Flexbox e responsividade com CSS

Prof. Tiago Lopes Telecken

tiago.telecken@riogrande.ifrs.edu.br

IFRS – Rio Grande

overflow

div {overflow:valor}

- **visible** – é o padrão. O overflow transborda a caixa (normalmente causa um erro no visual da página)
- **hidden** – o overflow não é mostrado
- **scroll** – o elemento passa a ter uma barra de rolagem
- **auto** - Similar ao scroll, mas só aparece quando é necessário

@import

- Serve para importar um código css para o atual arquivo
- Também serve para importar fontes
- Deve ser o primeiro comando do arquivo

```
@import 'estilo_base.css';
```

@import

- Organize seus arquivos css com o @import
- Em um site algumas partes do visual se repetem e os comandos CSS correspondentes se repetem também.
  - Por exemplo o css do cabeçalho, do rodapé da estrutura do site, do reset
  - Estes códigos css são iguais em várias (ou todas) páginas
  - E eles devem ser consistentes em todas as páginas. Se você alterar alguma coisa em um cabeçalho deve alterar em todos

@import

- Por isso é importante separar códigos que se repetem em arquivos separados
  - Crie um arquivo css para o reset (Ex. reset.css)
  - Crie um arquivo css com o código que irá se repetir em todas páginas (Ex. geral.css)
- Cada página do site deverá importar estes arquivos e a seguir ter seu código. Por exemplo, o arquivo css da página de produtos deverá se chamar produto. css e o seu código deverá ser:

```
@import ´reset.css´
@import ǵeral.css´
// A seguir o restante do código css do arquivo produto.css
```

@import

- Desta maneira se você tiver que alterar algo no rodapé basta altera-lo em 1 lugar (no arquivo geral .css)
- Sem uma organização como esta você teria que fazer a mesma alteração em todas as páginas.
- Pode-se fazer organizações mais sofisticadas como um arquivo css para cada parte do site. Por exemplo:
  - reset.css
  - header.css
  - footer.css
  - layout.css
  - ...

