# Layout CSS avançado com flexbox

## 1. Introdução ao Flexbox

Flexbox é uma maneira de organizar itens em linhas ou colunas. Esses itens serão flexíveis (ou seja, aumentarão ou diminuirão) com base em algumas regras simples que você pode definir. Para começar, vamos ver uma demonstração simples. Para todos os exercícios aqui, reserve um tempo para inspecionar o código e realmente entender o que está acontecendo. Na verdade, trabalhar você mesmo com o código tornará muito mais fácil reter essas informações.

index.html

```
<div class="flex-container">
    <div class="one"></div>
    <div class="two"></div>
    <div class="three"></div>
  </div>
```

styles.css

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

  /* este seletor seleciona todas as divs dentro de
.flex-container */
  .flex-container div {
    background: peachpuff;
    border: 4px solid brown;
    height: 100px;
    /* flex: 1; */
  }
```

Nós vamos entrar em exatamente o que está acontecendo aqui em breve. Mas, por enquanto, vamos descomentar as duas declarações CSS relacionadas ao flex acima removendo as tags `/*` e `*/` que as cercam e, em seguida, confira o resultado.

Todos os 3 divs agora devem ser organizados horizontalmente. Se você redimensionar o quadro de resultados com os botões “1x”, “.5x” e “.25x” você também verá que os divs serão 'flexionados'. Eles preencherão a área disponível e cada um terá a mesma largura.

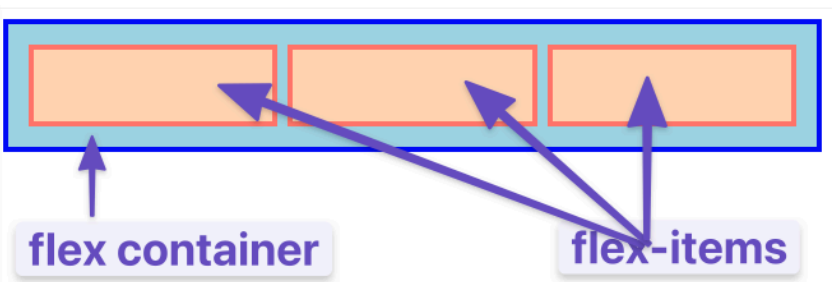
Se você adicionar outro div ao HTML, dentro do `.flex-container`, ele aparecerá ao lado dos outros, e tudo será flexível para caber na área disponível.

### 1.1. Contêineres Flex e Itens Flex

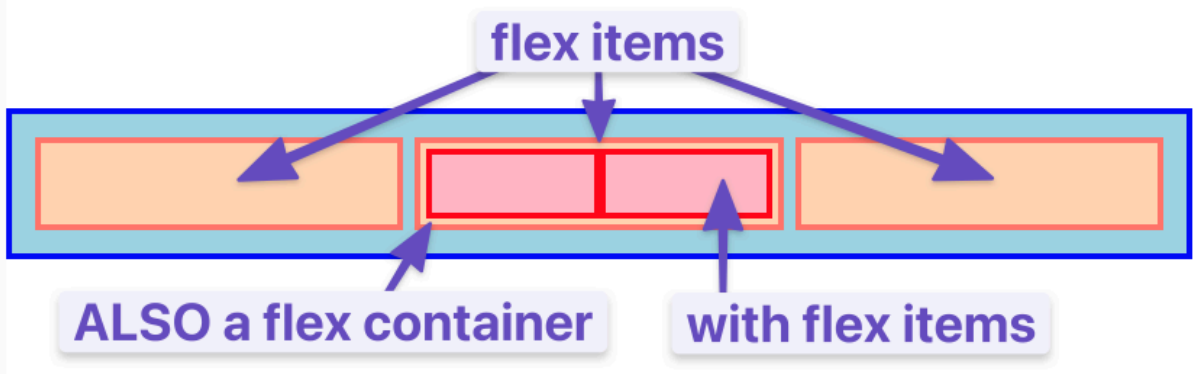
Como você viu, flexbox não é apenas uma única propriedade CSS, mas uma caixa de ferramentas inteira de propriedades que você pode usar para colocar as coisas onde precisar delas. Algumas dessas propriedades pertencem ao contêiner flex, enquanto algumas vão aos itens flex. Este é um conceito simples, mas importante.

Um contêiner flex é qualquer elemento que tenha `display: flex` nele. Um item flex é qualquer elemento que vive diretamente dentro de um contêiner flex.

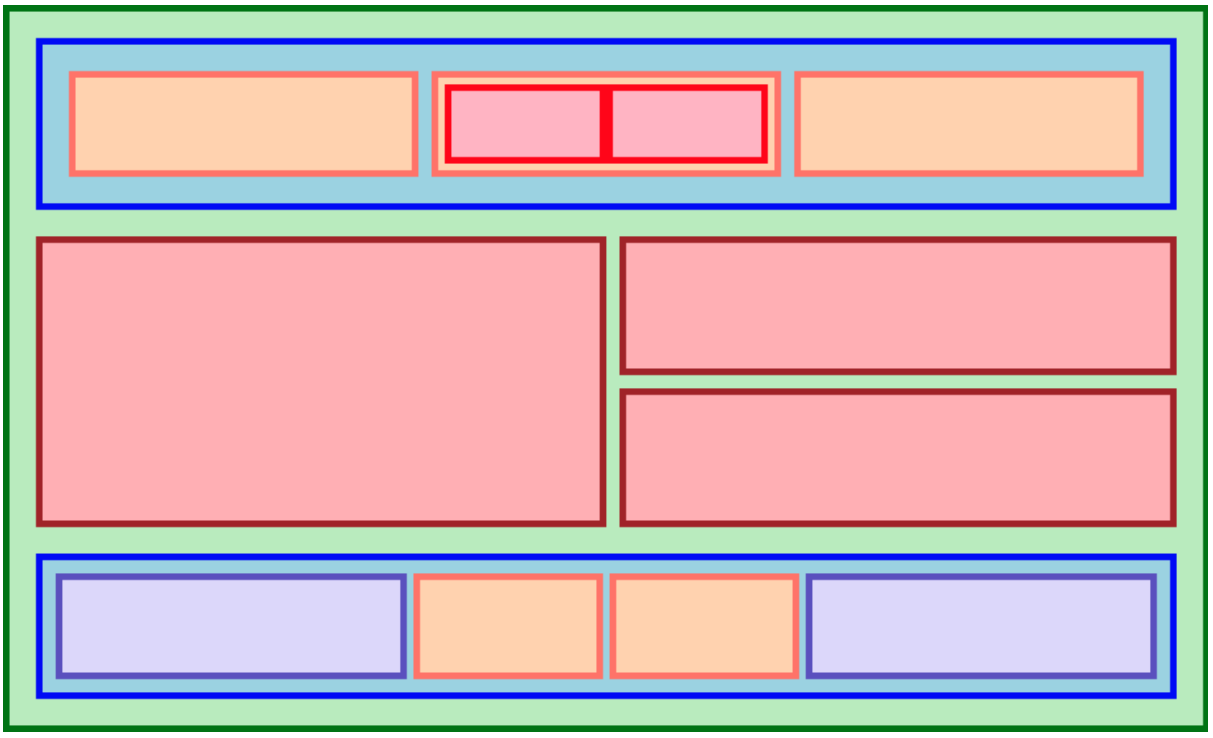
```
.container {
  background: lightblue;
  padding: 16px;
  border: 4px solid blue;
  gap: 8px;
  display: flex;
}

