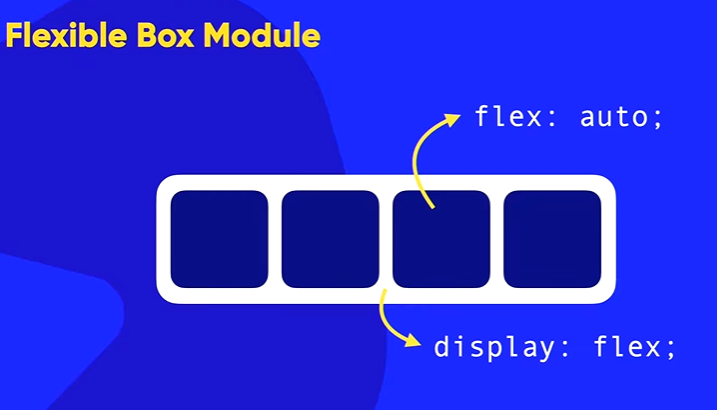
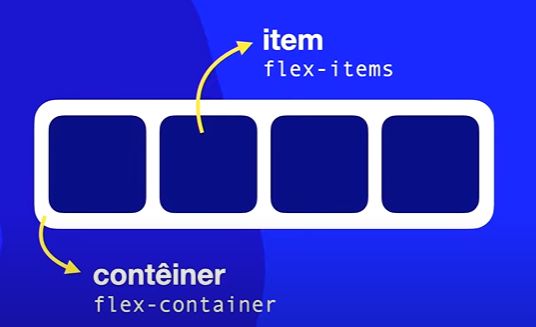
**Flexbox CSS:**

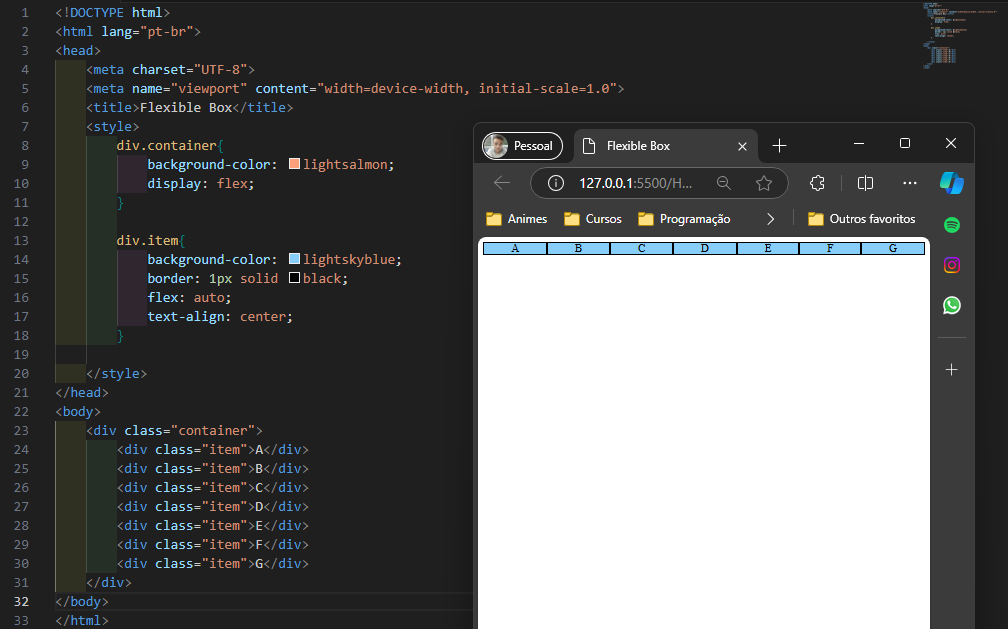
Surgiu para **melhorar** a responsividade, surgindo: “Qual o formato da agua?” Depende do recipiente. (no caso contêiner).



**Se adptando de acordo com o Pai! O container**. Ele que leva o **display:flex; SEMPRE!**



Exemplo:



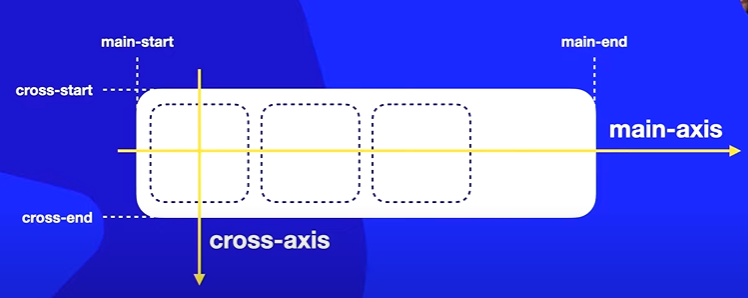
**Direções e Eixos: Apenas do PAI! (container).**

**Flex-Direction: (para o pai)**

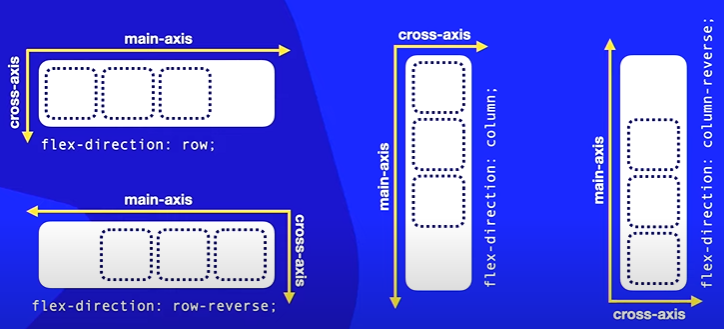
1. Padrão: flex-direction: row; (linha – esquerda para direita).
2. Reverso: flex-direction: row-reverse; (linha – direita para esquerda).
3. Vertical: flex-direction: column; (cima para baixo).
4. Reverso: flex-direction: column-reverse; (baixo para cima).

Quando escolhe a direção, são criados **2 eixos (axis) importantes. (Main e Cross)**

1. **Main-axis;** (eixo principal)
   1. **Main-start;**
   2. **Main-end;**
2. **Cross-axis;** (eixo transversal)
   1. **Cross-start;**
   2. **Cross-end;**



Resumo das outras flex- direction:

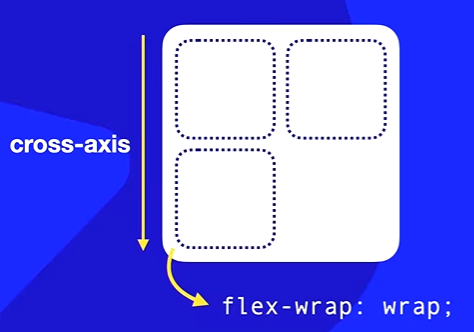


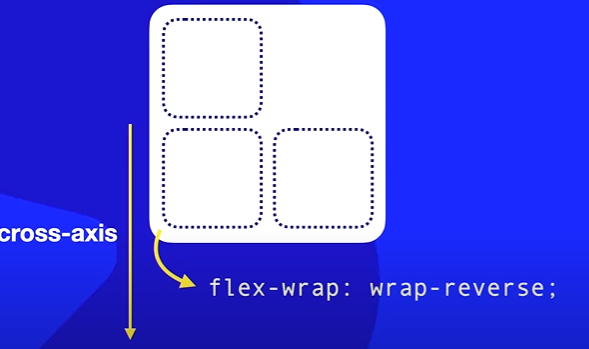
Observa-se que os items são sempre colocado no inicio do main!