Display, visibility

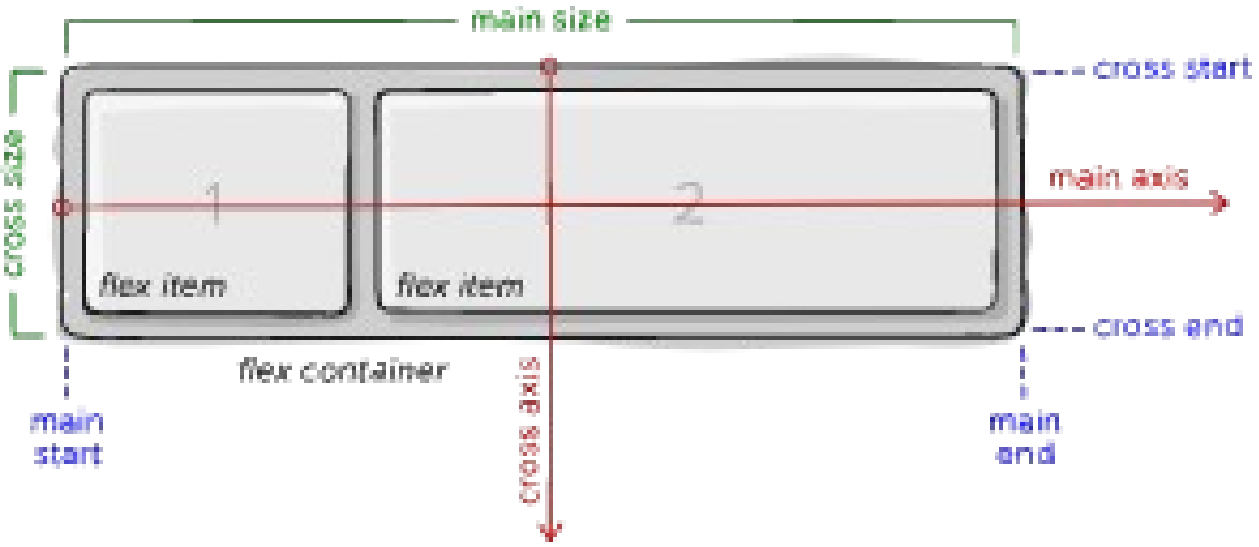
{ display:valor }

- **Block**: o elemento será posicionado em uma nova linha. É o comportamento padrão da maioria dos elementos(div, h1, input, form…)
- **In-line:**os elementos serão colocados lado a lado. As propriedades width (largura) e height (altura) serão ignoradas. É o comportamento padrão das letras e trchos de <b> e <i> por exemplo.
- **Inline-block:** os elementos serão colocados lado a lado. As propriedades width (largura) e height (altura) não serão ignoradas
- **None**: o elemento fica invisível e não ocupa espaço no layout da página

{ visibility: valor}

- **Hidden:**o elemento fica invisível mas ocupa espaço no layout da página
- **Visible:**o elemento fica visível. Comportamento padrão

Flexbox




```
<div class="flexcontainer">
      <div class="item">1</div>
      <div class="item">2</div>
</div>
--------
.flexcontainer { display:flex }
```

Flexbox

- **Flex container**: é o elemento que contém os itens. O elemento pai.
- **Flex item**: são os elementos filhos do flex container
- Eixos:
  - **Main axis**: eixo principal, direção depende da propriedade flex-direction
  - **Cross axis**: eixo transversal, direção depende da propriedade flex-direction
- **Main size e cross size**: tamanho principal e tamanho transversal, **width** ou **height** ... vai depender da direção dos eixos.

Propriedades do elemento pai, o container

Display: se for **flex** ou **inline-flex** coloca os elementos internos em um contexto flex.

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

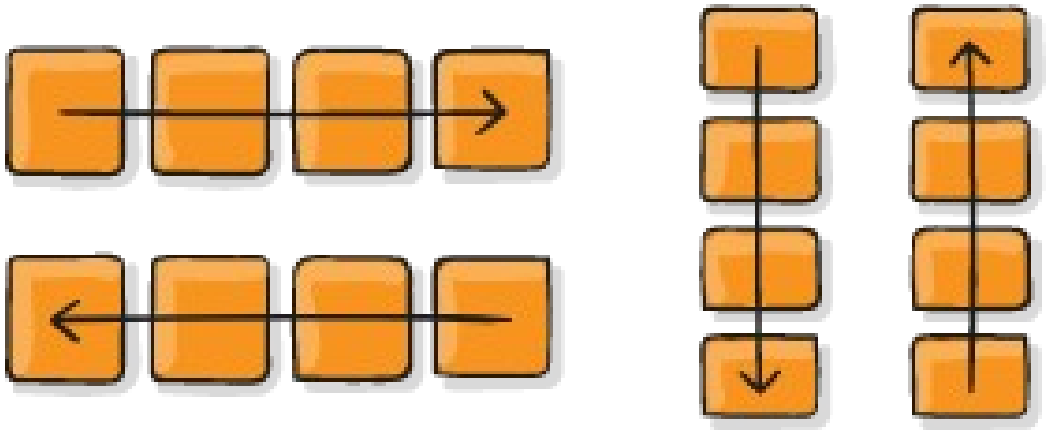
Lembrando valores do display:

Block: o elemento é mostrado em bloco. Começa e termina com uma linha nova.
Inline: segue o fluxo e o tamanho da linha em que está. (como letras em um parágrafo)
Inline-block: é um bloco (com altura e largura próprios) que tenta se colocar ao lado do elemento anterior sem quebras de linha (como um menu horizontal de botões).

Então…
Display flex da ao container o comportamento de block
Display inline-flex da ao container o comportamento de inline-block
Nesses 2 casos, o comportamento block ou inline-block afeta só o container, o elemento pai. Os itens, ou elementos filhos, nos 2 casos estão em um contexto flex.

