3 - Flex Layout Model

linkedin.com/in/barbaraeste/
github.com/babieste/

Introdução

O CSS 2 definiu quatro modos de layout:

1. Block layout;
2. Inline layout;
3. Table layout;
4. Positioned layout.

O CSS 3 introduz um novo modo de layout, o **flex layout**, que foi projetado para organizar aplicações e webpages mais complexas.

Superficialmente, o flex layout é similar ao block layout. Ele carece de propriedades mais complexas voltadas para a manipulação de textos, mas em troca fornece ferramentas simples e poderosas para distribuir espaço e alinhar conteúdo de maneiras complexas.

No flex layout model, os filhos de um container flexível podem ser dispostos em qualquer direção e podem "flexibilizar" seus tamanhos, aumentando para preencher o espaço não utilizado ou diminuindo para evitar transbordarem.

O conteúdo de um flex container:

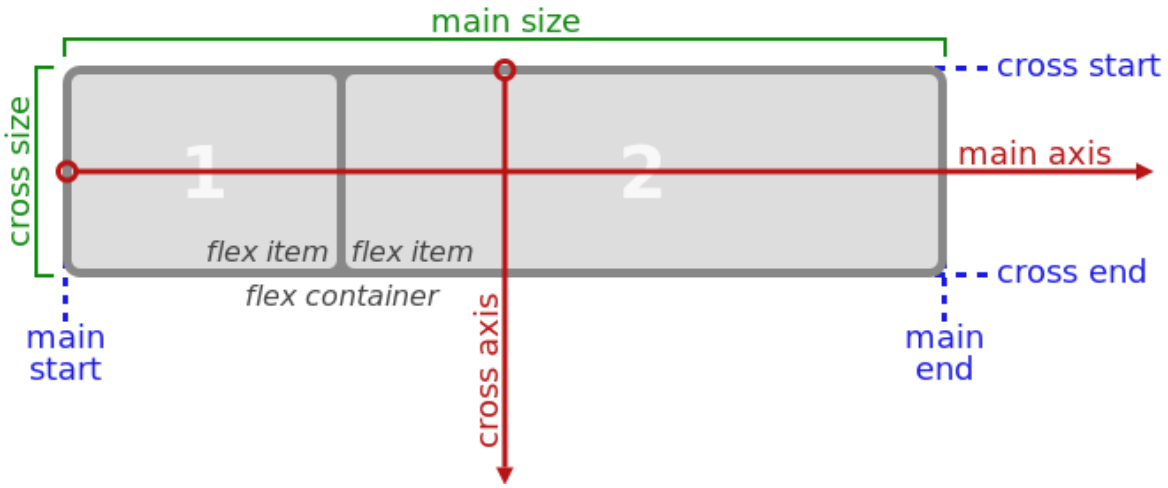
1. Podem ser dispostos em qualquer direção de texto (flow direction): para a esquerda, para a direita, para baixo ou até mesmo para cima;
2. Podem ter sua ordem de exibição invertida ou reorganizada no nível visual;
3. Pode ser disposto linearmente ao longo de um único eixo principal (main axis) ou agrupado em múltiplas linhas ao longo de um eixo transversal (cross axis);
4. Conseguem "flexibilizar" seus tamanhos para corresponder ao espaço disponível;
5. Podem ser alinhados com respeito ao seu contêiner ou à si mesmos no eixo transversal;
6. Podem ser colapsados/decolapsados dinamicamente no eixo principal enquanto preservam o tamanho do contêiner no eixo transversal.

Flex Layout Box Model

Um **flex container** é o box gerado por um elemento com um valor de '**flex**' ou '**inline-flex**' na propriedade '**display**'. Os filhos de um flex container são chamados de **flex items** e são dispostos utilizando o flex layout model.

Diferentemente do block e inline layout, cujos cálculos de layout são direcionados nas direções do flow do texto, o flex layout é direcionado nas direções flex (**flex**

directions). As propriedades '**flex-flow**' e '**writing-mode**' definem como estas direções são mapeadas para as direções físicas (cima/baixo/esquerda/direita), eixos (vertical/horizontal) e tamanhos (largura/altura);



Uma ilustração dos vários termos aplicados à direções e tamanhos, aplicados à um flex container orientado à linha

1. **main axis**
   main dimension
   O **eixo principal** (main axis) de um flex container é o eixo primário no qual flex items são dispostos. Estende-se na **dimensão principal** (main dimension).

2. **main-start**
   main-end
   Os flex items são dispostos dentro do container começando no lado **main-start** em direção ao lado **main-end**.

3. **main size**
   main size property
   A **largura** ou a **altura** de um flex container ou um flex item, dependendo de qual dimensão é a principal (main dimension) define o **tamanho principal** (main size) do box. A propriedade que define o tamanho principal é '**width**' ou '**height**', dependendo, novamente, de qual é a dimensão principal.

   De forma similar, as propriedades que definem o **tamanho principal mínimo ou máximo** (min/max main size properties) serão '**min-width**'/'**max-width**' ou '**min-height**'/'**max-height**', dependendo de qual é a dimensão principal.

4. **cross axis**
   cross dimension
   O **eixo perpendicular** ou **transversal** ao eixo principal é denominado **cross axis**. Estende-se na **direção transversal** (cross dimension).

5. **cross-start**
   cross-end

Linhas flex (flex lines) são preenchidas com itens e dispostas no container começando no lado **cross-start** e indo em direção ao lado **cross-end**.

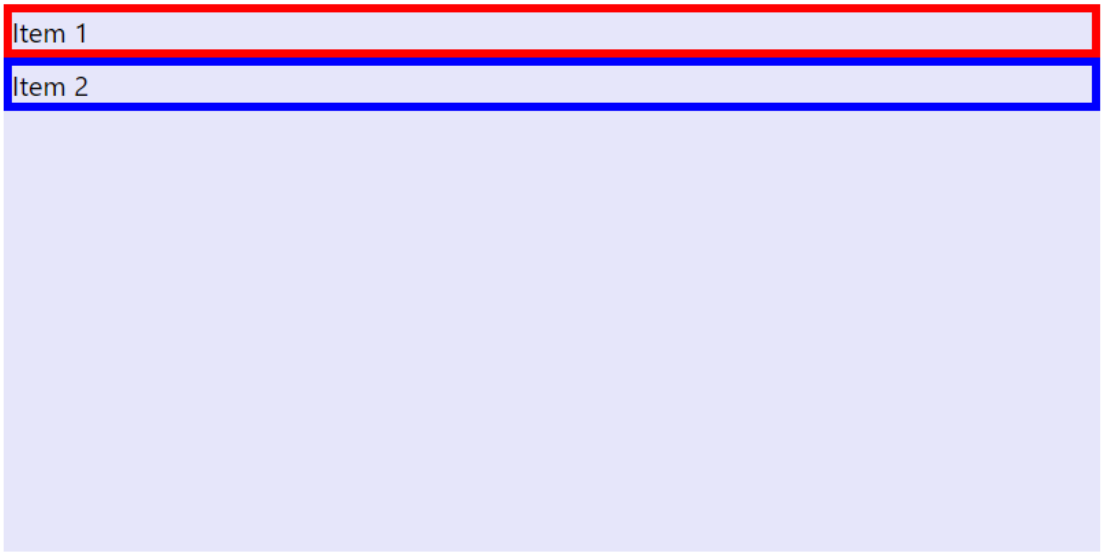
6. **cross size**
   cross size property
   A **largura** ou a **altura** de um flex container ou flex item, dependendo de qual dimensão é a transversal (cross dimension), define o **tamanho transversal** (cross size) do box. A propriedade que define o cross size será então '**width**' ou '**height**', dependendo de qual é a dimensão transversal.

