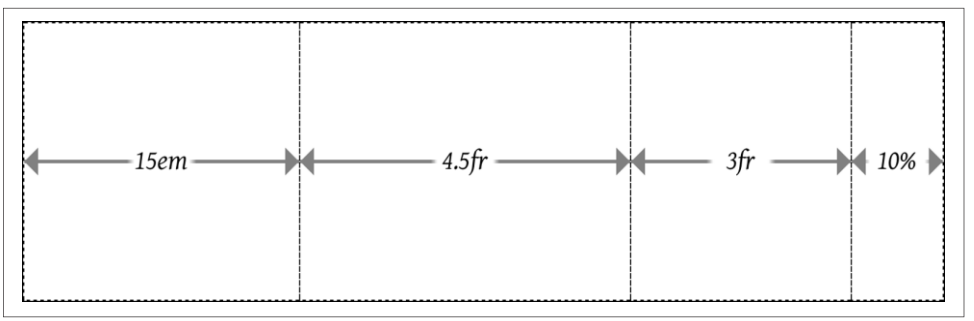


Figura 14. Dimensionamento flexível da coluna

Sim, reconhecidamente, coloquei um polegar nas escalas da Figura 14: o valor total e de largura fr foram projetados para produzir números bons e redondos para as várias colunas. Isso é puramente para ajudar a compreensão, é claro. Se você quiser trabalhar no processo com números menos organizados, considere o uso de 92,5em ou 1234px para o valor de largura no exemplo anterior.

Nos casos em que você deseja definir um tamanho mínimo ou máximo para uma determinada faixa, minxmax() pode ser bastante útil em algumas situações. Para estender o exemplo anterior, suponha que a coluna third nunca deve ter menos de 5em de largura, não importa o quê. O CSS seria então

```
grid-template-columns: 15em 4. 5fr minmax (5em, 3fr) 10%;
```

Agora, o layout terá duas colunas flexíveis em seu meio, até o ponto em que a terceira coluna atinge 5em de largura. Abaixo desse ponto, o layout terá três colunas inflexíveis (15em, 5em e 10% de largura, respectivamente) e uma única coluna flexível que obterá todo o espaço restante, se houver algum. Depois de executar a matemática, verifica-se que até 30,5556em de largura, a grade terá uma coluna

flexível. Acima dessa largura, haverá duas dessas colunas.

Você pode pensar que isso funciona de outra maneira — por exemplo, se você quisesse tornar uma faixa de coluna flexível até um certo ponto e, em seguida, ficar fixo depois, você declararia um valor fr mínimo . Isso não funcionará, infelizmente, porque as unidades fr não são todasdevidas na posição *min* de uma expressão minmax(). Portanto, qualquer valor fr fornecido como mínimo invalidará a declaração. (Em implementações anteriores, um mínimo fr foi definido como zero. Em implentações futuras, os valores fr provavelmente poderão ser tamanhos mínimos.)

Falando em definir como zero, vejamos uma situação em que o valor mínimo é explicitamente definido como 0, como esta:

```
grid-template-columns: 15em 1fr minmax (0,500px) 10%;
```

A Figura 15 ilustra a largura de grade mais estreita na qual a terceira coluna pode permanecer com 500 pixels de largura. Qualquer mais estreita, e a coluna minmaxed será mais estreita do que 500 pixels. Qualquer mais larga, e a segunda coluna, a coluna fr, crescerá além da largura zero, enquanto a terceira coluna permanecerá com 500 pixels de largura.

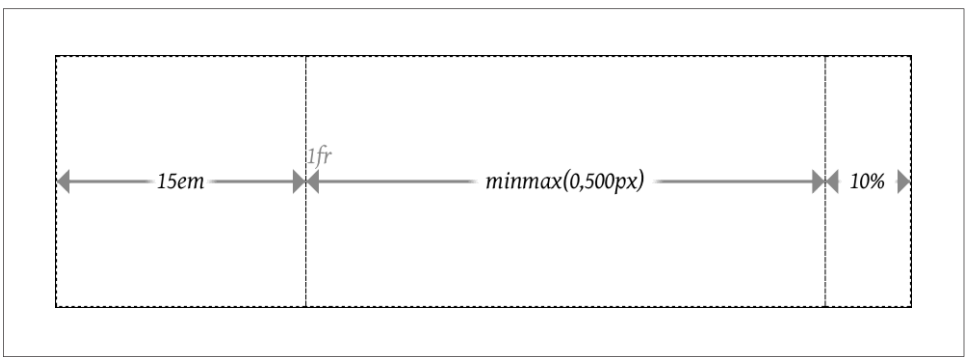


Figura 15. Dimensionamento mínimo da coluna

Se você olhar de perto, verá o rótulo 1fr ao lado do limite entre as colunas 15em e minmax (0,500px). Isso está lá porque o 1fr é colocado com sua borda esquerda na linha da grade da segunda coluna e não tem largura, porque não há espaço para flexionar. Da mesma forma, o minmax é colocado na linha de grade da terceira coluna. É que, nesta situação específica, as linhas de grade da segunda e terceira coluna estão no mesmo lugar (e é por isso que a coluna 1fr tem largura zero).

Se você já se deparar com um caso em que o valor mínimo é maior que o valor máximo, então a coisa toda é substituída pelo valor mínimoimum. Assim, minmax (500px, 200px) seria tratado como um simples 500px. Você provavelmente não faria isso tão obviamente, mas esse recurso é útil ao misturar coisas como porcentagens e frações. Assim, você poderia ter uma coluna que é minmax (10%, 1fr) que seria flexível até o ponto em que a coluna flexível fosse inferior a 10% da largura do container da grade , momento em que ela ficaria em 10%.

Unidades fracionárias e minmaxes são utilizáveis em linhas tão facilmente quanto colunas; é just que as linhas raramente são dimensionadas dessa maneira. Você poderia facilmente imaginar a configuração de um layout onde o mastro e o rodapé são faixas fixas, enquanto o conteúdo é flexível até um certo ponto. Isso pode ser algo parecido com isto:

```
grid-template-rows: 3em minmax (5em,1 fr) 2em;
```

Isso funciona bem, mas é muito mais provável que você queira dimensionar essa linha pela altura de seu conteúdo, não por uma fração da altura do contêiner de grade. A próxima seção mostra exatamente como fazer isso acontecer.

Faixas com reconhecimento de conteúdo

Uma coisa é configurar faixas de grade que ocupam frações do espaço disponível para eles, ou que ocupam quantidades fixas de espaço. Mas e se você quiser alinhar um monte de pedaços de uma página e não puder garantir o quão largos ou altos eles podem ficar? É aqui que entram o conteúdo mínimo e o conteúdo máximo.

O que essas palavras-chave significam é simples de afirmar, mas não necessariamente simples de descrever na íntegra. max-content significa, com efeito, "ocupar a quantidade máxima de espaço necessária para esse conteúdo". Para grandes blocos de texto (como uma postagem de blog), isso geralmente significaria ocupar o máximo de espaço disponível, a fim de maximizar o espaço para esse conteúdo.

min-content, por outro lado, significa "ocupar o espaço mínimo necessário para esse conteúdo". Com o texto, isso significa espremer a largura até o ponto em que a palavra mais longa (ou o elemento embutido mais largo, se houver coisas como imagens ou entradas de formulário) fica em uma linha por si só. Isso levaria a muitas quebras de linha em um elemento de grade muito magro e muito alto.

O que é tão poderoso sobre essas palavras-chave de dimensionamento é que elas se aplicam a toda a faixa de grade que elas definem. Por exemplo, se você dimensionar uma coluna para ser conteúdo máximo, a faixa de coluna inteira será tão ampla quanto o conteúdo mais amplo dentro dela. Isso é mais fácil de ilustrar com uma grade de imagens (12 neste caso) com a grade declarada da seguinte forma e mostrada na Figura 16.

```
#galeria {display: grade;
    grid-template-columns: max-content max-content max-content max-
    content; grid-template-rows: max-content max-content max-content;}
```

Olhando para as colunas, podemos ver que cada faixa de coluna é tão larga quanto a imagem mais larga dentro dessa faixa. Onde um monte de imagens de retrato se alinharam, a coluna é mais estreita; onde uma imagem de paisagem apareceu, a coluna foi feita larga o suficiente para cabê-la. A mesma coisa aconteceu com as linhas. Cada linha é tão alta quanto a imagem mais alta dentro dela, portanto, onde

quer que uma linha tenha todas as imagens curtas, a linha também é curta.

A vantagem aqui é que isso funciona para qualquer tipo de conteúdo, não importa o que esteja lá. Então, digamos que adicionamos legendas às fotos. Todas as colunas e linhas serão redimensionadas conforme necessário para lidar com texto e imagens, como mostra a Figura 17.

Obviamente, esse não é um design completo – as imagens estão fora do lugar e não há tentativa de restringir as larguras das legendas. Na verdade, isso é exatamente o que devemos esperar dos valores de conteúdo máximo para a largura da colunas. Como significa "tornar esta coluna larga o suficiente para conter todo o seu conteúdo", é o que temos.

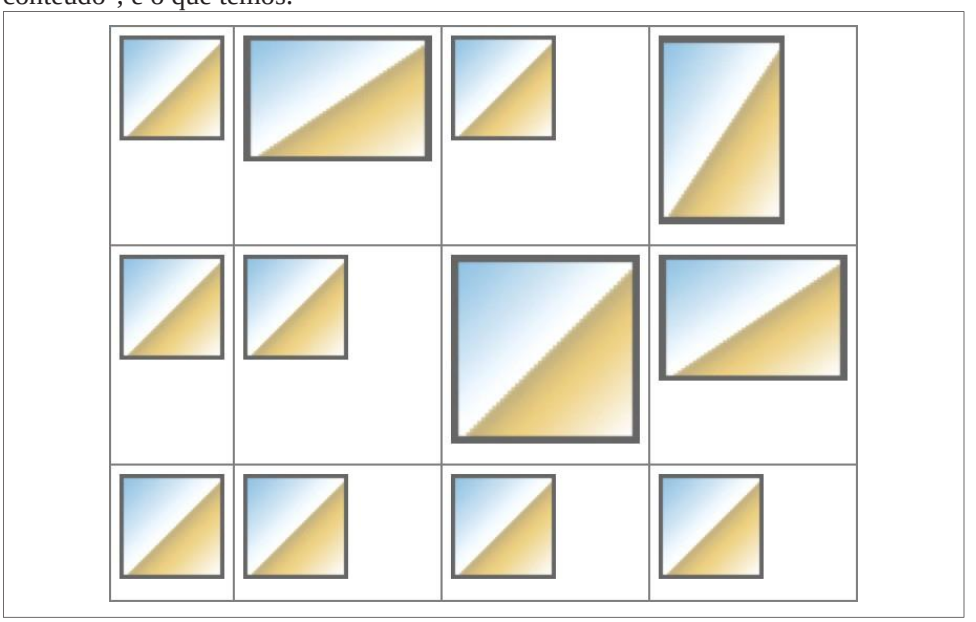


Figura 16. Dimensionando faixas de grade por conteúdo

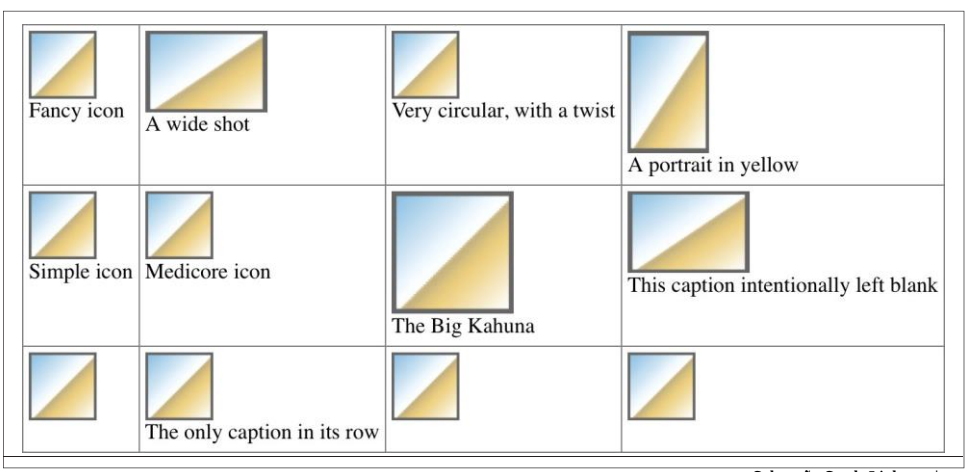


Figura 17. Dimensionando faixas de grade em torno de conteúdo misto

O que é importante perceber é que isso se manterá mesmo que as trilhas da grade tenham que se derramar para fora do contêiner da grade. Isso significa que, mesmo que tivéssemos atribuído algo como largura: 250px ao contêiner de grade, as imagens e legendas seriam dispostas da mesma forma. É por isso que coisas como max-content tendem a aparecer em instruções minmax(). Considere o seguinte, onde grades com e sem minmax() aparecem lado a lado. Em ambos os casos, o contêiner de grade é representado por um fundo laranja (consulte a Figura 18):

```
#g1 {display: grade;
     grid-template-columns: max-content max-content max-content    max-content;
     }
#g2 {display: grade;
     grid-template-columns: minmax (0, max-content) minmax (0, max-
          content) minmax (0, max-content)  minmax (0, max -conteúdo);
     }
```

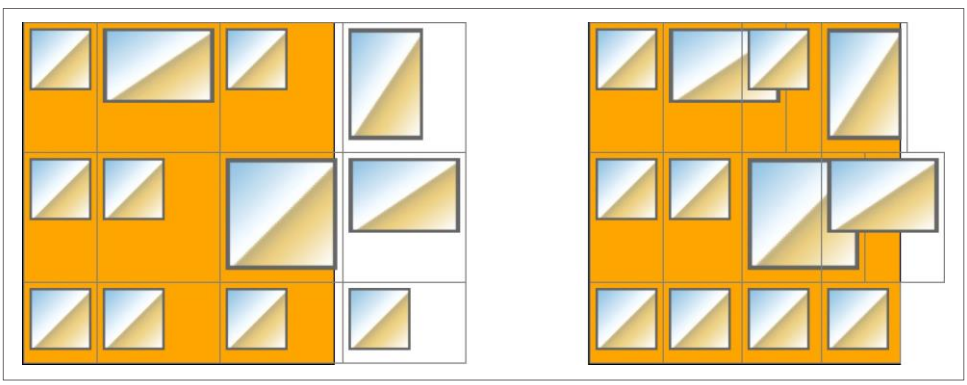


Figura 18. Dimensionamento de trilhos de grade com e sem minmax()

Em primeiro lugar, os itens de grade contêm completamente seu conteúdo, mas eles se espalham para fora do contêiner de grade. No segundo, o minmax() direciona o navegador para manter as colunas dentro do intervalo de 0 e max-content, para que todas elas sejam ajustadas ao contêiner de grade, se possível. Uma variante disso seria declarar minmax (min-content, max-content), o que leva a um resultado ligeiramente diferente da abordagem 0, max-content.

Se você está se perguntando o que acontece se você min-content tanto as colunas quanto as linhas,i t's é praticamente o mesmo que aplicar min-content às colunas e deixar as linhas sozinhas. Isso acontece porque a especificação de grade direciona os navegadores para resolver o dimensionamento da coluna primeiro e o dimensionamento da linha depois disso.

Há mais uma palavra-chave que você pode usar com o dimensionamento da faixa de grade, que é auto. No mínimo, ele é tratado como o tamanho mínimo para o item de grade, conforme definido por largura mínima ou altura mínima. Como máximo, é tratado da mesma forma que o conteúdo máximo. Você pode pensar que isso

podeser usado apenas em instruções minmax(), mas esse não é o caso. Você pode usá-lo em qualquer lugar, e ele assumirá uma função mínima ou máxima. Qual deles ele assume depende dos outros valores de faixa ao seu redor, de maneiras que são francamente complicadas demais para entrar aqui. Tal como acontece com tantos outros aspectos do CSS, o uso de auto é essencialmente deixar o navegador fazer o que quiser. Às vezes tudo bem, mas em geral você provavelmente vai querer evitá-lo.

Há uma ressalva para essa última afirmação: os valores automáticos permitem que os itens de grade sejam redimensionados pelos laços de alinhamento de conteúdo e de conteúdo de justificação , um tópico que discutiremos em uma seção posterior. Como os valores automáticos são os únicos valores de dimensionamento de faixa que permitem isso, pode haver boas razões para usar o auto, afinal.

Falando de coisas a evitar, você provavelmente já se perguntou sobre os valores repetitivos do modelo de grade e o que acontece se você precisar de mais de três ou quatro faixas de grade. Você terá que escrever cada largura de faixa individualmente? Na verdade, não, como veremos na próxima seção.

Repetindo linhas de grade

Se você tem uma situação em que deseja configurar um monte de linhas de grade, provavelmente não quer ter que digitar cada uma delas. Felizmente, repeat() está aqui para se certificar de que você não precisa.

Digamos que queremos configurar uma linha de grade de coluna a cada 5 ems e configurar 10 faixas de coluna. Veja como fazer isso:

```
#grid {display: grade;
       grid-template-columns: repeat(10, 5em);}
```

É isso. Terminado. Dez faixas de coluna, cada uma com 5em de largura, para um total de 50 ems de faixas de coluna. Com certeza bate digitando 5em 10 vezes!

Qualquer valor de dimensionamento de faixa pode ser usado em uma repetição, de min-content e max-content a valores fr a auto, e assim por diante, e você pode reunir mais de um valor de dimensionamento. Suponha que desejemos definir uma estrutura de coluna de tal forma que haja uma faixa 2em, depois uma faixa 1fr e, em seguida, outra faixa 1fr – e, além disso, queremos repetir esse padrão três vezes. Veja como fazer isso, com o resultado mostrado na Figura 19:

```
#grid {display: grade;
       grid-template-columns: repeat(3, 2em 1 fr 1 fr);}
```

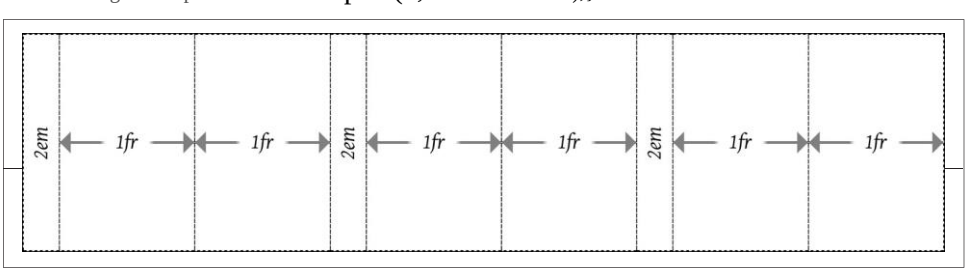


Figura 19. Repetindo um padrão de faixa

Observe como a última faixa de coluna é uma faixa de 1fr, enquanto a primeira faixa de coluna tem 2em de largura. Este é um efeito da forma como o repeat() foi escrito. É fácil adicionar outra faixa 2em no final, a fim de equilibrar as coisas, simplesmente fazendo esta alteração:

```
#grid {display: grade;
       grid-template-columns: repeat(3, 2em 1 fr 1 fr) 2em;}
```

Viu aquele extra de 2em no final do valor? Isso adiciona mais uma faixa de coluna após os três padrões repetidos. Isso destaca o fato de que a repetição pode ser combinada com quaisquer outros valores de dimensionamento de faixa – até mesmo outras repetições – na construção de uma grade. A única coisa que você *não pode* fazer é aninhar uma repetição dentro de outra repetição.

Fora isso, praticamente qualquer coisa vai dentro de um valor repeat(). Aqui está um exemplo retirado diretamente da especificação da grade:

```
#grade {
     exibição: grade;
     grid-template-columns: repeat(4, 10px [col-start] 250px [col-end]) 10px;}
```

Nesse caso, há quatro repetições de uma faixa de 10 pixels, uma linha de grade nomeada, uma faixa de 250 pixels e, em seguida, outra linha de grade nomeada. Então, após as quatro repetições, uma faixa final de coluna de 10 pixels. Sim, isso significa que haverá quatro linhas de grade de coluna nomeadas col-start e outras quatro chamadas col-end, como mostra a Figura 20. Isso é aceitável; os nomes de linhas de grade não precisam ser exclusivos. (Veremos como lidar com esse tipo de coisa em uma seção posterior.)

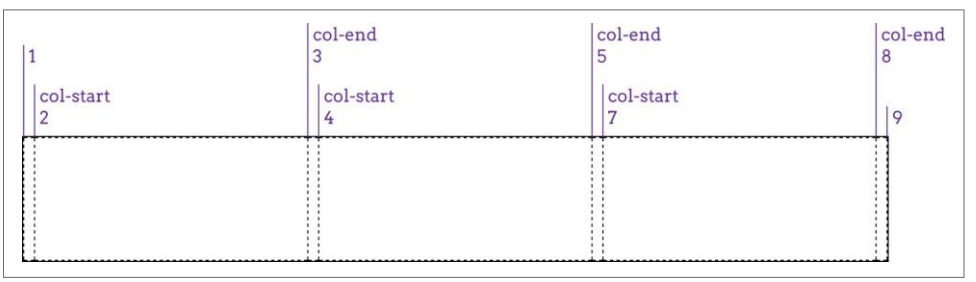


Figura 20. Colunas repetidas com linhas *de grade nomeadas*

Uma coisa a lembrar, se você for repetir linhas nomeadas, é que, se você colocar duas linhas nomeadas uma ao lado da outra, elas serão mescladas em uma única linha de grade de nome duplo. Em outras palavras, as duas declarações a seguir são equivalentes:

```
grid-template-rows: repeat(3, [top] 5em [bottom]);
  grid-template-rows: [top] 5em [bottom top] 5em [top bottom] 5em [bottom];
```

Se você está preocupado em ter o mesmo nome aplicado a linhas

de grade multi-ple, não seja: não há nada que o impeça, e isso pode até ser útil em alguns casos. Exploraremos maneiras de lidar com essas situações em uma seção posterior.