.item {
  background: peachpuff;
  border: 4px solid salmon;
  height: 48px;
  flex: 1;
  padding: 4px;
}
```




Um tanto confuso, qualquer elemento pode ser um contêiner flex e um item flex. Dito de outra forma, você também pode colocar display: flex em um item flex e então



Criar e aninhar vários contêineres e itens flexíveis é a principal maneira de construir layouts complexos. A imagem a seguir foi obtida usando apenas flexbox para organizar, dimensionar e colocar os vários elementos. Flexbox é uma ferramenta muito poderosa.



## 2. Aumentando e diminuindo

Vamos olhar um pouco mais de perto o que realmente aconteceu quando você colocou `flex: 1` nesses itens flex na última lição.

### 2.1. Abreviação(shorthand) flex

A declaração `flex` é na verdade uma abreviação de 3 propriedades que você pode definir em um item flex. Essas propriedades afetam como os itens flexíveis se dimensionam dentro de seu contêiner. Você já viu algumas propriedades abreviadas antes, mas ainda não as definimos oficialmente.

*As propriedades abreviadas (shorthands) são propriedades CSS que permitem definir os valores de várias outras propriedades CSS simultaneamente. Usando uma propriedade abreviada, você pode escrever folhas de estilo mais concisas (e geralmente mais legíveis), economizando tempo e energia.*

Neste caso, `flex` é na verdade uma abreviação para `flex-grow`, `flex-shrink` e `flex-basis`.