   De forma similar, as propriedades que definem **o tamanho transversal mínimo ou máximo** (min/max cross size properties) serão '**min-width**'/'**max-width**' ou '**min-height**'/'**max-height**', dependendo de qual é a dimensão transversal.

Dado o seguinte código:

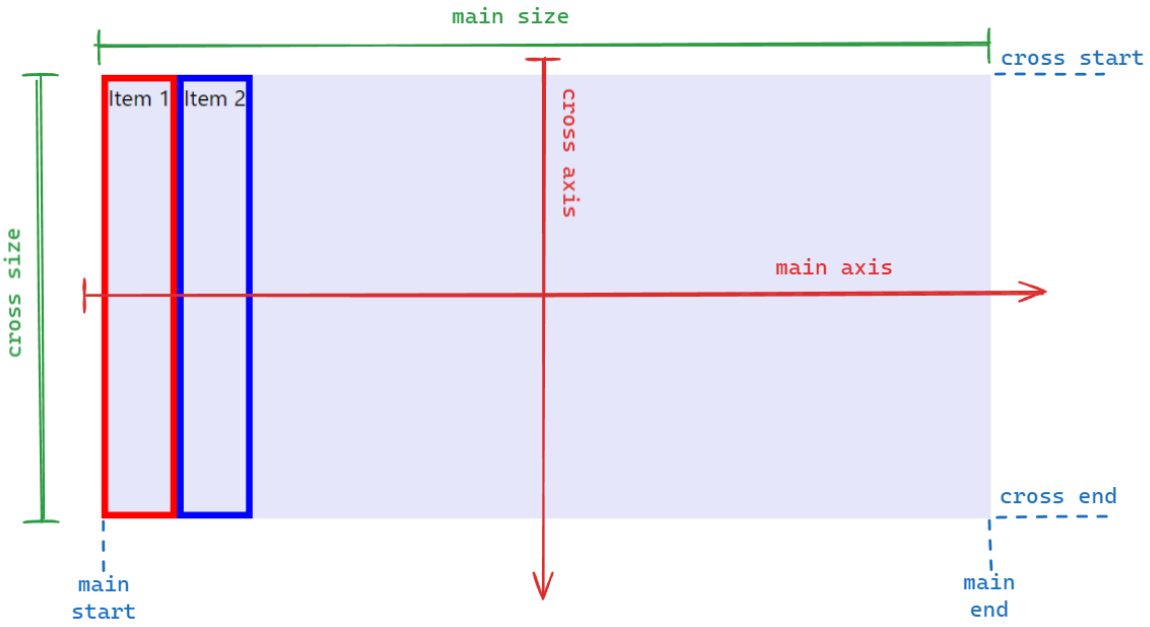
```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
  </head>
  <body>
    <div id="container" style="background-color: lavender; width: 40rem; height: 20rem">
      <div id="item-1" style="box-sizing: border-box; border: 5px solid red">Item 1</div>
      <div id="item-2" style="box-sizing: border-box; border: 5px solid blue">Item 2</div>
    </div>
  </body>
</html>
```

O flow layout inicial é:



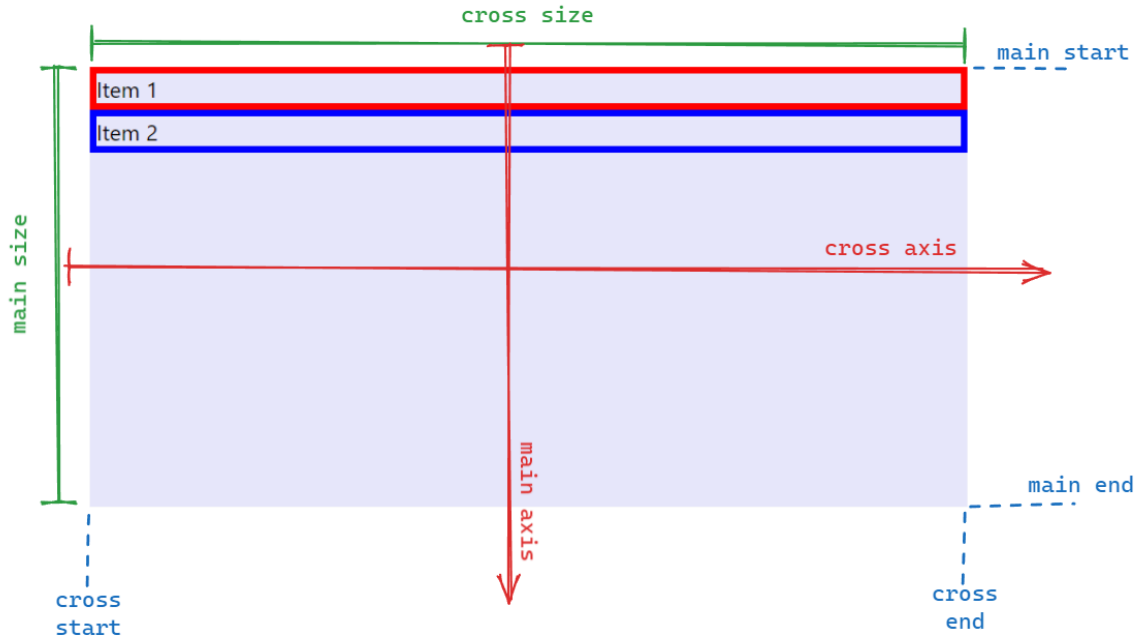
Ao aplicarmos ‘**display: flex**’ para a DIV **#container**, ficamos com:

A direção principal padrão de um flex container é ‘**row**’, logo temos as seguintes terminologias:



O main size é definido pela propriedade ‘width’, e o cross size é definido pela propriedade ‘height’

Caso definirmos a direção principal como ‘**flex-direction: column**’, temos as seguintes terminologias:



O main size é definido pela propriedade ‘height’ e o cross size é definido pela propriedade ‘width’

Flex Container

Um **flex container** é gerado pelos valores ‘**flex**’ e ‘**inline-flex**’ para a propriedade ‘**display**’:

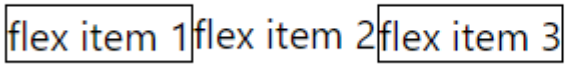
1. ‘**flex**’: o valor causa o elemento a gerar um flex container box que é block-level quando colocado no flow layout;
2. ‘**inline-flex**’: o valor causa o elemento a gerar um flex container box que é inline-level quando colocado no flow layout.

Um flex container estabelece um novo **flex formatting context** para seu conteúdo. Flex containers formam um **contaning block** para seu conteúdo da mesma maneira que block containers.

Flex Items

De maneira simplificada, os flex items de um flex container são boxes representando seu conteúdo. Cada descendente de um flex container se torna um flex item, e cada sequência contínua de texto é envolta em block box flex item anônimo.

```
<div style="display: flex">
  <div style="border: 1px solid">flex item 1</div>
  flex item 2
  <span style="border: 1px solid">flex item 3</span>
</div>
```



Um flex item estabelece um **contexto de formatação independente** para seu conteúdo. Contudo, flex items são **flex-level boxes**: eles participam no contexto de formatação de seu contêiner.

As margens adjacentes de flex items não colapsam. Valores em porcentagem para margens e padding são calculados com relação ao inline size do containing block, assim como em block boxes.

Auto-margens expandem para absorder espaço extra na direção correspondente: elas podem ser usadas para alinhamento ou para separar itens adjacentes.

Os flex items são renderizados exatamente da mesma forma que os inline block boxes, exceto que a ordem visual do documento pode ser modificada pela propriedade '**order**', e os valores de '**z-index**' diferentes de '**auto**' criam um **contexto de empilhamento** (stacking context), mesmo se a posição for 'static' (comportando-se exatamente como se a posição fosse 'relative').