Esteiras de enchimento automático

Há até uma maneira de configurar um padrão simples e repeti-lo até que o contêiner de grade seja preenchido. Isso não tem a mesma complexidade que o repeat() regular (pelo menos não ainda), mas ainda pode ser bastante útil.

Por exemplo, suponha que desejemos que o padrão de linha anterior se repita quantas vezes o contêiner de grade aceitar confortavelmente:

```
grid-template-rows: repeat(auto-fill, [top] 5em [bottom]);
```

Isso definirá uma linha a cada 5 ems até que não haja mais espaço. Assim, para um contêiner de grade com 11 ems de altura, o seguinte é equivalente:

```
grid-template-rows: [top] 5em [bottom top] 5em [bottom ];
```

Se a altura do contêiner de grade for aumentada além de 15 ems, mas for inferior a 20 ems, esta é uma declaração equivalente:

```
grid-template-rows: [top] 5em [bottom top] 5em [top bottom] 5em [bottom];
```

A limitação com a repetição automática é que ela pode ter apenas um nome de linha de grade opcional, um tamanho de faixa fixo e outro nome de linha de grade opcional. Então, o representante [superior] 5em [inferior] se ressente do padrão de valor máximo. Você pode soltar as linhas nomeadas e apenas revogar t 5em, ou apenas soltar um dos nomes. Não é possível repetir vários tamanhos de pista fixos , nem você pode repetir tamanhos de pista flexíveis. (O que faz sentido: quantos vezes um navegador repetiria 1fr para preencher um contêiner de grade?)

Você pode desejar poder repetir automaticamente vários tamanhos de faixa para definir "calhas" em torno de suas colunas de conteúdo. Isso é desnecessário porque as grades têm um conceito de (e propriedades para definir) calhas de trilha, que abordaremos em uma seção posterior.

Além disso, você pode ter apenas uma repetição automática em um determinado modelo de faixa. Assim, *não* seria admissível o seguinte:

```
grid-template-columns: repeat(auto-fill, 4em) repeat(auto-fill, 100px);
```

Faz sentido que isso não seja permitido, uma vez que você olhe para ele. Se o primeiro encher o contêiner de grade com faixas de coluna de 4 em, onde há espaço para faixas de coluna repetidas de 100 pixels ?

No entanto, você *pode* combinar faixas de repetição fixa com faixas de preenchimento automático. Por exemplo, você pode começar com três colunas

largas e, em seguida, preencher o restante do contêiner de grade com trilhas estreitas (supondo que haja espaço para elas). Isso seria mais ou menos assim:

```
grid-template-columns: repeat(3, 20em) repeat(auto-fill, 2em);
```

Você também pode inverter isso:

```
grid-template-columns: repeat(auto-fill, 2em) repeat(3, 20em);
```

Isso funciona porque o algoritmo de layout de grade atribui espaço às faixas fixas primeiro e, em seguida, preenche qualquer espaço restante com faixas repetidas automaticamente. O resultado final desse exemplo é ter uma ou mais faixas 2em preenchidas automaticamente e, em seguida, três faixas 20 em.

Com o preenchimento automático, você sempre obterá pelo menos uma coluna, mesmo que ela não caiba no contêiner de grade pelo motivo. Você também terá quantas faixas se encaixarem, mesmo que algumas das faixas não tenham conteúdo nelas. Por exemplo, suponha que você configure um preenchimento automático que colocasse cinco colunas, mas apenas as três primeiras delas realmente acabaram com itens de grade nelas. Os outros dois permaneceriam no lugar, mantendo o espaço de layout aberto.

Se você usar o ajuste automático, por outro lado, as faixas que não contêm itens de grade serão descartadas. Suponha o seguinte:

```
grid-template-columns: repeat(auto-fit, 20em);
```

Se houver ro om para cinco faixas decoluna no contêiner de grade (ou seja, tiver mais de 100 ems de largura), mas duas faixas não tiverem nenhum item de grade para entrar nelas, essas faixas de grade vazias serão descartadas, deixando as três colunas controles que contêm itens de grade. O espaço restante é tratado de acordo com os valores de align-content e justify-content (discutidos em uma seção posterior).

Há uma razão pela qual esta seção não teve números: no início de 2016, o preenchimento automático e o ajuste automático não haviam sido implementados publicamente, embora a maioria dos navegadores que suportam a grade tenha anunciado planos para suportá-los em breve. Teste bem antes de usar!

Áreas de grade

Você se lembra da arte ASCII? Bem, está de volta, e graças ao grid-template-areas propriedade, você pode usá-lo para criar modelos de grade completos!

grid-template-areas

Valores	nenhum |
Valor inicial:	nenh
Aplica-se	Contêineres de
Herdada:	Nã
Valor calculado :	Conforme

Poderíamos passar por uma descrição prolixa de como isso funciona, mas é muito mais divertido apenas mostrá-lo. A regra a seguir tem o resultado mostrado na Figura 21:

```
#grid {display: grade; grid-
    template-areas:
        "h h h h"
        "l c c r" "l f f
        f";}
```

Figura 21. Um conjunto simples de áreas de grade

Isso mesmo: as letras nos valores da cadeia de caracteres são usadas para definir como as áreas da grade são moldadas. Realmente! E você nem está restrito a letras únicas! Por exemplo , poderíamos expandir o exemplo anterior da seguinte forma:

```
#grid {display: grade; grid-
    template-areas:
        "cabeçalho  cabeçalh  cabeçalh   cabeçalho"
                    o          o
        "Lado       conteúdo   conteúd    lado
        esquerdo                o         direito"
```

```
"Lado esquerdo rodapé rodapé rodapé";}
```

O layout da grade seria o mesmo mostrado na Figura 21, embora o nome de cada área fosse diferente (por exemplo, rodapé em vez de f).

Na definição de áreas de modelo, o espaço em branco é recolhido, para que você possa usá-lo (como fiz no exemplo anterior) para alinhar visualmente colunas de nomes no valor de grid-template-areas. Você pode alinhá-los com espaços ou abas, o que mais irritar seustrabalhadores. Ou você pode apenas usar um único espaço para separar cada identificador e não se preocupar com os nomes alinhados uns com os outros. Você nem precisa quebrar a linha entre as cordas; as seguintes obras tão bem quanto uma versão impressa bonita:

```
grid-template-areas: "h h h h" "l c c r" "l f f f";
```

O que você não pode fazer é mesclar essas cadeias de caracteres separadas em uma única cadeia de caracteres e fazer com que isso signifique a mesma coisa. Cada nova cadeia de caracteres (conforme delimitado pelas aspas duplas)

define uma nova linha na grade. Assim, o exemplo anterior, como os exemplos anteriores, define três linhas. Se os fundirmos todos em uma única cadeia de caracteres, assim:

```
grid-template-areas: "h
    h h h h
     l c c r
     l f  f  f";
```

... então teríamos uma única linha de doze colunas, começando com a área de quatro colunas h e terminando com a área de três colunas f. As quebras de linha não são significativas de forma alguma, exceto como espaço em branco que separa um identificador de outro.

Se você olhar para esses valores de perto, você pode vir a perceber que cada identifier individual representa uma célula de grade. Vamos trazer de volta nosso primeiro exemplo desta seção e considerar o resultado mostrado na Figura 22:

```
#grid {display: grade; grid-
    template-areas:
        "h h h h"
        "l c c r" "l f f
        f";}
```

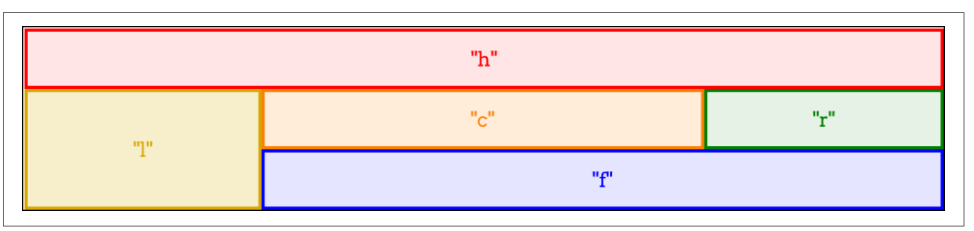


Figura 22. Células de grade com identificadores

Esse é exatamente o mesmo resultado de layout, mas aqui, mostramos como cada identificador de grade no valor grid-template-areas corresponde a uma célula de grade. Depois que todas as células são identificadas , o navegador mescla todas as células adjacentes com o mesmo nome em uma única área que encerra todas elas, desde que descrevam uma forma retangular! Se você tentar configurar áreas mais complicadas, o modelo inteiro será inválido. Assim, o seguinte resultaria em nenhuma área de grade sendo definida:

```
#grid {display: grade;  grid-
    template-areas:
        "h h h h"
        "l c c r" "l l f
        f";}
```

Viu como l delineia uma forma de "L"? Essa simples alteração faz com que todo o valor de áreas de modelo de grade seja descartado como inválido. Uma versão futura do layout de grade pode permitir formas não retangulares, mas, por enquanto, é isso que temos.

Se você tiver uma situação em que deseja definir apenas algumas células de grade para fazer parte de áreas de grade, mas deixara dela sem rótulo, você pode usar uma ou mais . caracteres a serem preenchidos para essas células sem nome. Digamos que você queira apenas definir algumas áreas de cabeçalho, rodapé e barra lateral e deixar o resto sem nome. Isso seria mais ou menos assim, com o resultado mostrado na Figura 23:

```
#grid {display: grade;  grid-
    template-areas:
        "cabeçal  cabeçal  cabeçal  cabeçalho
        ho        ho       ho       "
        "esquerd  ...      ...      certo"
        a
        "rodapé   rodapé   rodapé   rodapé";}
```

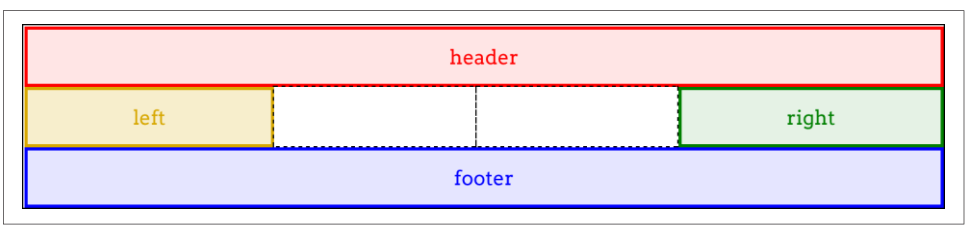


Figura 23. Uma grade *com* algumas *células* de *grade sem nome*

As duas células de grade no centro da grade não fazem parte de uma área nomeada, tendo sido representadas no modelo por *tokens* de *célula nula* (o . identificadores). Onde cada um desses ... sequências aparece, poderíamos ter usado um ou mais tokens nulos - então à esquerda. direita ou esquerda direita teriam funcionado tão bem quanto.

Você pode ser tão simples ou criativo com os nomes de suas células quanto quiser. Se você quiser chamar seu cabeçalho de steve e seu podólogo de rodapé, vá em

frente. Você pode até usar qualquer caractere de código Uni- acima do ponto de código U+0080, de modo que ConHugeCo©®™ e åwësømë são identificadores de área totalmente válidos.

Agora, para dimensionar as faixas de grade criadas por essas áreas, trazemos nossos velhos amigos grid-template-columns e grid-template-rows. Vamos adicionar ambos ao exemplo anterior, com o resultado mostrado na Figura 24:

```
#grid {display: grade;  grid-
    template-areas:
        "cabeçalho cabeçalho cabeçalho"
        "esquerda  ...       ...       direita"
        "rodapé    rodapé rodapé";
    grid-template-columns: 1 fr 20em  20em 1 fr;
    grid-template-rows: 40px 10em 3em;}
```

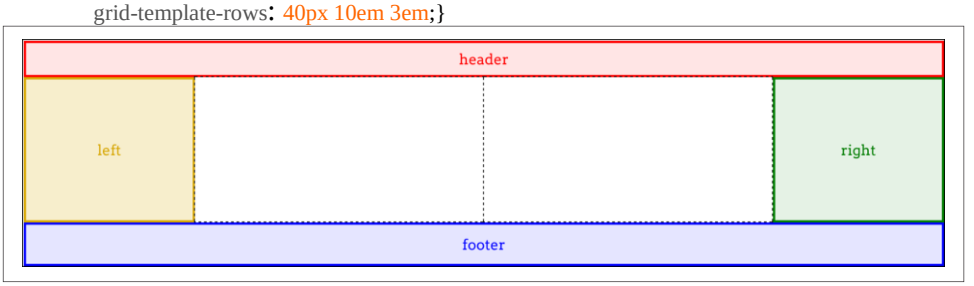


Figura 24. Áreas nomeadas e faixas dimensionadas

Assim, as colunas e linhas criadas pela nomeação das áreas de grade recebem tamanhos de faixa. Se dermos mais tamanhos de pista do que há faixas de área, isso simplesmente adicionará mais faixas além das áreas nomeadas. Portanto, o CSS a seguir levará ao resultado mostrado na Figura 25:

```
#grid {display: grade;  grid-
    template-areas:
        "cabeçalho cabeçalho cabeçalho"
        "esquerda  ...       ...       direita"
        "rodapé    rodapé rodapé";
    grid-template-columns: 1 fr 20em  20em 1 fr  1 fr;
    grid-template-rows: 40px 10em 3em 20px;}
```

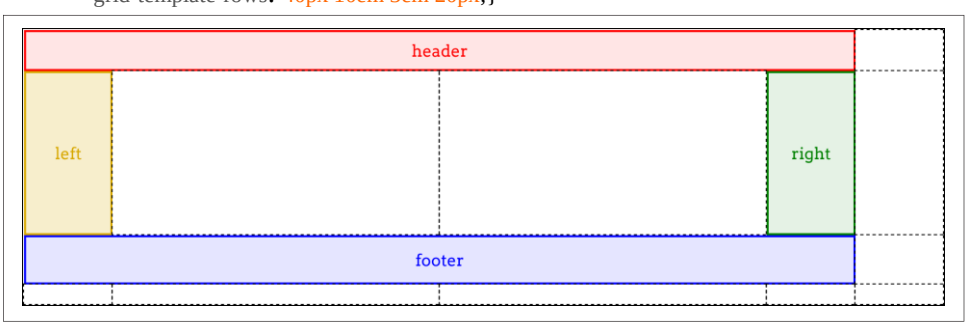


Figura 25. Adicionando mais faixas além das áreas nomeadas

Então, dado que estamos nomeando áreas, que tal misturar algumas linhas de grade nomeadas? Acontece que já temos: nomear uma área de grade adiciona automaticamente nomes às linhas de grade em seu início e fim. Para a área de cabeçalho, há um nome implícito de início de cabeçalho em sua primeira linha de grade de coluna e sua primeira linha de grade de linha , e de extremidade de cabeçalho para suas linhas de segunda coluna *e* de grade de linha . Para a área do rodapé, os nomes de início e fim do rodapé foram atribuídos automaticamente às suas linhas de grade.

As linhas de grade se estendempor toda a área da grade, então muitos desses nomes são coincidências. A Figura 26 mostra a nomenclatura das linhas criadas pelo seguinte modelo:

```
grid-template-areas:
    "cabeçalho cabeçalho cabeçalho "
    "esquerda …          …       certo"
    "rodapé    rodapé    rodapé  rodapé";
```

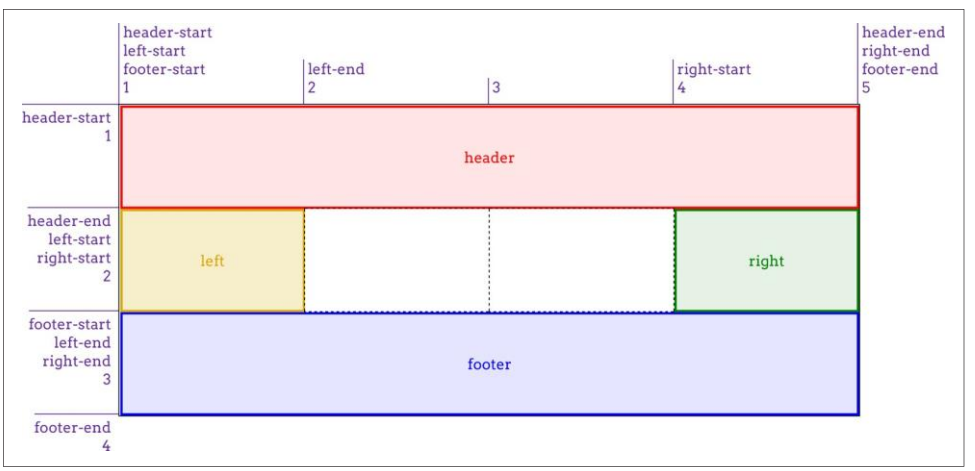


Figura 26. Nomes de linhas de grade implícitos tornados explícitos

Agora vamos misturá-lo ainda mais, adicionando alguns nomes de linha de grade explícitos ao nosso CSS. Dadas as regras a seguir, a primeira linha de grade de coluna na grade adicionaria o nome start e a segunda linha de grade line na grade adicionaria o conteúdo do nome.

```
#grid {display: grade;  grid-
    template-areas:
        "cabeçalho cabeçalho cabeçalho"
        "esquerda  ...       ...       direita"
        "rodapé    rodapé rodapé";
    grid-template-columns: [begin] 1 fr 20em 20em  1 fr 1
    fr;  grid-template-rows:  40px [conteúdo] 1fr 3em 20px;}
```

Novamente: esses nomes de linha de grade são *adicionados* aos nomes de linha de grade implícitos criados pelas áreas nomeadas. Curiosamente, os nomes de linha de

grade nunca substituem outros nomes de linha de grade. Em vez disso, eles continuam se acumulando.

Ainda mais interessante, esse mecanismo de nome implícito é executado ao contrário. Suponha que você não use grid-template-areas de forma alguma, mas configure algumas linhas de grade nomeadas assim , conforme ilustrado na Figura 27:

```
grid-template-columns:
    [cabeçalho-início rodapé-início] 1fr
    [content-start] 1 fr [content-end] 1 fr
    [header-end footer-end];
grid-template-rows:
    [header-start] 3em
    [início do conteúdo do início do cabeçalho] 1fr [content-end
    footer-start] 3em [footer-end];
```

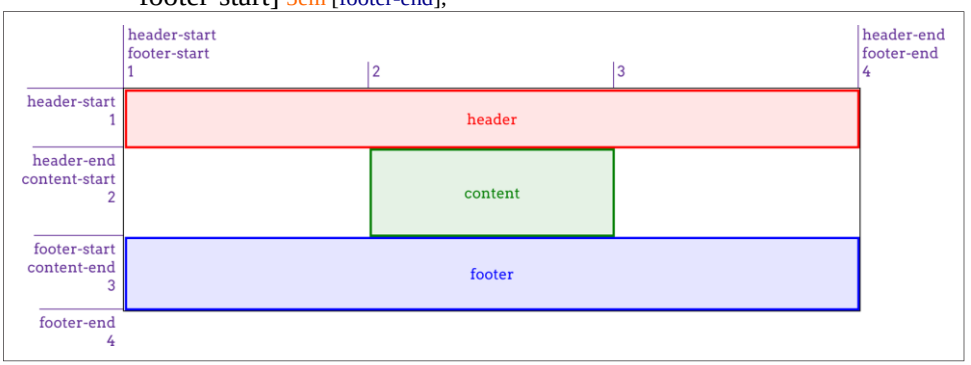


Figura 27. Nomes de áreas de grade implícitas explicitados

Como as linhas de grade usam a forma de name-start/name-end, as áreas de grade que elas definem são implicitamente nomeadas. Para ser franco, é mais desajeitado do que fazê-lo de outra maneira, mas a capacidade está lá no caso de você querer.

Tenha em mente que você não precisa que todas as quatro linhas de grade sejam nomeadas para criar uma área de grade nomeada, embora você provavelmente precise de todas elas para criar uma área de grade nomeada onde você deseja que ela esteja. Considere o seguinte exemplo simples:

```
grid-template-columns: 1 fr [content-start] 1 fr [ content-end] 1 fr;
grid-template-rows: 3em 1 fr 3em;
```

Isso ainda criará uma área de grade chamada conteúdo. É que a área nomeada será colocada em uma nova linha depois de todas as linhas definidas. O que é estranho é que uma linha extra vazia aparecerá após as linhas definidas, mas antes da linha que contém o conteúdo. Este foi confirmado como sendo o comportamento pretendido. Assim, se você tentar criar uma área nomeada nomeando as linhas de grade e perder uma ou mais delas, sua área nomeada será efetivamente pendurada para um lado da grade, em vez de fazer parte da estrutura geral do grid.

Então, novamente, você provavelmente deve se ater a nomear explicitamente as áreas da grade e deixar que os nomes das linhas da grade aconteçam implicitamente, em oposição ao contrário.

Anexando elementos à grade

Acredite ou não, chegamos até aqui sem falar sobre como os itens de grade são realmente anexados a uma grade, uma vez que ela tenha sido definida.