```
div {
  flex: 1;
}
```

Na captura de tela acima, `flex: 1` equivale a: `flex-grow: 1`, `flex-shrink: 1`, `flex-basis: 0`.

Muitas vezes você vê a abreviação flex definida com apenas um valor. Nesse caso, esse valor é aplicado ao `flex-grow`. Então, quando colocamos `flex: 1` em nossas divs, na verdade estávamos especificando uma abreviação de `flex: 1 1 0`.

**Flex-Grow**

`flex-grow` espera um único número como seu valor, e esse número é usado como o "fator de crescimento" do flex-item. Quando aplicamos `flex: 1` a cada div dentro de nosso contêiner, estávamos dizendo a cada div para crescer a mesma quantidade. O resultado disso é que cada div acaba exatamente do mesmo tamanho. Se, em vez disso, adicionarmos `flex: 2` a apenas uma das divs, essa div cresceria para 2x o tamanho das outras.

No exemplo a seguir, a abreviação `flex` tem valores para `flex-shrink` e `flex-basis` especificados com seus valores padrão.

index.html

```
<div class="flex-container">
```

```
    <div class="one"></div>
    <div class="two"></div>
    <div class="three"></div>
  </div>
```

styles.css

```
.flex-container {
    display: flex;
  }

  /* este seletor seleciona todas as divs dentro de
.flex-container */
  .flex-container div {
    background: peachpuff;
    border: 4px solid brown;
    height: 100px;
    flex: 1 1 0%;
  }

  /* apenas div.two está selecionado aqui */
  .flex-container .two {
    flex: 2 1 0%;
  }
```

**Flex-Shrink**

`flex-shrink` é semelhante a `flex-grow`, mas define o "fator de redução" de um item flex. `flex-shrink` acaba sendo aplicado apenas se o tamanho de todos os itens flexíveis for maior que seu contêiner pai. Por exemplo, se nossos 3 divs acima tivessem uma declaração de largura como: `width: 100px`, e `.flex-container` fosse menor que `300px`, nossos divs teriam que encolher para caber.

O fator de redução padrão é `flex-shrink: 1`, o que significa que todos os itens serão reduzidos uniformemente. Se você não deseja que um item seja reduzido, você pode especificar `flex-shrink: 0`;. Você também pode especificar números

mais altos para fazer com que certos itens encolham a uma taxa maior do que o normal.

Aqui está um exemplo. Observe que também alteramos o `flex-basis` por motivos que serão explicados em breve. Se você encolher a janela do seu navegador, notará que `.two` nunca fica menor que a largura especificada de 250px, mesmo que a regra `flex-grow` especifique que cada elemento deve ter o mesmo tamanho.

index.html

```
<div class="flex-container">
    <div class="one"></div>
    <div class="two"></div>
    <div class="three"></div>
  </div>
```

styles.css

```
.flex-container {
    display: flex;
  }

  /* este seletor seleciona todas as divs dentro de
.flex-container */
  .flex-container div {
    background: peachpuff;
    border: 4px solid brown;
    height: 100px;
    width: 250px;
    flex: 1 1 auto;
  }

