

Fundo

O Flexbox Layoutmódulo (Flexible Box) ([uma Recomendação Candidata do W3C](#) (<https://www.w3.org/TR/css-flexbox/>) em outubro de 2017) visa fornecer uma maneira mais eficiente de dispor, alinhar e distribuir o espaço entre os itens em um contêiner, mesmo quando seu tamanho é desconhecido e/ou dinâmico (portanto, a palavra “flex”).

A ideia principal por trás do layout flexível é dar ao contêiner a capacidade de alterar a largura/altura (e a ordem) de seus itens para melhor preencher o espaço disponível (principalmente para acomodar todos os tipos de dispositivos de exibição e tamanhos de tela). Um contêiner flexível expande os itens para preencher o espaço livre disponível ou os reduz para evitar o estouro.

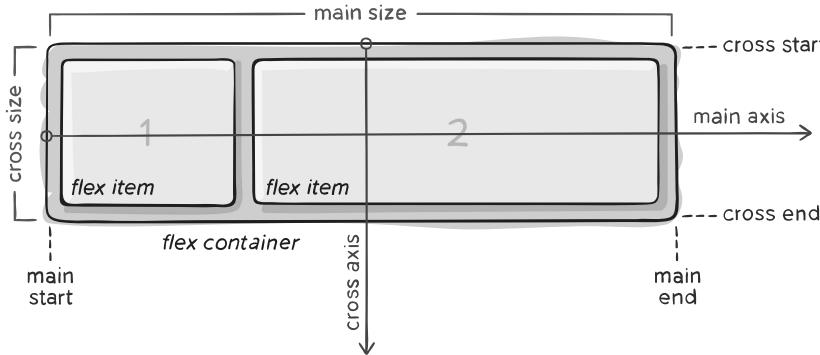
Mais importante ainda, o layout do flexbox é independente de direção, em oposição aos layouts regulares (bloco que é baseado na vertical e inline que é baseado na horizontal). Embora funcionem bem para páginas, eles carecem de flexibilidade (sem trocadilhos) para oferecer suporte a aplicativos grandes ou complexos (especialmente quando se trata de mudança de orientação, redimensionamento, alongamento, encolhimento etc.).

Observação: o layout Flexbox é mais adequado para os componentes de um aplicativo e layouts de pequena escala, enquanto o layout [Grid](#) (<https://css-tricks.com/snippets/css/complete-guide-grid/>) é destinado a layouts de grande escala.

Noções básicas e terminologia

Como o flexbox é um módulo inteiro e não uma única propriedade, ele envolve muitas coisas, incluindo todo o conjunto de propriedades. Alguns deles devem ser definidos no contêiner (elemento pai, conhecido como “flex container”), enquanto outros devem ser definidos nos filhos (ditos “itens flexíveis”).

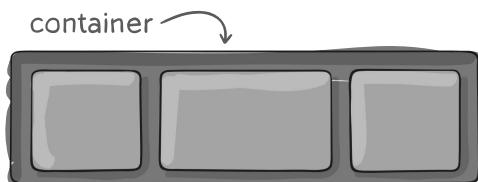
Se o layout “regular” for baseado em direções de fluxo em bloco e em linha, o layout flexível será baseado em “direções de fluxo flexível”. Por favor, dê uma olhada nesta figura da especificação, explicando a ideia principal por trás do layout flexível.



Os itens serão dispostos seguindo o `main axis`(de `main-start`para `main-end`) ou o eixo transversal (de `cross-start`para `cross-end`).