Usando linhas de coluna e linha

Existem algumas maneiras de fazer isso, dependendo se você deseja se referir a áreas de gradeou grid. Começaremos com quatro propriedades simples que anexam um elementar às linhas de grade.

grid-row-start, grid-row-end, grid-column-start , fim de coluna de grade

Valores	| automática | < > personalizado| [<inteiro> &&& <personalizado>?] | [palmo && [<inteiro> || <personalizado-ident>]]
Valor inicial:	Auto
Aplica-se	Itens de grade e elementos | absolutamente posicionados |, se o bloco | que o contém durante micrômetro contêiner de grade
Herdada:	Nã
Valor calculado :	Conforme

O que essas propriedades fazem é permitir que você diga: "Quero que a borda do elemento seja anexada à linha de grade tal e assim". Tal como acontece com grande parte do layout de grade, é muito mais fácil mostrar do que descrever, então reflita sobre os seguintes estilos e seu resultado (Figura 28):

```
.grid {display: grade; largura: 50em;
      linhas de modelo de grade: repetir(5, 5em);
       grid-template-columns: repeat(10, 5em);}
.one {
       grid-row-start: 2; grade-linha-fim: 4;
       grid-column-start: 2; fim da coluna da grade: 4;}
.two {
       grid-row-start: 1; grade-linha-fim: 3;
       grid-column-start: 5; fim da coluna da grade: 10;}
.three {
      grid-row-start: 4;
       início-coluna-de-grade: 6;}
```

Aqui, estamos usando números de linha de grade para dizer onde e como os elementos devem ser colocados dentro da grade. Os números das colunas contam da esquerda para a direita e os números das linhas de cima para baixo. Observe que, se você omitir as linhas de grade finais, como foi o caso de .three, as próximas linhas de grade em sequência serão usadas para as linhas finais .

Assim , a regra para .três no exemplo anterior é exatamente equivalente à seguinte:

```
.three {
    grid-row-start: 4; grade-linha-fim: 5;
    grid-column-start: 6; fim da coluna da grade: 7;}
```

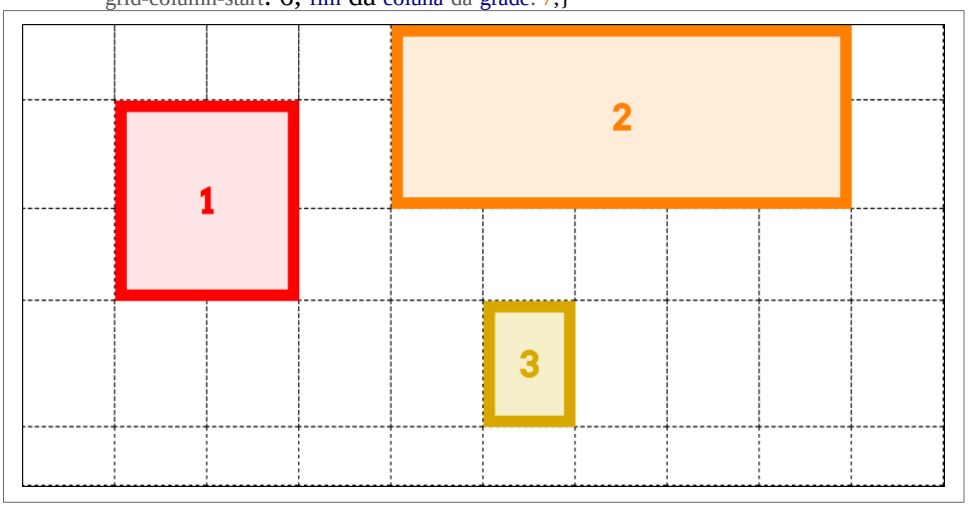


Figura 28. Anexando elementos a linhas de grade

Há outra maneira de dizer a mesma coisa, como acontece: você pode substituir os valores finais por span 1, ou mesmo apenas span simples, assim:

```
.three {
    grid-row-start: 4; grid-row-end: span 1;
    grid-column-start: 6; grid-column-end: extensão;}
```

Se você fornecer um número, você está dizendo: "span através de tantas células de grade". Assim, poderíamos reescrever nosso exemplo anterior como este e obter exatamente o mesmo resultado:

```
#grid {display: grade;
    linhas de modelo de grade: repetir(5, 5em);
    grid-template-columns: repeat(10, 5em);}
.one {
    grid-row-start: 2; grid-row-end: extensão 2;
    grid-column-start: 2; grid-column-end: span 2;}
.two {
    grid-row-start: 1; grid-row-end: extensão 2;
    grid-column-start: 5; grid-column-end: span 5;}
.three {
    grid-row-start: 4; grid-row-end: span 1;
```

```
    grid-column-start: 6; grid-column-end: extensão;}
```

Se você deixar de fora um número para a extensão, ele será definido como 1. Você não pode usar zero ou núm plices negativos para a extensão; apenas inteiros positivos.

Uma característica interessante do span é que você pode usá-lo para linhas de grade final *e* inicial. O comportamento preciso da extensão é que ele conta linhas de grade na direção "longe" da linha de grade onde começa. Em outras palavras, se você definir uma linha de grade inicial e definir a linha de grade final como um valor de extensão, ela pesquisará em direção ao final da grade. Conversamente, se você definir uma linha de grade final e tornar a linha inicial um valor de extensão, ela procurará em direção ao início da grade.

Isso significa que as seguintes regras terão o resultado mostrado na Figura 29:

```
#grid {display: grade;
     linhas de grade: repetir(4, 2em); colunas de grade: repeat(5, 5em);}
.box01 {grid-row-start: 1; grid-column-start: 3; grid-column-end: span 2;}
.box02 {grid-row-start: 2; grid-column-start: span 2; fim da coluna da grade: 3;}
.box03 {grid-row-start: 3; grid-column-start: 1; grid-column-end: span 5;}
.box04 {grid-row-start: 4; grid-column-start: span 1; fim da coluna de grade: 5;}
```

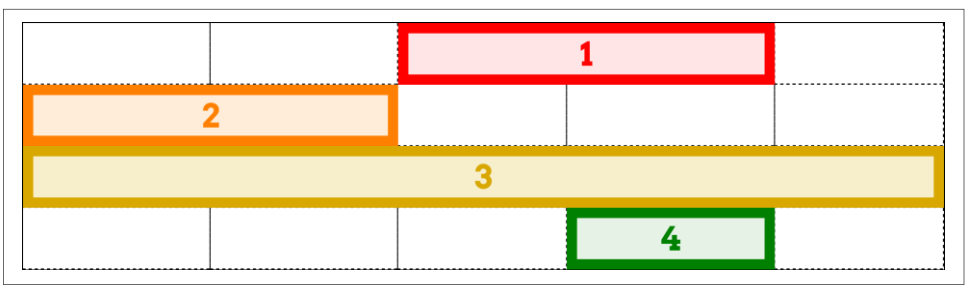


Figura 29. Abrangendo linhas de grade

Em contraste com a numeração de extensão, você não está restrito a inteiros positivos para seus valores reais de linha de grade. Os números negativos simplesmente contarão para trás a partir do final das linhas de grade explicitamente definidas. Assim, para colocar um elemento na célula de grade inferior direita de uma grade definida, independentemente de quantas colunas ou linhas ele possa ter, você pode simplesmente dizer o seguinte:

```
grid-column-start: -1;
grid-row-start: -1;
```

Observe que isso não se aplica a nenhuma faixa de grade implícita, um conceito que abordaremos daqui a pouco, mas apenas às linhas de grade que você define explicitamente por meio de uma das linhas de grade do modelo de grade-* (por exemplo, linhas de modelo de grade).

Não estamos restritos a números de linhas de grade, as isso acontece. Se houver linhas de grade nomeadas, podemos nos referir a elas em vez de (ou em conjunto

com) números. Se você tiver instâncias multi-ple de um nome de linha de grade , poderá usar números para identificar qual instância do nome da linha de grade você está falando uma luta. Assim, para começar a partir da quarta instância de uma grade de linha chamada mast-slice, você pode dizer mast-slice 4. Dê uma olhada no seguinte, ilustrado na Figura 30, para ter uma ideia de como isso funciona.

```
#grid {display: grade;
      linhas de modelo de grade: repeat(5, [R] 4em);
       grid-template-columns: 2em repeat(5, [col-A] 5em  [col-B] 5em)  2em;}
.one {
       grid-row-start: R  2; grade-linha-fim:  5;
      grid-column-start: col-B;  grid-column-end:  span 2;}.two {
        grid-row-start: R; grid-row-end: vão R  2;
        grid-column-start: col-A  3;  grid-column-end: span 2 col-A;}
.three {
       grid-row-start: 9;
      grid-column-start: col-A -2;}
```

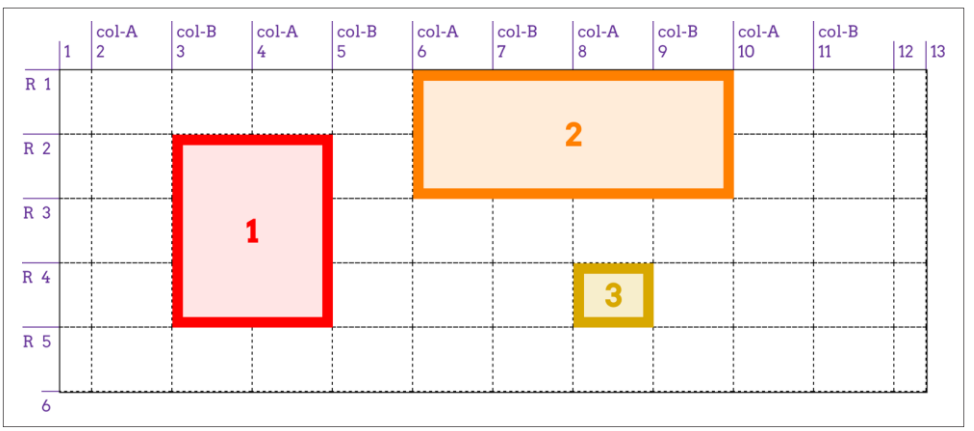


Figura 30. Anexando elementos a linhas de grade nomeadas

Observe como a extensão muda quando adicionamos um nome: onde dissemos a extensão 2 col-A, que fez com que o item de grade se estendesse de seu ponto de partida (o terceiro col-A) através de outro col-A e terminasse no col-A depois disso. Isso significa que o item de grade realmente abrange quatro células de grade , já que col-A aparece em todas as outras linhas de grade de coluna.

Novamente, os números negativos contam para trás a partir do final de uma sequência, de modo que col-A-2 nos obtém a penúltima instância de uma linha de grade chamada col-A. Como não há valores de linha final declarados para .three, ambos são definidos como abrangendo 1. Isso significa que o seguinte é exatamente equivalente ao .three no exemplo anterior:

```
.three {
       grid-row-start: 9;  grid-row-end:  span 1;
      grid-column-start: col-A -2;  grid-row-end:  span 1;}
```

Há uma maneira alternativa de usar nomes com linhas de grade nomeadas —

especificamente, as linhas de grade nomeadas que são implicitamente criadas por áreas de grade. Por exemplo, considere os seguintes estilos, ilustrados na Figura 31:

```
grid-template-areas:
    "              cabeçalho              cabeçalho
    cabeçalho      " "conteúdo do lado
    esquerdo do lado direito" "rodapé do
                   rodapédo rodapé        do
                    lado esquerdo                ";
#masthead {grid-row-start: cabeçalho;
          grid-column-start: cabeçalho;  grid-row-end: cabeçalho;}
#sidebar {gri d-row-start: 2;  grade-linha-fim:  4;
           grid-column-start: lado esquerdo / span 1;}
#main {grid-row-start: conteúdo;  grid-row-end: conteúdo;
          grid-column-start:  conteúdo;}
#navbar {grid-row-start:  lado direito;  grd-row-end: 3;  grid-
          column-start: lado direito;}
#footer {  grid-row-start: 3;  grid-row-end:  span 1;
           grid-column-start: rodapé; grid-row-end:  rodapé;}
```

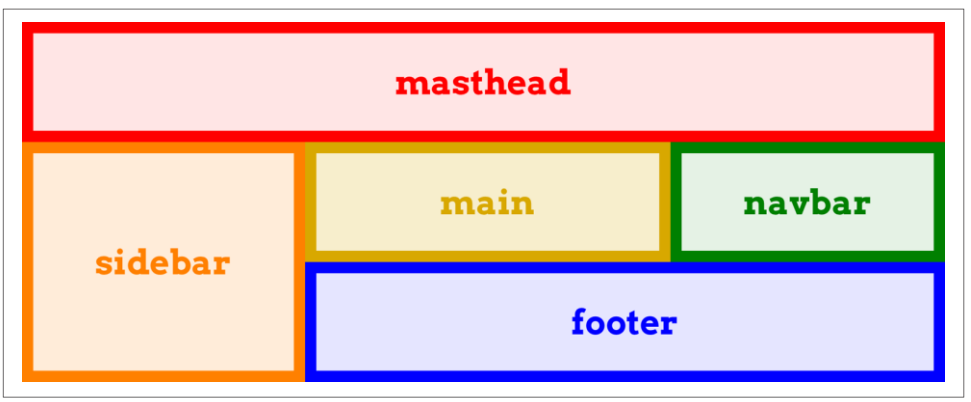


Figura 31. Outra maneira de anexar elementos a linhas de grade nomeadas

O que acontece se você fornecer um identificador personalizado (ou seja, um nome que você definiu) é que o navegador procura uma linha de grade com esse nome *mais* -start ou -end adicionado, dependendo se você está atribuindo uma linha inicial ou uma linha final. Assim, o seguinte umequivalente re:

```
grid-column-start: cabeçalho;  grid-column-end: cabeçalho;
grid-column-start: cabeçalho-início;  grid-column-end: header-end;
```

Isso funciona porque, como foi mencionado com grid-template-areas, a criação explícita de uma área de grade cria implicitamente as linhas de grade nomeadas -start e -end que a cercam.

A possibilidade de valor final, auto, é interessante. De acordo com a especificação Layout de Grade, se uma das propriedades start/end da linha de grade estiver definida como automática, isso indica "posicionamento automático, uma extensão

automática ou uma extensão padrão de uma". Na prática, o que isso tende a significar é que a linha de grade que é escolhida é governada pelo *grid flow*, um conceito que ainda temos que cobrir (mas em breve!). Para uma linha inicial, auto geralmente significa que a próxima coluna ou linha disponível será usada. Para uma linha final, auto geralmente significa uma extensão de uma célula. Em ambos os casos, a palavra "geralmente" é usada intencionalmente: como em qualquer mecanismo automático, não há absolutos.

Taquigrafias de linha e coluna

Existem duas propriedades abreviadas que permitem anexar de forma mais compacta um elemento às linhas de grade.

linha de grade, coluna

Valores	<linha de grade> [/ <linha de
Valor inicial:	Auto
Aplica-se	Itens de grade e elementos | absolutamente posicionados |, se o bloco | que o contém durante micrômetro contêiner de grade
Herdada:	Nã
Valor calculado :	Conforme
Nota:	É equivalente à sintaxe | fazer valor para grid-column-start Et Al.

O principal benefício dessas propriedades é que elas tornam muito mais simples declarar as linhas de grade inicial e final a serem usadas para o layout de um item de grade. Por exemplo:

```
#grid {display: grade;
      linhas de modelo de grade: repetir(10, [R] 1,5em);
      grid-template-columns: 2em repeat(5, [col-A] 5em [col-B] 5em) 2em;}
.one {
      fila de grade: R 3 / 7;
      coluna de grade: col-B / span 2;}
.two {
      linha de grade: R /vão R 2;
      coluna de grade: col-A 3 / span 2 col-A;}
.three {
      linha de grade: 9;
      coluna de grade: col-A -2;}
```

Isso é muito mais fácil de ler do que ter cada valor inicial e final em seu próprio suporte, honestamente. Além de ser mais compacto, o comportamento dessas

propriedades é mais ou menos o que você esperaria. Se você tiver dois bits separados por um solidus, a primeira parte define a linha do grid inicial e a segunda parte define a linha do grid final.

Se você tem apenas um bit sem solidus, ele define a linha do grid de partida. A linha de grid final depende do que você disse para a linha de partida. Se você fornecerum nome para a linha de grade de partida, a linha de grade final receberá o mesmo nome. Assim, os seguintes elementos são equivalentes:

```
coluna-grade: col-B;
grade-coluna: col-B / col-B;
```

Isso, é claro, se estenderá de uma instância desse nome de linha de grade para a próxima, independentemente de quantas células de grade sejam estendidas.

Nos casos em que apenas um nome é dado para a coluna de grade ou linha de grade, ele também é usado para a segunda (a linha final). Se um único número for fornecido, o segundo número (a linha final) será definido como automático. Isso significa que os seguintes pares são equivalentes:

```
fila de grade: 2;
grade-linha: 2 / auto;

grade-coluna: cabeçalho;
grid-column: cabeçalho / cabeçalho;
```

Há um comportamento sutil incorporado ao processamento de nomes de linhas de grade em linhas de grade e colunas de grade que pertencem a linhas de grade implicitamente nomeadas. Se você se lembrar, definir uma área de grade nomeada criará linhas de grade -start e -end. Ou seja, dada uma área de grade com um name de rodapé, existem linhas de grade de início de rodapé implicitamente criadas em sua parte superior e esquerda, e linhas de grade de extremidade de rodapé em sua parte inferior e direita.

Nesse caso, se você se referir a essas linhas de grade pelo nome da área, o elemento ainda será colocado corretamente. Assim, os seguintes estilos têm o resultado mostrado na Figura 32:

```
#grid {display: grade; grid-
    template-areas:
        "cabeçalho do
        cabeçalho"
        "conteúdo da barra
        lateral" "rodapé do
        rodapé";
      grid-template-rows: auto 1 fr auto;
     grid-template-columns: 25% 75%;}
#header {grid-row: cabeçalho / cabeçalho; coluna de grade: cabeçalho;}
```

```
#footer {grid-row: rodapé; grid-column: footer-start / footer-end;}
```

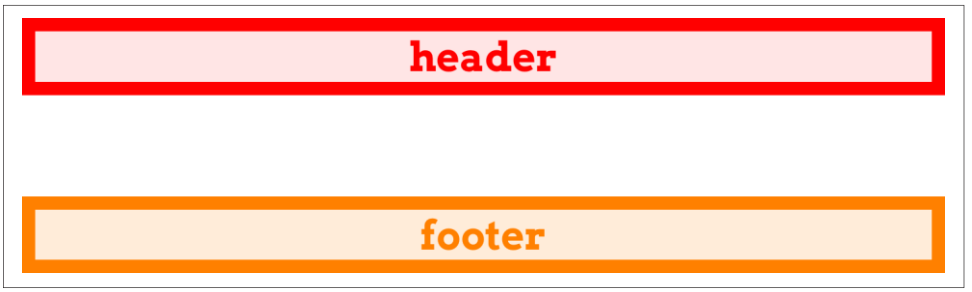


Figura 32. Anexando a linhas de grade *implícitas por meio* de *nomes de área de grade*

Você sempre pode se referir explicitamente às linhas de grade implicitamente nomeadas, mas se você apenas se referir à área da grade, as coisas ainda funcionarão. Se você se referir a um nome de linha de grade que não corresponde a uma área de grade, ele voltará ao comportamento discutido anteriormente. Em detalhes, é o mesmo que dizer o nome da linha 1, então os dois a seguir são equivalentes:

```
grade-coluna: jane / doe; grade-
coluna: jane 1 / doe 1;
```

É por isso que é arriscado nomear as linhas de grade da mesma forma que as áreas de grade. Considere o seguinte:

```
grid-template-areas:
    "cabeçalho do
    cabeçalho"
    "conteúdo da barra
    lateral" "rodapé
    do rodapé" "legal
    legal ";
  grid-template-rows: auto 1 fr [rodapé] auto
[rodapé]; grid-template-columns: 25% 75%;
```

Isso define explicitamente as linhas de grade chamadas de rodapé acima da linha "rodapé" e abaixo da linha "legal"... e agora há problemas pela frente. Suponha que adicionemos isto:

```
#footer {grid-column: rodapé; linha de grade: rodapé;}
```

Para as linhas de coluna, não há problema. rodapé é expandido para rodapé / rodapé. O navegador procura uma área de grade com esse nome e a encontra, por isso traduz rodapé / rodapé para rodapé-início / rodapé-fim. O elemento #footer é anexado a essas linhas de grade implícitas.

Para a linha de grade, tudo começa da mesma forma. footer torna-se footer / footer, que é traduzido para footer-start / footer-end. Mas isso significa que o #footer só será tão alto quanto a linha de "rodapé". Ele *não* se estenderá até a segunda linha de grade de rodapé explicitamente nomeada abaixo da linha "legal", porque a

conversão de rodapé para rodapé (devido à correspondência entre o nome da linha de grade e o nome da área de grade) tem precedência.

O resultado de tudo isso: geralmente é uma má ideia usar o mesmo nome para áreas de grade e linhas de grade. Você pode ser capaz de se safar em alguns cenários, mas é quase sempre melhor manter seus nomes de linha e área distintos, de modo a evitar tropeçar em conflitos de resolução de nomes.

A Grade Implícita

Até este ponto, nos preocupamos apenas com grades explicitamente definidas: falamos sobre as faixas de linha e coluna que definimos por meio de propriedades como colunas de modelo de grade e como anexar itens de grade às células nessas faixas.

Mas o que acontece se tentarmos colocar um item de grade, ou mesmo apenas parte de um item de grade, além dessa grade explicitamente criada? Por exemplo, considere a seguinte grade simples:

```
#grid {display: grade;
      grid-template-rows: 2em 2em;
       grid-template-columns: repeat(6, 4em);}
```

Duas linhas, seis colunas. Simples o suficiente. Mas suponha que definamos um item de grade para se sentar na primeira coluna e ir da primeira linha de grade para a quarta:

```
.box01 {grid-column: 1; linha de grade: 1 / 3;}
```

E agora? Há apenas duas linhas delimitadas por três linhas de grade, e dissemos ao navegador para ir além disso, da linha 1 à linha 4.