  .flex-container .two {
    flex-shrink: 0;
  }
```

Uma implicação importante a ser observada aqui é que, quando você especifica `flex-grow` ou `flex-shrink`, os itens flex não respeitam necessariamente os valores fornecidos por `width`. No exemplo acima, todos os 3 divs recebem uma largura de 250px, mas quando seu pai é grande o suficiente, eles crescem para preenchê-lo. Da mesma forma, quando o pai é muito pequeno, o comportamento padrão é encolher para caber. Isso não é um bug, mas pode ser um comportamento confuso se você não estiver esperando por isso.

**Flex-Basis**

`flex-basis` simplesmente define o tamanho inicial de um item flex, então qualquer tipo de efeito de `flex-grow` ou `flex-shrink` começa a partir desse patamar. O valor abreviado é padronizado para `flex-basis: 0%`. A razão pela qual tivemos que alterá-lo para `auto` no exemplo de `flex-shrink` é que com a base definida como `0`, esses itens ignorariam a largura do item e tudo diminuiria uniformemente. Usar auto como base flexível diz ao item para verificar uma declaração de largura (`width: 250px`).

**Nota importante sobre Flex-Basis:**
*" Há uma diferença entre o valor padrão de `flex-basis` e a forma como a abreviação `flex` o define se nenhum `flex-basis` for fornecido. O valor padrão real para `flex-basis` é auto, mas quando você especifica `flex: 1` em um elemento, ele interpreta isso como `flex: 1 1 0`. Se você quiser apenas ajustar o `flex-grow` de um item, você pode simplesmente fazê-lo diretamente, sem abreviatura. Ou você pode ser mais detalhado e usar a abreviação completa de 3 valores `flex: 1 1 auto`, que também é equivalente a usar `flex: auto`."*

**O que é flex: auto?**
"Se você notou, mencionamos um novo flex abreviado `flex: auto` na nota anterior. No entanto, não o introduzimos totalmente. `flex: auto` é uma das abreviações de flex. Quando `auto` é definido como uma palavra-chave flex, é equivalente aos valores de `flex-grow: 1`, `flex-shrink: 1` e `flex-basis: auto` ou para `flex: 1 1 auto` usando a abreviação flex. Observe que `flex: auto` não é o valor padrão ao usar a abreviação flex, apesar do nome ser "auto", o que pode ser um pouco confuso no início. Você encontrará e aprenderá mais sobre `flex: auto` e seus possíveis casos de uso ao ler a seção de atribuição."

**Na prática...**

você provavelmente não usará valores complexos para `flex-grow`, `flex-shrink` ou `flex-basis`. Geralmente, é mais provável que você use declarações como `flex: 1`; para fazer divs crescerem uniformemente e `flex-shrink: 0` para evitar que certas divs diminuam.

É possível que fique chique configurar layouts onde algumas colunas se relacionam entre si em uma proporção específica, por isso é útil saber que você pode usar outros valores, mas esses são relativamente raros.

### 3. Eixos

A coisa mais confusa sobre o flexbox é que ele pode funcionar horizontalmente ou verticalmente, e a maneira como algumas regras funcionam muda um pouco dependendo da direção em que você está trabalhando.

A direção padrão para um contêiner flexível é horizontal ou `row`, mas você pode alterar a direção para vertical ou `column`. A direção pode ser especificada em CSS assim:

```
.flex-container {
  flex-direction: column;
}
```

Não importa em qual direção você esteja usando, você precisa pensar em seus contêineres flexíveis como tendo 2 eixos: o eixo principal e o eixo transversal. É a direção desses eixos que muda quando a direção flexível é alterada. Na maioria das circunstâncias, `flex-direction: row` coloca o eixo principal na horizontal (da esquerda para a direita) e `column` coloca o eixo principal na vertical (de cima para baixo).

Em outras palavras, em nosso primeiro exemplo, colocamos `display: flex` em uma div e ela organizou seus filhos horizontalmente. Esta é uma demonstração de `flex-direction: row`, a configuração padrão. O exemplo a seguir é muito semelhante. Se você descomentar a linha que diz `flex-direction: column`, esses divs serão empilhados verticalmente.

index.html

```
```

```
<div class="flex-container">
    <div class="one"></div>
    <div class="two"></div>
    <div class="three"></div>
  </div>
```

styles.css

