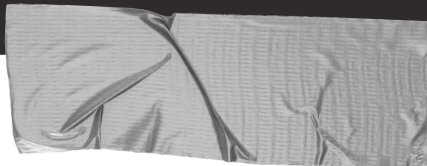


---

---

# Módulo VIII - HTML Semântico e FlexBox

FlexBox



## O que é Flexbox?

O layout Flexbox tem como objetivo oferecer uma forma mais precisa de posicionar, alinhar e distribuir o espaço entre itens em um container mesmo que seus tamanhos sejam desconhecidos ou dinâmicos (flex).

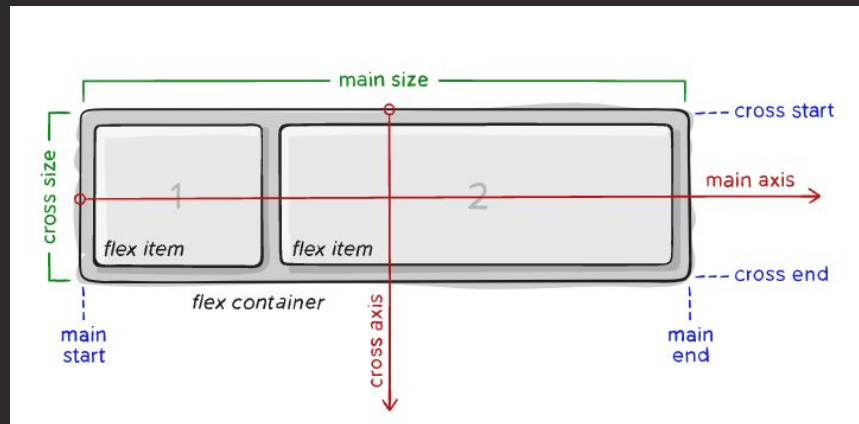
A ideia principal por trás do layout flex é dar ao container a habilidade de alterar a largura e altura dos itens para melhor preencher o espaço disponível, garantindo assim uma melhor visualização independente do dispositivo que renderiza a página. Um container flex expande os itens para preencher o espaço disponível ou os encolhe para evitar o overflow.

Importante mencionar que o flexbox não pressupõe direção, diferentemente dos layouts regulares como o block, que é vertical, e o inline que é horizontal.

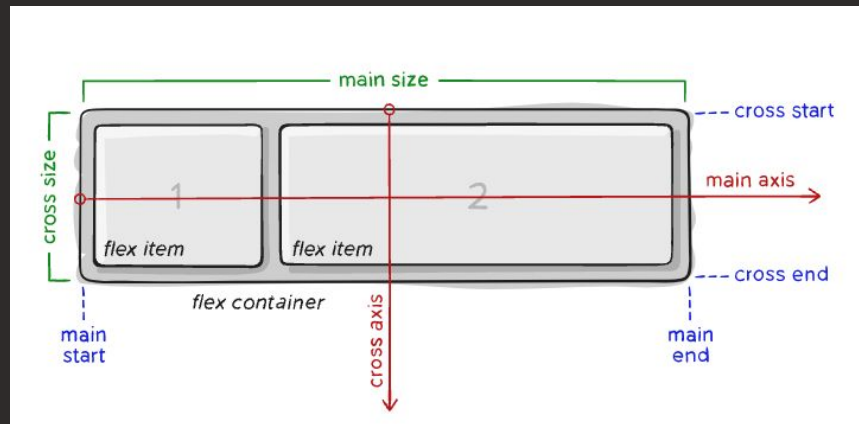
—

O que eu **preciso**  
**saber** para lidar e usar  
bem o layout **Flexbox**?