O que acontece é que outra linha de linha é criada para lidar com a situação. Essa linha de grade e a nova faixa de linha que ela cria fazem parte da grade *implícita*. Aqui estão alguns exemplos de itens de grade que criam linhas de grade implícitas (e faixas) e como elas são dispostas (consulte a Figura 33):

```
.box01 {grid-column: 1; linha de grade: 1 / 3;}
.box02 {grid-column: 2; linha de grade: 3 / span 2;}
.box03 {grid-column: 3; linha de grade: span 2 / 3;}
.box04 {grid-column: 4; linha de grade: span 2 / 5;}
.box05 {grid-column: 5; linha de grade: span 4 / 5;}
.box06 {grid-column: 6; linha de grade: -1 / span 3;}
```

```
.box07 {grid-column: 7; linha de grade: span 3 / -1;}
```

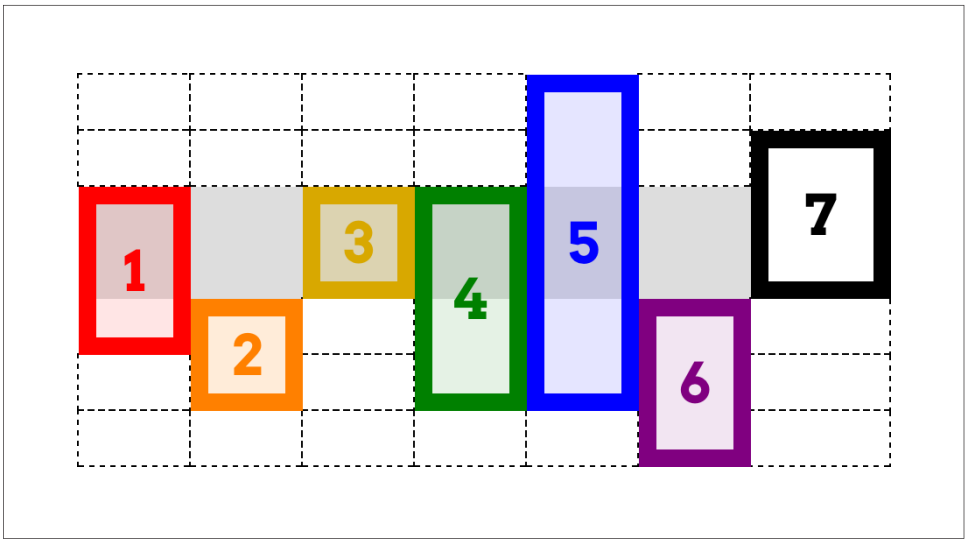


Figura 33. Criando linhas de grade e faixas implícitas

Há muita coisa acontecendo lá, então vamos detalhar . Em primeiro lugar, a grade explícita é repre-enviada pela caixa cinza-claro atrás das várias caixas numeradas; todas as linhas tracejadas representam a grade implícita.

Então, e aquelas caixas numeradas? box1 adiciona uma linha de grade extra após o final da grade explícita, como discutimos anteriormente. box2 começa na última linha da grade explícita e se estende por duas linhas, de modo que adiciona ainda outra linha de grade implícita. caixa 3 termina na última linha de grade explícita (linha 3) e se estende para trás duas linhas, começando assim na primeira grade explícita line.

box4 é onde as coisas realmente ficam interessantes. Ele termina na linha da quinta linha, ou seja, a segunda linha de grade implícita. Ele se estende por três linhas e, no entanto, ainda começa na mesma linha de grade que o box3. Isso acontece porque os vãos precisam começar a contar dentro da grade explícita. Uma vez que eles começam, eles podem continuar na grade implícita (como happed com box2), mas eles *não podem* começar a contar dentro da grade implícita.

Assim, a caixa 4 termina na linha 5, mas sua extensão começa com a linha de grade 3 e conta duas linhas (extensão 2) para chegar à linha 1. Da mesma forma, a caixa 5 termina na linha 5 e se estende por quatro linhas, o que significa que ela começa na linha –2. Lembre-se: a contagem de extensão deve *começar* na grade explícita. Não precisa parar por aí.

Depois disso, a caixa 6 é bastante direta: ela começa na última linha explícita (linha 3) e se estende até a sexta linha de linha — adicionando ainda outra linha implícita. O objetivo de tê-lo aqui é mostrar que as referências negativas de linha de grade são em relação à grade implícitae contar a partir de seu final. Eles *não* se referem a linhas

implícitas indexadas negativamente que são colocadas antes do início da grade explícita.

Se você quiser iniciar um elemento em uma linha de grade implícita antes do início da grade explícita, a maneira de fazer isso é mostrada pela caixa 7: coloque sua linha final em algum lugar da grade explícita e vá de volta além do início da grade explícita. E você deve ter notado: a caixa 7 ocupa uma faixa de coluna implícita. A grade original foi configurada para criar seis colunas, o que significas sete linhas de coluna, sendo a sétima o final da grade explícita. Quando a caixa 7 recebeu a coluna de grade: 7, isso era equivalente à coluna de grade: 7 / span 1 (uma vez que uma linha final ausente é sempre assumida como span 1). Isso exigiu a criação de uma linha de coluna implícita para manter o item de grade na sétima coluna implícita.

Agora vamos pegar esses princípios e adicionar linhas de grade nomeadas à mistura. Veja-se o seguinte, ilustrado em Figure 34:

```
#grid {display: grade;
    grid-template-rows: [begin] 2em [middle] 2em [ end ];
    grid-template-columns: repeat(5, 5em);}
.box01 {grid-column: 1; linha de grade: 2 / extensão final 2;}
.box02 {grid-column: 2; linha-de-grade: 2 / span final;}
.box03 {grid-column: 3; linha de grade: 1 / span 3 meio;}
.box04 {grid-column: 4; linha de grade: span start 2 / end;}
.box05 {grid-column: 5; linha de grade: span 2 meio / início;}
```

O que você pode ver no trabalho lá, em vários dos exemplos, é o que acontece com nomes de linha de grade na grade implícita: cada linha criada implicitamente tem o nome que está sendo caçado. Tomemos a caixa 2, por exemplo. É dada uma linha final de final, mas não há nenhuma linha com esse nome. Assim, a pesquisa de extensão vai até o final da grade explícita e, não tendo encontrado o nome que está procurando, cria uma nova linha de grade, à qual anexa o nome final.

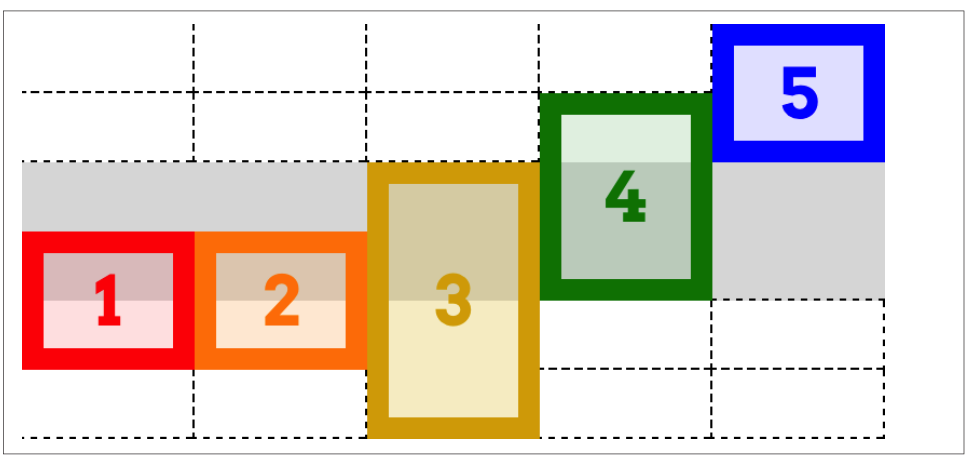


Figura 34. Linhas de grade e faixas implícitas nomeadas

Da mesma forma, a caixa 3 começa na primeira linha de linha explícita e, em seguida,

precisa abranger três linhas nomeadas do meio. Ele procura para a frente e encontra um, em seguida, vai à procura dos outros dois. Não encontrando nenhuma, ele anexa o nome do meio à primeira linha de grade implícita e, em seguida, faz o mesmo para a segunda linha de grade implícita. Assim, ele termina duas linhas de grade implícitas após o final da grade explícita.

Quando você vai direto ao ponto, a grade implícita é um mecanismo de fallback deliciosamente barroco. Geralmente, é uma prática recomendada manter a grade explícita e garantir que a grade explícita cubra tudo o que você deseja fazer. Se você achar que precisa de outra linha, não apenas corra para fora da borda da grade – ajuste os valores do seu modelo de grade!

Tratamento de erros

Há alguns casos que precisam ser cobertos, pois eles se enquadram no guarda-chuva geral de "o que as grades fazem quando as coisas ficam em forma de pera".

Primeiro, e se você acidentalmente colocar a linha de partida após a linha de chegada? Diga, algo assim:

```
grid-row-start: 5;
grade-linha-fim: 2;
```

Tudo o que acontece é provavelmente o que se quis dizer em primeiro lugar: os valores são trocados. Assim, você acaba com isso:

```
grid-row-start: 2;
grade-linha-fim: 5;
```

Em segundo lugar, e se as linhas de início e de fim forem declaradas como extensões de alguma variedade? Por exemplo:

```
grid-column-start:
extensão; grade-coluna-
fim: extensão 3;
```

Se isso acontecer, o valor final será descartado e substituído por automático. Isso significa que você acabaria com isso:

```
grid-column-start: extensão; /* 'span' é igual a 'span 1' */
grid-column-end: auto;
```

Isso faria com que o item de grade tivesse sua borda final colocada automaticamente, de acordo com o fluxo de grade atual (um assunto que exploraremos em breve), e a borda inicial fosse colocada uma linha de grade mais cedo.

Terceiro, e se a única coisa que direciona o posicionamento do item de grade for uma extensão nomeada? Em outras palavras:

```
grid-row-start: rodapé de
extensão; grid-row-end: auto;
```

Isso não é permitido, portanto, o rodapé da extensão neste caso é substituído pela

extensão 1.

Usando áreas

Anexar por linhas de linha e linhas de coluna é ótimo, mas e se você pudesse se referir a uma área de grade com uma única propriedade? Eis aí: área de grade.

área de

Valores	<grid-line> [/ < de grade>]{0,3}
Valor inicial:	Ver propriedades |
Aplica-se	Itens de grade e elementos | absolutamente posicionados |, se o bloco | que o contém durante micrômetro contêiner de grade
Herdada:	Nã
Valor calculado :	Conforme
Nota:	É equivalente à sintaxe | fazer valor para grid-column-start Et Al.

Vamos começar com o uso mais fácil da área de grade: atribuir um elemento a uma área de grade definida anteriormente . Faz sentido, certo? Vamos trazer de volta nosso velho amigo grid-template-areas, juntá-lo com grid-area e alguma marcação, e ver o que resulta mágicamente (como mostrado na Figura 35):

```
#grid {display: grade;  grid-
    template-areas:
        "              cabeçalho              cabeçalho
        cabeçalho" "conteúdo do lado esquerdo do
        lado direito" "rodapé do rodapé do
                    rodapédo                  lado
        esquerdo       "            ;}
#masthead {grid-area: cabeçalho;}
#sidebar {grid-area: leftside;}  #main
{grid-area: content;}  #navbar {grid-
area: rightside;}  #footer {grid-area:
rodapé;}

<div id="grid">
    <div id="masthead">… </div>
    <div id="main">… </div>
    <div id="navbar">… </div>
    <div id="sidebar">… </div>
    <div id="footer">… </div>
```

```
</div>
```

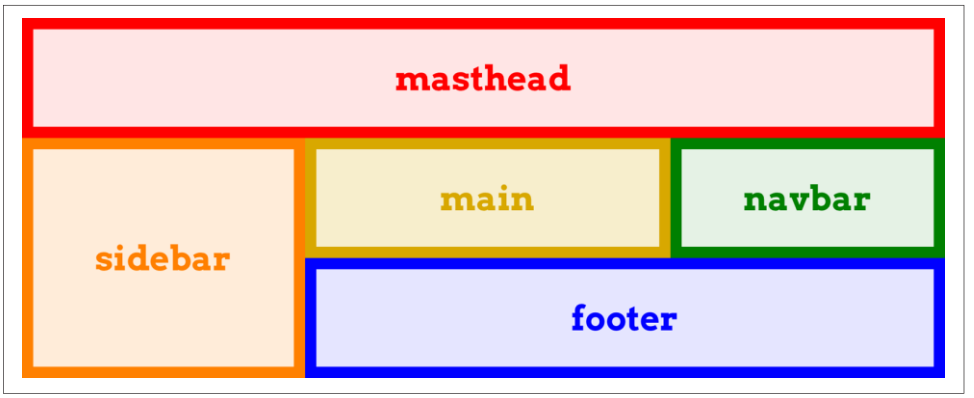


Figura 35. Atribuindo elementos a áreas de grade

Isso é tudo o que é preciso: configure algumas áreas de grade nomeadas para definir seu layout e, em seguida, solte itens de grade nelas com a área de grade. Tão simples e, no entanto, tão poderoso.

Como você deve ter notado, o dimensionamento das faixas de coluna e linha foi omitido desse CSS. Isso foi feito inteiramente por uma questão de clareza. Em um design real, a regra provavelmente ficaria mais parecida com esta:

```
grid-template-areas:
    "          cabeçalhocabeçalho
    cabeçalho" "conteúdo do lado
    esquerdo do lado  direito" "rodapé
    do  rodapé do rodapé do  lado esquerdo";
grid-template-rows: 200px 1fr 3em;
 grid-template-columns: 20em 1 fr 1 fr 10em;
```

Há outra maneira de usar a área de grade que se refere a linhas de grade em vez de áreas de grade. Aviso justo: é provável que seja confuso no início, por algumas razões.

Aqui está um exemplo de um modelo de grade que define algumas linhas de grade e algumas regras de área de grade que fazem referência às linhas, conforme ilustrado na Figura 36:

```
#grid {display: grade;
    linhas de modelo de
    grade:
        [r1-início]  1fr [r1-end r2-start] 2 fr [r2-end];  grid-
    template-columns:
        [col-start]  1fr [col-end main-start] 1fr [main-end];}
.box01 {grid-area: r1 / main /  r1  / main;}
.box02 {grid-area: r2-start /  col-start /  r2-end  /  main-end;}
```

```
.box03 {área de grade: 1 / 1 / 2 / 2;}
```

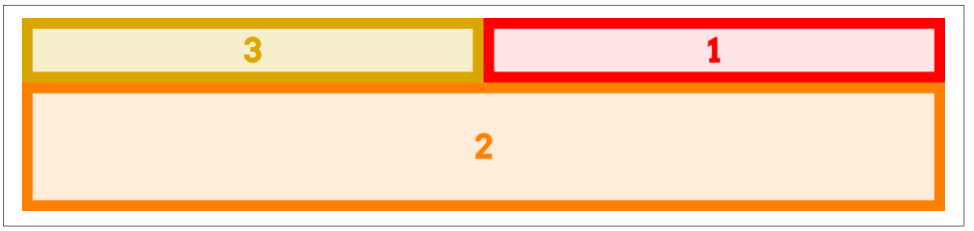


Figura 36. Atribuindo elementos a linhas de grade

Como você pode ver, os elementos foram colocados conforme as instruções. Observe a ordem dos valores da linha de grade, no entanto. Eles são listados na ordem de início de linha, início de coluna, fim de linha, fim de coluna. Se você diagramar isso em sua cabeça, perceberá rapidamente que os valores vão no sentido anti-horário ao redor do item da grade – o exato oposto do TRBL (Top, Right, Bot- tom, Left) padrão ao qual estamos acostumados a partir de margens, preenchimento, bordas e assim por diante. Além disso, isso significa que as referências de coluna e linha não são agrupadas, mas são divididas.

Sim, isso é intencional. Não, não sei porquê.

Se você fornecer menos de quatro valores, os valores ausentes serão retirados daqueles que você fornecer. Se houver apenas três valores, o grid-column-end ausente será o mesmo que grid-column-start se for um nome; se a linha inicial for um número, a linha final será definida como automática. O mesmo se aplica se você fornecer apenas dois valores, exceto que o grid-row-end agora ausente é copiado de grid-row-start se for um nome; caso contrário, ele será definido como automático.

A partir disso, você provavelmente pode adivinhar o que acontece se apenas um valor for fornecido: se for um nome, use-o para todos os quatro valores; se for um número, o resto é definido como automático. Esse padrão de replicação de um a quatro é, na verdade, como dar um único nome de área de grade se traduz em fazer com que o item de grade preencha essa área. Os seguintes são equivalentes:

```
área de grade: rodapé;
área de grade: rodapé / rodapé / rodapé / rodapé;
```

Agora, lembre-se do comportamento discutido na seção anterior sobre coluna de grade e linha de grade: se o nome de uma linha de grade corresponder ao nome de uma área de grade, ele será traduzido em uma variante -start ou -end , conforme apropriado. Isso significa que o exemplo anterior é traduzido para o seguinte:

```
grid-area: footer-start / footer-start / footer-end / footer-end;
```

E é assim que um único nome de área de grade faz com que um elemento seja colocado na área de grade correspondente.

Sobreposição de item de grade

Uma coisa que temos sido muito cuidadosos em fazer em nossos layouts de grid até agora é evitar excesso de volta. Um pouco como o posicionamento, é absolutamente (entendeu?) possível fazer com que os itens da grade se sobreponham uns aos outros. Tomemos um caso simples, ilustrado na Figura 37:

```
#grid {display: grade;
      linhas de modelo de grade: 50% 50%;
     grid-template-columns: 50% 50%;}
.box01 {grid-area: 1 / 1 / 2 / 3;}
.box02 {grid-area: 1 / 2 / 3 / 2 ;}
```

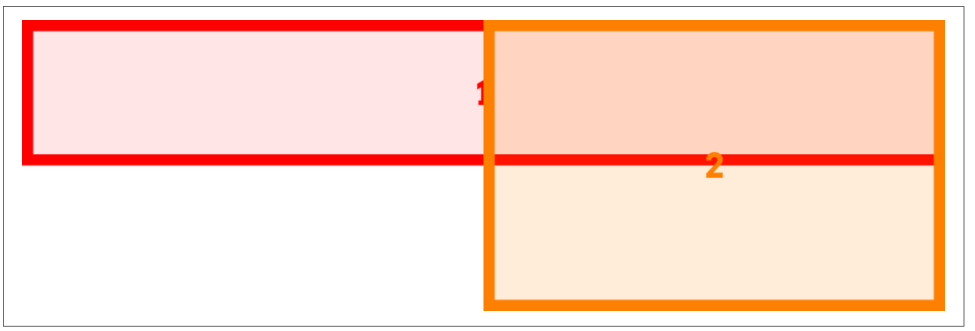


Figura 37. Itens de grade sobrepostos

Graças aos números de grade que foram fornecidos, os dois itens de grade se sobrepõem na célula de grade superior direita. O que está em cima do outro depende do comportamento de camadas discutido mais tarde, mas, por enquanto, apenas tome como dado que eles fazem camada ao se sobrepor.

Overlap não se restringe a situações que envolvem números brutos do grid, é claro. No seguinte caso de baixo, a barra lateral e o rodapé se sobreporão, como mostra a Figura 38. (Supondo que o rodapé venha mais tarde do que a barra lateral na marcação, então, na ausência de outros estilos, o rodapé estará na parte superior da barra lateral.)

```
#grid {display: grade; grid-
      template-areas:
           "cabeçalho do
           cabeçalho"
           "conteúdo da barra
           lateral" "rodapé
           do rodapé";}
#header {grid-area: header;}
#sidebar {grid-area: sidebar / sidebar / footer-end / sidebar;} #footer
{grid-area: rodapé;}
```

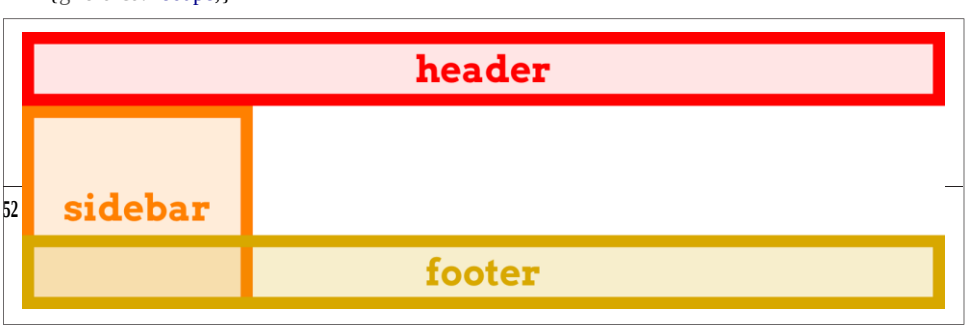


Figura 38. Sobreposição da barra lateral e do rodapé

Eu trago isso à tona em parte para alertá-lo sobre a possibilidade de sobreposição, e também para servir como uma transição para o próximo tópico. É um recurso que diferencia o layout da grade da posição, na medida em que às vezes pode ajudar a evitar a sobreposição: o conceito de *fluxo* de *grade*.

Fluxo de grade

Na maior parte, colocamos explicitamente itens de grade na grade. Se os itens não forem colocados explicitamente, eles serão automaticamente colocados na grade. Após o fluxo de grade em vigor, um item é colocado na primeira área que se encaixará nele. Ocaso st simples é simplesmente preencher uma faixa de grade em sequência, um item de grade após o outro, mas as coisas podem ficar muito mais complexas do que isso, especialmente se houver uma mistura de itens de grade colocados explicitamente e automaticamente – o último deve trabalhar em torno do primeiro.