- **eixo principal** – O eixo principal de um contêiner flexível é o eixo principal ao longo do qual os itens flexíveis são dispostos. Cuidado, não é necessariamente horizontal; depende da `flex-direction`propriedade (veja abaixo).
- **início principal | main-end** – Os itens flexíveis são colocados dentro do contêiner começando do `main-start` e indo para o `main-end`.
- **tamanho principal** – A largura ou altura de um item flexível, o que estiver na dimensão principal, é o tamanho principal do item. A propriedade de tamanho principal do item flexível é a propriedade 'largura' ou 'altura', o que estiver na dimensão principal.
- **eixo cruzado** – O eixo perpendicular ao eixo principal é chamado de eixo cruzado. Sua direção depende da direção do eixo principal.
- **início cruzado | cross-end** – As linhas flexíveis são preenchidas com itens e colocadas no contêiner começando no lado de início cruzado do contêiner flexível e indo em direção ao lado de extremidade cruzada.
- **tamanho cruzado** – A largura ou altura de um item flexível, o que estiver na dimensão cruzada, é o tamanho cruzado do item. A propriedade de tamanho cruzado é qualquer uma de 'largura' ou 'altura' que esteja na dimensão cruzada.

🔗 (#aa-flexbox-properties) Propriedades da caixa flexível



🔗 (#aa-properties-for-the-parentflex-container) Propriedades para o pai (contêiner flexível)

🔗 (#aa-display) mostrar

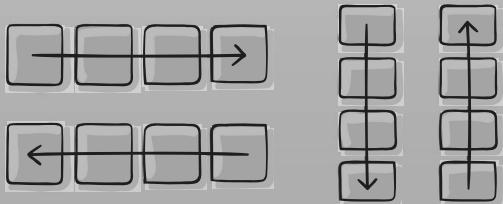
Isso define um contêiner flexível; inline ou bloco, dependendo do valor fornecido. Ele permite um contexto flexível para todos os seus filhos diretos.

```
.container {  
  display: flex; /* or inline-flex */  
}
```

CSS

Observe que as colunas CSS não têm efeito em um contêiner flexível.

🔗 (#aa-flex-direction) **direção flexível**



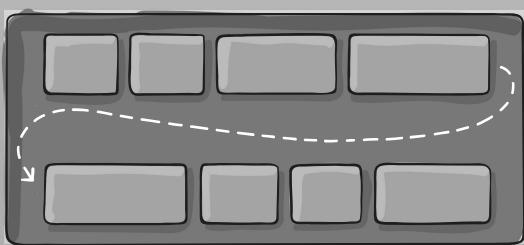
Isso estabelece o eixo principal, definindo assim a direção em que os itens flexíveis são colocados no contêiner flexível. Flexbox é (além do empacotamento opcional) um conceito de layout de direção única. Pense nos itens flexíveis como dispostos principalmente em linhas horizontais ou colunas verticais.

```
.container {  
  flex-direction: row | row-reverse | column | column-reverse;  
}
```

CSS

- **row**(padrão): da esquerda para a direita em ltr; direita para esquerda em rtl
- **row-reverse**: direita para esquerda em ltr; da esquerda para a direita em rtl
- **column**: o mesmo que, rowmas de cima para baixo
- **column-reverse**: o mesmo que, row-reversemás de baixo para cima

🔗 (#aa-flex-wrap) **envoltório flexível**



Por padrão, todos os itens flexíveis tentarão caber em uma linha. Você pode alterar isso e permitir que os itens sejam encapsulados conforme necessário com essa propriedade.

```
.container {  
  flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

CSS

- **nowrap**(padrão): todos os itens flexíveis estarão em uma linha

- **wrap**: os itens flexíveis serão agrupados em várias linhas, de cima para baixo.
- **wrap-reverse**: os itens flexíveis serão agrupados em várias linhas de baixo para cima.

Existem algumas [demonstrações visuais flex-wrap aqui](https://css-tricks.com/almanac/properties/f/flex-wrap/) (<https://css-tricks.com/almanac/properties/f/flex-wrap/>).

🔗 (#aa-flex-flow) fluxo flexível

Este é um atalho para as propriedades `flex-direction` `flex-wrap`, que juntas definem os eixos principal e transversal do contêiner flexível. O valor padrão é `row nowrap`.

```
.container {  
  flex-flow: column wrap;  
}
```

css

🔗 (#aa-justify-content) justificar-contúdo

`flex-start`



`flex-end`



`center`



`space-between`



`space-around`



`space-evenly`



Isso define o alinhamento ao longo do eixo principal. Ele ajuda a distribuir sobras de espaço livre extra quando todos os itens flexíveis em uma linha são inflexíveis ou são flexíveis, mas atingiram seu tamanho máximo. Também exerce algum controle sobre o alinhamento dos itens quando eles ultrapassam a linha.