```
.flex-container {
  display: flex;
  /* flex-direction: column; */
}

/* este seletor seleciona todas as divs dentro de
.flex-container */
.flex-container div {
  background: peachpuff;
  border: 4px solid brown;
  height: 80px;
  flex: 1 1 auto;
}
```

Uma coisa a notar é que neste exemplo, `flex-direction: column` não funcionaria como esperado se usássemos o atalho `flex: 1`. Experimente agora (ou seja, altere o valor flex no `flex: 1 1 auto`) . Você pode descobrir por que não funciona se `flex: 1` for usado? As divs colapsam, embora tenham claramente altura `height` definida ali.

A razão para isso é que a abreviação flex expande `flex-basis` para `0`, o que significa que todos os `flex-grow` e `flex-shrink` começariam seus cálculos a partir de `0`. Divs vazias por padrão têm 0 height, então para nossos itens flex serem preenchidos a altura de seu contêiner, eles não precisam ter nenhuma altura.

O exemplo acima corrigiu isso especificando `flex: 1 1 auto`, informando aos itens flex para serem padronizados com a altura `height` fornecida. Também poderíamos corrigi-lo colocando uma altura no `.flex-container` pai, ou usando `flex-grow: 1` em vez da abreviação.

Outro detalhe a ser observado: quando alteramos o direção flex para `column`, `flex-basis` refere-se a altura `height` ao invés de largura `width`. Dado o contexto, isso pode ser óbvio, mas é algo para se estar ciente.

Nós nos desviamos um pouco do ponto... Estávamos falando sobre direção flexível e eixos. Para trazê-lo ao assunto, o comportamento padrão é `flex-direction: row` que organiza as coisas horizontalmente. A razão pela qual isso geralmente funciona bem sem alterar outros detalhes no CSS é porque os elementos em nível de bloco são padronizados para a largura total de seu pai. Alterar as coisas para vertical usando `flex-direction: column` adiciona complexidade porque os elementos de nível de bloco são padronizados para a altura de seu conteúdo e, neste caso, não há conteúdo.

## 4. Alinhamento

Os itens dentro de um contêiner flex podem alinhar tanto vertical quanto horizontalmente.

Vejamos um exemplo:

index.html

```
<div class="container">
    <div class="item"></div>
    <div class="item"></div>
    <div class="item"></div>
</div>
```

styles.css

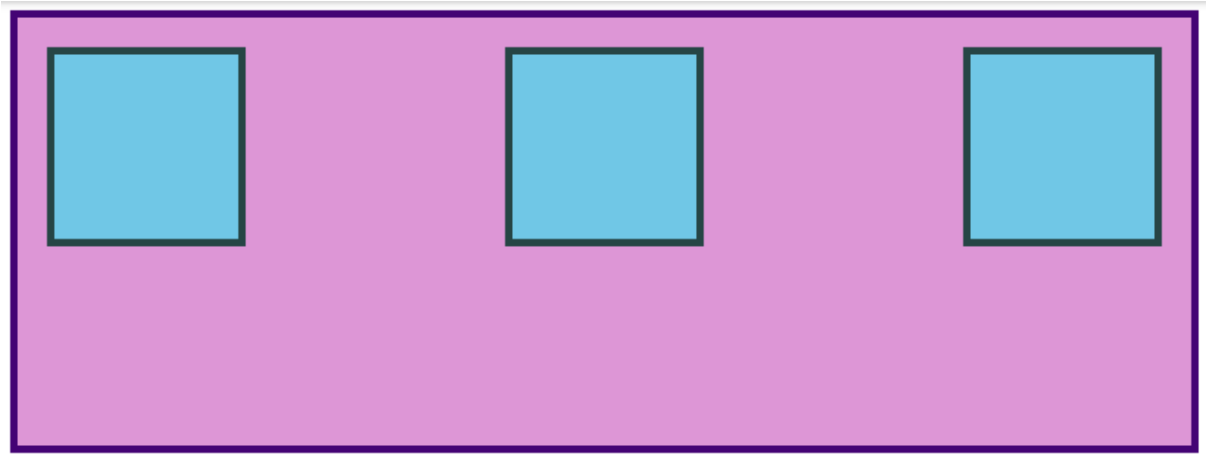
```
.container {
  height: 140px;
  padding: 16px;
  background: plum;
  border: 4px solid indigo;
  display: flex;
}
```

