

Flexbox e Grid

Labenu_



Sumário

Labenu_



O que vamos ver hoje?

- Para alegria de muitos, a aula de hoje falará sobre **flexbox** e **grid**



Introdução

Labenu_



Dispondo e organizando elementos

- Uma tarefa comum no CSS é dispor elementos na tela
- Ex.
 - Centralizar elementos
 - Header e footer
 - Menu lateral
 - Card de informações



Antes...

- O CSS não fornecia muitas ferramentas para lidar com esse tipo de problema
- Existiam formas não muito intuitivas de posicionar as coisas
 - display block e inline
 - position relative e absolute
 - float
 - table



Grid e flexbox

- Para resolver problemas como esse, foram criados os display Grid e Flex
- Propriedades modernas - navegadores muito antigos não suportam
- Ambos são muito parecidos, e muitas vezes os dois resolvem os mesmos problemas
- Usados para **posicionar** elementos em relação a um **container**



Container e filhos

- **Container:**
 - Bloco (frequentemente uma div) que será a base do layout
 - Determina os limites do layout
- **Filhos**
 - Elementos a serem posicionados dentro do container
 - Somente filhos **diretos** são afetados



Container e filhos

- Exemplo:



Flexbox

Labenu_



Flexbox

- Dispõe elementos em **uma direção** (vertical ou horizontal)
- Cria um layout **flexível** - elementos filhos se ajustam
- Muito usado para criar layouts responsivos e fluidos



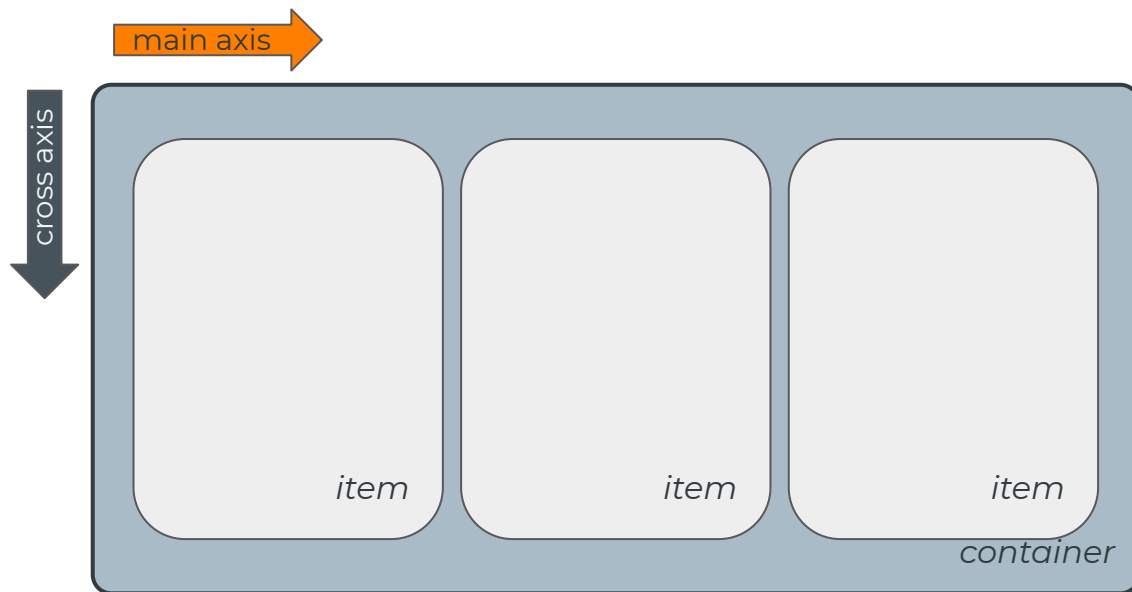
Flexbox

- Veremos os seguintes tópicos sobre o Flexbox:
 - A estrutura
 - Propriedades do container
 - Propriedades dos itens



Flexbox: Estrutura

- Possui o **eixo principal** (main axis) e o **eixo transversal** (cross axis)