Propriedades do elemento pai, o container

Flex-direction: define o eixo principal. Definindo a direção em que os flex-itens são colocados.



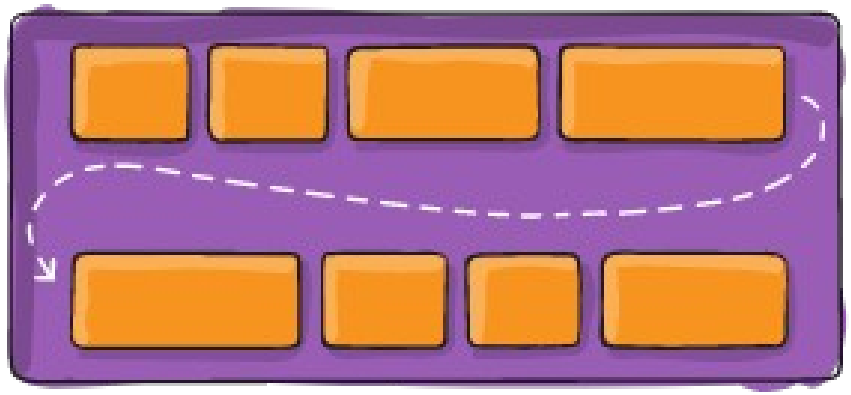
```
.container {  flex-direction: row }

// row | row-reverse | column | column-reverse;
```

Padrão: row

Propriedades do elemento pai, o container

Flex-wrap: por padrão os flex items vão ficar em uma linha só. Com esta propriedade pode-se permitir que os itens quebrem para uma próxima linha se for necessário.

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

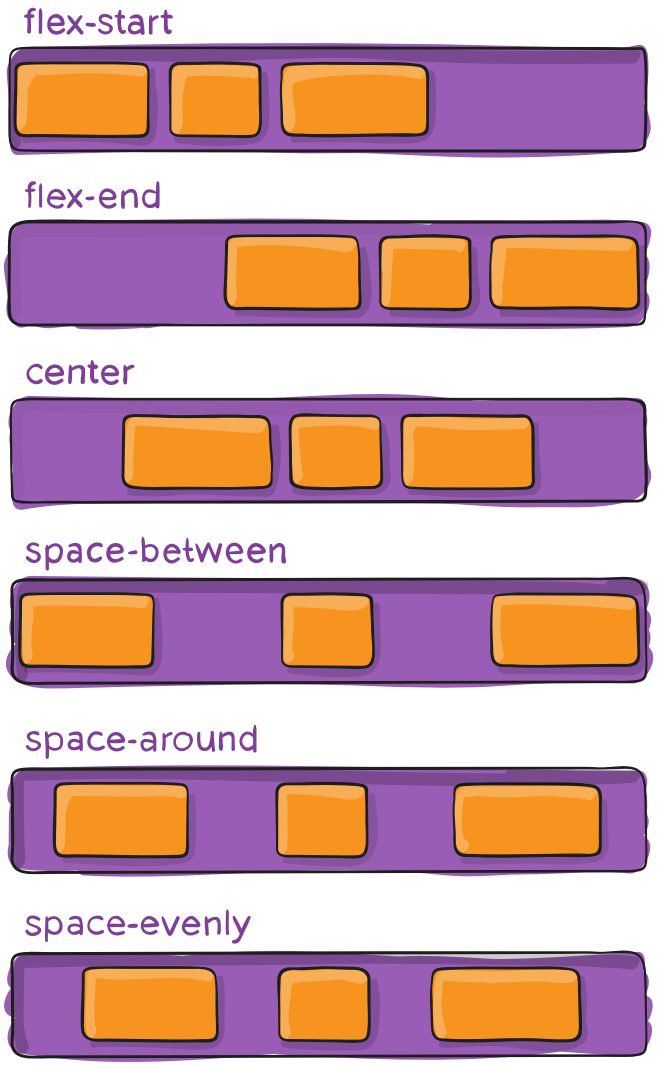


Propriedades do elemento pai, o container

Justify-content:
Define a distribuição dos itens ao longo do eixo principal.
Varia conforme a direção do eixo principal

```
.flex-container {  justify-content: flex-start }

// flex-start | flex-end | center |
   space-between | space-around | space-evenly;
```