```
.item {
  width: 60px;
  height: 60px;
  border: 4px solid darkslategray;
  background: skyblue;
}
```

Você deve ser capaz de prever o que acontece se você colocar `flex: 1` no seletor `.item`. Faça este teste antes de seguirmos em frente!

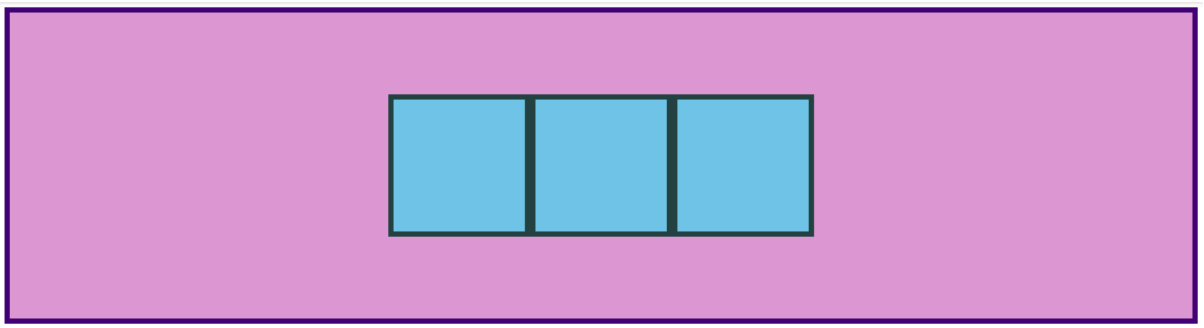
Adicionar `flex: 1` a `.item` faz com que cada um dos itens cresça para preencher o espaço disponível, mas e se quiséssemos que eles ficassem com a mesma largura, mas se distribuíssem de forma diferente dentro do container? Nós podemos fazer isso!

Remova `flex:` 1 de `.item` e adicione `justify-content: space-between` a `.container`. Fazê-lo deve dar-lhe algo assim:



`justify-content` alinha os itens no eixo principal. Existem alguns valores que você pode usar aqui. Você aprenderá o restante deles abaixo, mas por enquanto tente alterá-lo para `center`, que deve centralizar as caixas ao longo do eixo principal.

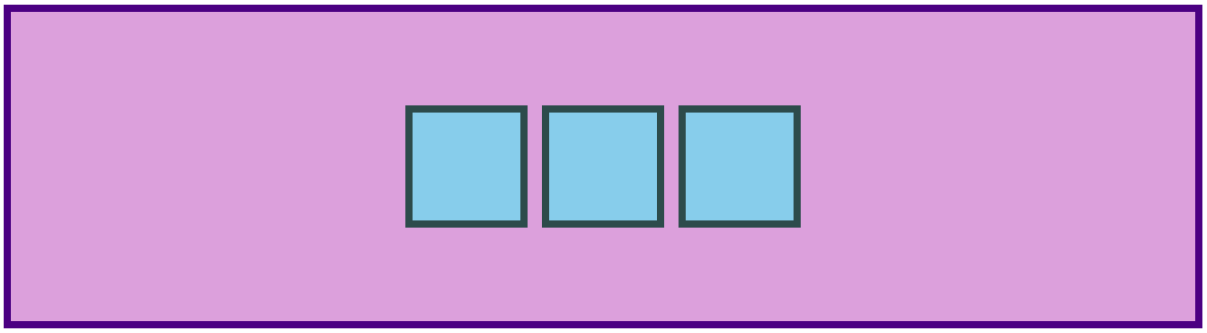
Para alterar o posicionamento dos itens ao longo do eixo transversal, use `align-items`. Tente colocar as caixas no centro do contêiner adicionando `align-items: center` a `.container`. O resultado desejado fica assim:



Como `justify-content` e `align-items` são baseados no eixo principal e transversal do seu contêiner, seu comportamento muda quando você altera a direção flexível de um contêiner flex. Por exemplo, quando você altera `flex-direction` para `column`, `justify-content` é alinhado verticalmente e `align-items` é alinhado horizontalmente. O comportamento mais comum, no entanto, é o padrão, ou seja, `justify-content` alinha os itens horizontalmente (porque o eixo principal padrão é horizontal) e `align-items` os alinha verticalmente. Um dos maiores pontos de discórdia que os iniciantes têm com o flexbox é a confusão quando esse comportamento muda.

**Gap**

Mais um recurso muito útil do flex é a propriedade `gap`. Definir `gap` em um contêiner flexível simplesmente adiciona um espaço especificado entre os itens flexíveis, muito semelhante a adicionar uma margem aos próprios itens. `gap` é uma propriedade nova, então ainda não aparece em muitos recursos, mas funciona de forma confiável em todos os navegadores modernos, por isso é seguro de usar e muito útil! Adicionando `gap: 8px` ao exemplo centralizado acima produz o resultado abaixo.



**Propriedades Flexbox**

- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content

***flex-direction:*** A flex-direction é usada para definir a direção do item flexível. O eixo padrão é horizontal no flexbox, então os itens fluem em uma linha.

Sintaxe:

```
// Empinhamento dos itens flex em coluna
flex-direction: column;

// Empinhamento dos itens flex de baixo para cima
flex-direction: column-reverse;

// Empinhamento dos itens flex em linha
flex-direction: row;

// Empinhamento dos itens flex da direita para esquerda
flex-direction: row-reverse;
```

index.html

```
<div class="threeway_flex">
    <div>Item 1</div>
    <div>Item 2</div>
    <div>Item 3</div>
    <div>Item 4</div>
    <div>Item 5</div>
</div>
```

styles.css

```css
.threeway_flex {
  display: flex;
  flex-direction: row;
  background-color: green;
  text-align:center;
}

.threeway_flex > div {
  background-color: #f4f4f4;
  width: 100px;
  height:100px;
  margin: 10px;
  font-size: 40px;
}
```

***flex-wrap:*** A propriedade flex-wrap é usada para definir o wrap de flex-items. Se a propriedade flex-wrap for definida como wrap, a janela do navegador definirá a caixa. Se a janela do navegador for menor que os elementos, os elementos descem para a próxima linha.

### Sintaxe:

```
// Cria um invólucro em torno dos itens quando necessário
flex-wrap: wrap;

// Não cria um invólucro em torno dos itens
flex-wrap: nowrap;
```

index.html

```html
<div class="threeway_flex">
    <div>Item 1</div>
    <div>Item 2</div>
    <div>Item 3</div>
    <div>Item 4</div>
    <div>Item 5</div>
    <div>Item 6</div>
    <div>Item 7</div>
    <div>Item 8</div>
```

```
    <div>Item 9</div>
</div>
```

styles.css

```
.threeway_flex {
  display: flex;
  flex-wrap: wrap;
  background-color: green;
  text-align:center;
}

.threeway_flex > div {
  background-color: #f4f4f4;
  width: 100px;
  height:100px;
  margin: 10px;
  font-size: 40px;
}
```

***flex-flow:***
Nota: O `flex-flow` é um atalho (shorthand) para `flex-direction` e `flex-wrap`.

Sintaxe:

```
flex-flow: row wrap;
```

***justify-content:*** A propriedade `justify-content` é usada para alinhar os itens flex de acordo com o eixo principal dentro de um container flexbox.

```
// Alinha os itens flex no centro
justify-content: center;

// O espaço é distribuído em torno dos itens flexbox
//e também adiciona espaço antes do primeiro item e depois do último
justify-content: space-around;
```

```
// Itens são igualmente distribuídos na linha
justify-content: space-between;

// Alinha os itens flex no início do contêiner
justify-content: flex-start;

// Alinha os itens flexíveis no final do contêiner
justify-content: flex-end;
```

index.html

```
<b>justify-content: center </b>
<div class="flex1">
    <div class="flex-items">1</div>
    <div class="flex-items">2</div>
    <div class="flex-items">3</div>
</div>
<br>
<b>justify-content: space-around </b>
<div class="flex2">
    <div class="flex-items">1</div>
    <div class="flex-items">2</div>
    <div class="flex-items">3</div>
</div>
<br>
<b>justify-content: space-between </b>
<div class="flex3">
    <div class="flex-items">1</div>
    <div class="flex-items">2</div>
    <div class="flex-items">3</div>
</div>
<br>
<b>justify-content: flex-start </b>
<div class="flex4">
    <div class="flex-items">1</div>
    <div class="flex-items">2</div>
    <div class="flex-items">3</div>
```