Tamanho mínimo automático

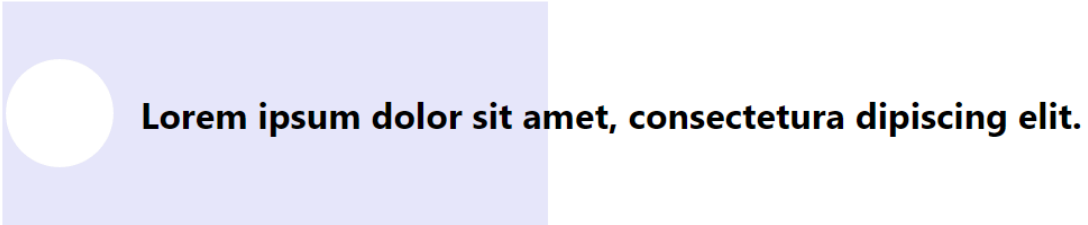
Normalmente, o menor tamanho que um box pode ter que não resulta em transbordo é denominado **min-content size**. No modelo de layout flexbox, o valor utilizado para o tamanho mínimo automático ('**min-width = auto**') de um flex item que não é um scroll contêiner é baseado no conteúdo (**content-base minimum size**).

No geral, o tamanho mínimo baseado no conteúdo de um flex item é o menor valor entre a sugestão de tamanho de conteúdo (**content size suggestion**) e a sugestão de tamanho específicado (**specified size suggestion**).

Considere um cenário onde há um componente com um título que pode ser longo. Para prevenir o elemento de ficar muito largo, queremos truncar o título e mostrar um ellipsis:

```html
<head>
  <style>
    .container {
      align-items: center;
      background-color: lavender;
      display: flex;
      gap: 0.5rem;
      width: 10rem;
    }

    .container__avatar {
      background-color: white;
      border-radius: 50%;
      flex: none;
      height: 2rem;
      width: 2rem;
    }

    .container__content {}

    .container__title {
      overflow: hidden;
      text-overflow: ellipsis;
      white-space: nowrap;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="container__avatar"></div>
    <div class="container__content">
      <h6 class="container__title">
        Lorem ipsum dolor sit amet, consectetura dipiscing elit.
      </h6>
    </div>
  </div>
</body>
```

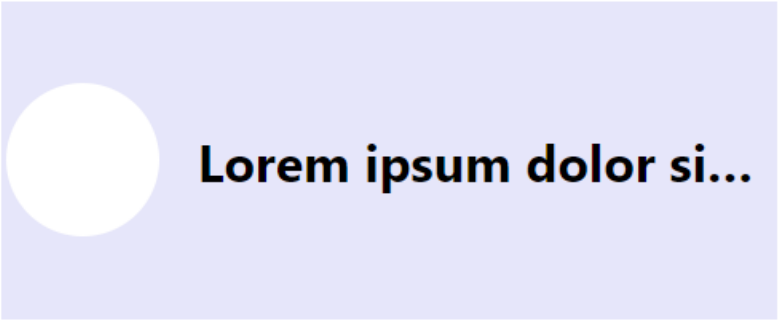
Todavia o efeito surtido não é o que esperávamos:



Porque o conteúdo em **.container__content** não está encolhendo?

O algoritmo do flexbox se recusa a reduzir um filho abaixo do tamanho mínimo. Quando o texto está em um elemento, o tamanho mínimo é determinado pelo tamanho da sequência mais longa de caracteres que não pode ser quebrada. Este tamanho é o denominado **content-size suggestion**.

Para corrigir o problema, podemos definir ‘**min-width: 0**’ no **.container__content**, o que faz com que a **specified size suggestion** seja então o menor valor:



Note que, quando o content-based sizing é utilizado num flex item, a layout engine precisa atravessar todo o conteúdo para encontrar seu tamanho mínimo, o que não acontece quando o autor define um mínimo explícito. Para itens com grandes quantidades de conteúdo, pode se tornar uma preocupação de performance.

Ordenação e Orientação

O conteúdo de um flex container pode ser disposto em qualquer direção ou ordem. Estas funcionalidades são alcançadas através das propriedades ‘**flex-direction**’, ‘**flex-wrap**’, e ‘**order**’.

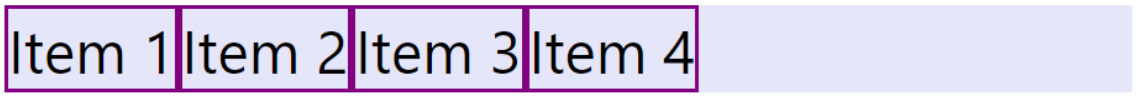
Flex flow direction: a propriedade ‘flex-direction’

nome: 'flex-direction'
valor(es): 'row' | 'row-reverse' | 'column' | 'column-reverse'
valor inicial: 'row'
aplica-se a: flex containers
herdável: não?

A propriedade ‘**flex-direction**’ determina como flex items são dispostos no flex container, definindo a direção do **eixo principal** do contêiner (main axis). Os valores aplicáveis são:

1. ‘**row**’: o **main axis** do flex container possui a mesma orientação do **inline axis** do modo de escrita em efeito. As direções ‘**main-start**’ e ‘**main-end**’ são equivalentes à ‘**inline-start**’ e ‘**inline-end**’, respectivamente;

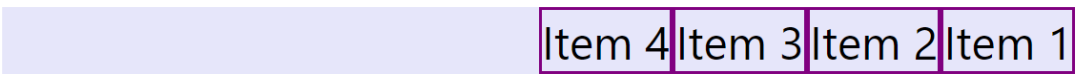
```html
<head>
  <style>
    .box {
      border: 1px solid purple;
    }

    .container {
      background-color: lavender;
      display: flex;
      flex-direction: row;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="box">Item 1</div>
    <div class="box">Item 2</div>
    <div class="box">Item 3</div>
    <div class="box">Item 4</div>
  </div>
</body>
```




```html
<head>
  <style>
    html {
      writing-mode: vertical-lr;
    }

    .box {
      border: 1px solid purple;
    }

    .container {
      background-color: lavender;
      display: flex;
      flex-direction: row;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="box">Item 1</div>
    <div class="box">Item 2</div>
    <div class="box">Item 3</div>
    <div class="box">Item 4</div>
  </div>
</body>
```

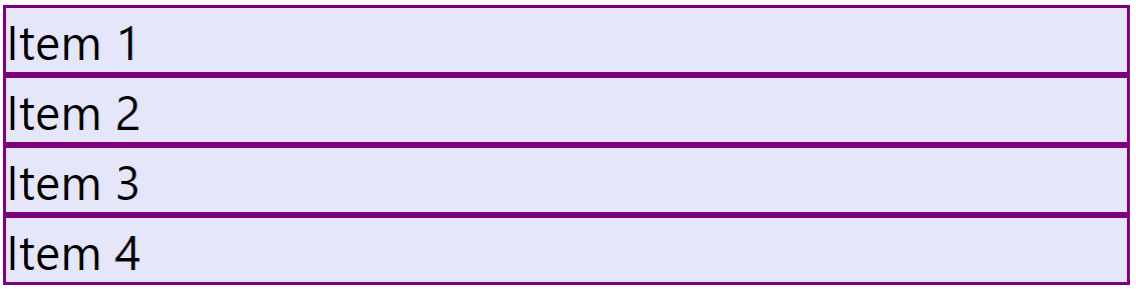
2. ‘**row-reverse**’: igual à ‘**row**’, mas as direções ‘**main-start**’ e ‘**main-end**’ são trocadas;

```
<head>
  <style>
    .box {
      border: 1px solid purple;
    }

    .container {
      background-color: lavender;
      display: flex;
      flex-direction: row-reverse;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="box">Item 1</div>
    <div class="box">Item 2</div>
    <div class="box">Item 3</div>
    <div class="box">Item 4</div>
  </div>
</body>
```