- **main axis** – O eixo principal de um contêiner flexível é o eixo primário ao longo do qual os itens flexíveis estão dispostos. Atenção ao fato de que não é, necessariamente, um eixo horizontal, vai depender de como os elementos estão dispostos.
- **main-start | main-end** – Os itens flex são colocados dentro do container começando do main-start e indo para o main-end.
- **main size** – A largura ou altura de um item flexível, independente do que esteja dentro dele, é o main size. Se essa propriedade não for especificada, o main size será o tamanho dos itens dentro dele.



- **cross axis** – O eixo perpendicular ao eixo principal (main axis) é chamado de eixo transversal (cross axis) e sua direção depende da direção do eixo principal.
- **cross-start | cross-end** – As linhas flexíveis são preenchidas com itens e colocadas no contêiner começando no cross-start do contêiner flexível e indo em direção ao cross-end.
- **cross size** – A largura ou a altura de um item flexível, independente do que esteja dentro dele, é o cross size do item. Se essa propriedade não for especificada, o cross size será o tamanho dos itens dentro dele.



# Ok, mas quais são as propriedades do Flexbox?



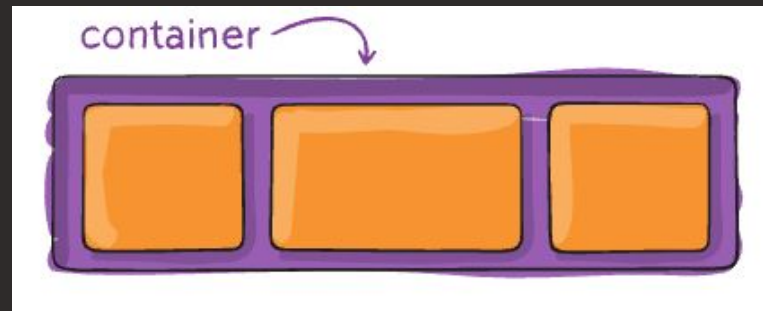
Vimos o básico da terminologia do layout flexbox. Agora veremos quais são as propriedades que utilizamos para os elementos pais e filhos dos contêineres flex.

## Propriedades do elemento Pai

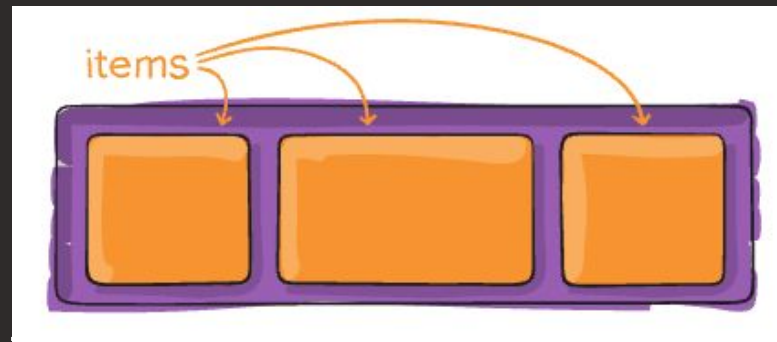
### → *display:*

Define um container flex, inline ou block dependendo do valor fornecido, habilita um layout flex para todos os seus filhos diretos.