```
  </div>
  <br>
  <b>justify-content: flex-end </b>
  <div class="flex5">
      <div class="flex-items">1</div>
      <div class="flex-items">2</div>
      <div class="flex-items">3</div>
  </div>
```

styles.css

```
.flex1 {
  display: flex;
  justify-content: center;
  background-color: green;
}

.flex2 {
  display: flex;
  justify-content: space-around;
  background-color: green;
}

.flex3 {
  display: flex;
  justify-content: space-between;
  background-color: green;
}

.flex4 {
  display: flex;
  justify-content: flex-start;
  background-color: green;
}

.flex5 {
```

```
  display: flex;
  justify-content: flex-end;
  background-color: green;
}

.flex-items {
  background-color: #f4f4f4;
  width: 100px;
  height:50px;
  margin: 10px;
  text-align: center;
  font-size: 40px;
}
```

***align-items:*** Esta propriedade é usada para alinhar itens flexíveis verticalmente de acordo com o eixo transversal.

Sintaxe:

```
// Alinha os itens flex no meio do contêiner
align-items: center;

// os itens estão alinhados, como suas linhas de base se alinham
align-items: baseline;

// Estica os itens flex para preencher o container
// (mas respeita min-width/max-width)
align-items: stretch;

// Alinha os itens flex na parte superior do contêiner
align-items: flex-start;

// Alinha os itens flex na parte inferior do contêiner
align-items: flex-end;
```

index.html

```
 <b>align-items: center </b>
<div class="flex1">
    <div class="flex-items">1</div>
    <div class="flex-items">2</div>
    <div class="flex-items">3</div>
</div>
<br>
<b>align-items: baseline </b>
<div class="flex2">
    <div class="flex-items">1</div>
    <div class="flex-items">2</div>
    <div class="flex-items">3</div>
</div>
<br>
<b>align-items: stretch </b>
<div class="flex3">
    <div class="flex-items">1</div>
    <div class="flex-items">2</div>
    <div class="flex-items">3</div>
</div>
<br>
<b>align-items: flex-start </b>
<div class="flex4">
    <div class="flex-items">1</div>
    <div class="flex-items">2</div>
    <div class="flex-items">3</div>
</div>
<br>
<b>align-items: flex-end </b>
<div class="flex5">
    <div class="flex-items">1</div>
    <div class="flex-items">2</div>
    <div class="flex-items">3</div>
</div>
```

styles.css

```
.flex1 {
  display: flex;
  height: 200px;
  align-items: center;
  background-color: green;
}

.flex2 {
  display: flex;
  height: 200px;
  align-items: baseline;
  background-color: green;
  }

.flex3 {
  display: flex;
  height: 200px;
  align-items: stretch;
  background-color: green;
}

.flex4 {
  display: flex;
  height: 200px;
  align-items: flex-start;
  background-color: green;
}

.flex5 {
  display: flex;
  height: 200px;
  align-items: flex-end;
  background-color: green;
}
```

```css
.flex-items {
  background-color: #f4f4f4;
  width: 100px;
  margin: 10px;
  text-align: center;
  font-size: 50px;
}
```

**align-content:** Esta propriedade define como cada linha flex é alinhada dentro de um flexbox e só é aplicável se `flex-wrap: wrap` for aplicado, ou seja, se houver várias linhas de itens flexbox presentes.

Sintaxe:

```
// Exibe as linhas flex com espaço igual entre elas
align-content: space-between;

// Exibe as linhas flex no início do container
align-content: flex-start;

// Exibe as linhas flex no final do container
align-content: flex-end;

// O espaço será Distribuído igualmente
// em torno das linhas flex
align-content: space-around;

// Estica as linhas flex
align-content: stretch;
```

index.html

```html
  <div class="main-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
    <div>6</div>
```

```
    <div>7</div>
    <div>8</div>
    <div>9</div>
    <div>10</div>
</div>
```

styles.css

```
.main-container {
  display: flex;
  height: 400px;
  flex-wrap: wrap;
  align-content: space-between;
  background-color: green;
}

.main-container div {
  background-color: #f4f4f4;
  width: 100px;
  margin: 10px;
  text-align: center;
  font-size: 50px;
}
```