Taqui são principalmente dois modelos de fluxo de grade, *linha* primeiro e *coluna primeiro*, embora você possa aprimorar qualquer um deles especificando um fluxo denso. Tudo isso é feito com o fluxo automático da grade de propriedades.

fluxo

Valores	[linha | coluna] || denso |
Valor inicial:	rem
Aplica-se	Contêineres de
Herdada:	Nã
Valor calculado :	Conforme

Para ver como esses valores funcionam, considere a seguinte marcação:

```
<ol id="grade">
<li>1</li>
<li>2</li>
<li>3</li>
<li>4</li>
<li>5</li>
</ol>
```

Para essa marcação, vamos aplicar os seguintes estilos:

```
#grid {display: grade;  largura: 45em;  altura: 8em;
      grid-auto-flow:  linha;}
#grid  li {grid-row: auto;  grid-column: auto;}
```

Supondo uma grade com uma linha de coluna a cada 15 ems e uma linha de linha a cada 4 ems, obtemos o resultado mostrado na Figura 39.

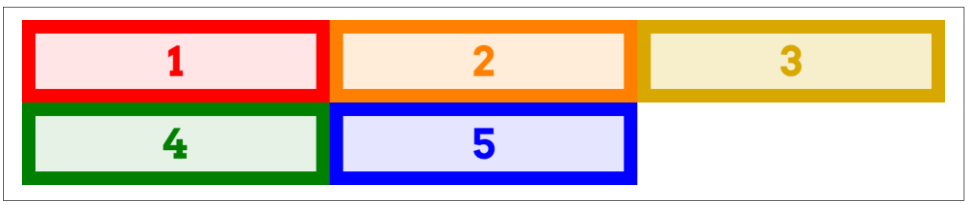


Figura 39. Fluxo de grade orientado a linha

Isso provavelmente parece bastante normal, o mesmo tipo de coisa que você obteria se flutuasse todas as caixas, ou se todas elas fossem blocos embutidos. É por isso que linha é o valor padrão. Agora, vamos tentar alternar o valor do fluxo automático de grade para coluna, como mostra a Figura 40:

```
#grid {display: grade;  largura: 45em;  altura: 8em;
      grid-auto-flow:  coluna;}
#grid  li {grid-row: auto;  grid-column: auto;}
```

Assim, com a linha grid-auto-flow:, cada linha é preenchida antes de começar na próxima linha. Com a coluna grid-auto-flow:, cada coluna é preenchida primeiro.

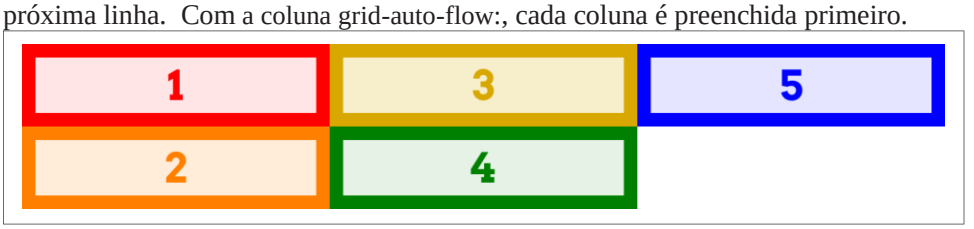


Figura 40. Fluxo de grade orientado a colunas

O que precisa ser enfatizado aqui é que os itens da lista não foram explicitamente dimensionados. Por padrão, eles foram redimensionados para anexar às linhas de grade definidas. Isso pode ser substituído atribuindo um dimensionamento explícito aos elementos. Por exemplo, se fizermos com que os itens da lista tenham 7 ems de largura e 1,5 ems de altura, obteremos o resultado mostrado na Figura 41:

```
#grid {display: grade;  largura: 45em;  altura: 8em;
      grid-auto-flow:  coluna;}
#grid  li {grid-row: auto;    grid-coluna: auto;
      largura: 7em;  altura: 1.5em;}
```

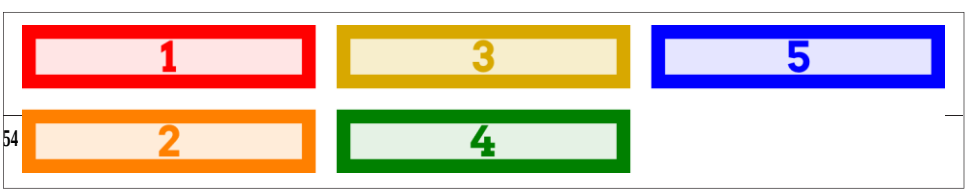


Figura 41. Itens de grade explicitamente dimensionados

Se você comparar isso com a figura anterior, verá que os itens de grade correspondentes começam no mesmo lugar em cada figura; eles simplesmente não terminam nos mesmos lugares. Isso ilustra que o que é realmente colocado no fluxo de grade são as áreas de grade, às quais os itens grid são então anexados.

Isso é importante ter em mente se você auto-fluxo de elementos que são mais largos do que a coluna atribuída ou mais altos do que a linha atribuída, como pode muito facilmente acontecer ao transformar imagens ou outros elementos de tamanho intrínseco em itens grid. Digamos que queremos colocar um monte de imagens, cada uma de um tamanho diferente, em uma grade configurada para ter uma linha de col- umn a cada 50 pixels horizontais e uma linha de linha a cada 50 pixels verticais. Essa grade é ilustrada na Figura 42, juntamente com os resultados do fluxo de uma série de imagens para essa grade por linha ou coluna.

```
#grid {display: grade;
      linhas de modelo de grade: repeat(3, 50px);
      grid-template-columns: repeat(4, 50px);
      grade-auto-linhas: 50px;
      grade-auto-colunas: 50px;
}
img {grid-row: auto; grid-column: auto;}
```

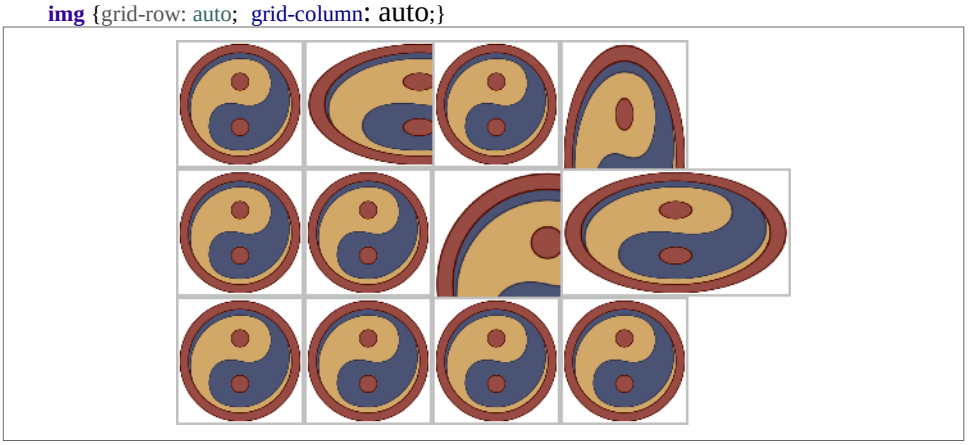


Figura 42. Imagens que fluem em grades

Percebeu como algumas das imagens se sobrepõem a outras? Isso porque cada imagem é anexada à próxima linha de grade no fluxo, sem levar em conta a presença de outros itens de grade. Não configuramos imagens para abranger mais de uma faixa de grade quando eles precisavam, então a sobreposição ocorreu.

Isso pode ser gerenciado com nomes de classe ou outros identificadores. Poderíamos classificar as imagens como altas ou largas (ou ambas) e especificar que elas obtivessem mais faixas de grade. Aqui estão alguns CSS para adicionar ao exemplo anterior, com o resultado mostrado na Figura 43:

```
img.wide {grid-column: auto / span 2;}
img.tall {grid-row: auto / span 2;}
```

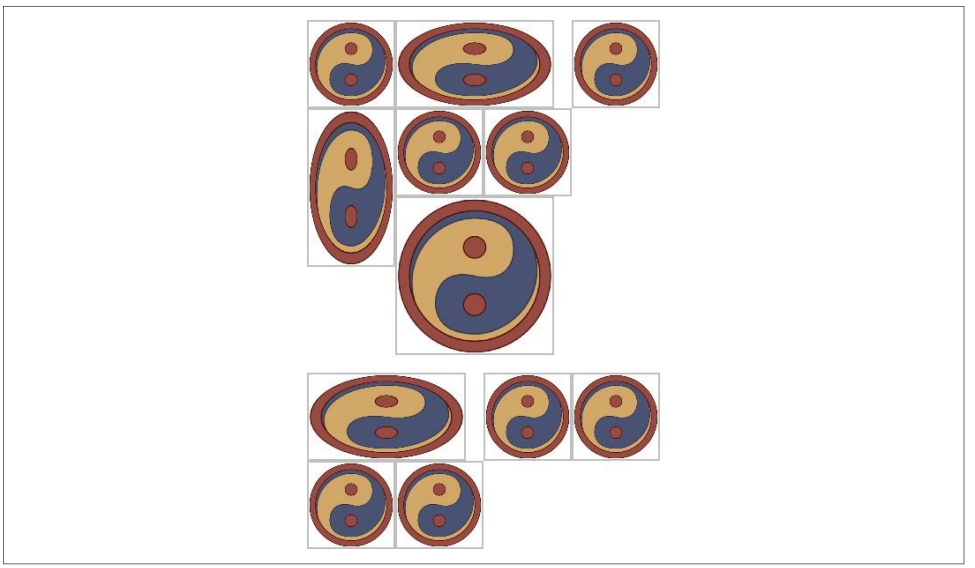


Figura 43. Dando às imagens mais espaço de rastreamento

Isso faz com que as imagens continuem se espalhando pela página, mas não há sobreposição.

No entanto, observe como há lacunas nessa última grade? Isso aconteceu porque a colocação de alguns itens de grade através das linhas de grade não deixou espaço suficiente para outros itens no fluxo. Para ilustrar isso, e os dois padrões de fluxo, mais claramente, vamos tentar um exemplo com caixas numeradas (Figura 44).

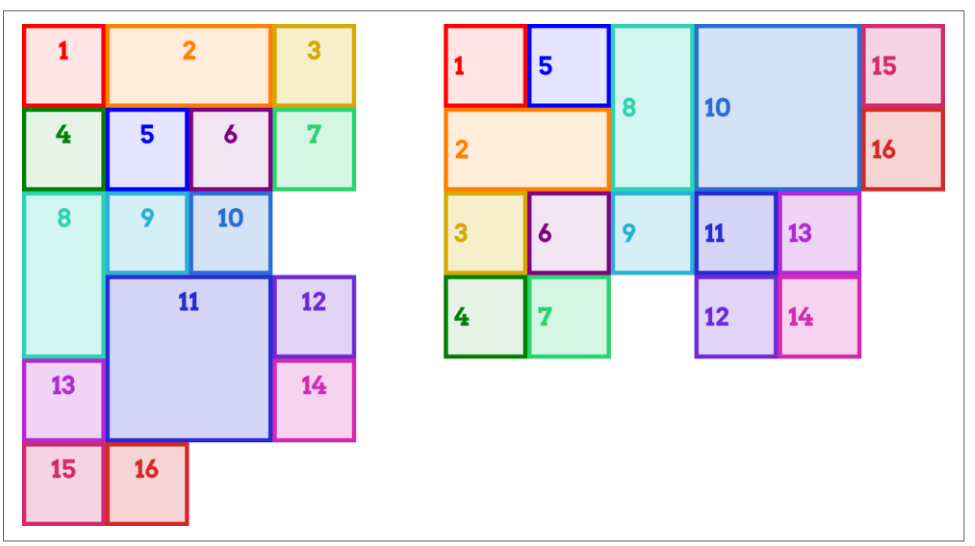


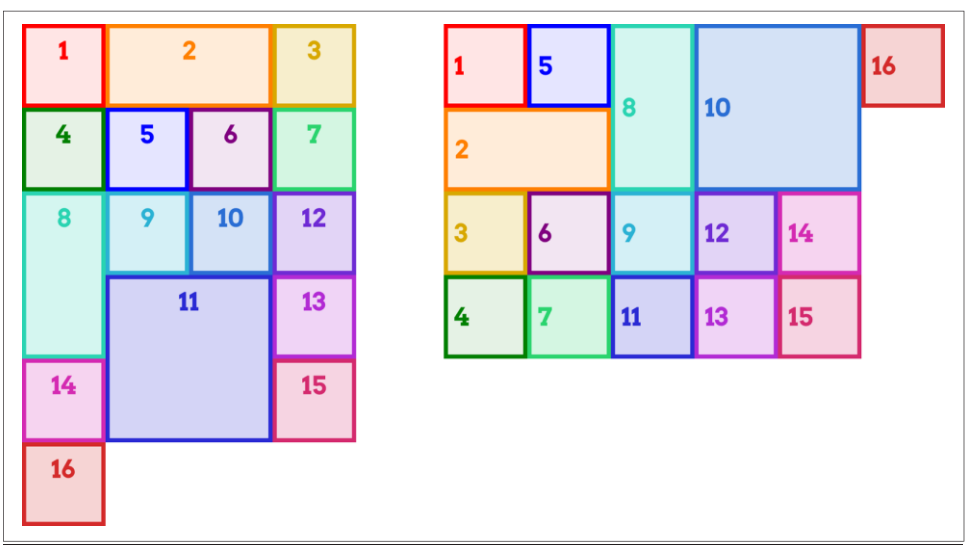
Figura 44. Ilustrando padrões de fluxo

Siga pelas linhas da primeira grade, contando junto com os números. Neste fluxo particular, os itens da grade são dispostos quase como se fossem flutuadores para a esquerda. Quase, mas não é bem assim: observe que o item 13 da grade está, na verdade, à esquerda do item 1 1 da grade. Isso nunca aconteceria com flutuadores, mas pode acontecer com o fluxo da rede. A maneira como o fluxo de linha (se é que podemos chamá-lo assim) funciona é que você atravessa cada linha da esquerda para a direita e, se houver espaço para um item de grade, você o coloca lá. Se uma célula de grade tiver sido ocupada por outro item de grade, pule-a. Assim, a célula ao lado do item 10 não foi preenchida, porque não havia espaço para o item 11. O item 13 foi para a esquerda do item 11 porque havia espaço para ele quando a fila foi alcançada.

Os mesmos mecanismos básicos valem para o fluxo de colunas, exceto que, neste caso, você trabalha de cima para baixo. Assim, a célula abaixo do item 9 está vazia porque o item 10 não caberia lá. Ele foi para a próxima coluna e abrangeu quatro células de grade. Os itens depois dele, since eles eram apenas uma célula de grade em tamanho, preenchido nas células depois dele em ordem de coluna.

O fluxo de grade funciona da esquerda para a direita, de cima para baixo em idiomas que têm esse padrão de escrita. Em idiomas da direita para a esquerda, como árabe e hebraico, o fluxo orientado a linhas seria da direita para a esquerda, não da esquerda para a direita.

Se você estava apenas agora desejando uma maneira de embalar itens de grade o mais densamente possível, independentemente de como isso afetou o pedido, uma boa notícia: você pode! Basta adicionar a palavra-chave densa ao seu valor de fluxo automático de grade, e é exatamente isso que acontecerá. Podemos ver o resultado na Figura 45, que mostra os resultados do grid-auto-flow: linha densa e grid-auto-flow:



coluna densa lado a lado.
Figura 45. Ilustrando padrões de fluxo densos

Na primeira grade, o item 12 aparece na linha acima do item 11 porque havia uma célula que se encaixava nele. Pelo mesmo motivo, o item 11 aparece à esquerda do item 10 na segunda grade.

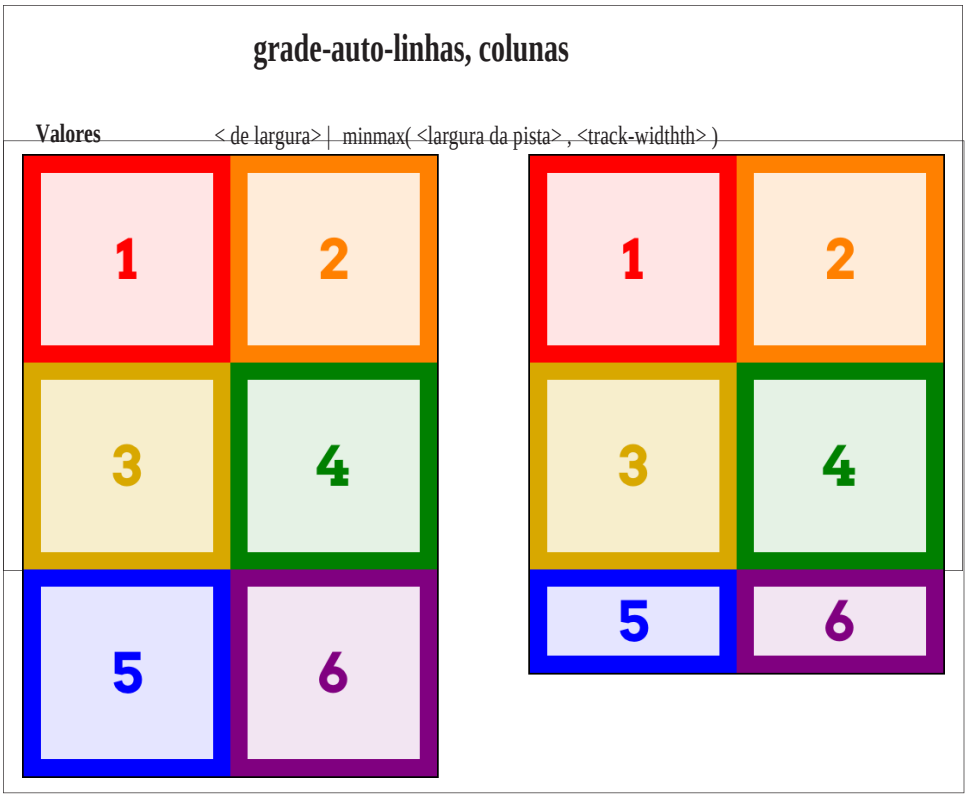
Com efeito, o que acontece com o fluxo de grade denso é que, para cada item de grade, o navegador verifica *toda* a grade na direção de fluxo dada (linha ou coluna), começando pelo ponto de partida do fluxo (o canto superior esquerdo, em idiomas LTR - da esquerda para a direita -), até encontrar um local onde esse item de grade se encaixará. Isso pode tornar coisas como galerias de fotos mais compactas, umnd funciona muito bem, desde que você não tenha uma ordem específica em que as imagens precisam aparecer.

Agora que exploramos o fluxo de grade, tenho uma confissão a fazer: para fazer com que os últimos dois itens de grade pareçam certos, incluí alguns CSS que não mostrei a você. Sem ele, os itens pendurados na borda da grade teriam parecido um pouco diferentes dos outros itens – muito mais curtos no fluxo orientado a linha e muito mais próximos no fluxo orientado a colunas. Veremos o porquê e o CSS que usei na próxima seção.

Linhas de grade automáticas

Até agora, vimos quase inteiramente itens de grade colocados em uma grade que foi explicitamente definida. Mas na última seção, tivemos situações em que os itens de grade saíram da borda da grade explicitamente definida. O que haparece quando um item de grade sai da borda? Linhas ou colunas são adicionadas conforme necessário para satisfazer as diretivas de layout dos itens em questão (consulte "A Grade Implícita" na página 37). Portanto, se um item com uma extensão de linha de 3 for adicionado após o final de uma grade orientada a linha, três novas linhas serão adicionadas após a grade explícita.

Por padrão, essas linhas adicionadas automaticamente são o tamanho mínimo absoluto necessário. Se você quiser exercer um pouco mais de controle sobre seu dimensionamento, então grade-auto-rows e grid-auto-columns são para você.



Para qualquer faixa de linha ou coluna criada automaticamente, você pode fornecer um único tamanho de faixa ou um par minto de tamanhos de faixa. Vamos dar uma olhada em uma versão reduzida do exemplo de fluxo de grade da seção anterior: vamos configurar uma grade 2 x 2 e tentar colocar cinco itens nela. Na verdade, vamos fazer isso duas vezes: uma vez com linhas automáticas de grade e outra sem, como ilustrado na Figura 46:

```
.grid {display: grade;
      linhas de modelo de grade: 80px
      80px; grid-template-columns: 80px
      80px;}
#g1 {grid-auto-rows: 80px;}
```

Como você pode ver, sem dimensionar a linha criada automaticamente, o item de grade é colocado em uma linha que é exatamente tão alta quanto o conteúdo do item de grade, e não um pixel a mais. Ainda é

tão larga quanto a coluna em que é colocada, porque ela tem um tamanho (80px). A linha, sem uma altura explícita, tem como padrão automático, com o resultado mostrado.