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



Elemento pai



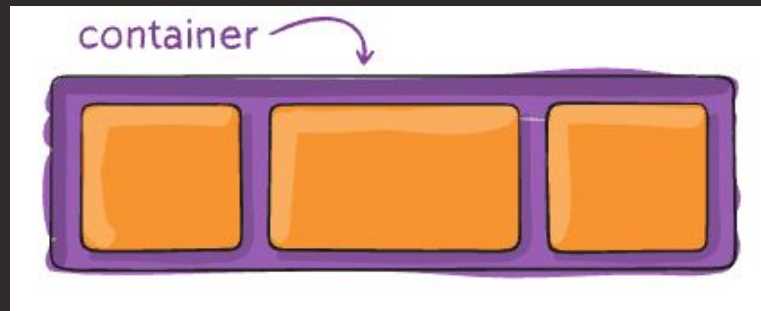
Elementos filhos

## Propriedades do elemento Pai

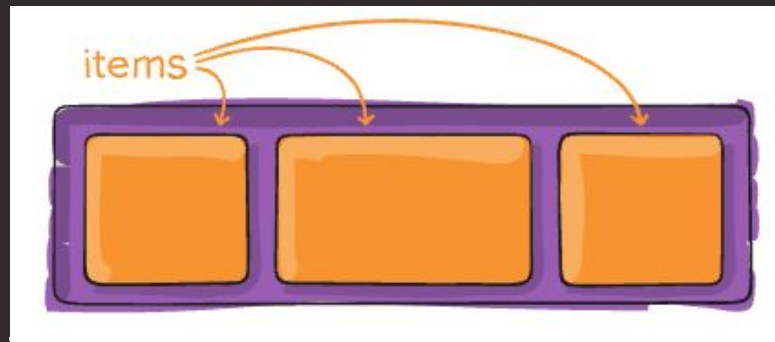
### → *flex-direction*:

Estabelece o eixo principal, definindo assim a direção que os itens flex são colocados no contêiner flex.

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



Elemento pai



Elementos filhos

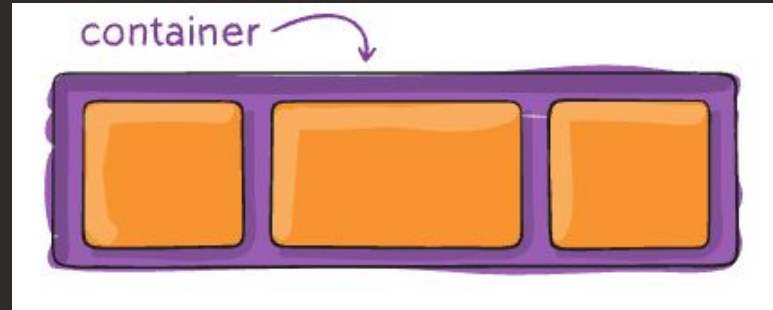


## Propriedades do elemento Pai

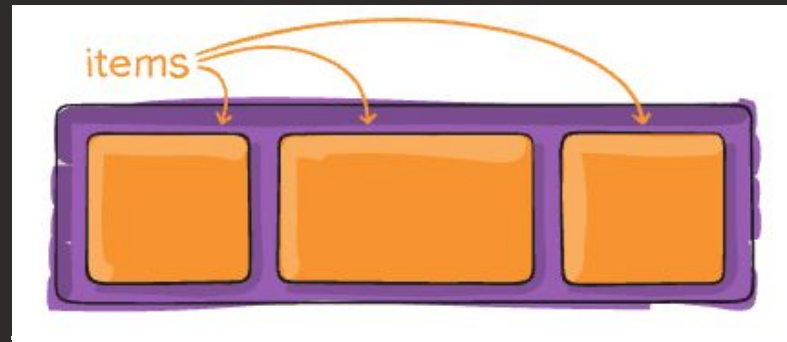
### → *flex-wrap*:

Por padrão, todos os itens flexíveis tentarão caber em uma linha. Você pode alterar isso e permitir que os itens sejam agrupados conforme a necessidade com esta propriedade.