**Empacotamento e fluxo:**

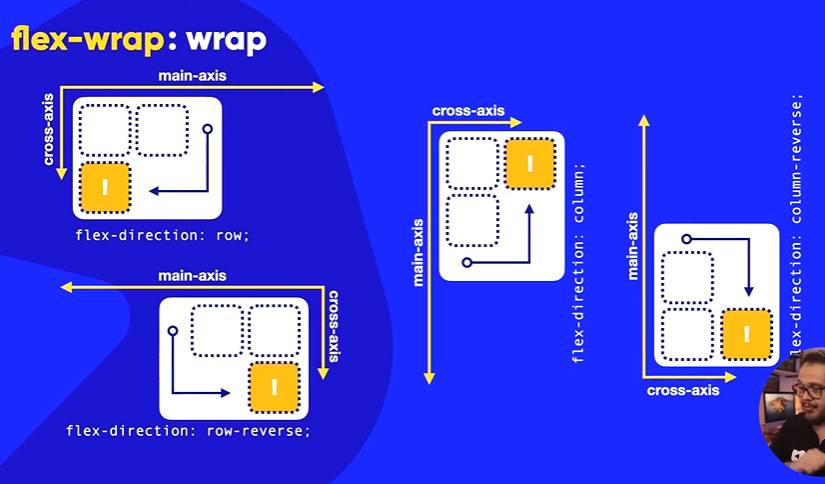
**Flex-wrap:**

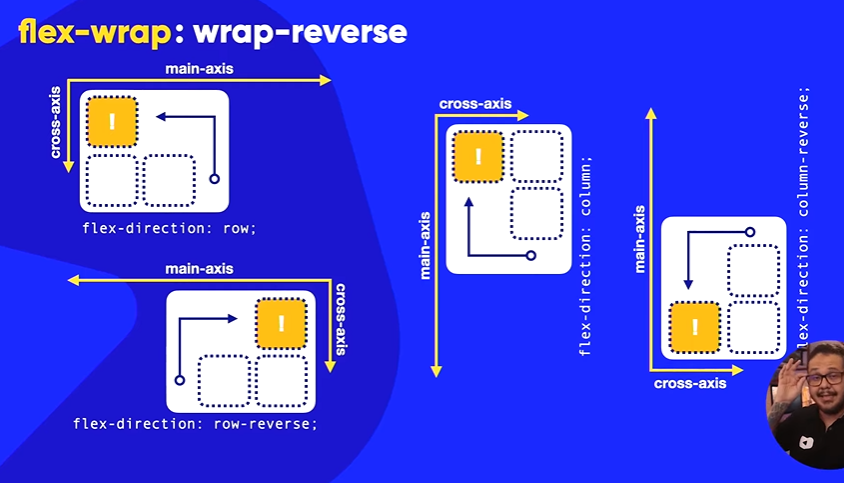
1. **Padrão**: **flex-wrap: nowrap**; (não quebra os blocos- fazendo encolher tudo).
2. **flex-wrap: wrap**; (quebra o ultimo elemento, no sentido do cross-axis).
3. **flex-wrap: wrap-reverse**; (quebra o ultimo elemento, no sentido oposto do cross-axis).



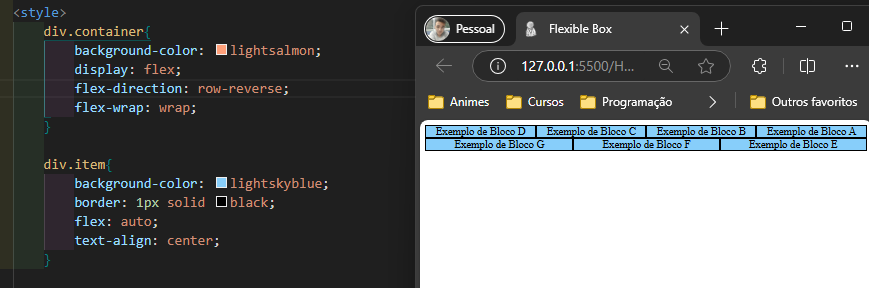


**Resumo:**

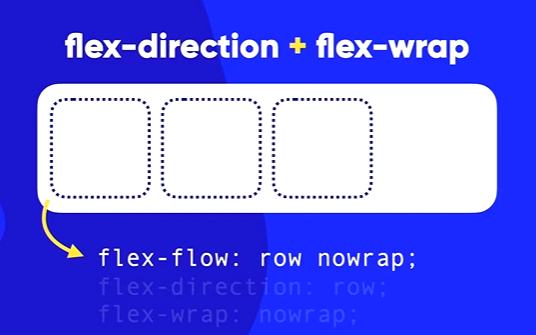




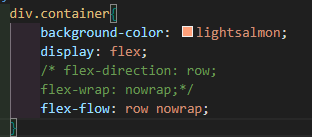
Exemplo:



E para juntar tudo em um só comando?



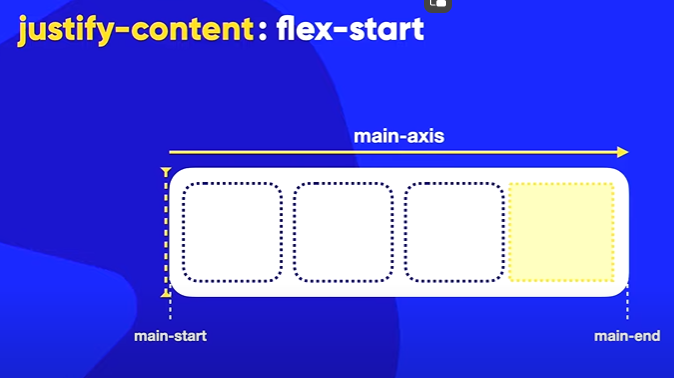
Primeiro vem o direction e depois o wrap.