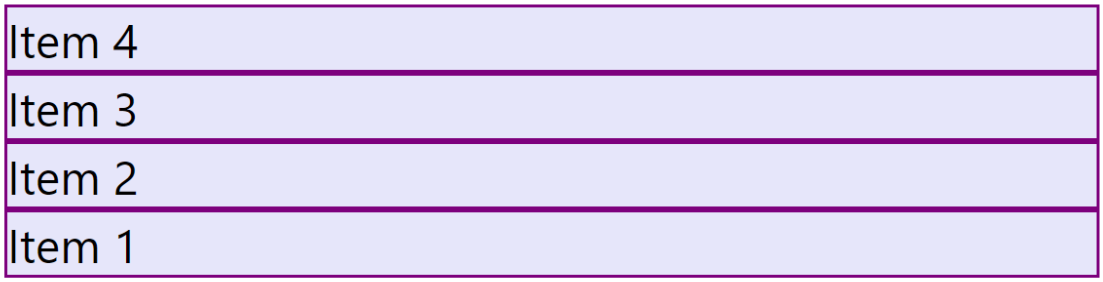
3. ‘**column**’: o main axis do flex container possui a mesma orientação que o block axis do modo de escrita em efeito. As direções ‘**main-start**’ e ‘**main-end**’ são equivalentes à ‘**block-start**’ e ‘**block-end**’;

```html
<head>
  <style>
    .box {
      border: 1px solid purple;
    }

    .container {
      background-color: lavender;
      display: flex;
      flex-direction: column;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="box">Item 1</div>
    <div class="box">Item 2</div>
    <div class="box">Item 3</div>
    <div class="box">Item 4</div>
  </div>
</body>
```



4. ‘**column-reverse**’: igual à ‘**column**’, mas as direções ‘**main-start**’ e ‘**main-end**’ são inversas;

```html
<head>
  <style>
    .box {
      border: 1px solid purple;
    }

    .container {
      background-color: lavender;
      display: flex;
      flex-direction: column-reverse;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="box">Item 1</div>
    <div class="box">Item 2</div>
    <div class="box">Item 3</div>
    <div class="box">Item 4</div>
  </div>
</body>
```



A habilidade de ordenação do flex layout intencionalmente afeta apenas a renderização visual, mantendo a ordem de fala e de navegação baseada na ordem do documento. Isso permite que os autores manipulem a apresentação visual, deixando intacta a ordem do código fonte para user agents que não trabalham com CSS, como softwares text to speech.

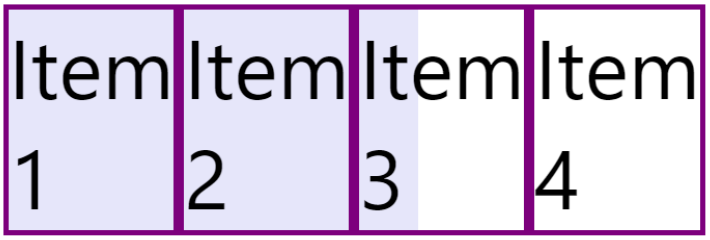
Flex Line Wrapping: a propriedade ‘flex-wrap’

nome: 'flex-wrap'
valor(es): 'nowrap' | 'wrap' | 'wrap-reverse'
valor inicial: 'nowrap'
aplica-se a: flex containers
herdável: não

A propriedade ‘**flex-wrap**’ controla se o flex container é **single-line** ou **multi-line**, e a direção da **cross axis**, o que determina a direção em que novas linhas são empilhadas. Os valores válidos são:

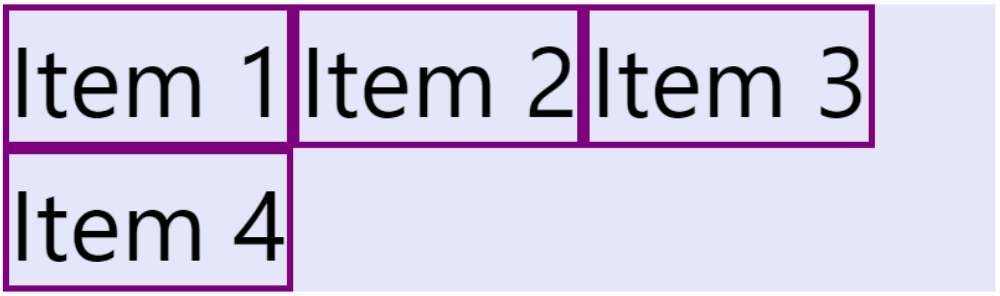
1. ‘**nowrap**’: o flex container é **single-line**;

```html
<head>
  <style>
    .box {
      border: 1px solid purple;
    }

    .container {
      background-color: lavender;
      display: flex;
      flex-wrap: nowrap;
      width: 7rem;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="box">Item 1</div>
    <div class="box">Item 2</div>
    <div class="box">Item 3</div>
    <div class="box">Item 4</div>
  </div>
</body>
```



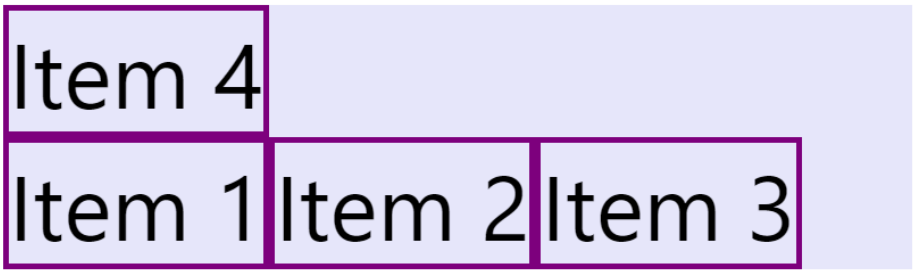
2. ‘**wrap**’: o flex container é **multi-line**;

```html
<head>
  <style>
    .box {
      border: 1px solid purple;
    }

    .container {
      background-color: lavender;
      display: flex;
      flex-wrap: wrap;
      width: 7rem;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="box">Item 1</div>
    <div class="box">Item 2</div>
    <div class="box">Item 3</div>
    <div class="box">Item 4</div>
  </div>
</body>
```



3. ‘**wrap-reverse**’: igual ao ‘**wrap**’.

```html
<head>
  <style>
    .box {
      border: 1px solid purple;
    }

    .container {
      background-color: lavender;
      display: flex;
      flex-wrap: wrap-reverse;
      width: 7rem;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="box">Item 1</div>
    <div class="box">Item 2</div>
    <div class="box">Item 3</div>
    <div class="box">Item 4</div>
  </div>
</body>
```



A propriedade '**flex-flow**' é um atalho que permite definir a '**flex-direction**' e '**flex-wrap**', que juntas definem os eixos principal e transversal do flex container.

Flex Lines

Flex items em um flex container são dispostos e alinhados em **flex lines**, contêineres hipotéticos utilizados pelo algoritmo de layout para agrupar e alinhar. Um flex container pode ser **single-line** ou **multi-line**, dependendo do valor da propriedade '**flex-wrap**':

1. Um **single-line flex container** dispõe todos os seus descendentes em uma única linha, mesmo que isso cause o overflow do conteúdo;

2. Um **multi-line flex container** quebra seus flex items em múltiplas linhas, parecido como texto é quebrado em uma nova linha caso seja muito longo para caber na linha atual. Quando linhas adicionais são criadas, elas são espilhadas no flex container no eixo transversal (cross axis) de acordo com

o valor de ‘flex-wrap’. Toda linha contém pelo menos um flex item, a menos que o contêiner esteja vazio.