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



Elemento pai



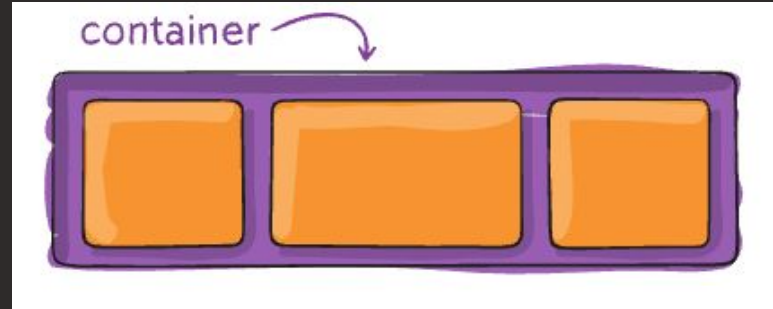
Elementos filhos

## Propriedades do elemento Pai

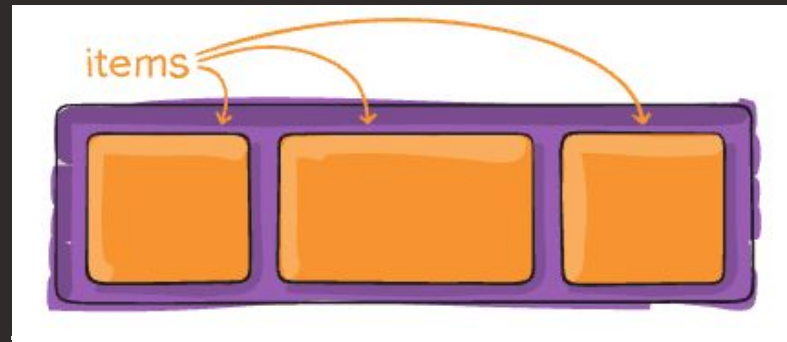
### → *flex-flow*:

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

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



Elemento pai



Elementos filhos

## Propriedades do elemento Pai

### → *justify-content:*

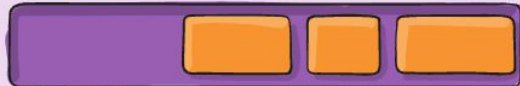
Define o alinhamento ao longo do eixo principal. Ajuda a distribuir espaço livre extra quando todos os itens flexíveis em uma linha são inflexíveis ou são flexíveis, mas atingiram seu tamanho máximo. Também exerce algum controle sobre o alinhamento dos itens quando eles transbordam a linha.

```
.container {  
  justify-content: flex-start | flex-end | center | space-between  
}
```