Figura 46. Grades com e sem dimensionamento automático de linha

Se o virarmos para colunas, os mesmos princípios básicos se aplicam (veja a Figura 47):

```
.grid {display: grade; grid-auto-flow: coluna;
    linhas de modelo de grade: 80px 80px;
    grid-template-columns: 80px 80px;} #g1
{grid-auto-columns: 80px;}
```

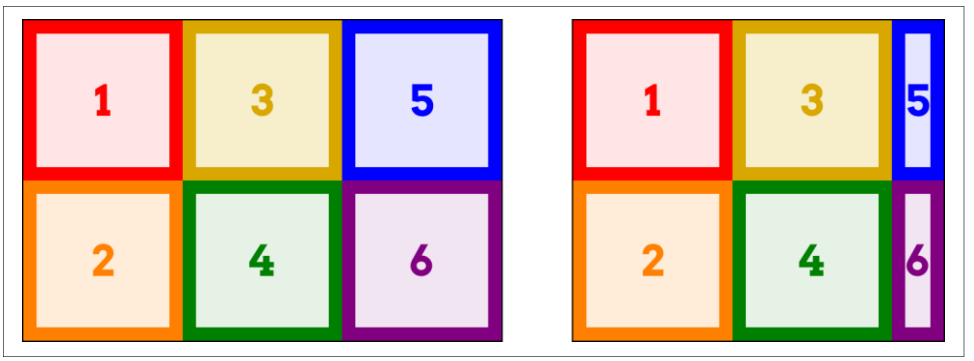


Figura 47. Grades com e sem dimensionamento automático de colunas

Nesse caso, como o fluxo é orientado a colunas, o último item de grade é colocado em uma nova coluna após o final da grade explícita. Na segunda grade, onde não há colunas automáticas de grade, esse quinto item é tão alto quanto sua linha (80px), mas tem uma largura automática, por isso é tão largo quanto precisa ser e não mais largo. É claro que, se um sexto item fosse adicionado e tivesse um conteúdo mais amplo, a coluna seria dimensionada para se ajustar a esse conteúdo, ampliando assim o quinto item.

Então, agora você sabe o que eu usei nas figuras de fluxo automático de grade na seção anterior: eu silenciosamente fiz as linhas automáticas e as colunas automáticas do mesmo tamanho que as colunas explicitamente dimensionadas, a fim de não ter o último par de itens parece estranho. Vamos trazer de volta uma dessas figuras, só que desta vez os estilos grid-auto-rows e grid-auto-columns serão removidos . Como você pode ver na Figura 48, os últimos itens em cada grade são mais curtos ou mais estreitos do que o resto, devido à falta de dimensionamento da via automática.

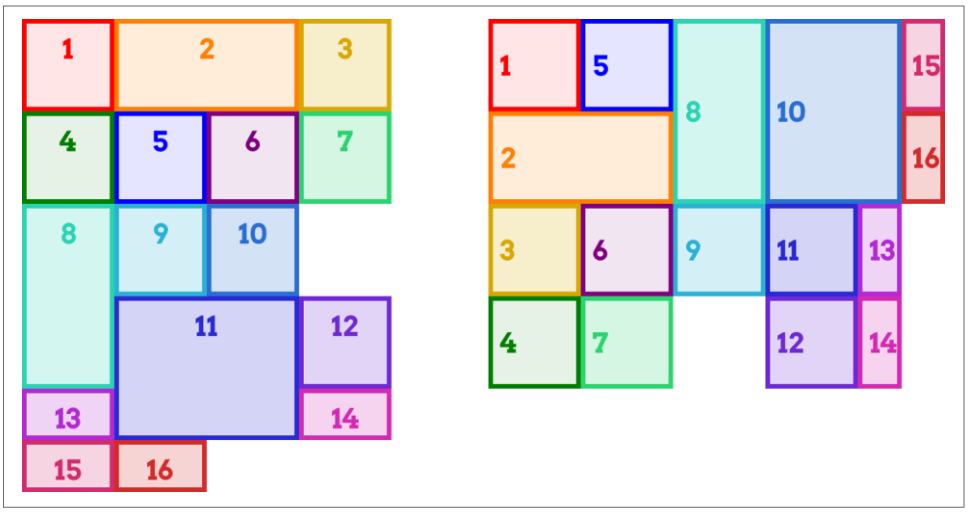


Figura 48. Uma figura anterior *com o dimensionamento da faixa automática removido*

E agora você sabe... o resto da história.

A taquigrafia da grade

Finalmente, chegamos à grade de propriedades abreviadas. Pode surpreendê-lo, no entanto, porque não é como outras propriedades taquigráficas.

gra

Valores	nenhum | | de sub-grade | [<grid-template-rows> / <grid-template-columns>] | [<linha- nomes>? < > < tamanho da faixa>? <nomes de linha>?]+ [/ <track-list>]? | [<fluxo automático da grade> [<grid-auto-rows> [/ <grade-auto-colunas>]?]?]]
Valor inicial:	Ver propriedades |
Aplica-se	Contêineres de
Herdada:	Nã
Valor calculado:	Consulte propriedades |

A sintaxe é um pouco indutora de enxaqueca, admito, mas vamos passar por ela um pedaço de cada vez.

Vamos ao elefante na sala imediatamente: a grade permite que você defina um modelo de grade *ou* defina o fluxo da grade e o dimensionamento da faixa

automática em uma sintaxe compacta. Você não pode fazer as duas coisas ao mesmo tempo.

Além disso, o que você não definir é redefinido para seus padrões, como é normal para uma propriedade abreviada. Portanto, se você definir o modelo de grade, os racks de fluxo e t automático serão retornados aos seus valores padrão. Isso inclui calhas de grade, um tópico que ainda nem abordamos . Você não pode definir as calhas com grade, mas ele irá redefini-las de qualquer maneira.

Sim, isso é intencional. Não, não sei porquê.

Então, vamos falar sobre a criação de um modelo de grade usando grade. Os valores podem ficar diabolicamente complexos e assumir alguns padrões fascinantes, mas podem ser muito úteis. Como exame, a seguinte regra é equivalente ao conjunto de regras que a segue:

```
grade:
    "header header  header" 3em ". barra
    lateral de conteúdo . " 1fr
    "rodapé rodapé rodapé  rodapé" 5em /
    2em 3 fr minmax (10em,1 fr) 2em;

grid-template-areas:
    "cabeçalho  cabeçalho cabeçalho"
    ". barra lateral de conteúdo . "
    "rodapé  rodapé rodapé"; grid-
template-rows: 3em 1fr 5em;
grid-template-columns: 2em 3 fr minmax(10em,1 fr) 2em;
```

Observe como o valor de grid-template-rows é dividido e espalhado pelas cadeias de caracteres de grid-template-areas. É assim que o dimensionamento de linha é tratado na grade quando você tem cadeias de caracteres de área de grade presentes. Tire essas cordas e você termina com o seguinte baixo:

```
grade:
    3em 1 fr 5em / 2em 3 fr minmax (10em, 1fr) 2em;
```

Em outras palavras, as faixas de linha sãoseparadas por um solidus (/) das trilhas da coluna.

Lembre-se de que, com a grade, os atalhos não declarados são redefinidos para seus padrões. Isso significa que as duas regras a seguir são equivalentes:

```
#layout {display: grade;
    grade: 3em 1 fr 5em / 2em 3 fr minmax ( 10em,1 fr) 2em;}

#layout {display: grade;
    grade: 3em 1 fr 5em / 2em 3 fr minmax (10em, 1fr)
    2em; grade-auto-linhas: auto;
     grid-auto-columns: auto;
    grid-auto-flow: linha;}
```

Portanto, certifique-se de que sua declaração de grade venha antes de qualquer outra coisa relacionada à definição da grade. Isso significa que, se quiséssemos um fluxo de coluna denso, escreveríamos algo assim:

```
#layout {display: grade;
    grade: 3em 1 fr 5em / 2em 3 fr minmax (10em, 1fr)
    2em; grid-auto-flow: coluna densa;}
```

Agora, vamos trazer as áreas de grade nomeadas de volta *e* adicionar alguns nomes de linha de grade de linha extra à mistura. Uma linha de grade nomeada que vai *acima* de uma faixa de linha é escrita *antes* da cadeia de caracteres, e uma linha de grade que vai *abaixo* da faixa de linha vem *depois* da cadeia de caracteres and qualquer faixa siz- ing. Então, digamos que queremos adicionar main-start e main-stop acima e abaixo da linha intermediária e page-end na parte inferior:

```
grade:
    "cabeçalho cabeçalho cabeçalho cabeçalho" 3em
    [início principal] ". barra lateral de conteúdo ". 1fr [main-
    stop] "rodapé rodapé rodapé rodapé" 5em [page-end]
    /
    2em 3 frminmax (10em, 1fr) 2em;
```

Isso cria a grade mostrada na Figura 49, com as linhas de grade nomeadas implicitamente criadas (por exemplo, início de rodapé) junto com as linhas de grade explicitamente nomeadas que escrevemos no CSS.

Você pode ver como a grade pode ficar muito complicada muito rapidamente. É um imposto de sintaxe muito poderoso, e é surpreendentemente fácil de se acostumar depois de ter tido apenas um pouco de prática. Por outro lado, também é fácil errar e fazer com que todo o valor seja inválido, evitando assim a aparência de qualquer grade.

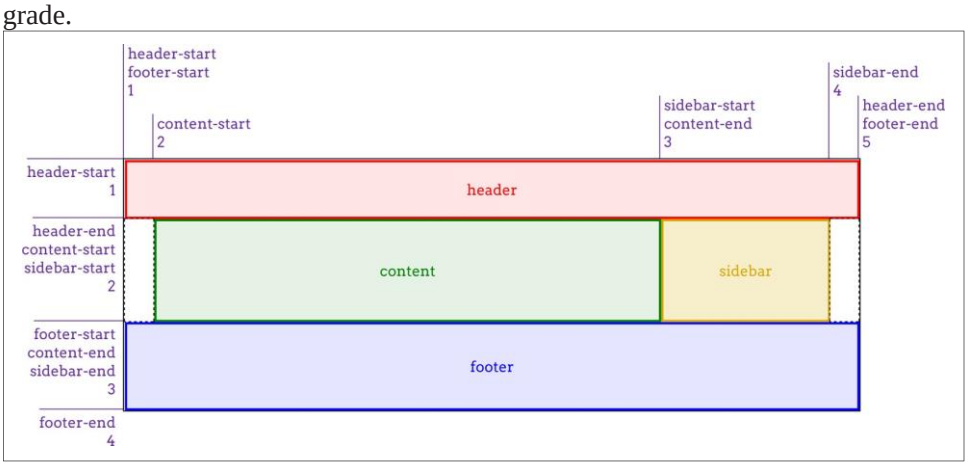


Figura 49. Criando uma grade *com* a *abreviatura de grade*

Para o outro uso da grade, é uma fusão de grid-auto-flow, grid-auto-rows, e grid-auto-columns. As seguintes regras são equivalentes:

```
#layout {grid-auto-flow: linhas densas;
    grade-auto-linhas: 2em;
    grid-auto-colunas: minmax(1em,3em);}
```

```
#layout {grade: linhas densas 2em / minmax(1em,3em);}
```

Isso certamente é muito menos digitação para o mesmo resultado! Mas, mais uma vez, tenho que lembrá-lo: se você escrever isso, todas as propriedades de faixa de coluna e linha serão definidas para seus padrões. Assim, as seguintes regras são equivalentes:

```
#layout {grade: linhas densas 2em / minmax(1em,3em);}

#layout {grade: linhas densas 2em / minmax(1em,3em);
         grid-template-rows: auto;
         grid-template-columns: auto;}
```

Então, mais uma vez, é importante garantir que sua taquigrafia venha antes de qualquer vínculo adequado que, de outra forma, poderia substituir.

Sub-grades

Há outro valor possível para a grade, que é a subgrade. Pode ser usado algo assim:

```
#grid {display: grade;
       grade: repetir (auto-enchimento, 2em) / repetir (10, 1% 8% 1%);}
.module {display: grade;
       grade: subgrade;}
```

O que acontece dentro de cada elemento do módulo é que seus itens de grade (ou seja, seus ele-mentos filhos) usam a grade definida por #grid para se alinharem.

Isso é potencialmente muito útil, porque você pode imaginar ter um módulo que abrange três dos padrões de coluna de seu pai e contém elementos filho que são alinhados e dispostos usando a grade "mestre". Isso é ilustrado na Figura 50.

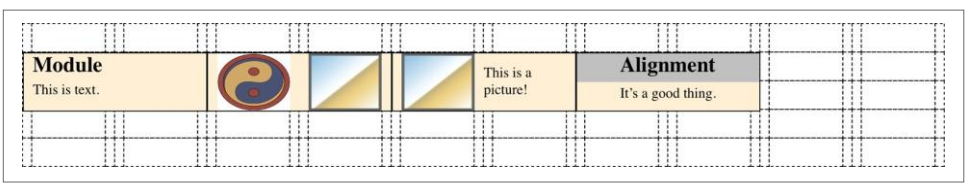


Figura 50. Alinhando itens subgradeados

O problema é que, no momento em que escrevo, a subgrade é um recurso "em risco" do layout da grade e pode ser descartada completamente. É por isso que classifica apenas esta pequena seção, em vez de um exame abrangente de minério.

Abrindo espaços de grade

Até agora, vimos muitos itens de grade emperrados uns contra os outros, sem espaço entre eles. Há várias maneiras de mitigar isso, como falaremos nesta seção, começando com calhas.

Calhas de grade (ou lacunas)

Simplificando, uma *calha* é um espaço entre duas faixas de grade. É criado como se expandisse a linha de grade entre eles para ter largura real. É muito parecido com o espaçamento entre bordas no estilo da tabela, tanto porque cria espaço entre as células de grade quanto porque você pode definir apenas um único valor de espaçamento para cada eixo, por meio das propriedades grid-row-gap e grid-column-gap.

lacuna de linha de grade, lacuna	
Valores	<
Valor inicial:	0
Aplica-se	Contêineres de
Herdada:	Nã
Valor calculado:	micrômetro comprimento absoluto

Logo de cara: como mostra a sintaxe do valor, você pode fornecer apenas um comprimento para essas propriedades; o que é menos claro é que os comprimentos devem ser não-negativos. Não é possível fornecer uma porcentagem, um fracionário value via fr, nem um minmax de algum tipo. Se você quiser que suas colunas sejam separadas por 1 em, então é bastante simples: grid-column-gap: 1em. Isso é praticamente o mais chique possível. Todas as colunas da grade serão separadas por 1 em, conforme ilustrado na Figura 51.

```
#grid {display: grade;
      grid-template-rows: 5em 5em;
       grid-template-columns: 15% 1 fr 1 fr;
      grid-column-gap: 1em;}
```

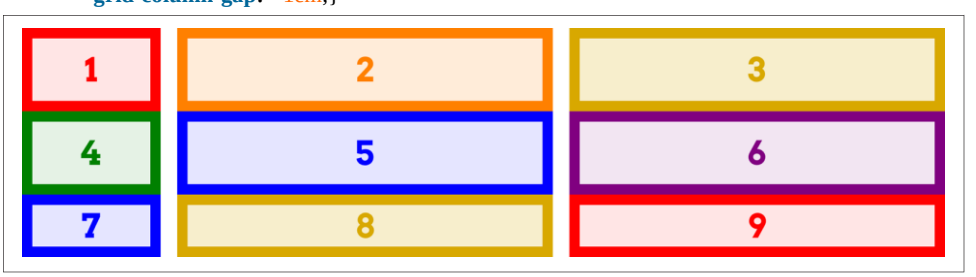


Figura 51. Criando calhas de coluna

Em termos de dimensionamento das faixas em uma grade, as calhas são tratadas

como se fossem faixas de grade. Assim , dados os seguintes estilos, as linhas de grade fracionária terão 140 pixels de altura cada.

```
#grid {display: grade; altura: 500px;
    grid-template-rows: 100px 1 fr 1 fr 75px;
    grid-row-gap: 15px;}
```

Obtemos 140 pixels para a altura de cada linha de fração porque há um total de 500 pixels de altura. A partir disso, subtraímos as duas faixas de linha (100 e 75) para obter 325. A partir desse resultado, subtraímos as três calhas de 15 pixels, que totalizam 45 pixels; isso rende 280 pixels. Isso dividido ao meio (porque as linhas fracionárias têm frações iguais) nos dá 140 pixels cada. Se o valor da calha fosse aumentado para 25px, as linhas fracionárias teriam 250 pixels para dividir entre elas, tornando cada uma com 125 pixels de altura.

É claro que o dimensionamento da pista pode ser muito mais complicado do que isso; o exemplo usou todos os pixels porque torna a matemática simples. Você sempre pode misturar unidades como quiser, incluindo minmaxing suas faixas de grade reais. Este é um dos principais pontos fortes do grid layout.

As calhas de grade podem ser alteradas de seu tamanho declarado pelos efeitos de alinhamento de conteúdo e justificação-conteúdo. Isso será abordado em uma seção posterior.

Existe, como você já deve ter suspeitado, uma abreviação que combina comprimentos de intervalo de linha e coluna em uma única propriedade.

lacuna

Valores	<grid-row-gap> <grid-column-gap>
Valor inicial:	0 0
Aplica-se	Contêineres de
Herdada:	Nã
Valor calculado :	Conforme

Não há muito mais a dizer do que isso, na verdade: forneça dois comprimentos não negativos, e você terá definido as calhas de linha e calhas de coluna, nessa ordem. Veja um exemplo, como mostra a Figura 52:

```
#grid {display: grade;
```

```
grid-template-rows: 5em 5em;
 grid-template-columns: 15% 1 fr 1 fr;
grid-gap: 12px 2em;}
```

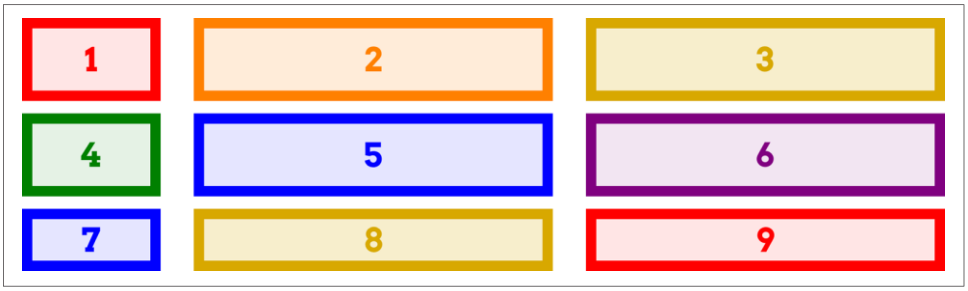


Figura 52. Definindo calhas de grade

Itens de grade e o modelo de caixa

Agora podemos criar uma grade, anexar itens à grade e até mesmo criar calhas entre as trilhas da grade. Mas o que acontece se estilizarmos o elemento que está ligado à grade com, digamos, margens? Ou se está absolutamente posicionado? Como essas coisas interagem coma grade?

Vamos tomar as margens primeiro. O princípio básico no trabalho é que um elemento é anexado à grade por suas bordas de margem. Isso significa que você pode empurrar as partes visíveis do elemento para dentro da área de grade que ele ocupa definindo margens positivas — e puxá-lo para fora com margens negativas. Por exemplo, esses estilos terão o resultado mostrado na Figura 53:

```
#grid {display: grade;
      grid-template-rows: repeat(2, 100px); grid-
     template-columns: repeat(2, 200px);}
.box02 {margem: 25px;}
.box03 {margem: -25px 0;}
```

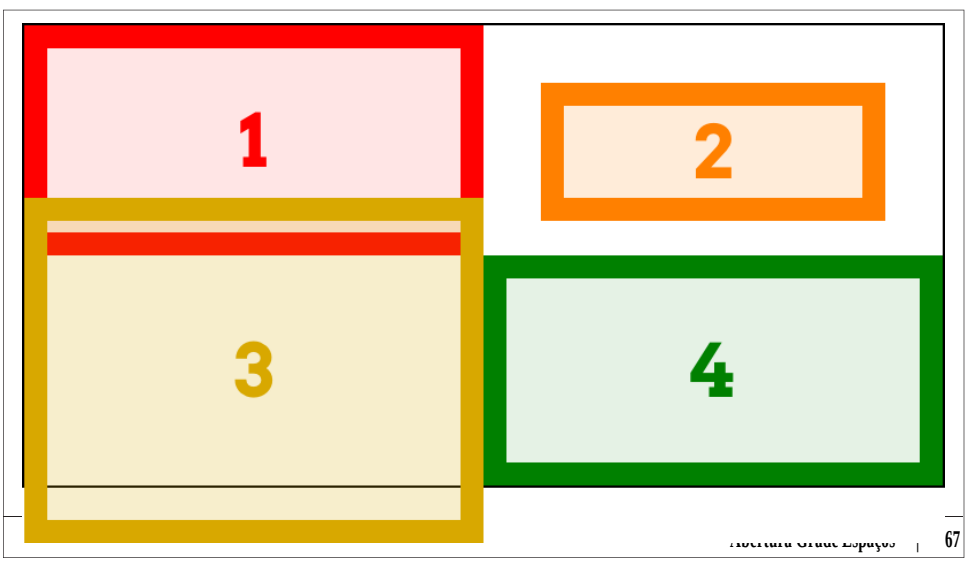


Figura 53. Itens de grade com margens

Isso funcionou como funcionou porque os itens tinham sua largura e altura definidas como automáticas, para que pudessem ser esticadas conforme necessário para que tudo funcionasse. Se a largura e/ou a altura tiverem valores não automáticos, elas acabarão substituindo as margens para fazer com que toda a matemática funcione. Isto é muito parecido com o que acontece com as margens direita e esquerda quando o dimensionamento elétrico é sobreposto: eventualmente, uma das margens é substituída.

Considere um elemento com os seguintes estilos colocados em uma área de grade de 200 pixels de largura por 100 pixels de altura:

```
. exel {largura: 150px; altura: 100px;
    preenchimento: 0; fronteira: 0;
    margem: 10px;}
```

Atravessando o elemento primeiro, ele tem 10 pixels de margem para cada lado, e sua largura é de 150px, dando um total de 170 pixels. Algo tem que dar e, neste caso, é a margem direita (em idiomas da esquerda para a direita), que é alterada para 40px para fazer com que a coisa funcione – 10 pixels na margem esquerda, 150 pixels na caixa de conteúdo e 40 pixels na caixa de conteúdo a margem direita é igual aos 200 pixels da largura da área da grade.