```
.container {  
  justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly | start | end | left | right ... + safe | unsafe;  
}
```

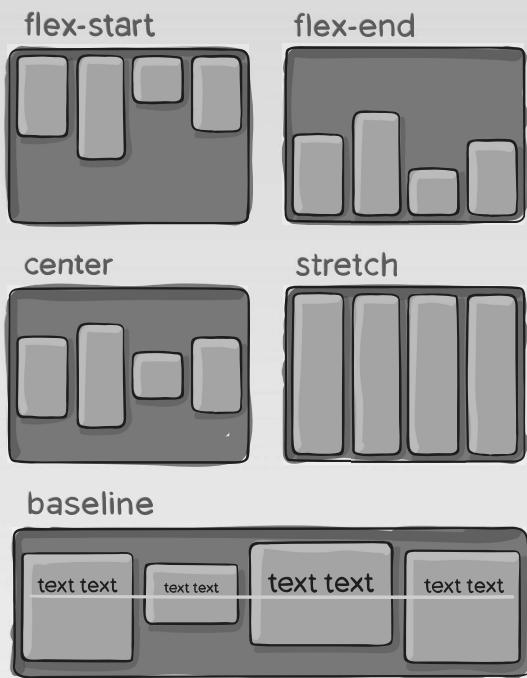
css

- **flex-start**(padrão): os itens são compactados no início da direção flexível.
- **flex-end**: os itens são embalados no final da direção flexível.
- **start**: os itens são embalados no início da writing-mode direção.
- **end**: os itens são embalados no final da writing-mode direção.
- **left**: os itens são empacotados na borda esquerda do contêiner, a menos que isso não faça sentido com o **flex-direction**, então ele se comporta como **start**.
- **right**: os itens são embalados na borda direita do contêiner, a menos que isso não faça sentido com o **flex-direction**, então ele se comporta como **end**.
- **center**: itens são centralizados ao longo da linha
- **space-between**: os itens são distribuídos uniformemente na linha; primeiro item está na linha de partida, último item na linha final
- **space-around**: os itens são distribuídos uniformemente na linha com espaço igual ao redor deles. Observe que visualmente os espaços não são iguais, pois todos os itens possuem espaço igual em ambos os lados. O primeiro item terá uma unidade de espaço contra a borda do contêiner, mas duas unidades de espaço entre o próximo item porque esse próximo item tem seu próprio espaçamento aplicável.
- **space-evenly**: os itens são distribuídos de forma que o espaçamento entre quaisquer dois itens (e o espaço até as bordas) seja igual.

Observe que o suporte do navegador para esses valores é diferenciado. Por exemplo, **space-between** nunca obtive suporte de algumas versões do Edge e **início/fim/esquerda/direita** ainda não estão no Chrome. MDN [tem gráficos detalhados](https://developer.mozilla.org/en-US/docs/Web/CSS/justify-content) (<https://developer.mozilla.org/en-US/docs/Web/CSS/justify-content>). Os valores mais seguros são **flex-start**, **flex-end** e **center**.

Há também duas palavras-chave adicionais que você pode associar a esses valores: **safe** e **unsafe**. O uso **safegarante** que, independentemente de como você faça esse tipo de posicionamento, você não pode empurrar um elemento de forma que ele seja renderizado fora da tela (por exemplo, fora do topo) de forma que o conteúdo também não possa ser rolado (chamado “perda de dados”).

🔗 (#aa-align-items) **itens de alinhamento**



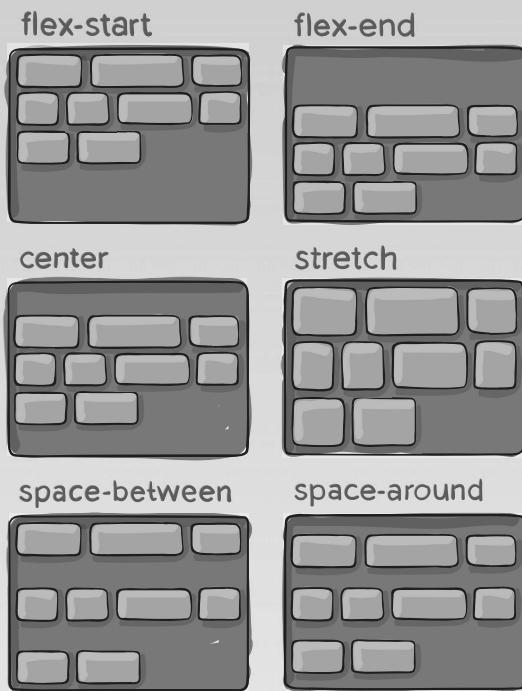
Isso define o comportamento padrão de como os itens flexíveis são dispostos ao longo do **eixo transversal** na linha atual. Pense nisso como a `justify-content` para o eixo transversal (perpendicular ao eixo principal).