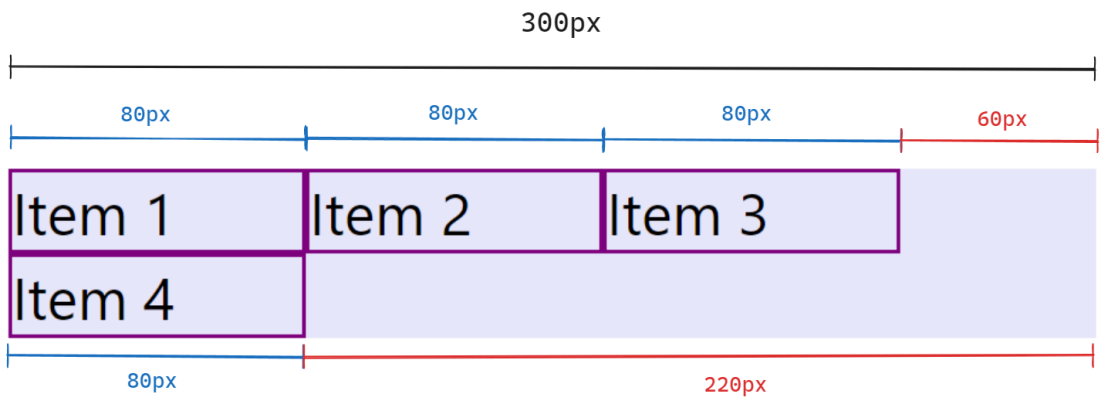
Uma vez que o conteúdo é disposto em linhas, cada linha é disposta de maneira independente. Tamanhos flexíveis e as propriedades ‘**justify-content**’ e ‘**align-self**’ consideram os itens uma linha por vez.

Em um multi-line flex container (mesmo os que possuem apenas uma única linha), o cross size de cada linha é o tamanho mínimo necessário para conter os flex items (depois de alinhados pelo ‘**align-self**’), e as linhas em si são alinhadas com a propriedade ‘**align-content**’. Em um single-line flex container, o cross size da linha é o cross size do flex container, e ‘**align-content**’ não tem efeito. O **main size** de uma linha é sempre igual ao main size do content box do flex container.

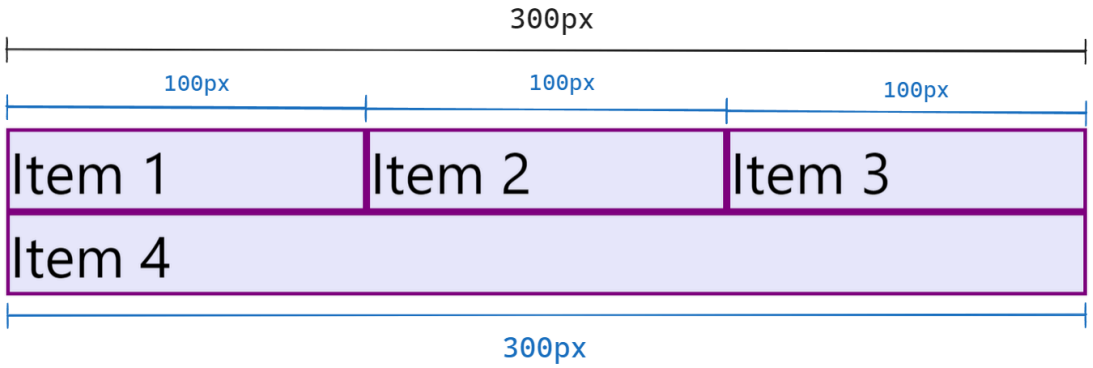
Dado o seguinte código:

```
<head>
  <style>
    .box {
      border: 1px solid purple;
      width: 80px;
    }

    .container {
      background-color: lavender;
      display: flex;
      flex-flow: row wrap;
      width: 300px;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="box">Item 1</div>
    <div class="box">Item 2</div>
    <div class="box">Item 3</div>
    <div class="box">Item 4</div>
  </div>
</body>
```

Como o container possui uma largura de 300px, apenas três itens cabem em uma única linha. Eles ocupam 240px, sobrando 60px de espaço disponível. Porque a propriedade ‘**flex-flow**’ especifica um contêiner multi-line, o flex container criará uma linha adicional para conter o último item:



Agora, todos os flex items receberam '**flex: auto**'. A primeira linha possui 60px de espaço restante, e todos os itens possuem a mesma flexibilidade, portanto cada um receberá 20px extras de largura, resultando em 100px. O item restante está por si próprio em uma linha separada, e irá esticar para a largura do flex container inteiro:



Flexibilidade

O aspecto que define o flex layout é a sua habilidade de fazer os flex items "flexibilizar", alterando sua largura/altura para preencher o espaço disponível na dimensão principal (main dimension). Isso é feito através da propriedade '**flex**'. Um flex container **distribui espaço livre** para seus descendentes (de maneira proporcional ao seu **flex grow factor**) para preencher o container, ou os **encolhe** (proporcional ao seu **flex shrink factor**) para prevenir o overflow.

Um flex item é **totalmente inflexível** se ambas suas propriedades '**flex-grow**' e '**flex-shrink**' são **zero**, caso contrário são denominados **flexíveis**.

A propriedade ‘flex’

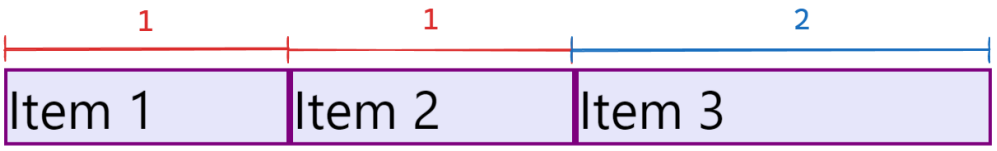
nome: 'flex'
valor(es): 'none' | [<flex-grow> <flex-shrink>? || <flex-basis>]
valor inicial: 0 1 auto
aplica-se a: flex items
herdável: não

A propriedade ‘**flex**’ determina os componentes de um **tamanho flexível**: os **flex factors** (**grow** e **shrink**) e a **flex basis**. Quando um box é um flex item, ‘**flex**’ é consultado ao invés da **main size property** para determinar o **main size** do box.

Componentes

flex-grow
Este número define a **flex grow factor**, que determina quanto o flex item irá **crescer** com relação ao resto dos itens no contêiner, quando o espaço disponível positivo é distribuído. Quando omitido, o valor é **1**.

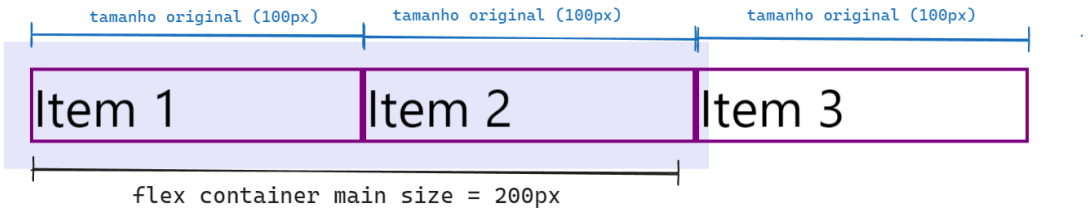
```
<head>
  <style>
    .box {
      border: 1px solid purple;
      flex-basis: auto;
      flex-grow: 1;
    }

    .container {
      background-color: lavender;
      display: flex;
      width: 300px;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="box">Item 1</div>
    <div class="box">Item 2</div>
    <div class="box" style="flex-grow: 2">Item 3</div>
  </div>
</body>
```