No eixo vertical, a margem inferior é redefinida para -10px. Isso compensa a margem superior e a altura do conteúdo, totalizando 110 pixels, quando a área da grade tem apenas 100 pix- els de altura.

As margens nos itens de grade são ignoradas ao calcular os tamanhos da trilha da grade. Isso significa que, independentemente de quão grande ou pequena você faça as margens de um item de grade, isso não alterará o tamanho de uma coluna de conteúdo mínimo, por exemplo, nem aumentará as margens em um item de grade fará com que as faixas de grade de tamanho fr mudem de tamanho.

Assim como no layout do bloco, você pode usar seletivamente as margens automáticas para decidir qual margem terá seu valor alterado para se ajustar. Suponha que quiséssemos que o item de grade se alinhasse à direita de sua área de grade. Ao definir a margem esquerda do item como automática, isso aconteceria:

```
. exel {largura: 150px; altura: 100px;
    preenchimento: 0; fronteira: 0;
    margem: 10px; margem esquerda: auto;}
```

Agora, o elemento adicionará 160 pixels para a margem direita e a caixa de conteúdo e, em seguida, dará a diferença entre isso e a largura da área da grade para a margem esquerda, já que foi explicitamente definido como automático. Isso tem o resultado

mostrado na Figura 54, onde há 10 pixels de margem em cada lado do item de exel, exceto a margem esquerda, que é (como acabamos de calcular) 40 pixels.

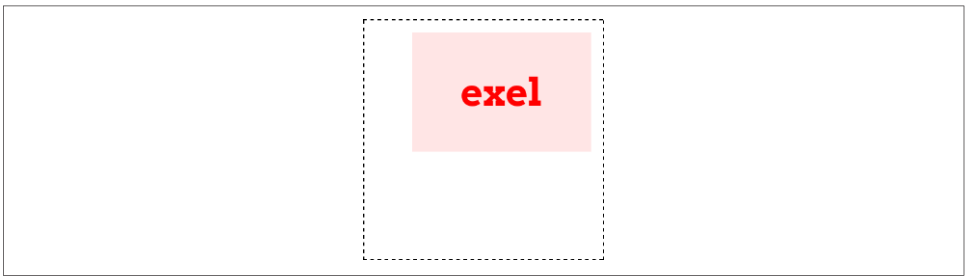


Figura 54. Usando margens automáticas para alinhar itens

Isso pode parecer familiar a partir do layout no nível do bloco, onde você pode usar as margens esquerda e direita automáticas para centralizar um elemento em seu bloco de contenção, desde que você tenha dado a ele uma largura explícita . Onde o layout da grade difere é que você pode fazer a mesma coisa no eixo vertical; ou seja, dado um elemento com uma altura absoluta, você pode verticalmente e cen- ter ele, definindo as margens superior e inferior para auto. A Figura 55 mostra uma variedade de efeitos de margem automática em elementos que naturalmente têm alturas e larguras explícitas: imagens.

```
.i01 {margem: 10px;}
.i02 {margem: 10px; margem esquerda: auto;}
.i03 {margin: auto 10px auto auto;}
.i04 {margem: auto;}
.i05 {margin: auto auto 0 0;}
.i06 {margem: 0 auto;}
```

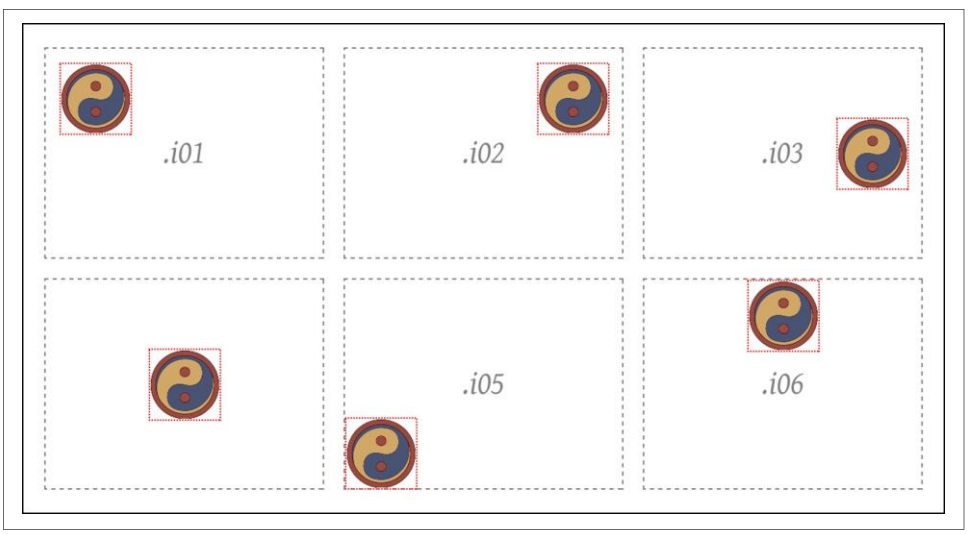


Figura 55. Vários alinhamentos de margem automática

Existem outras maneiras de alinhar itens de grade,

principalmente com propriedades como justify-self, que não dependem de ter valores explícitos de altura e largura. Estes serão abordados na próxima seção.

Isso é muito parecido com a forma como as margens e os tamanhos dos elementos operam quando os elementos estão absolutamente posicionados. O que nos leva à próxima pergunta: e se um item de grade *também* estiver absolutamente posicionado? Por exemplo:

```
. exel {grid-row: 2 / 4;     grade-coluna: 2 / 5;
    posição: absoluta;
    topo: 1em;  fundo: 15%;
    esquerda: 35px;  direita: 1rem;}
```

A resposta é, na verdade, bastante elegante: se você definiu inícios e finais da linha de grade, essa área de grade é usada como o bloco que contém o contexto e o posicionamento, e o item de grade é posicionado *dentro* desse contexto. Isso significa que as propriedades de deslocamento (top et al.) sãoconsideradas em relação à área de grade declarada. Assim, o CSS anterior teria o resultado mostrado na Figura 56.

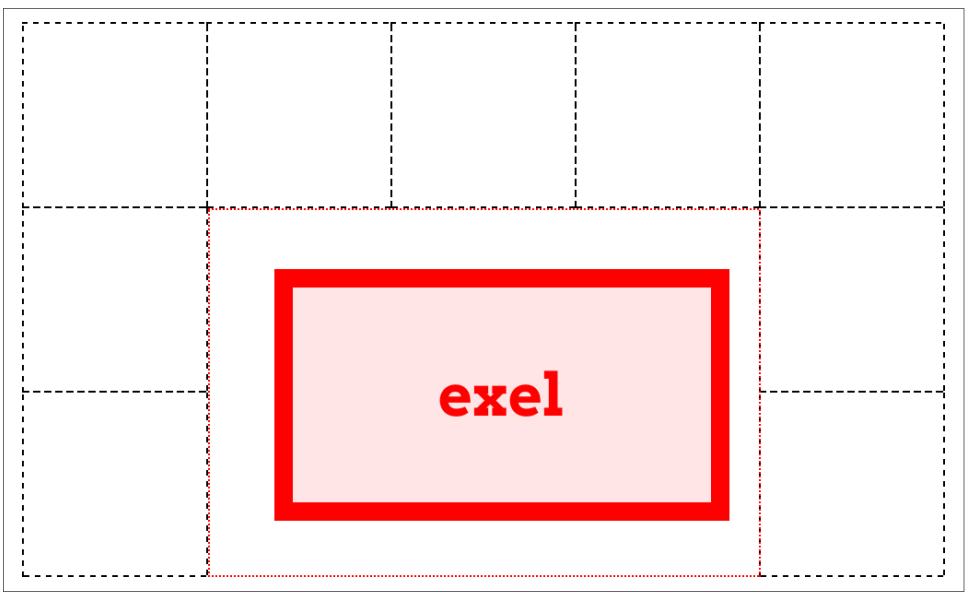


Figura 56. Posicionando absolutamente um item de grade

Tudo o que você sabe sobre elementos absolutamente posicionados em relação a deslocamentos, mar- gins, dimensionamento de elementos e assim por diante se aplica a esse contexto de formatação. É que neste caso, o contexto de formatação é definido por uma área de grade.

Há uma ruga que o posicionamento absoluto introduz: ele altera o comportamento do valor automático para propriedades de linha de grade. Se, por exemplo, você definir grid-column-end: auto para um item de grade absolutamente posicionado, a linha de grade final realmente criará uma linha de grade nova e especial que corresponda à

borda de preenchimento do próprio contêiner de grade. Isso é verdadeiro mesmo que a grade explícita seja menor que o contêiner de grade, como pode acontecer.

Para ver isso em ação, modificaremos o exemplo anterior da seguinte maneira, com o resultado mostrado na Figura 57:

```
. exel {grid-row: 1 / auto;      grade-coluna: 2 / auto;
      posição: absoluta;
      topo: 1em;  fundo: 15%;
      esquerda: 35px;  direita: 1rem;}
```

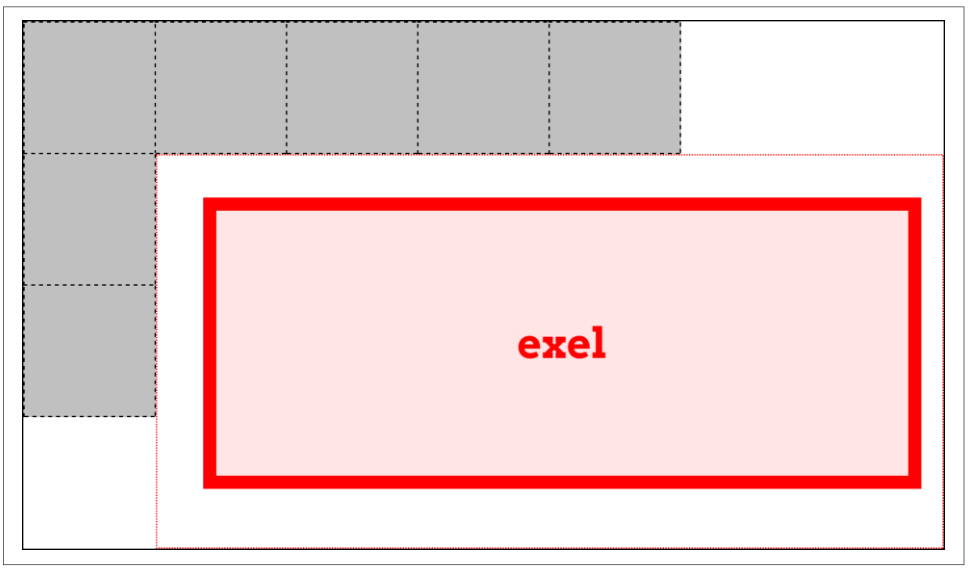


Figura 57. Valores automáticos e posicionamento absoluto

Observe como o contexto de posicionamento agora começa na parte superior do contêiner de grade e se estende até a borda direita do contêiner de grade, mesmo que a própria grade termine bem abaixo dessa borda.

Uma implicação desse comportamento é que, se você posicionar absolutamente um elemento que é um item de grade, mas não fornecer a ele nenhum valor inicial ou final de linha de grade, ele usará a borda de preenchimento interna do contêiner de grade como seu contexto de posicionamento. Ele faz isso sem ter que definir o contêiner de grade para a posição: relativo, ou qualquer um dos outros truques usuais para estabelecer um contexto de posicionamento.

Observe que os itens de grade absolutamente posicionados *não* participam da descoberta do dimensionamento da célula da grade e da pista. No quediz respeito ao layout grid , o item de grade posicionado não existe. Uma vez que a grade é configurada, o item de grade é posicionado em relação às linhas de grade que definem seu contexto de posicionamento.

No início de 2016, os navegadores não suportavam nenhum desses comportamentos de posicionamento absolutos. A única maneira de recriá-lo era posicionar relativamente o elemento que estabelece a área da grade e absolutamente colocar um elemento filho dentro dele. Foi assim que as figuras de posicionamento absoluto nesta seção foram criadas. O comportamento automático especial também não foi suportado.

Alinhamento e grades

Se você tem alguma familiaridade com o flexbox, provavelmente está ciente das várias propriedades de alinhamento e seus valores. Essas mesmas propriedades também estão disponíveis no layout da grade e têm efeitos muito semelhantes.

Primeiro, uma rápida atualização. As propriedades disponíveis e o que elas afetam estão somadas na Tabela 1.

Tabela 1. Justificar e alinhar valores

Propriedade	Alinha	Aplicado a
justificar-se	Um item de grade na direção embutida (horizontal)	Itens de grade
justify-items	Todos os itens de grade na direção embutida (horizontal)	Contêiner de grade
justificar-conteúdo	A grade inteira na direção embutida (horizontal)	Contêiner de grade
alinhar-se	Um item de grade na direção do bloco (vertical)	Itens de grade
itens de alinhamento	Todos os itens de grade na direção do bloco (vertical)	Contêiner de grade
alinhamento de conteúdo	Toda a grade na direção do bloco (vertical)	Contêiner de grade

Como mostra a Tabela 1, as várias propriedades justify-* alteram o alinhamento ao longo do eixo embutido — em inglês, essa será a direção horizontal. A diferença é se uma propriedade se aplica a um único item de grade, a todos os itens de grade em uma grade ou à grade inteira. Da mesma forma, as propriedades align-* afetam o alinhamento ao longo do eixo do bloco; em inglês, essa é a direção vertical.

Alinhando e justificando itens individuais

Émais fácil começar com as propriedades *-self, porque podemos ter uma grade mostrando valores de propriedade var-ious justify-self, enquanto uma segunda grade

mostra os efeitos desses mesmos valores quando usada por align-self. (Consulte a Figura 58.)

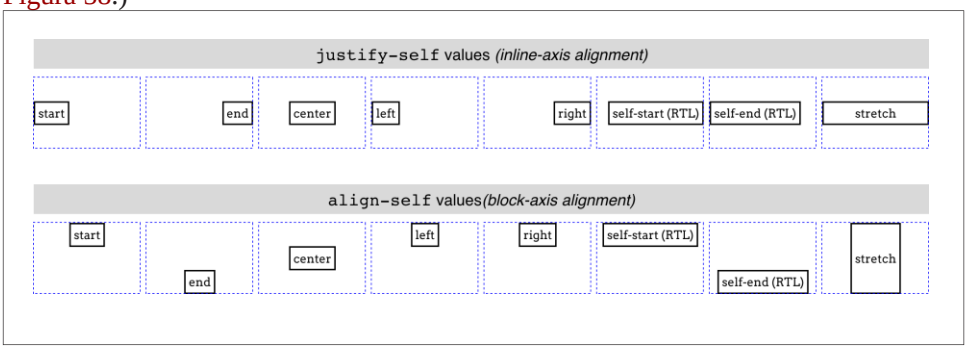


Figura 58. Auto-alinhamento nas direções em linha e de bloco

Cada item de grade na Figura 58 é mostrado com sua área de grade (a linha azul tracejada) e um rótulo que identifica o valor da propriedade que é aplicado a ele. Cada um merece um pouco de com- mentary.

Primeiro, porém, perceba que, para todos esses valores, qualquer elemento que não tenha uma largura ou altura explícita "encolherá" seu conteúdo, em vez de usar o comportamento default grid-item de preencher toda a área da grade.

start e end fazem com que o item de grade seja alinhado à borda inicial ou final de sua área de grade, o que faz sentido. Da mesma forma, o centro centraliza o item de grade dentro de sua área ao longo do eixo de alinhamento, *sem* a necessidade de declarar margens ou quaisquer outras propriedades, incluindo altura e largura.

esquerda e direita têm os resultados esperados para o alinhamento horizontal, mas se eles são aplicados aos elementos via align-self (que é o alinhamento vertical), eles são tratados como início.

auto-início e auto-fim são mais interessantes. O autoinício alinha um item de grade com a borda da área da grade que corresponde à borda *inicial do item de* grade. Assim, na Figura 58, as caixas de auto-início e auto-fim foram definidas para a direção: rtl. Isso os colocou para usar a direção do idioma da direita para a esquerda, o que significa que suas bordas iniciais eram suas bordas direitas e suas bordas finais eram as esquerdas. Você pode ver na primeira grade que esse auto-início alinhado à direita e auto-fim alinhado à esquerda . Na segunda grade, no entanto, a direção RTL é irrelevante para o alinhamento do eixo do bloco. Assim, o auto-início foi tratado como começo, e o auto-fim foi tratado como fim.

O último valor, alongamento, é interessante. Para entendê-lo, observe como as outras caixas em cada grade "shrink-wrap" para o seu conteúdo. o alongamento, por outro lado, direciona o elemento para se esticar de ponta a ponta na direção dada – alinhar-se: o alongamento faz com que o item da grade se estique verticalmente e justifique-se: o alongamento causa o alongamento horizontal . Isso é como você poderia esperar, mas tenha em mente que funciona apenas se as propriedades de

tamanho do elemento estiverem definidas como auto. Assim, dados os seguintes estilos, o primeiro exemplo se estenderá verticalmente, mas o segundo não:

```
.exel01 {align-self: alongamento; altura: auto;}
.exel02 {align-self: alongamento; altura: 50%;}
```

Como o segundo exemplo define um valor de altura que não é automático (que é o valor padrão), ele não pode ser redimensionado por trecho. O mesmo vale para o auto-justificador e a largura.

Existem mais dois valores que podem ser usados para alinhar itens de grade, mas eles são suficientemente interessantes para merecer sua própria explicação. Eles permitem o alinhamento da primeira ou da última linha de base de um item de grade com a linha de base mais alta ou mais baixa na faixa de grade. Por exemplo, suponha que você queira que um item de grade seja alinhado para que a linha de base de sua última linha seja alinhada com a última linha de base no item de grade mais alto que compartilha seu rastreamento de linha. Isso seria parecido com o seguinte:

```
. exel {align-self: last-baseline;}
```

Por outro lado, para alinhar sua primeira linha de base com a primeira linha de base mais baixa na mesma faixa de linha , você diria o seguinte:

```
. exel {align-self: linha de base;}
```

No início de 2016, nenhum navegador dava suporte ao alinhamento de linha de base e de última linha de base no layout de grade.

Em uma situação em que um elemento de grade não tem uma linha de base ou é solicitado a alinhar a linha de base em uma direção em que as linhas de base não podem ser comparadas, a linha de base é tratada como início e a última linha de base é tratada como fim.

Há dois valores que foram intencionalmente ignorados nesta seção: flex-start e flex-end. Esses valores devem ser usados somente no layout do flexbox e são definidos como equivalentes a iniciar e terminar em qualquer outro contexto de layout, incluindo o layout de grade.

Alinhando e justificando todos os itens

Agora vamos considerar itens de alinhamento e itens de justificação. Essas propriedades aceitam todos os mesmos valores que vimos na seção anterior e têm o mesmo efeito, exceto que se aplicam a todos os itens de grade em um determinado contêiner de grade e devem ser aplicadas a um con- tainer de grade em vez de a itens de grade

individuais.

Assim, você pode definir todos os itens de grade em uma grade para serem alinhados ao centro dentro de suas áreas de grade da seguinte maneira, com um resultado como o representado na Figura 59:

```
#grid {display: grade;
    alinhar-itens: centro; justificar-itens: centro;}
```

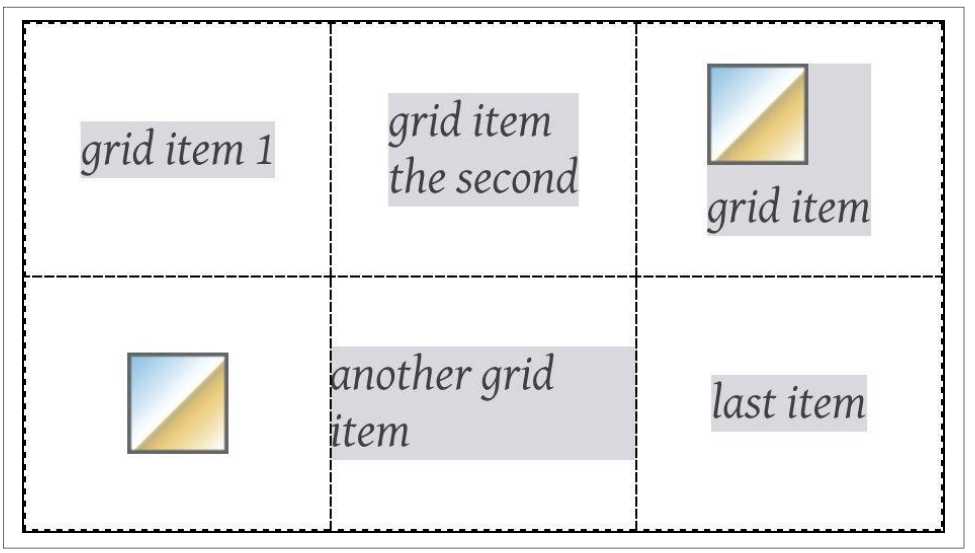


Figura 59. Centralizando todos os itens de grade

Como você pode ver, isso centraliza horizontal *e* verticalmente cada item de grade dentro de sua área de grade dada. Além disso, faz com que qualquer item de grade sem uma largura e altura explícitas "encolher" seu conteúdo em vez de se esticar para preencher sua área de grade. Se um item de grade tiver uma altura e largura explícitas, elas serão honradas e o item centralizado em sua área de grade.