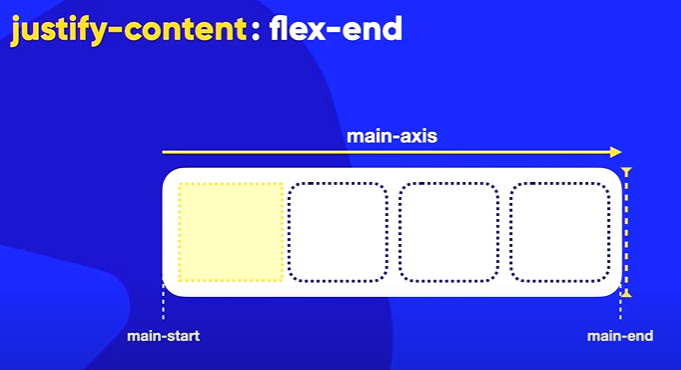
**Alinhamento nos eixos:**

**Justify-content: (alinhamento no eixo principal – main axis)**

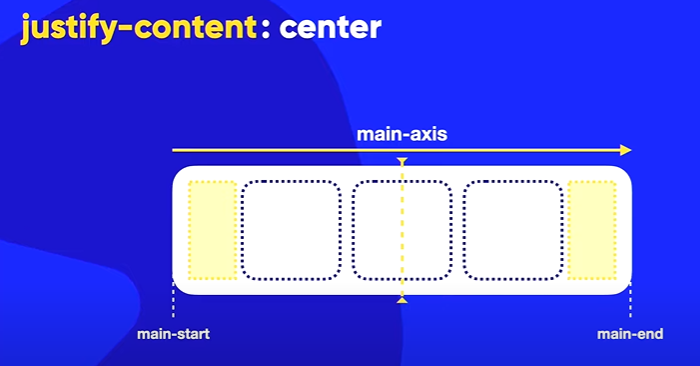
1. **Justify-content: flex-start**; (primeiro item grudado no main-start, com todos grudados nele, e o espaço em branco no main-end).



1. **Justify-content: flex-end**; (ultimo item grudado no main-end, com todos grudados nele, e o espaço em branco no main-start).

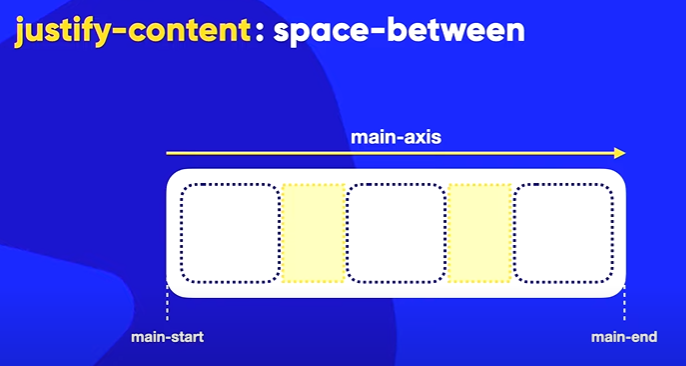


1. **Justify-content: center**; (calcula o centro do container e coloca todos os items juntos nesse centro e espaço em branco distribuído igual).

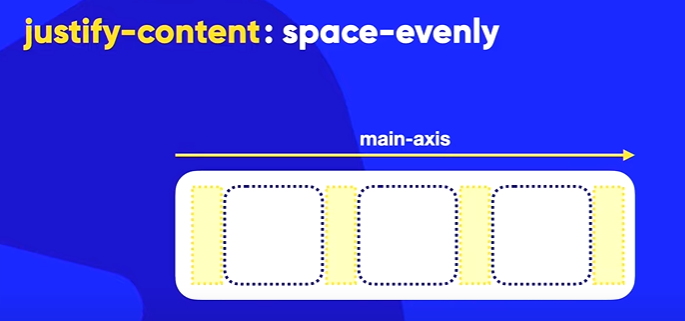


**Alinhamento de espaçamento:**

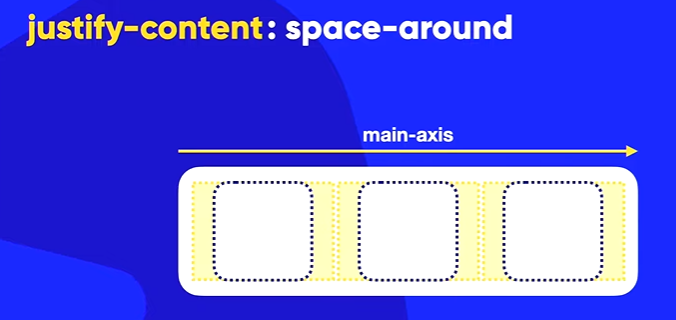
1. **Justify-content: space-between**; (coloca o primeiro item no main-start e o ultimo item no main-end e todos os outros no meio com espaçamento igual)

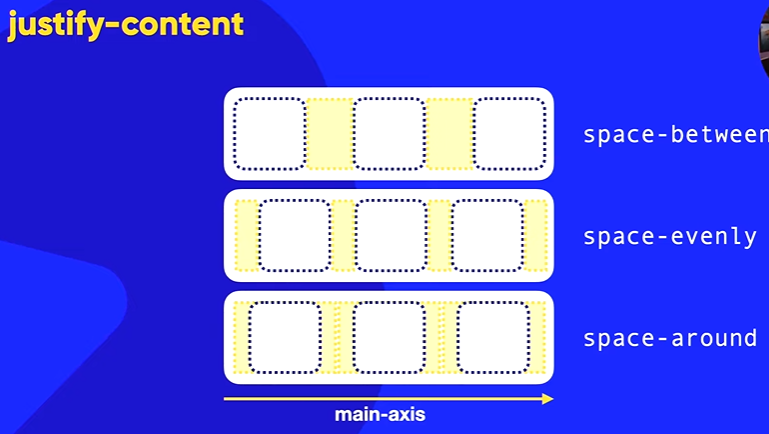


1. **Justify-content: space-evenly**; (coloca elementos dispostos dentro do container, de forma que antes e depois tem espaçamento igual).

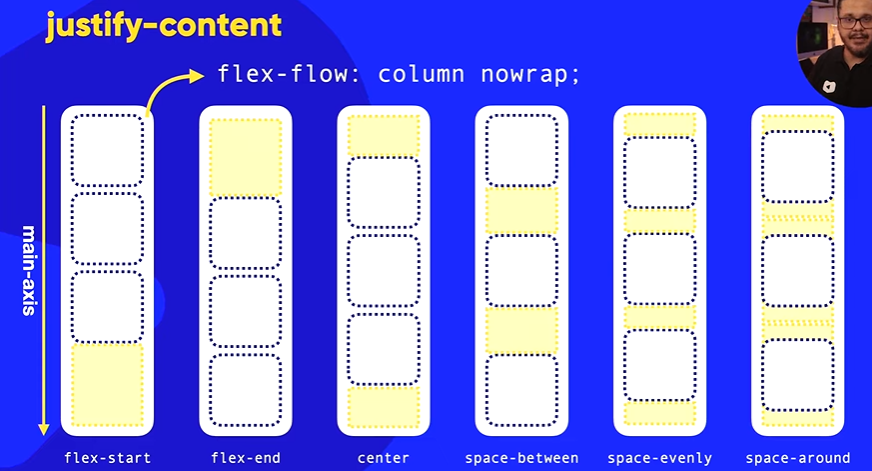


1. **Justify-content: space-around**; (pega todos os elementos, divide em pedaço igual e centraliza nos espaços).