Padrão: flex-start



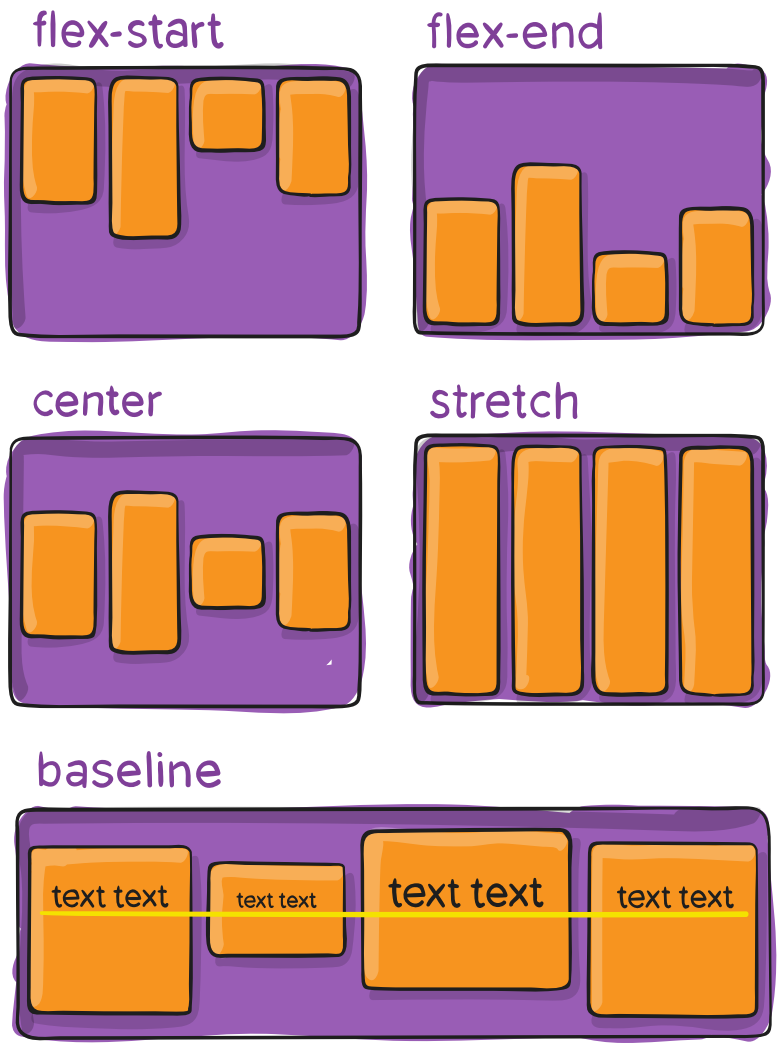
Propriedades do elemento pai, o container

Align-itens:
Define o alinhamento dos itens quanto ao eixo Transversal. Varia de acordo com o eixo.

.flex-container { align-itens: flex-start }

// stretch | flex-start | flex-end | center | baseline;