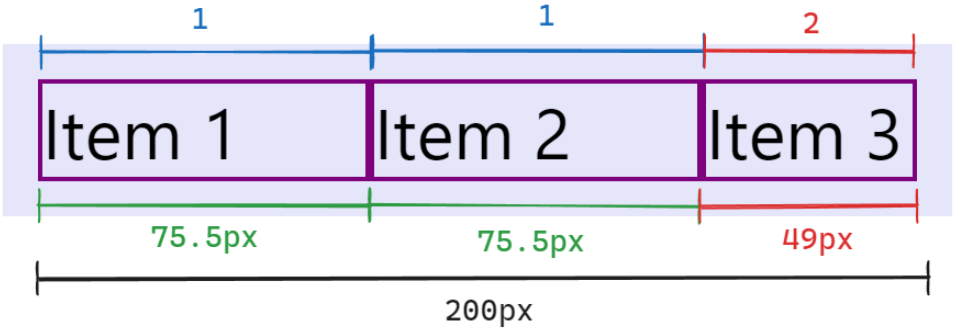


flex-shrink

Este número define o **flex shrink factor**, que define quanto o flex item irá **encolher** com relação ao resto dos itens quando o espaço disponível negativo é distribuído. Quando omitido, seu valor é **1**.

```
<head>
  <style>
    .box {
      border: 1px solid purple;
      flex-basis: 100px;
      flex-shrink: 1;
    }

    .container {
      background-color: lavender;
      display: flex;
      padding: 8px;
      width: 200px;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="box">Item 1</div>
    <div class="box">Item 2</div>
    <div class="box" style="flex-shrink: 2">Item 3</div>
  </div>
</body>
```





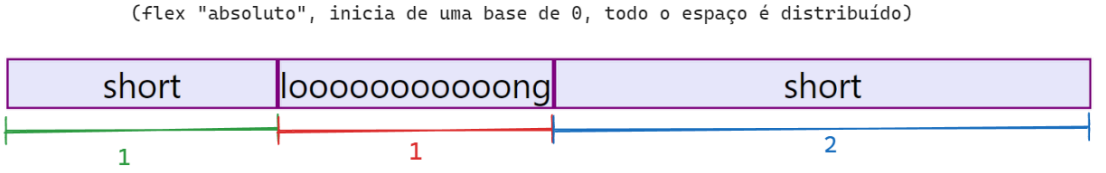
flex-basis

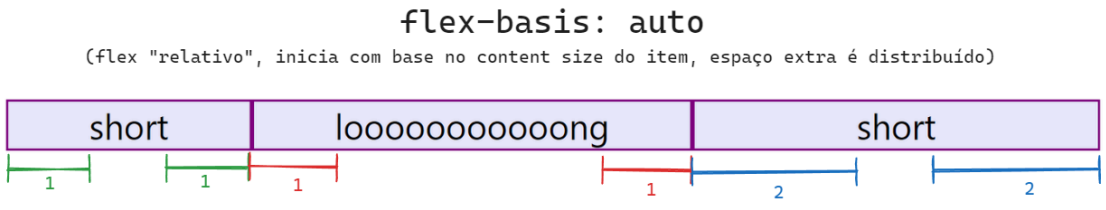
Esta propriedade define a **flex basis**, o **main size inicial** do flex item, antes do espaço livre disponível ser distribuído de acordo com os fatores flex. '**flex-basis**' aceita os mesmo valores que '**width**' e '**height**':

1. '**auto**': recupera o valor da **main size property** para ser utilizada como '**flex-basis**';
2. **<number>**: '**flex-basis**' é calculado da mesma maneira que '**width**' ou '**height**'.

Quando omitido, o valor inicial é **0**.

```html
<head>
  <style>
    .box {
      text-align: center;
      border: 1px solid purple;
      flex-basis: 0;
      flex-grow: 1;
    }

    .container {
      background-color: lavender;
      display: flex;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="box">short</div>
    <div class="box">looooooooooong</div>
    <div class="box" style="flex-grow: 2">short</div>
  </div>
</body>
```





Valores

‘**flex: initial**’
Equivalente a ‘**flex: 0 1 auto**’. Dimensiona o item com base nas propriedades ‘**width**’ e ‘**height**’. Se a **main size property** é ‘**auto**’, então o item será redimensionado com base no seu **conteúdo**. Faz com que o item seja inflexível quando há espaço disponível, mas permite encolher a um tamanho mínimo quando não há espaço suficiente.

‘**flex: auto**’
Equivalente a ‘**flex: 1 1 auto**’. Dimensiona o item com base nas propriedades ‘**width**’ e ‘**height**’, mas faz com que este seja **totalmente flexível**, para que possa “absorver” qualquer espaço disponível na **main axis**.

‘**flex: none**’
Equivalente a ‘**flex: 0 0 auto**’. Dimensiona o tem com base nas propriedades ‘**width**’ e ‘**height**’, mas faz com que este seja **totalmente inflexível**. É similar a ‘**initial**’, com exceção de que os itens não são permitidos encolher, resultando em overflow.

‘**flex: <positive-number>**’
Equivalente à ‘**flex: <positive-number> 1 0**’. Torna o flex item flexível e define a flex basis como zero, resultando em um item que recebe a proporção especificada de espaço disponível do flex container.

Alinhamento

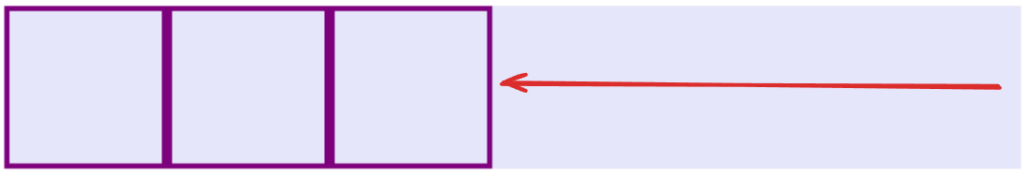
Depois que o conteúdo de um flex container terminou de ser dimensionado, este pode então ser alinhado dentro do contêiner.

Alinhamento no eixo principal: a propriedade ‘justify-content’

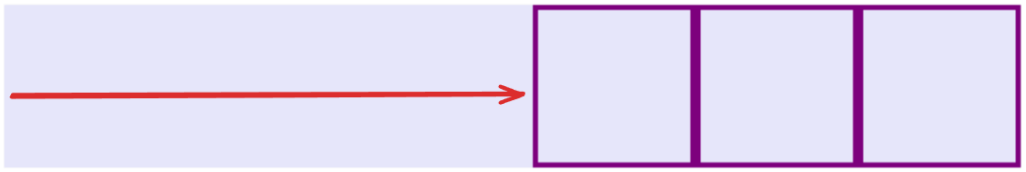
nome: 'justify-content'
valor(es): 'flex-start'|'flex-end'|'center'|'space-between'|'space-around'
valor inicial: 'flex-start'
aplica-se a: flex containers
herdável: não

A propriedade ‘**justify-content**’ alinha os flex items no **eixo principal** da linha atual do contêiner. Isso é feito depois que todos os tamanhos flexíveis são calculados.