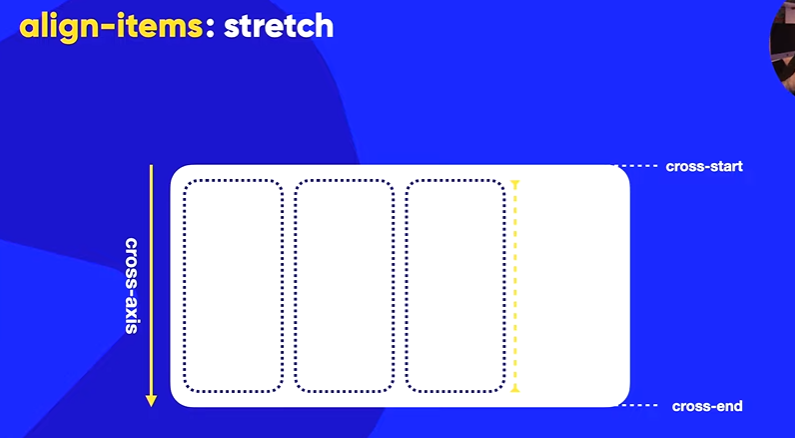


**E se fosse column?**

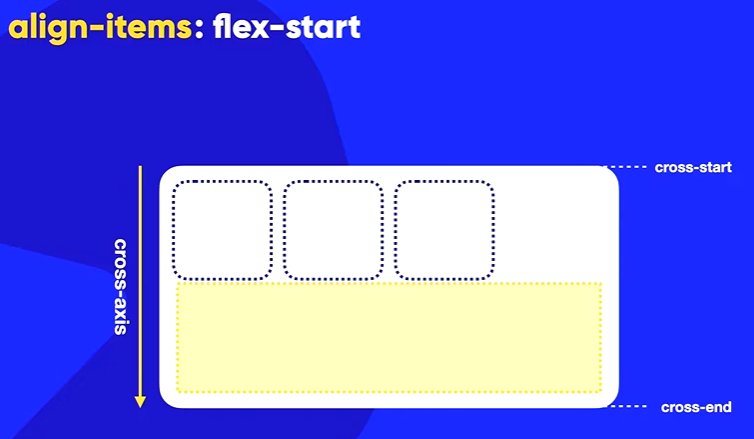


**Align-items: (alinhamento no eixo transversal –cross axis)**

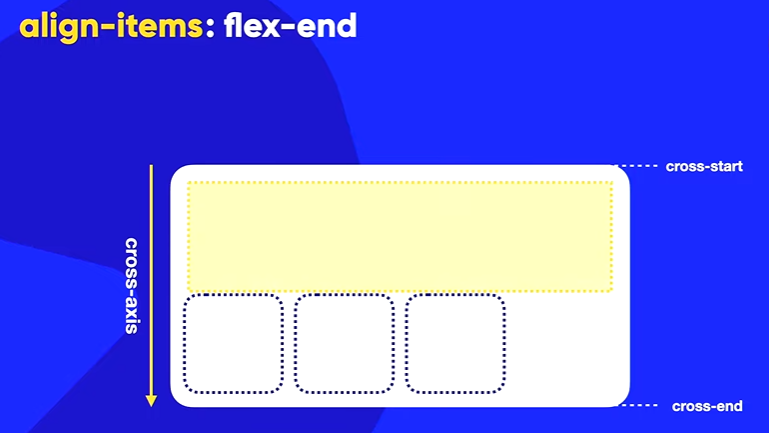
1. **Aling-items: stretch**; (os items se estica).



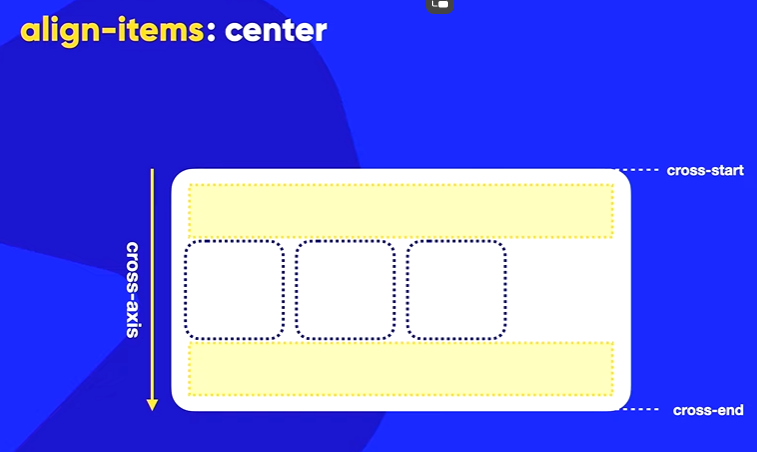
1. **Aling-items: flex-start**; (os items sempre grudam no cross-strat).



1. **Aling-items: flex-end**; (os items sempre grudam no cross-end).

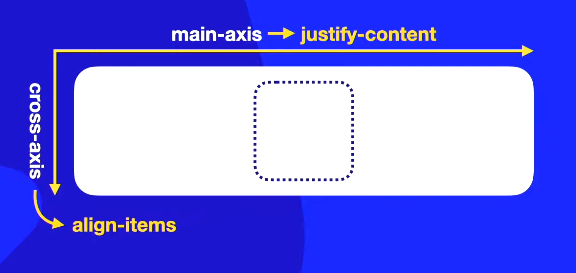


1. **Aling-items: center**; (os items ficam no centro).



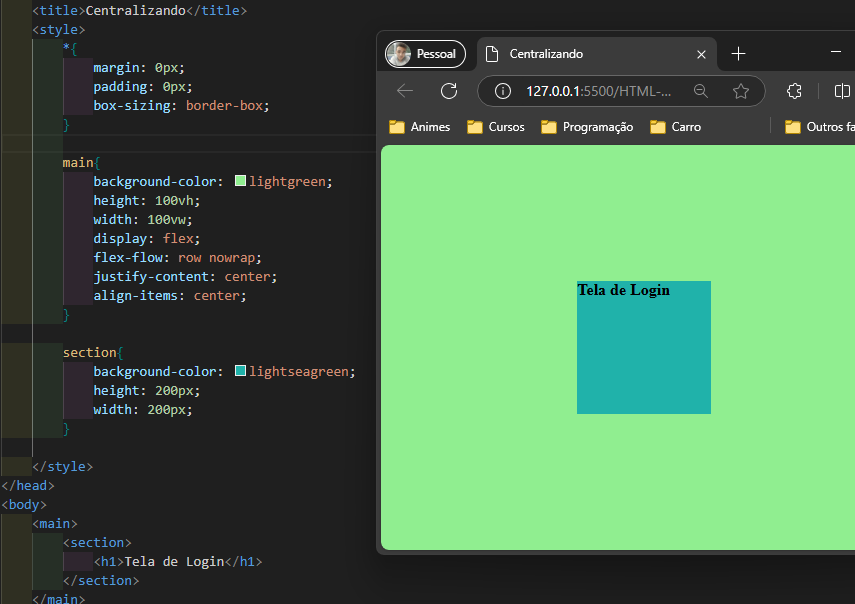
**Centralizando com flexbox:**

Ainda se aplicando apenas ao Pai (container).