Padrão: stretch



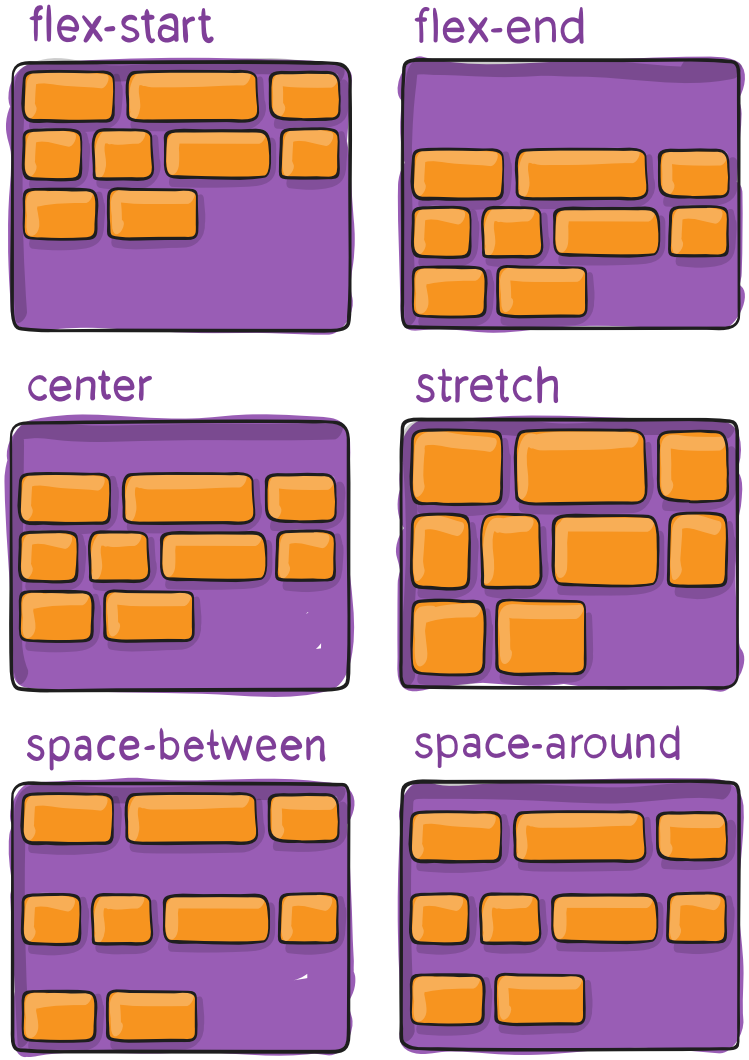
Propriedades do elemento pai, o container

align-content:
Organiza as linhas quando há espaço extra no eixo transversal.
Esta propriedade não tem efeito quando há somente uma linha de flex items no container.

.container { align-content: flex-start }

// flex-start | flex-end | center | space-between | space-around | stretch;

Padrão: stretch



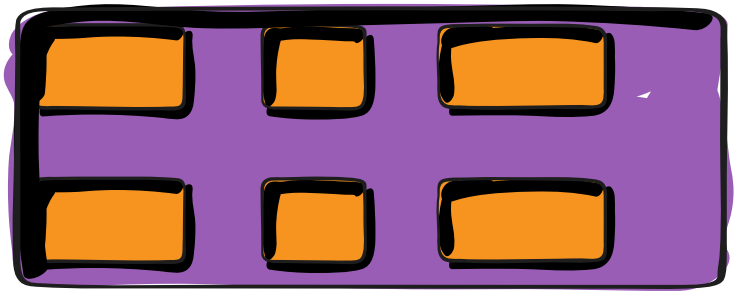
Propriedades do elemento pai, o container

gap:
Adiciona um espaçamento mínimo entre os elementos.
Também funciona no grid e em qualquer
outro layout multicoluna.

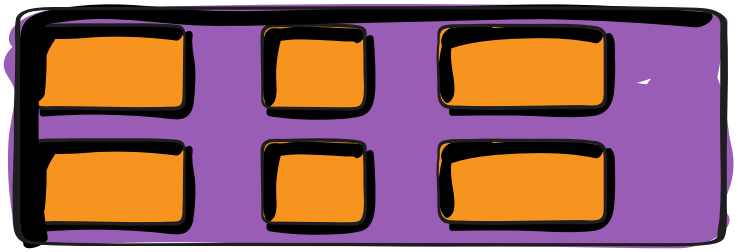
```
.container {
gap: 20px;
gap: 20px 30px //row-gap column gap;
row-gap:20px;
column-gap:30px;
 }
```