justify-content: flex-start

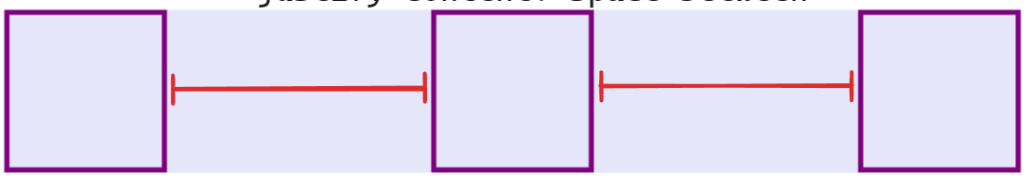


justify-content: flex-end

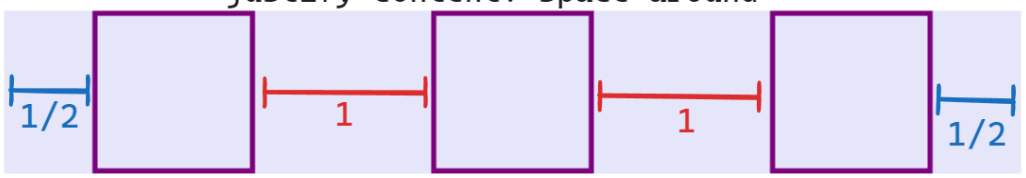


justify-content: center

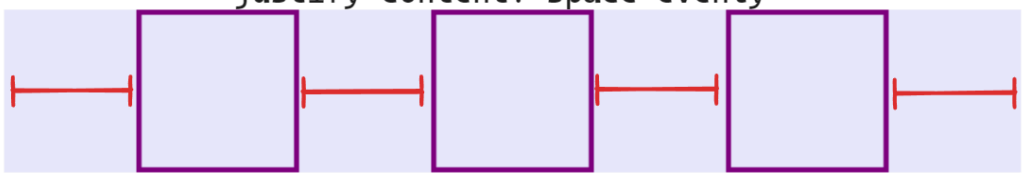
justify-content: space-between



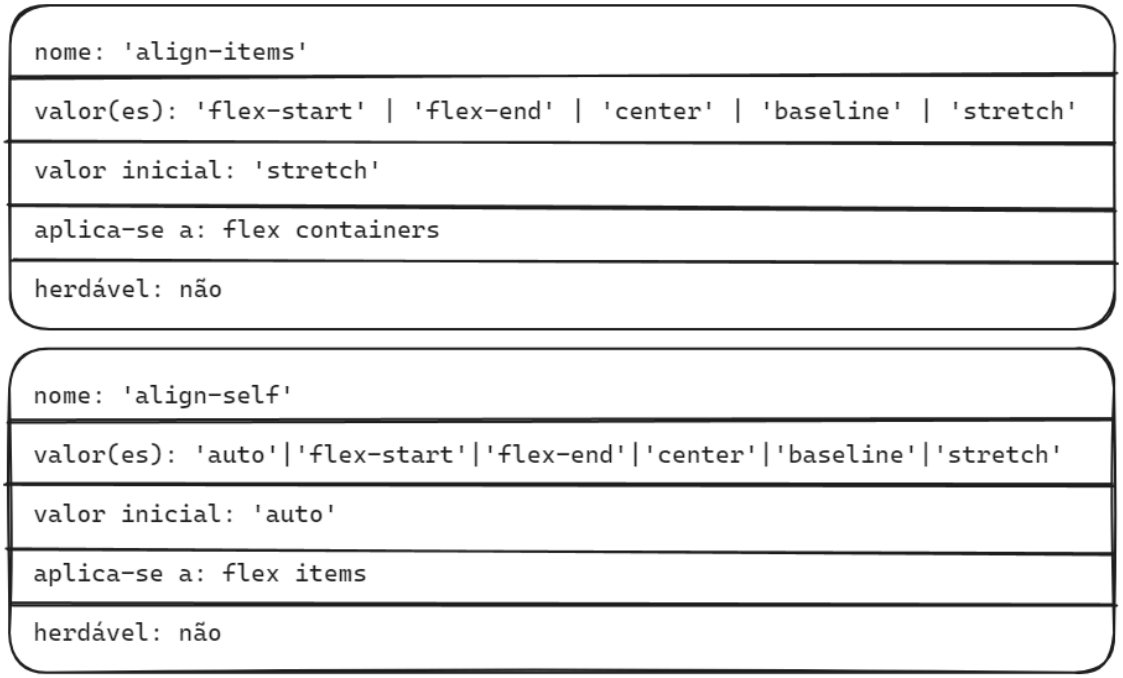
justify-content: space-around



justify-content: space-evenly

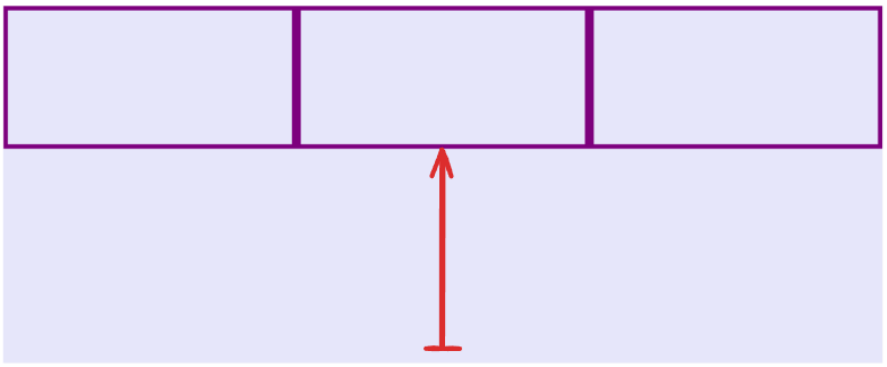


Alinhamento no eixo transversal: as propriedades ‘align-items’ e ‘align-self’

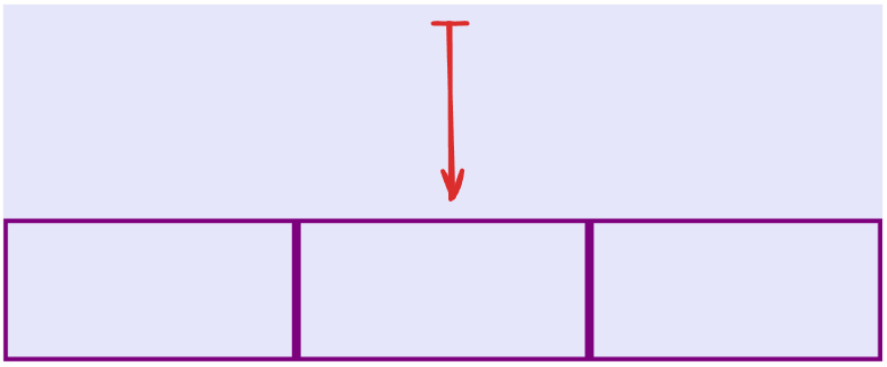


Flex items podem ser alinhados no eixo transversal da mesma maneira que ‘**justify-content**’, mas na direção perpendicular. ‘**align-items**’ define o alinhamento padrão de **todos** os itens, e ‘**align-self**’ permite que esse alinhamento padrão seja substituído **individualmente** para cada flex item.