Colocando ambos como center. (**sempre o pai tem que ter display flex!)**

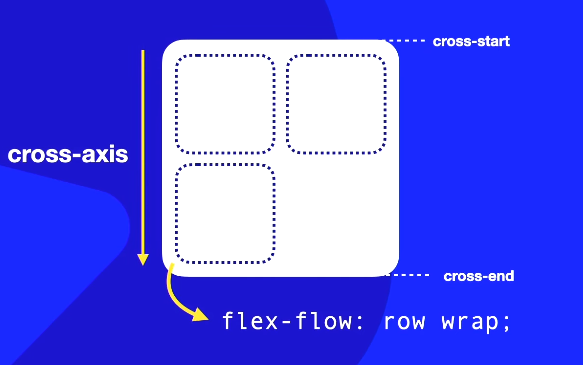
Exemplo:



E se quebrar os elementos? Como vamos alinhar?

**Alinhamento de conteúdo empacotado:**

**Empacotado**: é aquele que se quebra. Wrap!



**\*Cuidado**

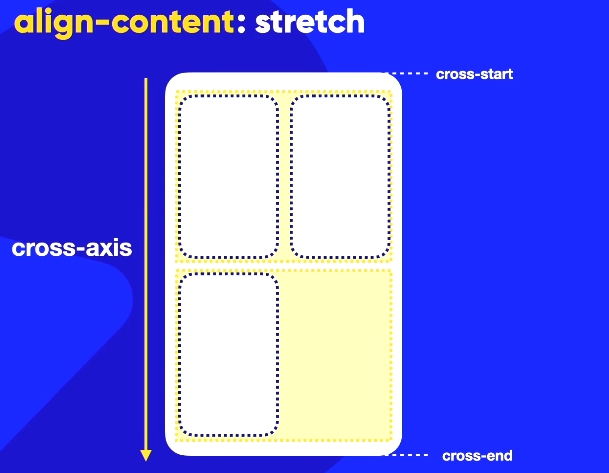
\* **Justify-content** alinha o conteúdo dentro do eixo principal!

\* **Aling-items** alinha o conteúdo dentro do eixo transversal!

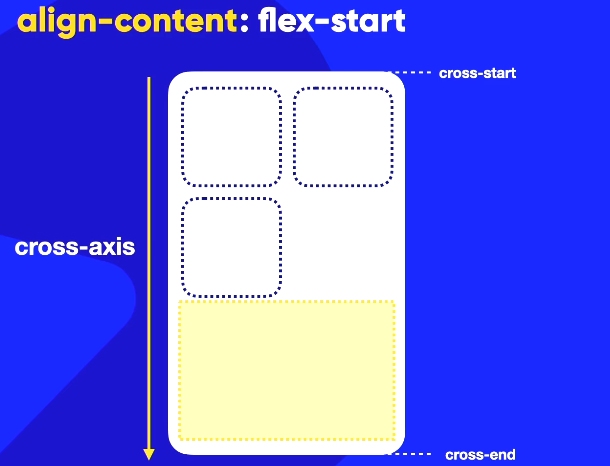
\***Aling-content** alinha o conteúdo dentro do eixo transversal mas quando eles estão com wrap! (empacotado).

**Aling-content: (sempre com wrap)**

1. **Align-content: stretch**; (Divide a quebra em partes iguais, e estica até o cross-end de cada quebra). Padrao



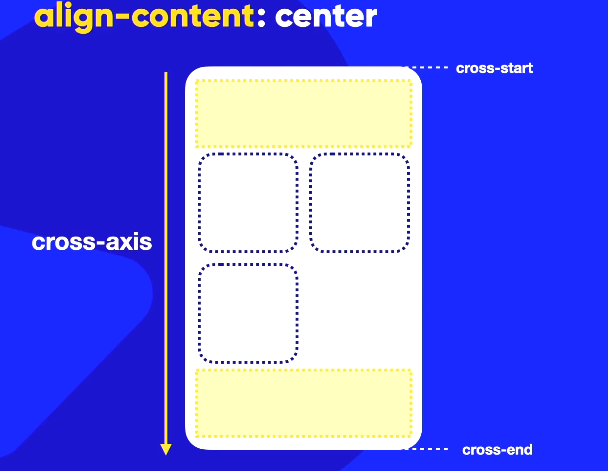
1. **Align-content: flex-start**; (coloca grudado no cross-start).



1. **Align-content: flex-end;** (coloca grudado no cross-end).

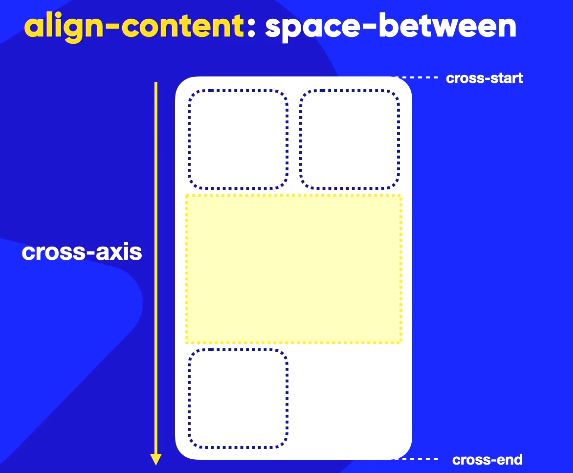


1. **Align-content: center**; (centraliza)