flex-start



flex-end



center



space-between



space-around



space-evenly



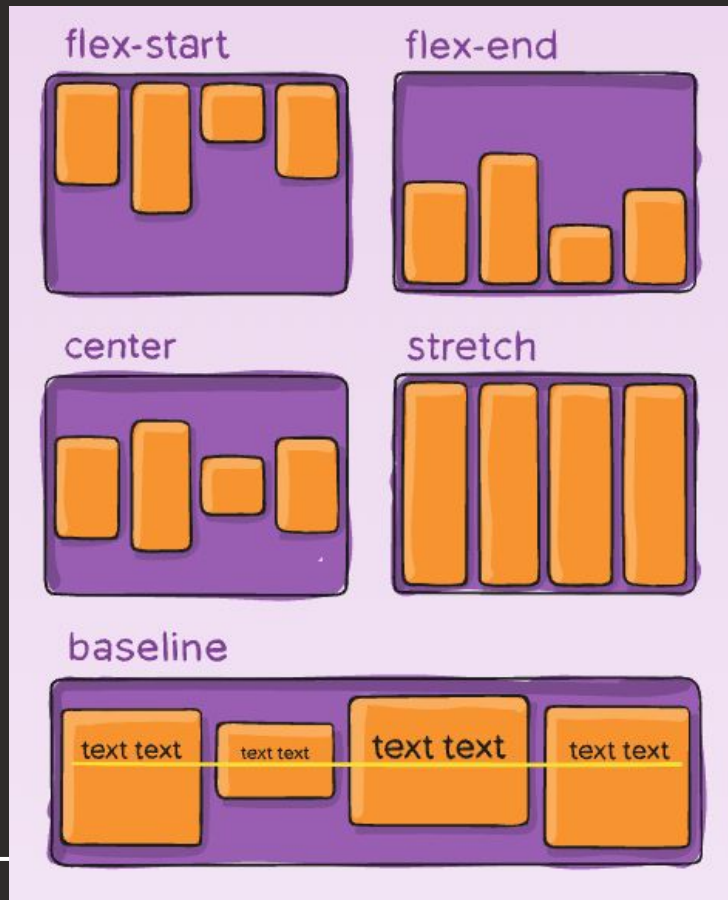
## Propriedades do elemento Pai

### → *align-items:*

Define o comportamento padrão de como os itens flexíveis são dispostos ao longo do cross-axis na linha atual. Pense nisso como a versão do justify-content para o cross-axis (perpendicular ao eixo principal).

```
.container {  
  align-items: stretch | flex-start | flex-end | center |
```

```
}
```

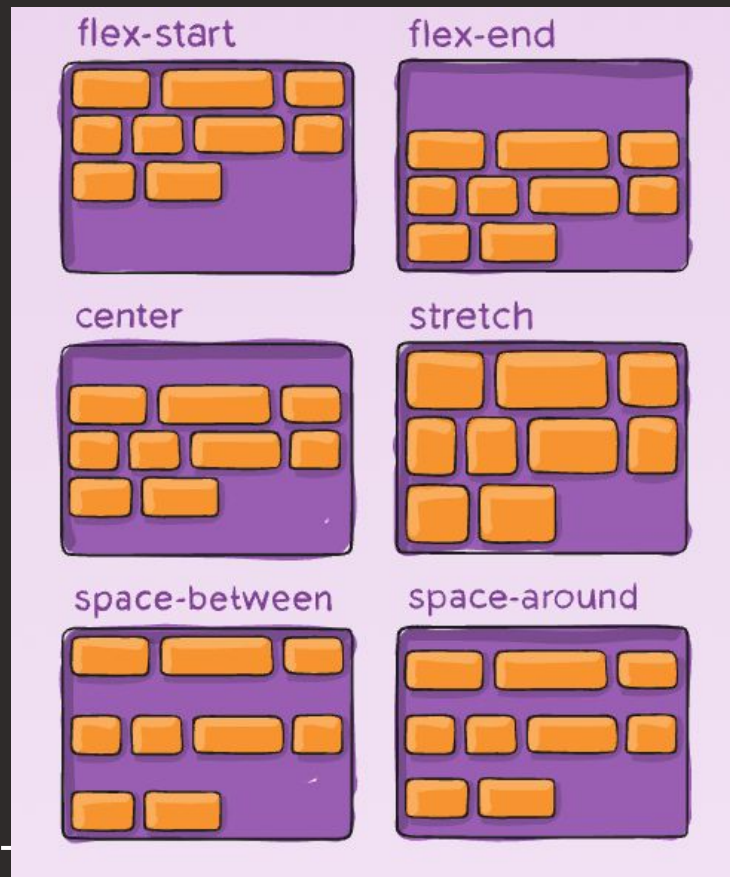


## Propriedades do elemento Pai

### → *align-content:*

Esta propriedade alinha as linhas de um contêiner flexível quando há espaço extra no cross-axis, semelhante ao modo como justify-content alinha itens individuais no eixo principal (main-axis).

```
.container {  
  align-content: flex-start | flex-end | center | space-between  
}
```



## Propriedades do elemento Pai

### → *gap, row-gap, column-gap*

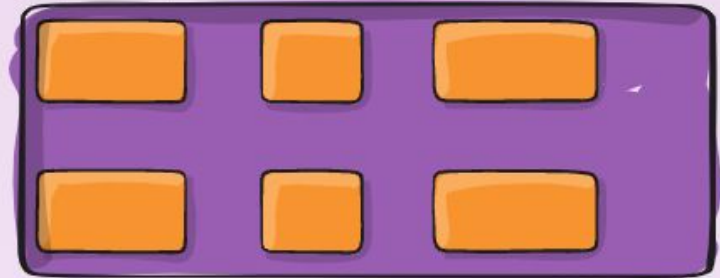
A propriedade `gap` controla explicitamente o espaço entre os itens flexíveis e aplica esse espaçamento apenas entre os itens e não nas bordas externas.

