Itens no Flexbox

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

“

Flexbox nos dá a possibilidade de aplicar **propriedades** diretamente em cada **item** para poder **manipulá-los separadamente** e ter maior controle

”

Índice

1 | order

order

Com esta propriedade, **controlamos** a **ordem** de cada item, independentemente da ordem original que tenham na estrutura HTML. Esta propriedade recebe como valor um **inteiro**, **positivo** ou **negativo**. Por padrão, todos os itens flex têm um `order: 0` implícito, mesmo que não seja especificado.

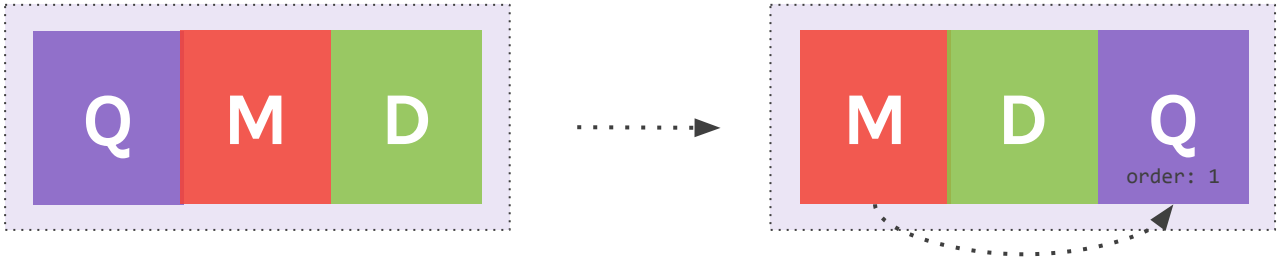
```css
.caixa {
    order: 1;
}
```

order: número positivo

Se atribuirmos à caixa Q (que tem a classe `caixa-q`) a propriedade `order` com valor `1`, ela **irá para o final da linha** por ser o número mais alto.

Lembremos que, por padrão, o valor da ordem de cada item é 0.

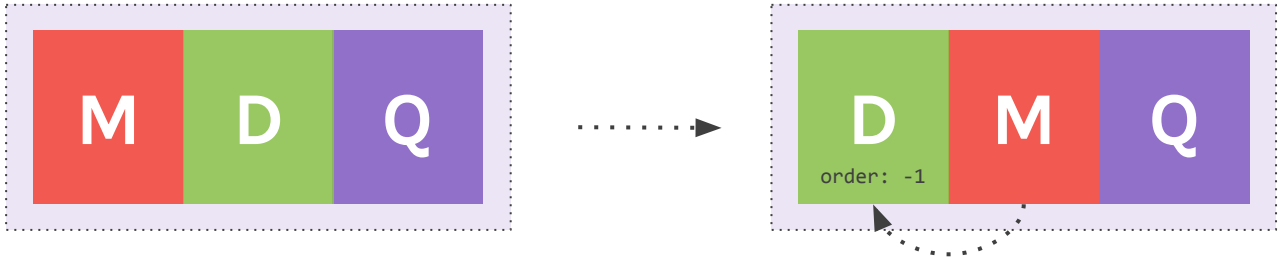
```css
.caixa-q {
    order: 1;
}
```



order: número negativo

Se agora atribuirmos a propriedade **order** à **caixa-d** com **-1** como valor, ela irá para o início da linha. Colocando por primeiro o item com o menor valor.

```css
.caixa-d {
    order: -1;
}
```

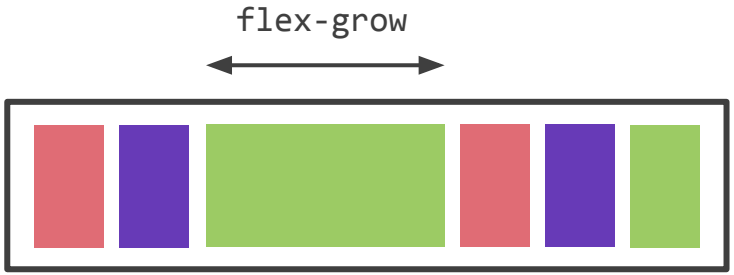


As caixas serão **ordenadas** respeitando a sequência de números **negativos** para **positivos**.