```
css
.container {
  align-items: stretch | flex-start | flex-end | center | baseline | first baseline | last baseline | start | end | self-start | self-end + ... ;
}
```

- **stretch**(padrão): esticar para encher o recipiente (ainda respeitando min-width/max-width)
- **flex-start/ start/ self-start**: os itens são colocados no início do eixo transversal. A diferença entre estes é sutil, e trata-se de respeitar as `flex-direction`regras ou as `writing-mode`regras.
- **flex-end/ end/ self-end**: os itens são colocados no final do eixo transversal. A diferença novamente é sutil e é sobre respeitar `flex-direction`regras versus `writing-mode`regras.
- **center**: os itens são centralizados no eixo cruzado
- **baseline**: itens são alinhados como suas linhas de base alinhadas

As palavras-chave modificadoras `safe` e `unsafe` podem ser usadas em conjunto com todas as outras palavras-chave (embora observe o [suporte do navegador](#) (<https://developer.mozilla.org/en-US/docs/Web/CSS/align-items>)) e ajudam a evitar o alinhamento de elementos de modo que o conteúdo se torne inacessível.

🔗 (#aa-align-content) **alinhar-contúdo**



Isso alinha as linhas de um contêiner flexível quando há espaço extra no eixo cruzado, semelhante a como `justify-content` alinha itens individuais no eixo principal.

Hey!

Observação: essa propriedade só tem efeito em contêineres flexíveis de várias linhas, em que `flex-wrap` definido como `wrap` ou `wrap-reverse`. Um contêiner flexível de linha única (ou seja, onde `flex-wrap` está definido como seu valor padrão, `no-wrap`) não refletirá `align-content`.

```
css
.container {
  align-content: flex-start | flex-end | center | space-between | space-around | space-evenly | stretch | start | end | baseline | first baseline
}
```

- `normal`(padrão): os itens são compactados em sua posição padrão, como se nenhum valor fosse definido.
- `flex-start/ start`: itens embalados no início do contêiner. O (mais apoiado) `flex-start` honra o `flex-direction` enquanto `start` honra a `writing-mode` direção.
- `flex-end/ end`: itens embalados até o final do container. O (mais suporte) `flex-end` homenageia o `flex-direction` while `end` homenageia a `writing-mode` direção.
- `center`: itens centralizados no contêiner
- `space-between`: itens distribuídos uniformemente; a primeira linha está no início do contêiner enquanto a última está no final
- `space-around`: itens distribuídos uniformemente com espaço igual ao redor de cada linha
- `space-evenly`: os itens são distribuídos uniformemente com espaço igual ao seu redor
- `stretch`: as linhas se estendem para ocupar o espaço restante

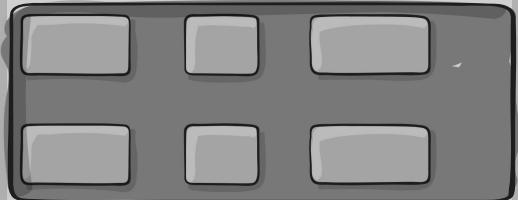
As palavras-chave modificadoras `safe` e `unsafe` podem ser usadas em conjunto com todas as outras palavras-chave (embora observe o [suporte do navegador](https://developer.mozilla.org/en-US/docs/Web/CSS/align-items) (<https://developer.mozilla.org/en-US/docs/Web/CSS/align-items>)) e ajudam a evitar o alinhamento de elementos de modo que o conteúdo se torne inacessível.

🔗 (#aa-gap-row-gap-column-gap) lacuna, lacuna entre linhas, lacuna entre colunas

gap: 10px



gap: 30px



gap: 10px 30px



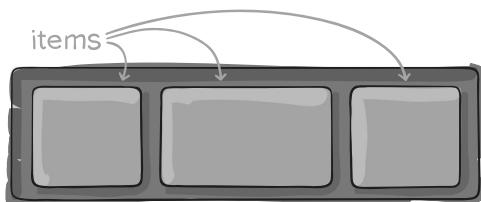
A gap propriedade (<https://css-tricks.com/almanac/properties/g/gap/>) controla explicitamente o espaço entre os itens flexíveis. Aplica esse espaçamento *apenas entre os itens* que não estão nas bordas externas.