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

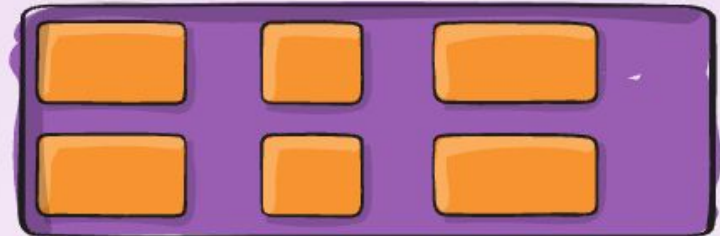
gap: 10px



gap: 30px



gap: 10px 30px

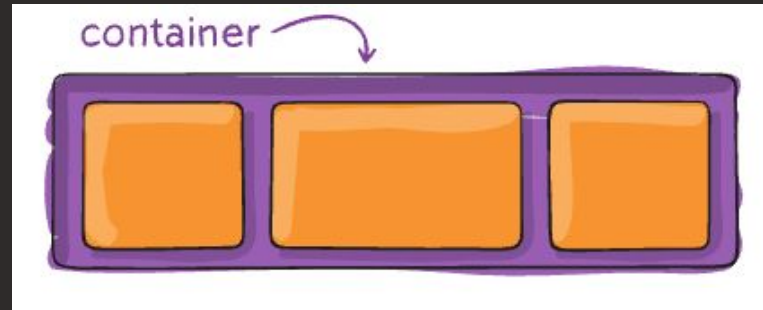
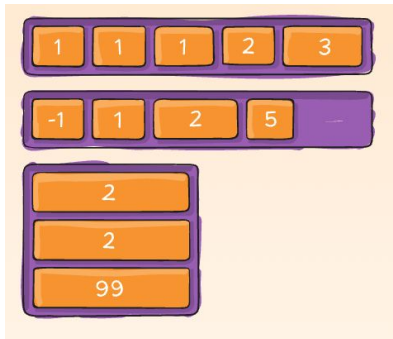


## Propriedades dos elementos filhos

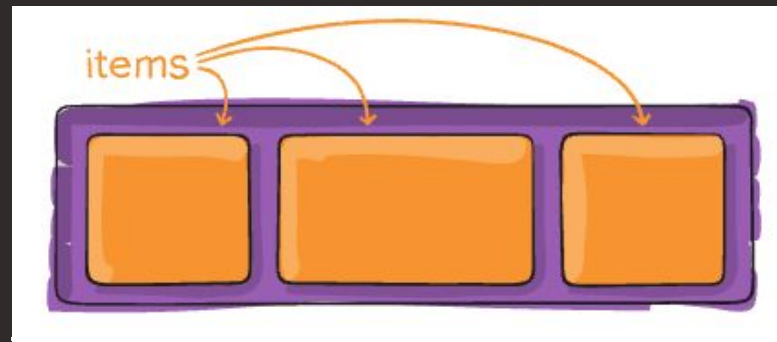
→ **order:**

Por padrão, os itens flexíveis são dispostos na ordem de origem (0), mas a propriedade *order* controla a ordem em que eles aparecem no contêiner flexível.

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



Elemento pai



Elementos filhos

## Propriedades dos elementos filhos

### → *flex-grow*:

Define a capacidade de um item flexível crescer, se necessário. Ele determina a quantidade de espaço disponível dentro do contêiner flexível que o item deve ocupar.