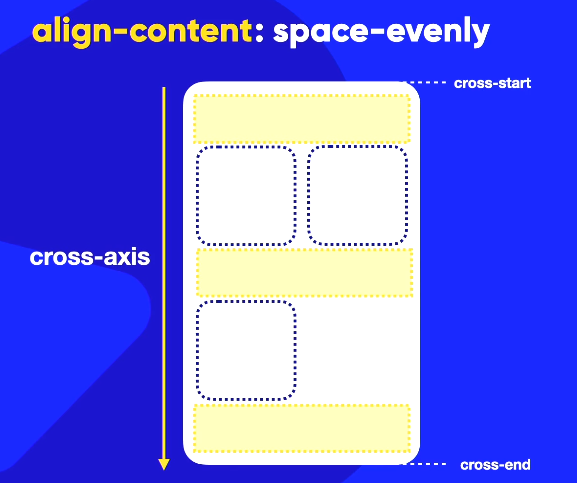


**Alinhamento de espaçamento:**

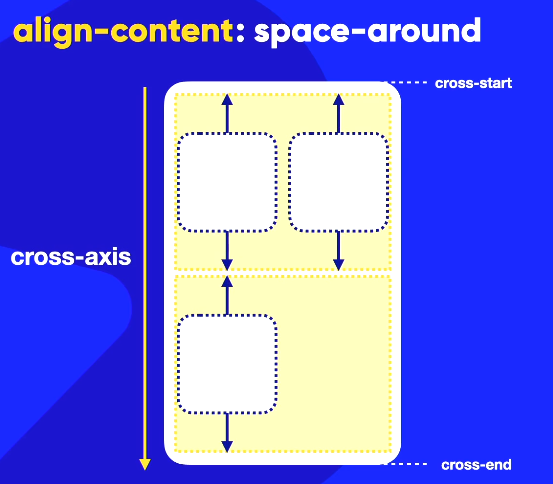
1. **Align-content: space-between**; (primeiros elemento próximo ao cross-start e os últimos próximo ao cross-end e o espaço dividir).



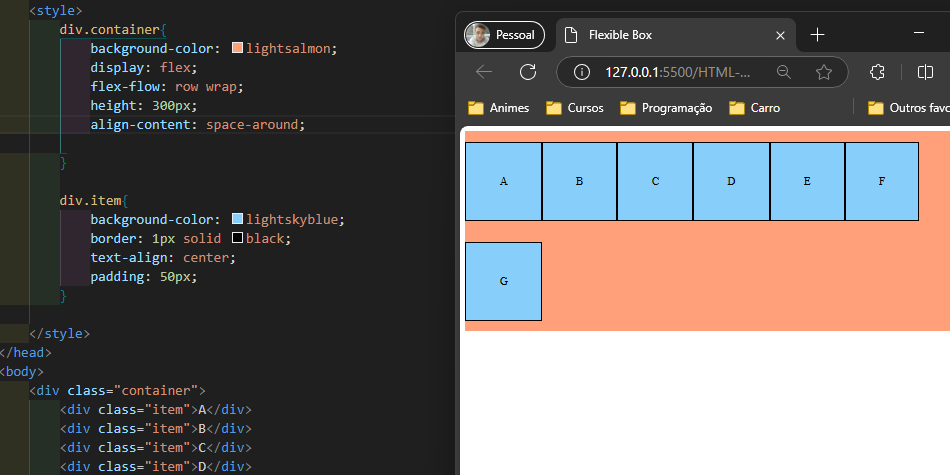
1. **Align-content: space-evenly**; (coloca elementos dispostos dentro do container, de forma que antes e depois tem espaçamento igual).



1. **Align-content: space-around**; (pega todos os elementos, divide em pedaço igual e centraliza nos espaços da divisão).



Exemplo:



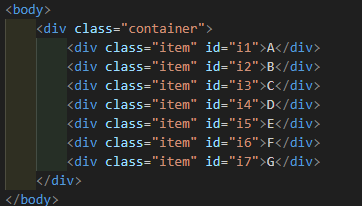
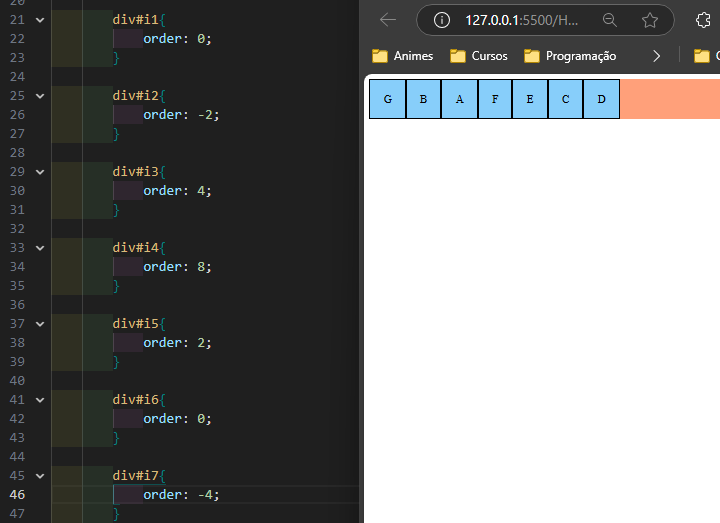
Agora que estudamos tudo sobre o pai(container), vamos estudar sobre os filhos (item)!

**Order:**

Todo item em flexbox que esta **dentro do pai tem order=0**; Mas se quiser mudar a ordem de exibição dos elementos, é só mudar o order. (em ordem crescente)

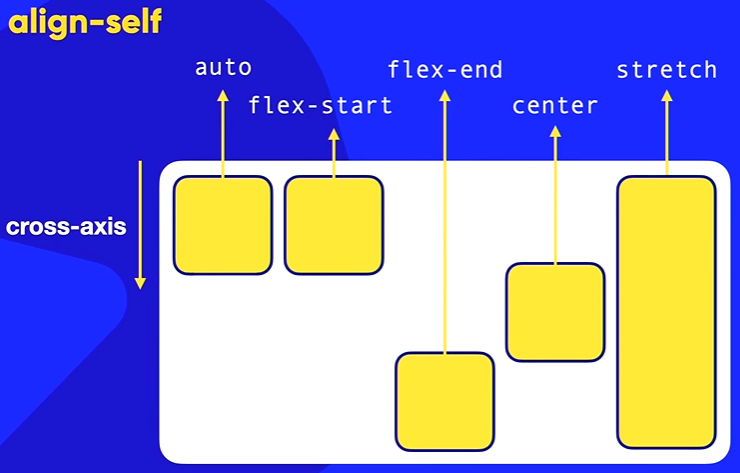


Ex:

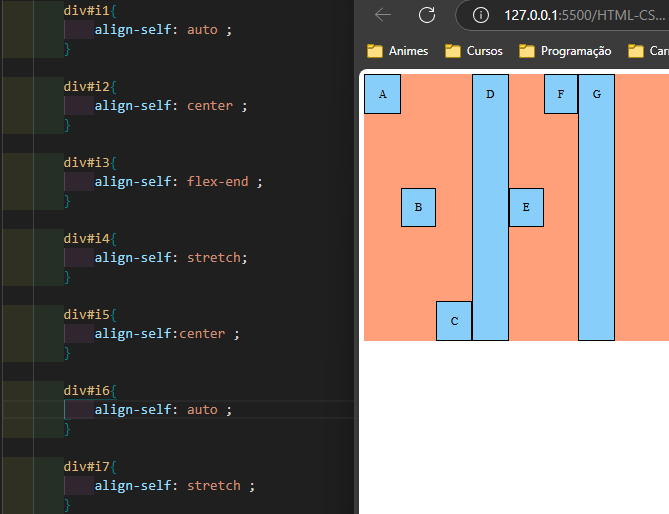


**Align-self:**

Se aplica aos items, e vai funcionar direto no eixo transversal (cross-axis).