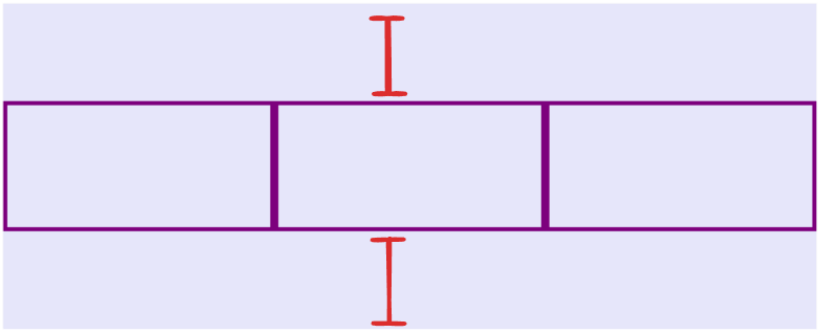
align-items: flex-start



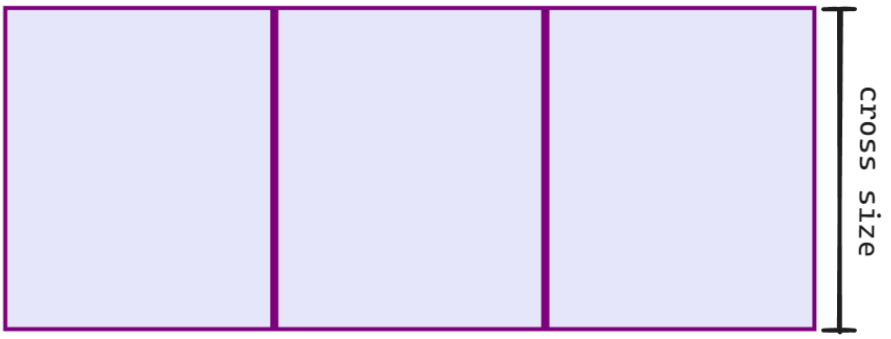
align-items: flex-end



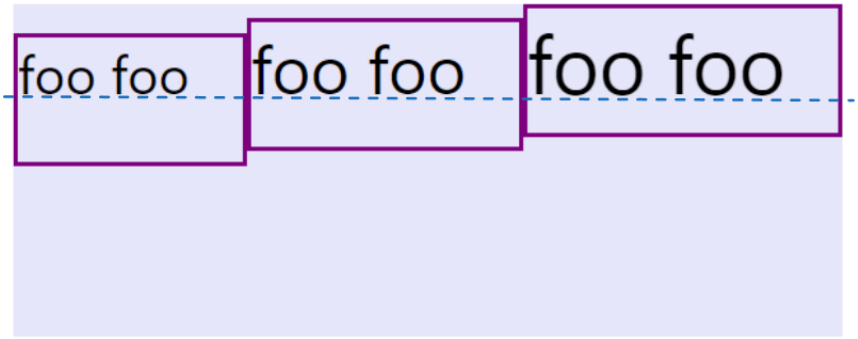
align-items: center



align-items: stretch



align-items: baseline

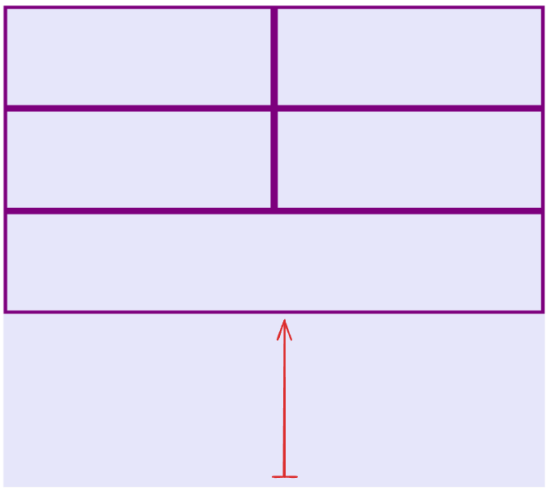


Alinhamento de Flex Lines: a propriedade ‘align-content’

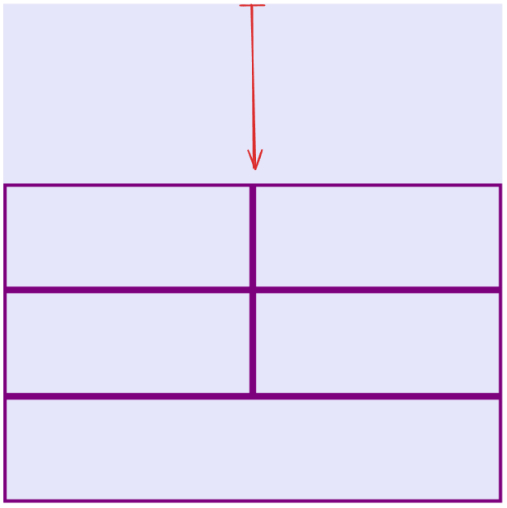
nome: 'align-content'
valor(es): 'flex-start'|'flex-end'|'center'|'space-between'|'space-around'|'stretch'
valor inicial: 'stretch'
aplica-se a: multi-line flex containers
herdável: não

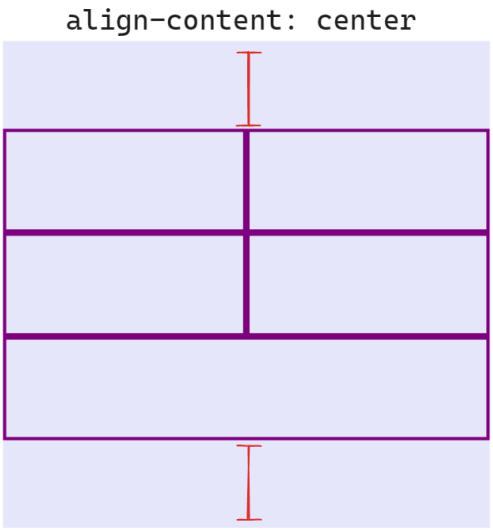
A propriedade ‘align-content’ alinha as linhas do flex container quando há espaço disponível no eixo transversal. Ela só possui efeito quando o contêiner possui mais de uma linha.

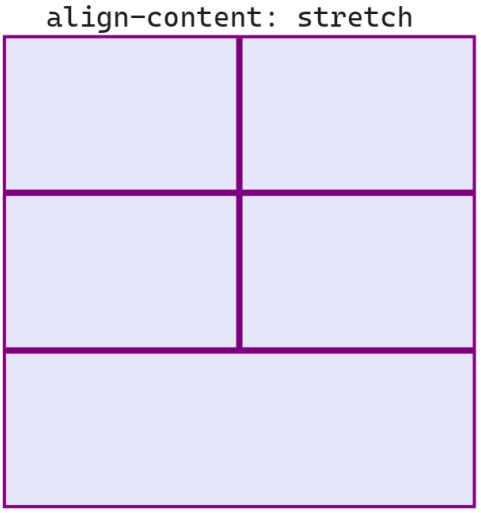
align-content: flex-start

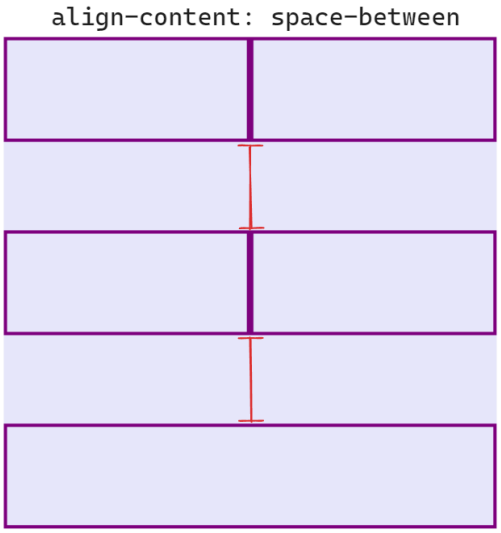


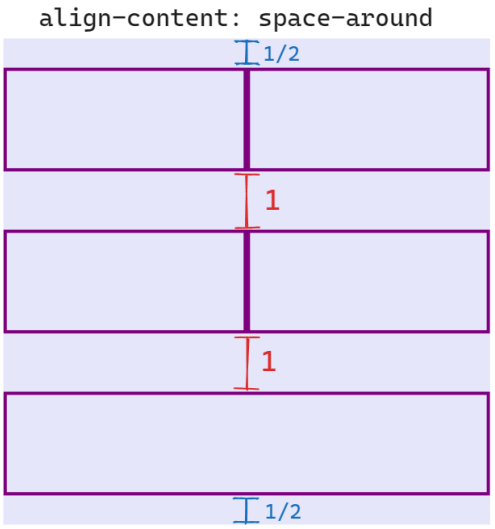
align-content: flex-end











Referências

https://www.w3.org/TR/css-flexbox-1/#intro
https://www.bigbinary.com/blog/understanding-the-automatic-minimum-size-of-flex-items