gap: 10px

gap: 30px



gap: 10px 30px



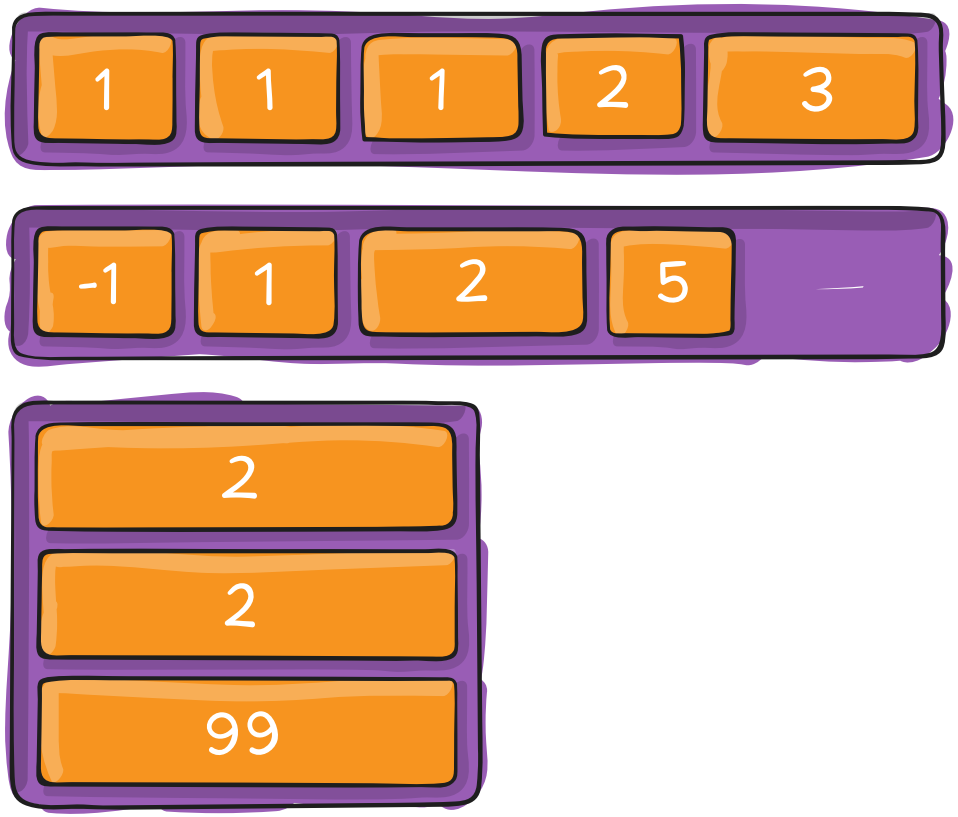
Propriedades dos elementos filho, os itens

order:
Por padrão os flex items são dispostos na tela na ordem do código.
A propriedade order pode controlar a ordem em que aparecerão no container.

.flex-item { order: 2 }

// <número>;

Padrão: 0



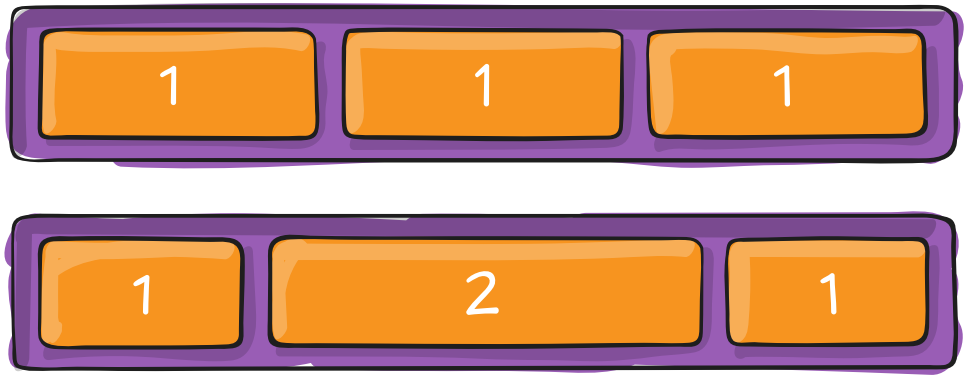
Propriedades dos elementos filho, os itens

flex-grow:
Define em que proporção um item cresce, caso necessário. O valor dessa propriedade é um valor numérico sem indicação de unidade, que serve para cálculo de proporção. Se um item tem flex-grow igual a 2 e os demais são iguais a 1 ele crescerá 2x mais que os demais.