Ex: Auto(herda do pai)



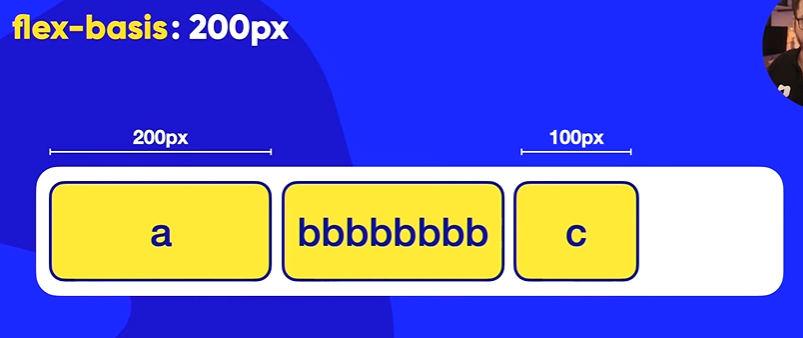
**Flex-basis:**

Mexe com o tamanho do elemento no item principal.

1. **flex-basis:auto**; (editado de acordo com o tamanho do conteúdo).Padrão.

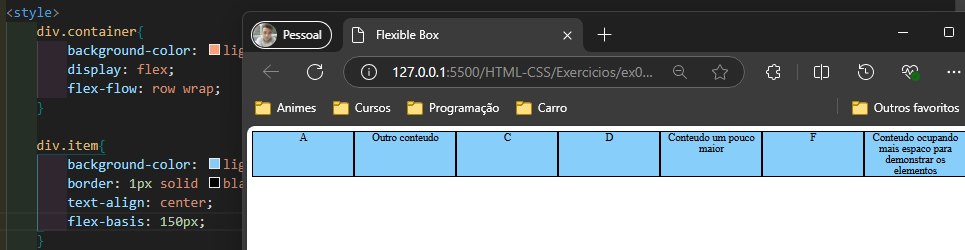


1. **flex-basis:...px**; (determino o tamanho). E posso colocar por item (so colocar id).

****

Vai ficar com 200px **enquanto der**, mas se não couber os 200px para cada, ele vai diminuir em tamanho igual. (dando para travar um elemento se quiser).

Ex:



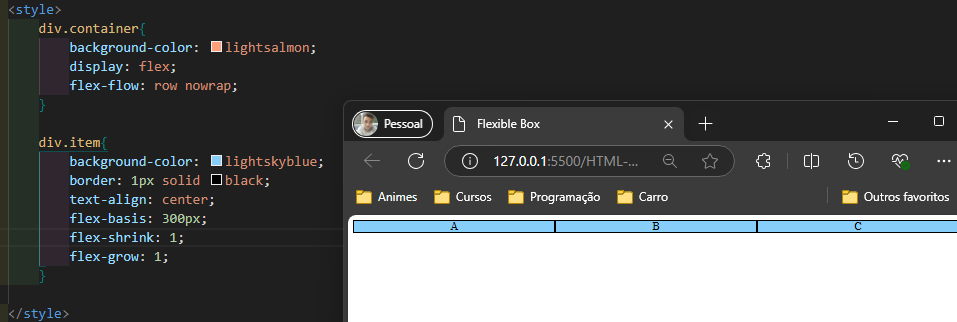
Como fixar um do tamanho que eu quero?

**Flex-shrink**: Enconher.

**Flex-grow**: Crescer.

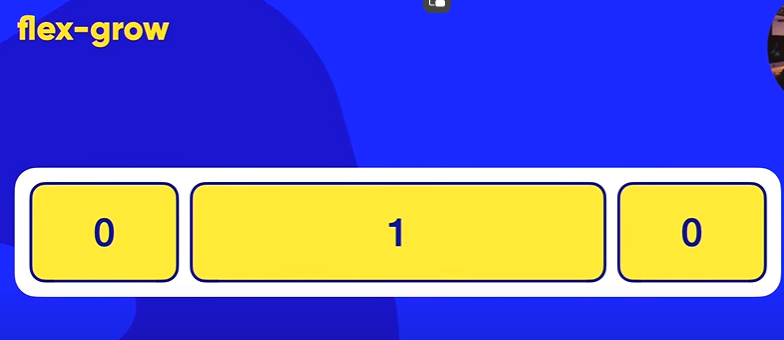
*Por padrão: flex-shrink:1; flex-grow:0;*

EX:



Mas ele não tem apenas 0 ou 1.