Se todos os itens tiverem *flex-grow* definido 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 */  
}
```



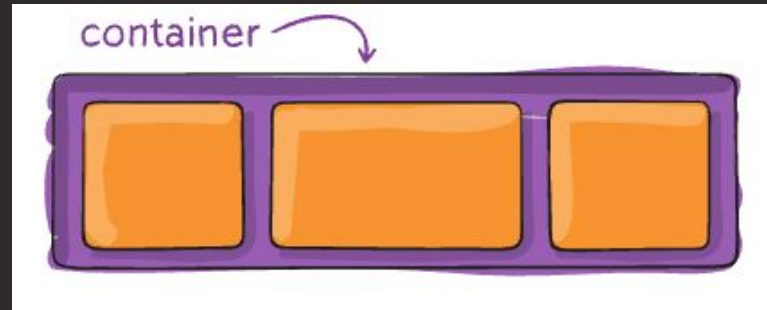


## Propriedades dos elementos filhos

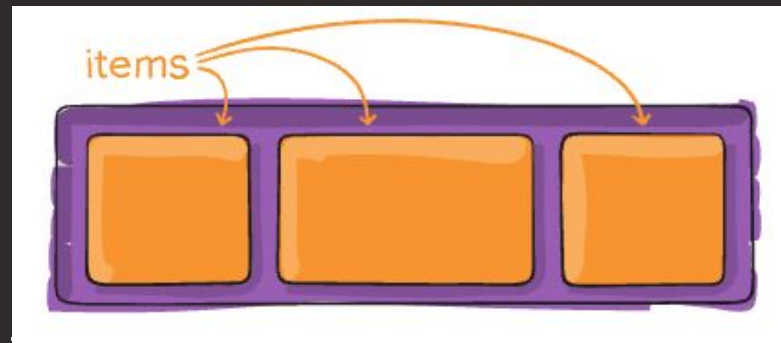
### → *flex-shrink*:

Define a capacidade de um item flexível diminuir, se necessário.

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



Elemento pai



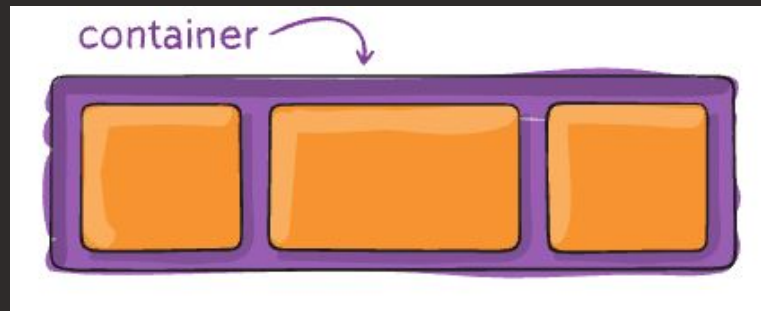
Elementos filhos

## Propriedades dos elementos filhos

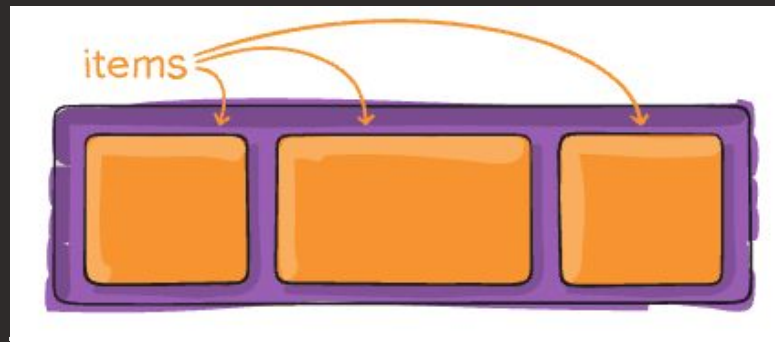
### → *flex-basis*:

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 palavra-chave auto significa "olhar para minha propriedade: largura ou altura". A palavra-chave content significa "dimensioná-la com base no conteúdo do item".

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



Elemento pai



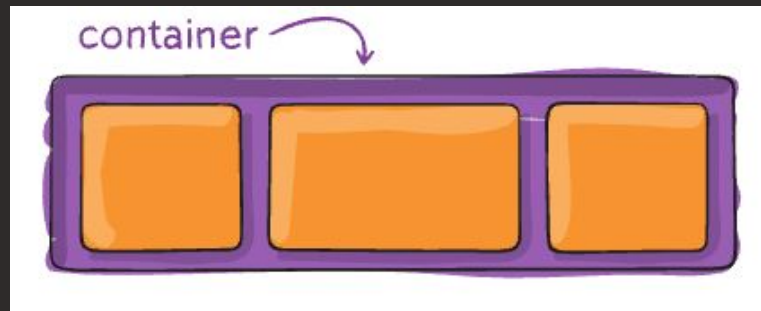
Elementos filhos

## Propriedades dos elementos filhos

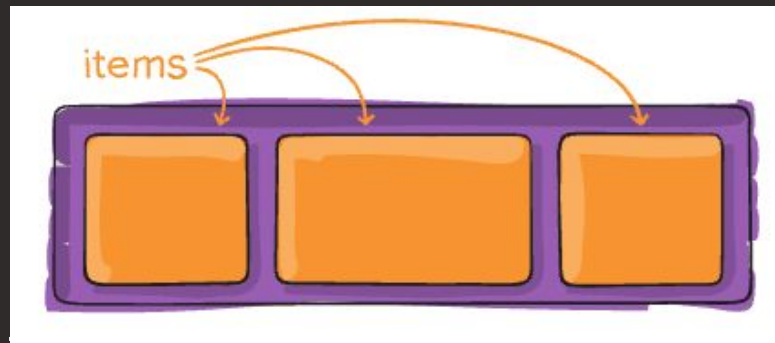
→ *flex:*

Este é o atalho para flex-grow, flex-shrink e flex-basis combinados. O segundo e terceiro parâmetros (flex-shrink e flex-basis) são opcionais. O padrão é o 1 auto, mas se você definir com um único valor numérico, como flex: 5;, que altera o flex-basis para 0%, então é como configurar flex-grow: 5; flex-shrink: 1; flex-basis: 0%;.