2 | flex-grow

flex-grow

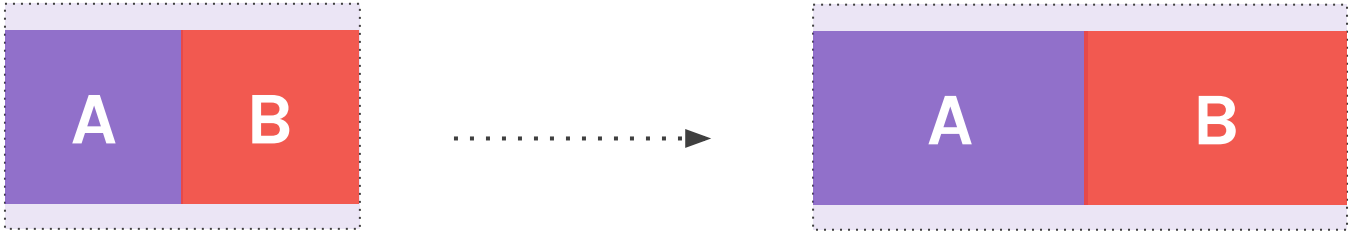
Com esta propriedade, definimos quanto um **item** pode **crescer** se houver **espaço livre** no contêiner. Define o crescimento flexível para o elemento.



flex-grow

Se **ambos os itens** tiverem a propriedade `flex-grow` com valor `1`, à medida que o contêiner crescer, eles cobrirão o espaço disponível em partes iguais.

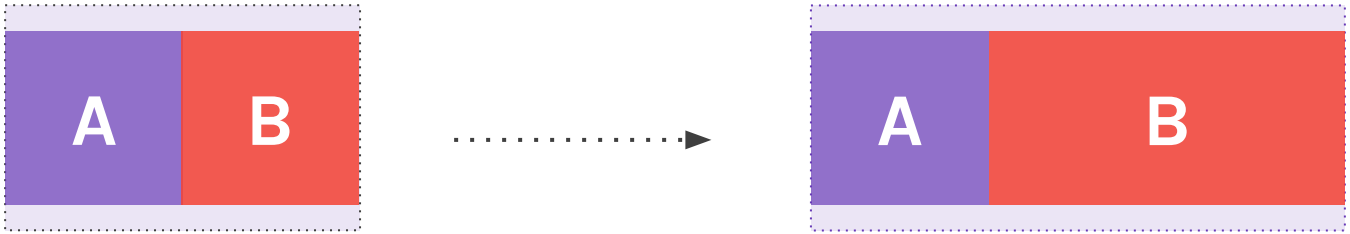
```css
.caixa-a, .caixa-b {
    flex-grow: 1;
}
```



flex-grow

Se um **único item** tiver a propriedade `flex-grow`, ele tentará ocupar o espaço livre disponível, conforme o contêiner cresce, de acordo com a proporção que definimos com o valor.

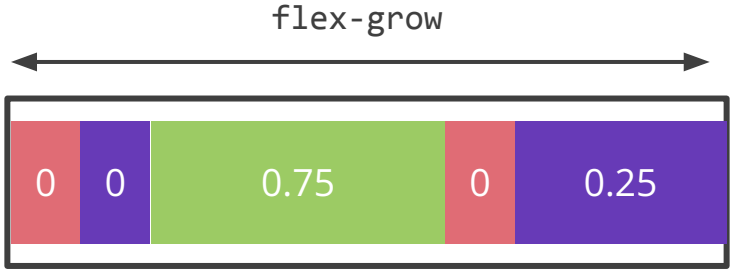
```css
.caixa-b {
    flex-grow: 1;
}
```



flex-grow

O número que atribuímos ao `flex-grow` determina quanto do espaço disponível dentro do contêiner flex esse item deve ocupar.

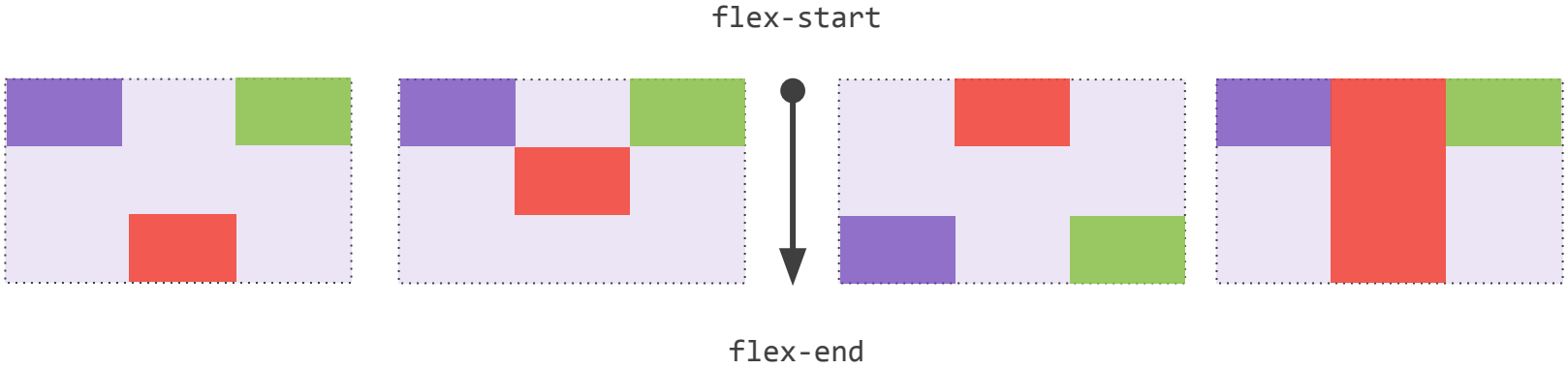
1 é igual a 100% do espaço disponível e 0 é igual a 0%. Podemos usar qualquer valor intermediário, como 0,25 para 25%.