**Flex-grow:**



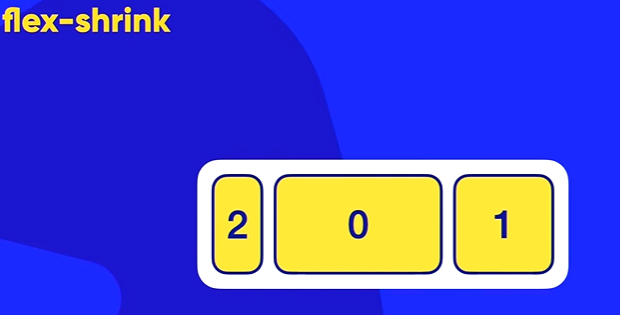


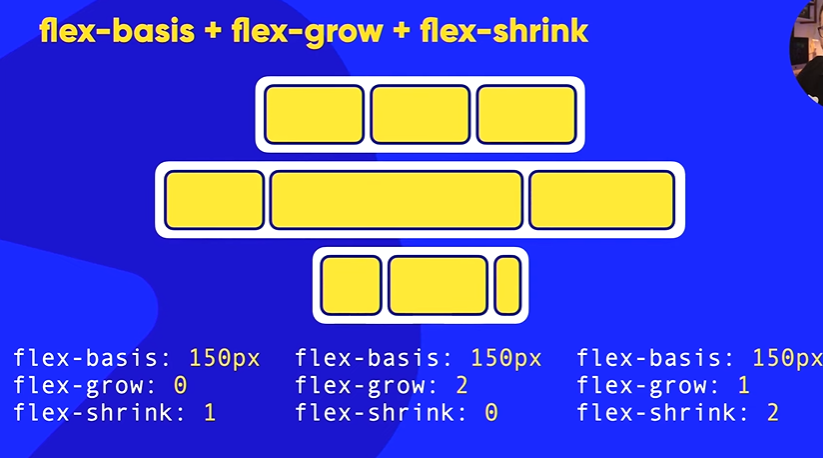
*\*cresceu o doblo do 1 elemento.*



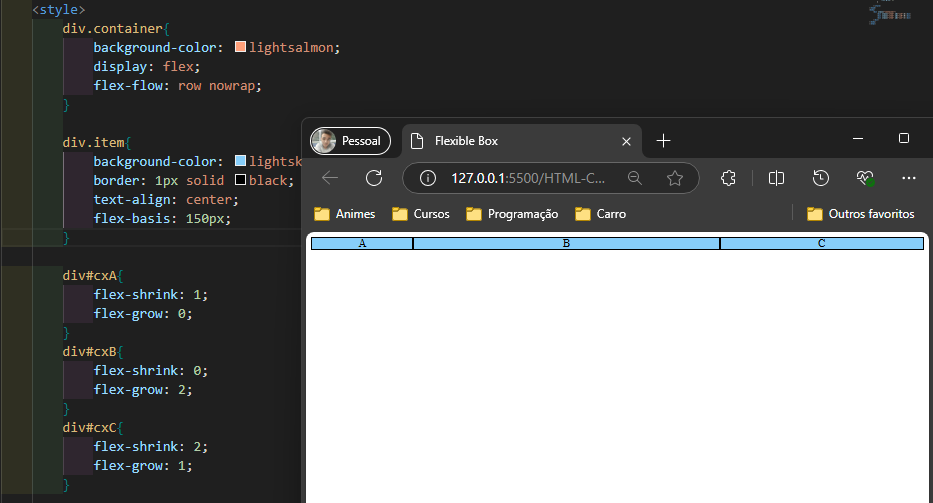
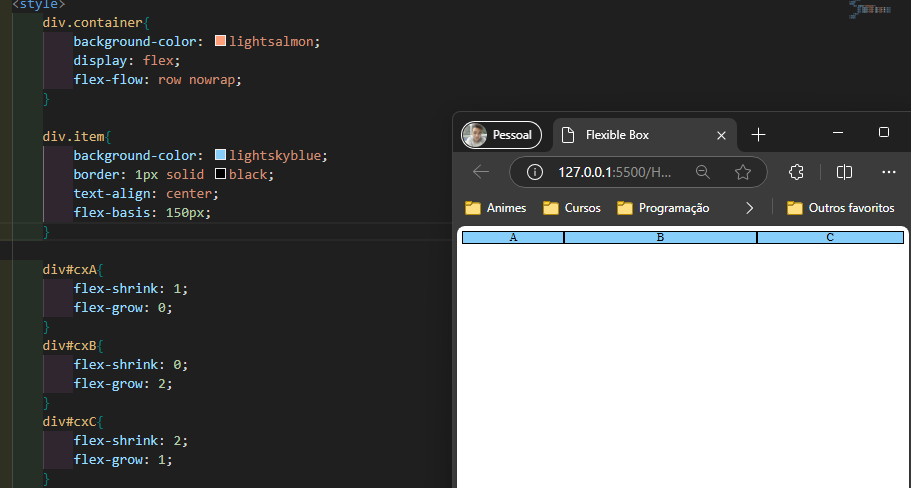
\*cresce com a mesma proporção.

**Flex-shrink:**



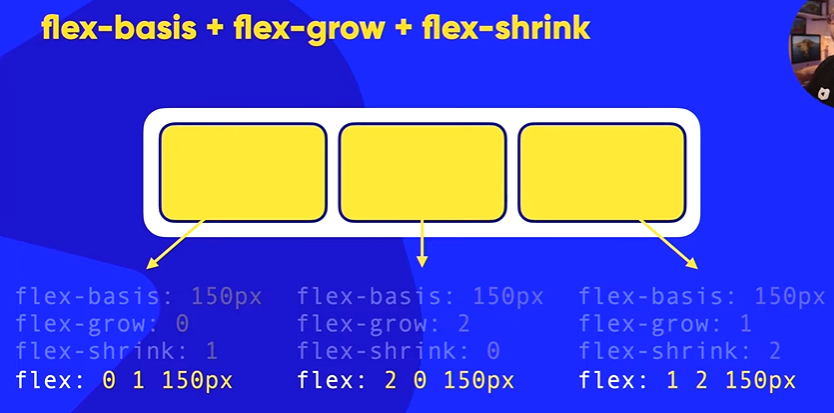


EX:

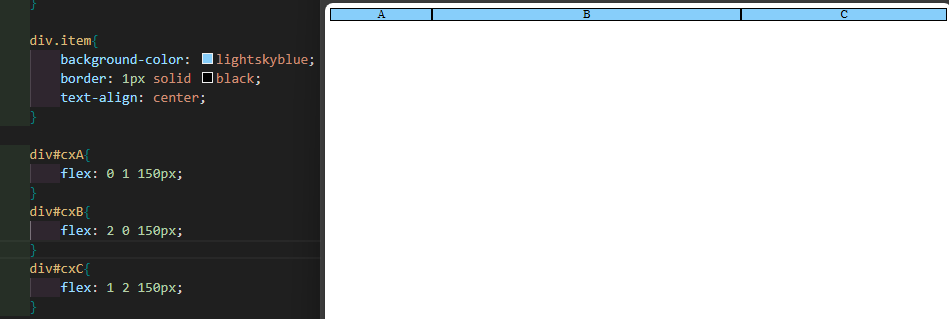


**Propriedade Flex: (apenas o item)**

*Flex= Grow, Shrink e Basis.*

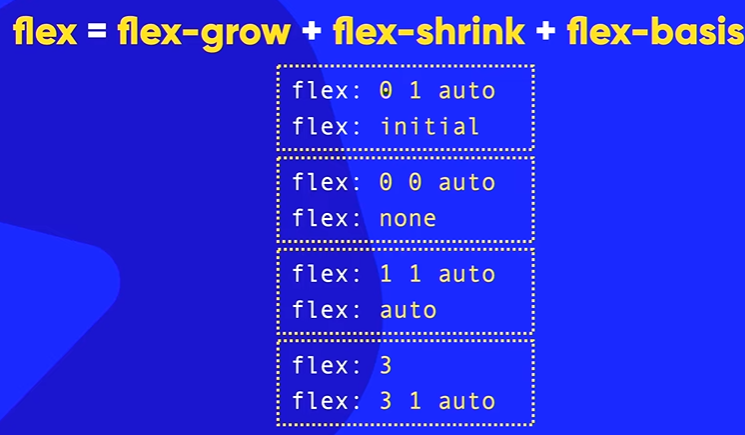


EX:



Resumindo ainda mais:

Padrao, nada flexível, tudo flexível:



O que seria então: Flex:3;

Significa que o apenas o grow é 3.