```
.item{  
  flex: 2 1 20%;  
}
```



Elemento pai



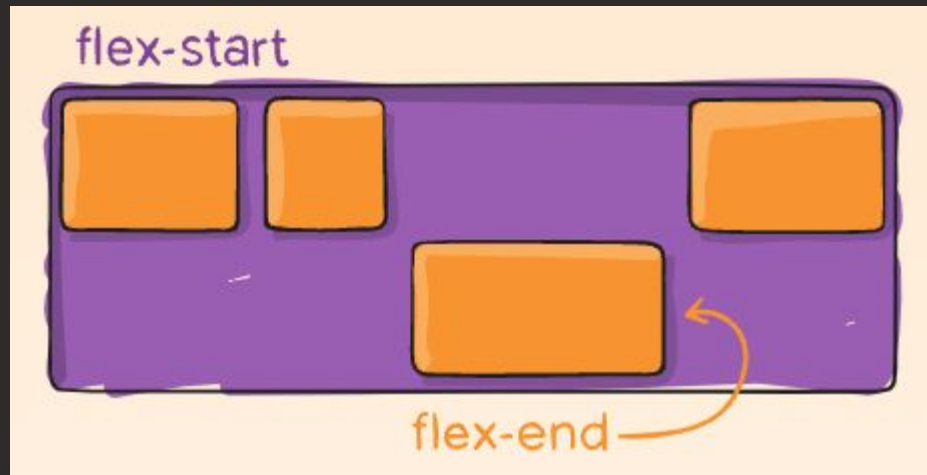
Elementos filhos

## Propriedades dos elementos filhos

### → *align-self*:

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

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



—

O layout Flexbox é a  
resposta e solução para  
**quase todos** problemas  
com responsividade!