3 | align-self

align-self

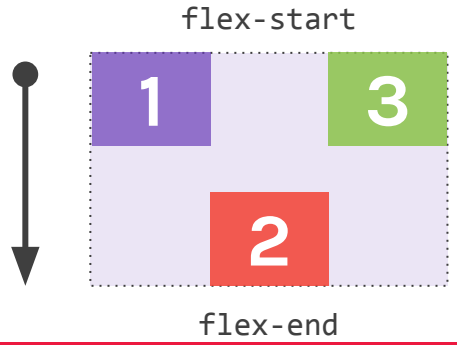
Permite **alinhar**, no **cross axis**, cada item ao qual aplicamos esta propriedade, independentemente do alinhamento que tenha sido definido no contêiner flex com **align-items**.



align-self: flex-end

Com `flex-end`, o **item** é alinhado ao **final** do eixo transversal.

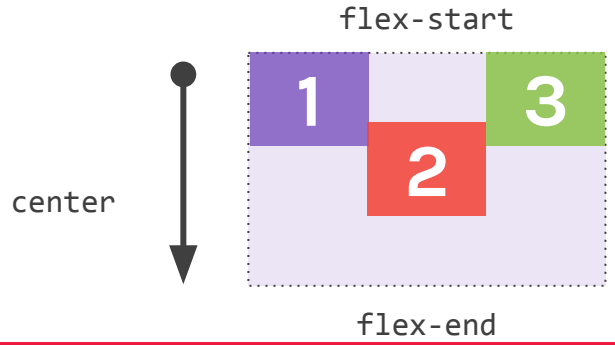
```css
.container-pai {
    align-items: flex-start;
}
.caixa-dois {
    align-self: flex-end;
}
```



align-self: center

Com `center`, o **item** é alinhado ao **centro** do eixo transversal.

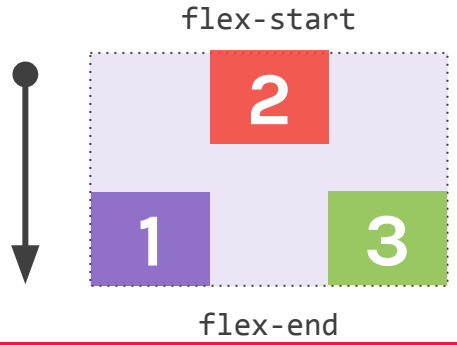
```css
.container-pai {
    align-items: flex-start;
}
.caixa-dois {
    align-self: center;
}
```



align-self: flex-start

Com `flex-start`, o **item** é alinhado **no início** do eixo transversal.

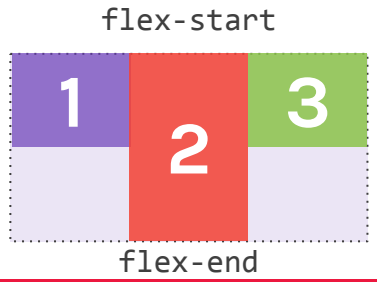
```css
.container-pai {
    align-items: flex-end;
}
.caixa-dois {
    align-self: flex-start;
}
```



align-self: stretch

Com `stretch`, o item se **ajusta** para abranger todo o **cross axis**, que é o comportamento padrão. Funciona desde que o elemento não tenha uma altura definida.

```css
.caixa-um, .caixa-tres {
    align-self: flex-start;
}
.caixa-dois {
    align-self: stretch;
}
```



Essas **propriedades serão aplicadas** aos itens flexíveis, desde que o **contêiner pai** seja um **contêiner flexível**.

DigitalHouse>