```
.container {  
  display: flex;  
  ...  
  gap: 10px;  
  gap: 10px 20px; /* row-gap column gap */  
  row-gap: 10px;  
  column-gap: 20px;  
}
```

CSS

O comportamento pode ser pensado como uma calha *mínima*, como se a calha fosse maior de alguma forma (por causa de algo como `justify-content: space-between;`), então a lacuna só teria efeito se esse espaço acabasse menor.

Não é exclusivo para flexbox, gap funciona em grade e layout de várias colunas também.



🔗 (#aa-properties-for-the-childrenflex-items) Propriedades para os filhos (itens flexíveis)

🔗 (#aa-order) ordem



Por padrão, os itens flexíveis são dispostos na ordem de origem. No entanto, a `order` propriedade controla a ordem em que aparecem no contêiner flexível.

```
.item {
  order: 5; /* default is 0 */
}
```

CSS

Itens com o mesmo `order` revertem para a ordem de origem.

🔗 (#aa-flex-grow) **flex-crescer**



Isso define a capacidade de um item flexível crescer, se necessário. Aceita um valor sem unidade que serve de proporção. Ele determina a quantidade de espaço disponível dentro do contêiner flexível que o item deve ocupar.

Se todos os itens forem `flex-grow` definidos como 1, o espaço restante no contêiner será distribuído igualmente para todos os filhos. Se um dos filhos tiver um valor de 2, esse filho ocuparia o dobro do espaço de qualquer um dos outros (ou tentará, pelo menos).

```
.item {
  flex-grow: 4; /* default 0 */
}
```

CSS

Números negativos são inválidos.

🔗 (#aa-flex-shrink) **flexionar**

Isso define a capacidade de um item flexível encolher, se necessário.

```
.item {  
  flex-shrink: 3; /* default 1 */  
}
```

CSS

Números negativos são inválidos.

⌚ (#aa-flex-basis) base flexível

Isso define o tamanho padrão de um elemento antes que o espaço restante seja distribuído. Pode ser um comprimento (por exemplo, 20%, 5rem, etc.) ou uma palavra-chave. A `autopalavra-chave` significa “olhar para minha propriedade de largura ou altura” (o que foi temporariamente feito pela `main-sizepalavra-chave` até ser obsoleto). A `contentpalavra-chave` significa “dimensioná-lo com base no conteúdo do item” – esta palavra-chave ainda não é bem suportada, então é difícil testar e mais difícil saber o que seus irmãos, `max-content`, `min-content` fazem `fit-content`.