Além de alinhar e justificar cada item de grade, é possível distribuir os itens de grade, ou mesmo justificar ou alinhar toda a grade, usando align-content e justify-content. Há um pequeno conjunto de valores distributivos para essas propriedades. A Figura 60 ilustra os efeitos de cada valor aplicado ao justificar-content, com cada grade compartilhando os seguintes estilos:

```
.grid {display: grade; preenchimento: 0.5em; margem: 0.5em 1em; largura:
        auto; grid-gap: 0.75em 0.5em; borda: 1px sólido;
        linhas de modelo de grade: 4em;
        grid-template-columns: repeat(5, 6em);}
```

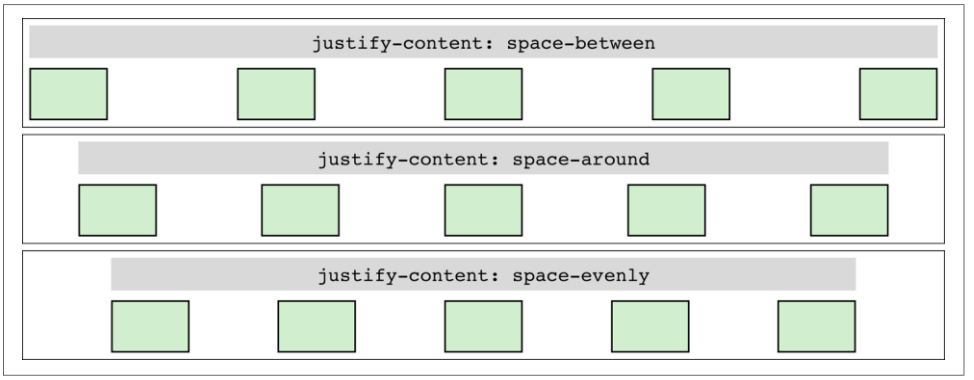


Figura 60. Distribuindo itens de grade horizontalmente

Isso funciona tão bem em trilhas de coluna quanto em faixas de linha, como a Figura 61 ilustra, desde que você alterne para o alinhamento de conteúdo. Desta vez, todas as grades compartilham estes estilos:

```
.grid {display: grade; preenchimento: 0.5em;
       grid-gap: 0.75em 0.5em; borda: 1px sólido;
       grid-template-rows: repeat(4, 3em);
       grid-template-columns: 5em;}
```

A maneira como essa distribuição funciona é que as trilhas da grade, incluindo quaisquer calhas, são todas dimensionadas como de costume. Em seguida, se houver algum espaço restante dentro do contêiner de grade, ou seja, se as trilhas de grade não alcançarem todo o caminho de uma borda do contêiner de grade até o outro
—então o espaço restante é distribuído de acordo com o valor de justificar-conteúdo (na horizontal) ou alinhamento de conteúdo (na vertical).

Esta distribuição de espaço é realizada através do redimensionamento das calhas da grade. Se não houver calhas declaradas, haverá calhas. Se já houver calhas, seus tamanhos são alterados conforme necessário para distribuir as trilhas da grade .

Observe que, como o espaço é distribuído somente quando as trilhas não preenchem o con- tainer da grade, as calhas só podem aumentar de tamanho. Se as faixas forem maiores que o contêiner , o que pode acontecer facilmente, não há espaço restante para distribuir (o espaço negativo acaba sendo indivisível).

Há outro valor de distribuição, muito novo até o momento em que escrevo, que não foi mostrado nas figuras anteriores. O alongamento pega qualquer espaço restante e o aplica igualmente às trilhas do grid, não às calhas. Portanto, se houver 400 pixels de espaço restante e oito faixas de grade , cada faixa de grade é aumentada em 50 pixels. As faixas de grade *não* são aumentadas proporcionalmente, mas igualmente. No início de 2016, não havia suporte do navegador para esse valor em termos de distribuição de grade.

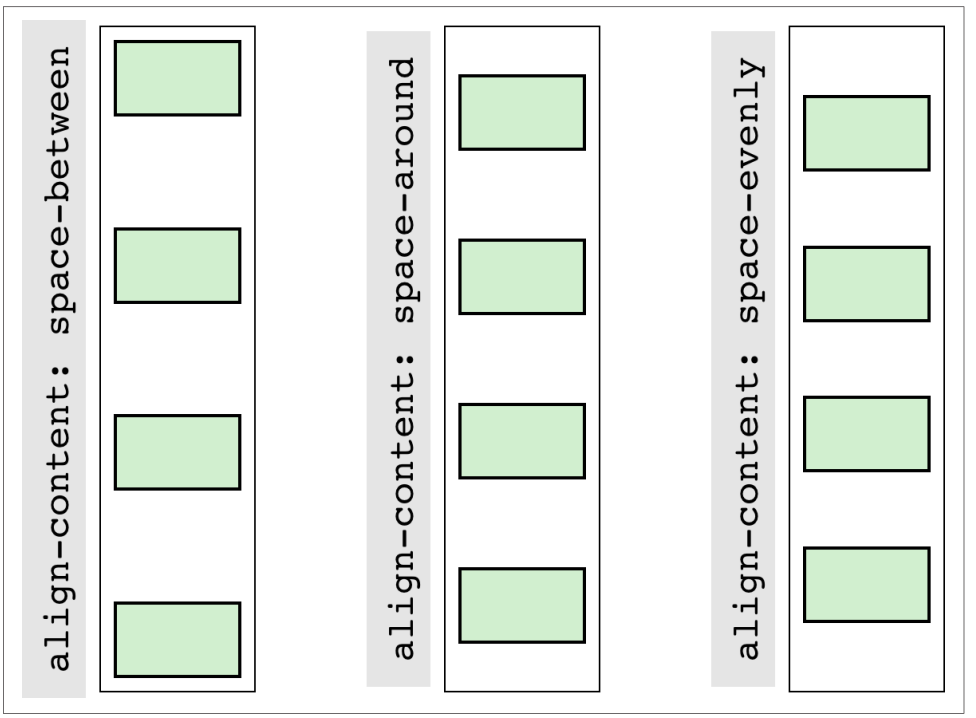


Figura 61. Distribuindo itens de grade verticalmente

Completaremos esta seção com exemplos de justificação, em vez de distribuição, de faixas de grade. A Figura 62 mostra as possibilidades ao justificar horizontalmente.

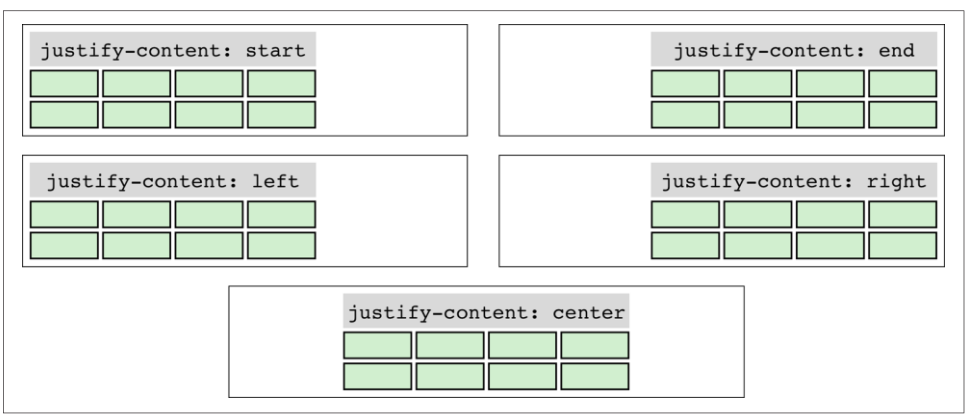
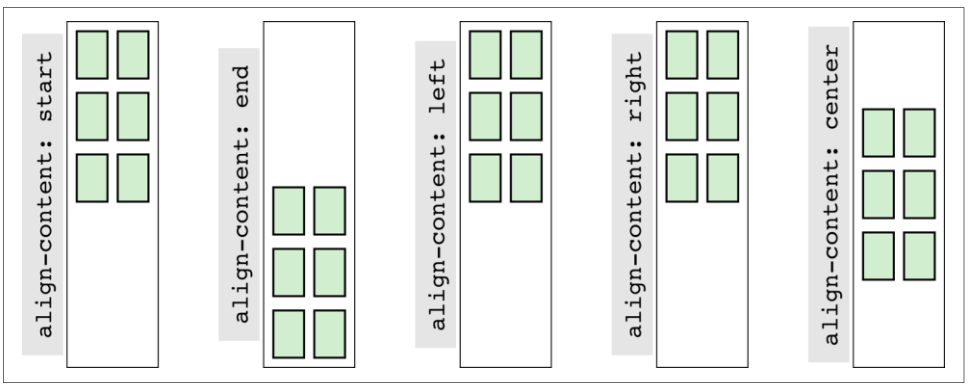


Figura 62. Justificando a grade horizontalmente

Nesses casos, o conjunto de faixas de grade é tomado como uma única unidade e justificado pelo valor de justificar-conteúdo. Esse alinhamento não afeta o alinhamento da grade individual

Itens; assim, você poderia finalizar toda a grade com o conteúdo justificado: terminar enquanto os itens de grade individuais são justificados à esquerda, ao centro ou ao início (entre asoutras opções) dentro de suas áreas de grade.

Como você poderia esperar agora, ser capaz de justificar o conteúdo horizontalmente significa que você pode alinhar o conteúdo verticalmente. A Figura 63 mostra cada valor



em ação.

Figura 63. Alinhando a grade verticalmente

É claro que esquerda e direita não fazem sentido em um contexto vertical, então elas são tratadas como
começar. Os outros têm o efeito que você esperaria de seus nomes.

Camadas e ordenação

Como vimos em uma seção anterior, é perfeitamente possível que os itens de grade se sobreponham uns aos outros, seja porque as margens negativas são usadas para puxar um item de grade além das bordas de sua área de grade ou simplesmente porque as áreas de grade de dois itens de grade diferentes compartilham a grade Células. Por padrão, os itens de grade se sobreporão visualmente na ordem de origem do documento: os itens de grade mais adiante na origem do documento aparecerão na frente dos itens de grade anterioresà origem do documento. Assim, vemos o seguinte resultado no que é representado na Figura 64. (Suponha que o número em cada nome de classe represente a ordem de origem do item de grade.)

```
#grid {display: grade; largura: 80%; altura: 20em;
      linhas de grade: repetir(10, 1fr); colunas de grade: repeat(10, 1fr);}
.box01 {linha de grade: 1 / span 4; coluna de grade: 1 / span 4;}
.box02 {linha de grade: 4 / span 4; coluna de grade: 4 / span 4;}
.box03 {grid-row: 7 / span 4; coluna de grade: 7 / span 4;}
.box04 {linha de grade: 4 / span 7; coluna de grade: 3 / span 2;}
.box05 {linha de grade: 2 / span 3; coluna de grade: 4 / span 5;}
```

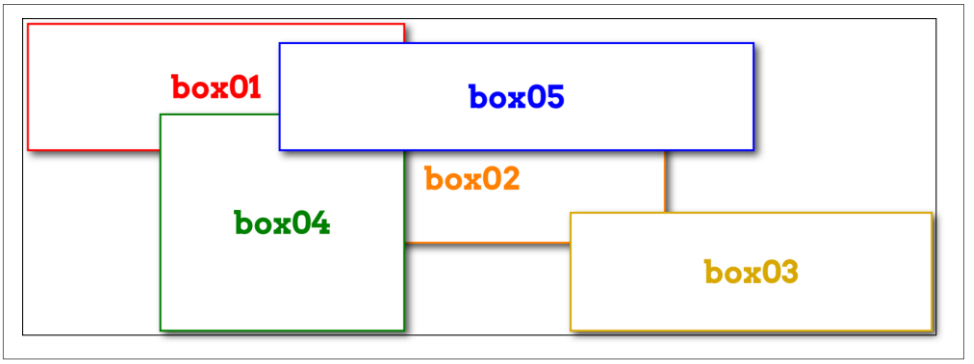


Figura 64. Itens de grade sobrepostos na ordem de origem

Se você quiser afirmar sua própria ordem de empilhamento, o z-index está aqui para ajudar. Assim como no posicionamento, o índice z coloca os elementos relativos uns aos outros no eixo z, que é perpendicular à superfície de exibição. Os valores positivos estão mais próximos de você e os valores negativos mais distantes. Então, para trazer a segunda caixa para o "topo", por assim dizer, tudo o que você precisa é dar-lhe um valor de índice z maior do que qualquer outro (com o resultado mostrado em Figura 65):

```
.box02 {z-índice: 10;}
```

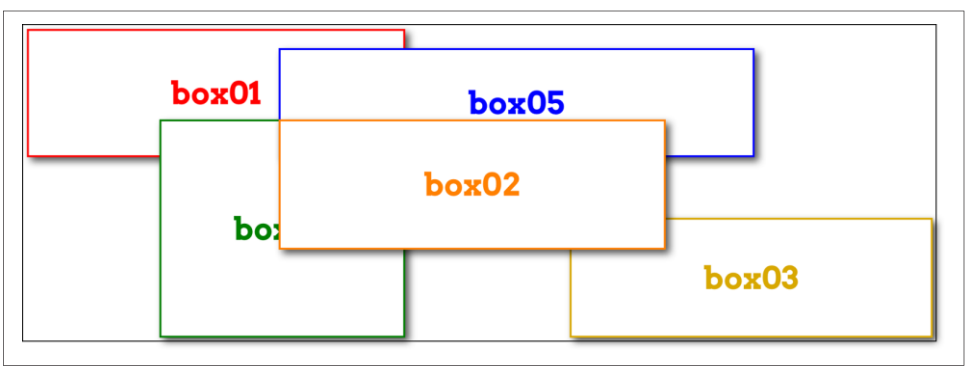


Figura 65. Elevar um item de grade

Outra maneira de afetar a ordenação de itens de grade é usando a propriedade order. Seu efeito é essencialmente o mesmo que no flexbox — você pode alterar a ordem dos itens de grade dentro de uma faixa de grade dando-lhes valores de ordem. Isso afeta não apenas o local dentro da pista, mas também a ordem de *pintura*, se eles se sobreporem. Por exemplo, poderíamos alterar o exemplo anterior de z-index para order, como mostrado aqui, e obter o mesmo resultado mostrado na Figura 65:

```
.box02 {ordem: 10;}
```

Nesse caso, a caixa 02 aparece "em cima" dos outros itens da grade porque seu pedido a coloca depois do resto deles. Assim, é desenhado por último. Da mesma forma, se esses itens de grade fossem todos colocado em sequência em uma faixa

de grade, o valor do pedido para box02 o colocaria no final da sequência. Isso é representado na Figura 66.

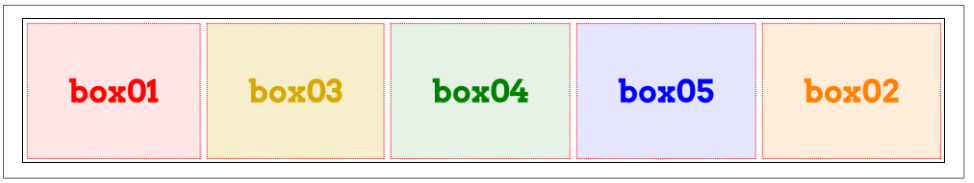


Figura 66. Alterando a ordem do item de grade

Lembre-se de que só porque você *pode* reorganizar a ordem dos itens da grade dessa maneira, isso não significa necessariamente que você *deve*. Como diz a especificação Layout de grade (seção 4.2):

> Assim como na reordenação de itens flexíveis, a propriedade order só deve ser usada quando a ordem visual precisa estar *fora de sincronia* com a ordem de fala e navegação; caso contrário, a fonte do documento subjacente deve ser reordenada.

Portanto, a única razão para usar a ordem para reorganizar o layout do item de grade é se você precisar ter a origem do documento em uma ordem e o layout na outra. Isso já é facilmente possível atribuindo itens de grade a áreas que não correspondem à ordem de origem, é claro.

Isso não quer dizer que a ordem seja inútil e deva ser sempre evitada; pode muito bem haver momentos em que faça sentido. Mas, a menos que você se encontre quase forçado a usá-lo por circunstâncias específicas, pense muito sobre se é a melhor solução.

Resumo

O layout da grade é complexo e poderoso, por isso não desanime se você se sentir sobrecarregado. Leva algum tempo para se acostumar com a forma como a rede opera, especialmente porque muitos de seus recursos não são nada parecidos com o que lidamos antes. Grande parte do p ower dessas fea-tures vem diretamente de sua novidade – mas, como qualquer ferramenta poderosa, pode ser difícil e frustrante aprender a usar. Fiquei frustrado e confuso enquanto escrevia sobre grade, descendo becos sem saída e sendo vítima de duas décadas de instintos que haviam sido aperfeiçoados em um CSS sem layout.

Espero ter sido capaz de guiá-lo além de algumas dessas armadilhas, mas ainda assim, lembre-se da sabedoria do Mestre Yoda: "Você deve desaprender o que aprendeu". Ao chegar ao layout de grade, nunca houve tanta necessidade de deixar de lado o que você acha que sabe sobre layout e aprender de novo. Com o tempo, sua paciência e persistência serão recompensadas.

Sobre o Autor

Eric A. Meyer trabalha com a Web desde o final de 1993 e é um especialista reconhecido internacionalmente nos assuntos de HTML, CSS e padrões da web. Um autor amplamente lido, ele também é o fundador da Complex Spiral Consulting, que conta entre seus clientes America Online; Apple Computer, Inc.; Banco Wells Fargo; e Mac- romedia, que descreveu Eric como "um parceiro crítico em nossos esforços para transformar o Mac- romedia Dreamweaver MX 2004 em uma ferramenta revolucionária para o design baseado em CSS".

A partir do início de 1994, Eric foi o designer visual e coordenador web do campus para o site da Case Western Reserve University, onde também escreveu uma série amplamente aclamada de três tutoriais HTML e foi coordenador de projeto para a versão on-line da *Encyclopedia of Cleveland History* e do *Dictionary of Cleveland Biog- raphy*, a primeira enciclopédia de história urbana publicada completa e livremente na Web.

Autor de Eric Meyer em CSS e *More Eric Meyer em* CSS (New Riders), CSS: *The Definitive Guide* (O'Reilly) e *CSS 2.0 Programmer's Reference* (Osborne/McGraw-Hill), bem como numerosos artigos para a O'Reilly Network, Web Techniques e Web Review, Eric também criou o CSS Browser Compatibility Charts e coordenou a criação e criação dos W3C's do CSS Test Suite fictício. Ele lecionou para uma ampla variedade de organizações, incluindo o Laboratório Nacional de Los Alamos, a Biblioteca Pública de Nova York, a Universidade de Cornell e a Universidade do Norte de Iowa. Eric também proferiu discursos e apresentações técnicasem inúmeras conferências, entre elas An Event Apart (que ele co-fundou), a série IW3C2 WWW, Web Design World, CMP, SXSW, a série de conferências User Interface e The Other Dreamweaver Conference.

Em seu tempo pessoal, Eric atua como acompanhante de listasda altamente ativa lista de discussão css-discussão, que ele co-fundou com John Allsopp da Civilização Ocidental, e que agora é apoiada por *evolt.org*. Eric mora em Cleveland, Ohio, que é umacidade muc h mais agradável do que você foi levado a acreditar. Por nove anos, ele foi o apresentador do "Your Father's Oldsmobile", um programa de rádio de big band ouvido semanalmente na WRUW 91.1 FM em Cleveland.

Você pode encontrar informações mais detalhadas na página pessoal de Eric.

Colofão

Os animais na capa do *Grid Layout em CSS* são salmão (*salmonidae*), que é uma família de peixes que consiste em muitas espécies diferentes. Dois dos salmões mais comuns são o salmão do Pacífico e o sal do Atlântico.

O salmão do Pacífico vive no norte do Oceano Pacífico ao largo das costas da América do Norte e da Ásia. Existem cinco subespécies de salmão do Pacífico, com um peso médio de 10 a 30 libras. Salmão do Pacífico nascem no outono em leitos de cascalho de córrego de água doce, onde

eles incubam durante o inverno e emergem como peixes de uma polegada de comprimento. Eles vivem por um ano ou dois em riachos ou lagos e depois seguem rio abaixo para o oceano. Lá eles vivem por alguns anos, antes de voltar rio acima para o seulocal de nascimento exa ct para desovar e depois morrer.

O salmão do Atlântico vive no norte do Oceano Atlântico, ao largo das costas da América do Norte e da Europa. Existem muitas subespécies de salmão do Atlântico, incluindo a truta e o char. Seu peso médio é de 10 a 20 libras. A família do salmão do Atlântico tem um ciclo de vida semelhante ao de seus primos do Pacífico e também viaja de leitos de cascalho de água doce para o mar. Uma grande diferença entre os dois, no entanto, é que o sal-mon do Atlântico não morre após a desova; ele pode voltar para o oceano e depois retornar ao riacho para desovar novamente, geralmente duas ou três vezes.

O salmão, em geral, é um peixe gracioso, de cor prateada, com manchas nas costas e barbatanas. Sua dieta consiste em plâncton, larvas de insetos, camarões e peixes menores. Acredita-se que seu olfato aguçado não aliado os ajude a navegar do oceano de volta ao local exato de seu nascimento, rio acima, passando por muitos obstáculos. Algumas espécies de salmão rema em sem litoral, vivendo toda a sua vida em água doce.

Os salmões são uma parte importante do ecossistema, pois seus corpos em decomposição fornecem fer- tilizer para os leitos dos córregos. Seus números têm diminuído ao longo dos anos, no entanto. Fatores no declínio da população de salmão incluem destruição de habitat, pesca, barragens que bloqueiam caminhos de desova, chuva ácida, secas, inundações e poluição.

A imagem da capa é uma gravura do século 19 do Arquivo Pictórico de Dover. As fontes de capa são URW Typewriter e Guardian Sans. A fonte do texto é Adobe Minion Pro; a fonte do título é Adobe Myriad Condensed; e a fonte do código é o Ubuntu Mono de Dalton Maag.