.flex-item { flex-grow: 2 }

// <número>;

Padrão: 0



Propriedades dos elementos filho, os itens

flex-shrink:
Define em que proporção um item encolhe, caso necessário.
.item { flex-shrink: 2 }
Padrão 1

flex-basis:
Define o tamanho padrão para um elemento
.item { flex-basis: auto }

// Pode ser um comprimento (20%, 50px, 5rem, etc) ou:
Mais alternativas em:

https://developer.mozilla.org/en-US/docs/Web/CSS/flex-basis

flex:
Shorthand para flex-grow, flex-shrink e flex-basis
.item { flex: 2 0 100px}
Padrão: 0 1 auto

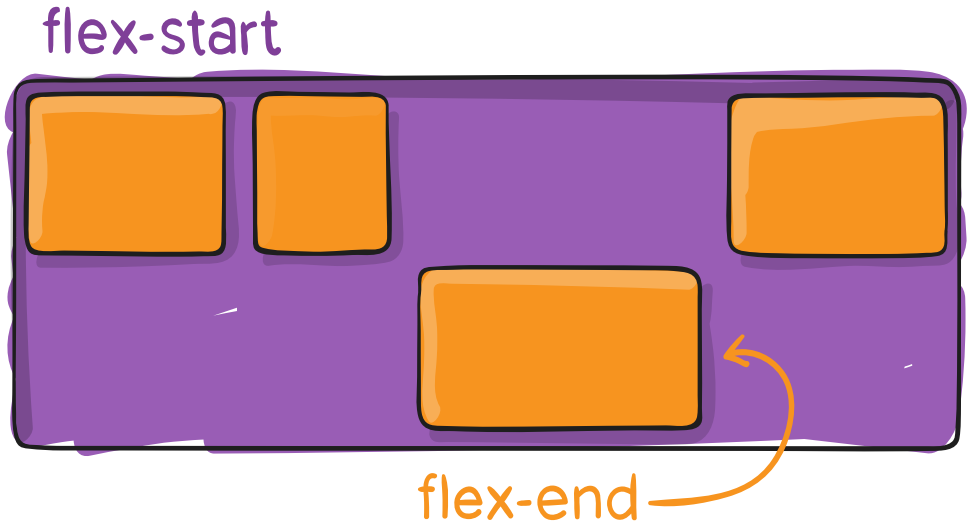
Propriedades dos elementos filho, os itens

align-self:
Permite que o alinhamento do container seja sobrescrito para ítens individuais.

.item { align-self: flex-end }

// auto | flex-start | flex-end | center | baseline | stretch;

Padrão: auto



+ Recursos de Responsividade

Max-width: define a largura máxima que um elemento pode ter. Em nenhuma situação, tamanho de tela, etc o elemento terá uma largura maior que a definida. O mesmo vale para as outras medidas limitadoras.
Min-width: define a largura mínima que um elemento pode ter.

Max-height: define a altura máxima que um elemento pode ter.
Min-height: define a altura mínima que um elemento pode ter.

```
div.caixa {
  Width:100%;
  max-width: 500px;
  min-width: 400px;

  height:100%;
  max-height: 190px;
  min-height: 130px;
}
```

Clamp, Calc

Podem ser usadas em qualquer propriedade que utilizem medidas
Podem misturar diferentes tipos de medidas.

clamp (MÍNIMO,PREFERIDO, MÁXIMO)

```
font-size: clamp(2rem, 4vw, 3rem)

font-size: clamp(18px, 2vw, 24px)
```

calc (FÓRMULA)

```
margin: calc (1%+2px)
```