```
.item {  
  flex-basis: 1 auto; /* default auto */  
}
```

CSS

Se for definido como `0`, o espaço extra ao redor do conteúdo não será contabilizado. Se for definido como `auto`, o espaço extra será distribuído com base em seu `flex-grow` valor. [Veja este gráfico](#). (<http://www.w3.org/TR/css3-flexbox/images/rel-vs-abs-flex.svg>)

⌚ (#aa-flex) flex

Esta é a abreviação de `flex-grow`, `flex-shrink` e `flex-basis` combinado. O segundo e terceiro parâmetros (`flex-shrink` e `flex-basis`) são opcionais. O padrão é `0 1 auto`, mas se você defini-lo com um único valor numérico, como `flex: 5;`, isso muda `flex-basis` para `0%`, então é como definir `flex-grow: 5;`, `flex-shrink: 1;` e `flex-basis: 0%;`.

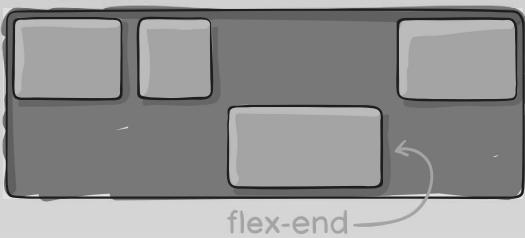
```
.item {  
  flex: none | [ '<'flex-grow'> <'flex-shrink'>? || '<'flex-basis'>' ]  
}
```

CSS

É recomendável usar essa propriedade abreviada em vez de definir as propriedades individuais. A abreviação define os outros valores de forma inteligente.

⌚ (#aa-align-self) alinhar-se

`flex-start`



Isso permite que o alinhamento padrão (ou aquele especificado por `align-items`) seja substituído por itens flexíveis individuais.

Consulte a `align-itemsexplicação` para entender os valores disponíveis.

```
.item {  
  align-self: auto | flex-start | flex-end | center | baseline | stretch;  
}
```

CSS

Observe que `float`, `clear` e `vertical-align` não têm efeito em um item flexível.

🔗 ↴ #aa-prefixing-flexbox) Prefixando Flexbox

O Flexbox requer algum prefixo de fornecedor para suportar o maior número possível de navegadores. Ele não inclui apenas propriedades anexadas com o prefixo do fornecedor, mas na verdade existem propriedades e nomes de valor totalmente diferentes. Isso ocorre porque a especificação do Flexbox mudou ao longo do tempo, criando versões “antigas”, “interpoladoras” e “novas” (<https://css-tricks.com/old-flexbox-and-new-flexbox/>).

Talvez a melhor maneira de lidar com isso seja escrever na nova (e final) sintaxe e executar seu CSS por meio do Autoprefixer (<https://css-tricks.com/autoprefixer/>) , que lida muito bem com os fallbacks.

Como alternativa, aqui está um Sass @mixin para ajudar com alguns dos prefixos, que também lhe dão uma ideia de que tipo de coisas precisam ser feitas:

```
@mixin flexbox() {  
  display: -webkit-box;  
  display: -moz-box;  
  display: -ms-flexbox;  
  display: -webkit-flex;  
  display: flex;  
}  
  
@mixin flex($values) {  
  -webkit-box-flex: $values;  
  -moz-box-flex: $values;  
  -webkit-flex: $values;  
  -ms-flex: $values;  
  flex: $values;  
}
```

SCSS

```
@mixin order($val) {  
  -webkit-box-ordinal-group: $val;  
  -moz-box-ordinal-group: $val;  
  -ms-flex-order: $val;  
  -webkit-order: $val;  
  order: $val;  
}  
  
.wrapper {  
  @include flexbox();  
}  
  
.item {  
  @include flex(1 200px);  
  @include order(2);  
}
```

↵ (#aa-examples) Exemplos

↵ (#aa-flexbox-tricks) Flexbox truques!

↵ (#aa-browser-support) Suporte ao navegador

Esses dados de suporte do navegador são de [Canius](http://caniuse.com/#feat=flexbox) (<http://caniuse.com/#feat=flexbox>) , que tem mais detalhes. Um número indica que o navegador oferece suporte ao recurso nessa versão e superior.

Área de Trabalho

Chrome: 21* Firefox: 28 IE: 11 Edge: 12 Safari: 6.1*

Celular / Tablet

Android Chrome: 109 Android Firefox: 109 Android: 4.4 iOS Safari: 7.0-7.1*

↵ (#aa-bugs) Insetos

↵ (#aa-related-properties) Propriedades relacionadas

↵ (#aa-more-information) Mais Informações

↵ (#aa-more-sources) Mais fontes