Flexbox: Propriedades do Container



- **Propriedades do container**

- display: flex
- flex-direction 🏆1
- justify-content 🥈2
- align-items 🏆3



Flexbox: Propriedades do Container



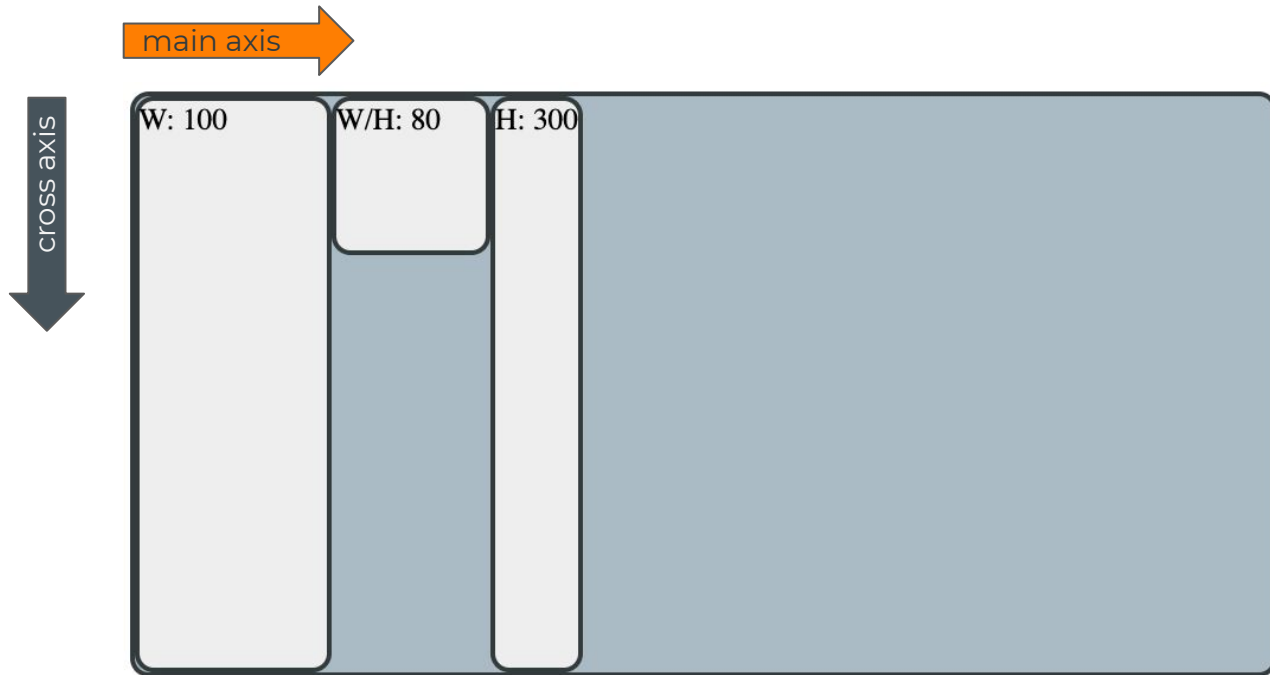
- **display: flex**
 - Um elemento container, com display: flex se comporta como um block-level
 - Afeta o comportamento dos elementos diretamente filhos



Flexbox: Propriedades do Container





- **display: flex**



Flexbox: Propriedades do Container



- **flex-direction**

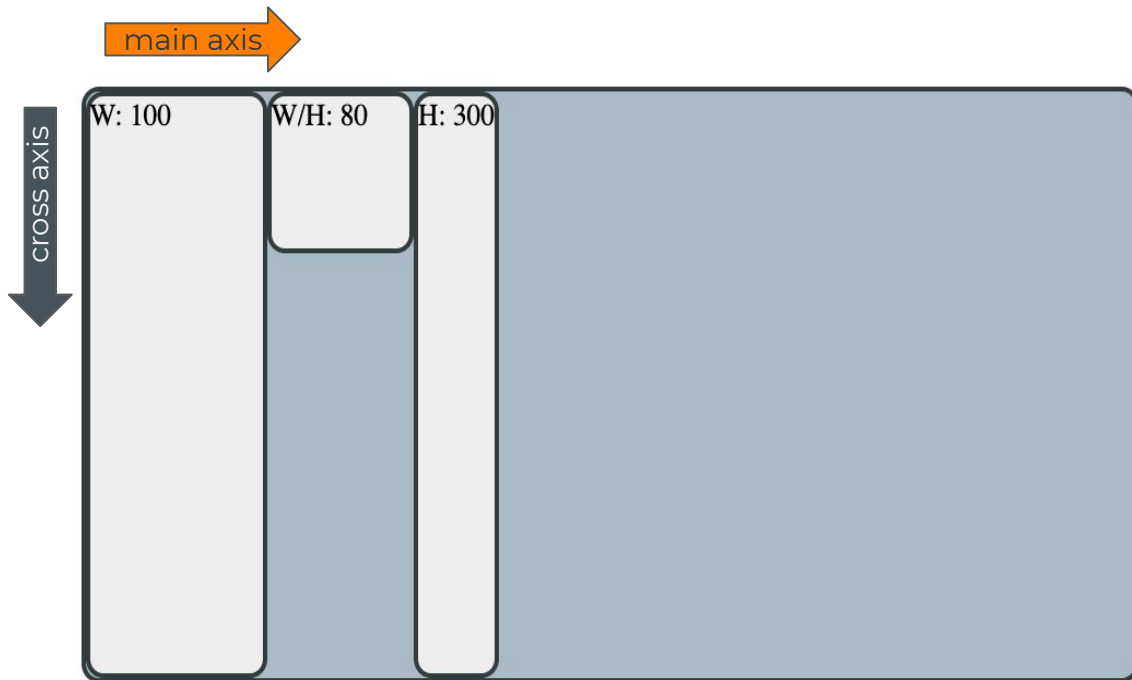
- Permite indicar a direção dos eixos
- Valores possíveis:
 - **row (padrão)** 
 - column 
 - row-reverse
 - column-reverse



Flexbox: Propriedades do Container



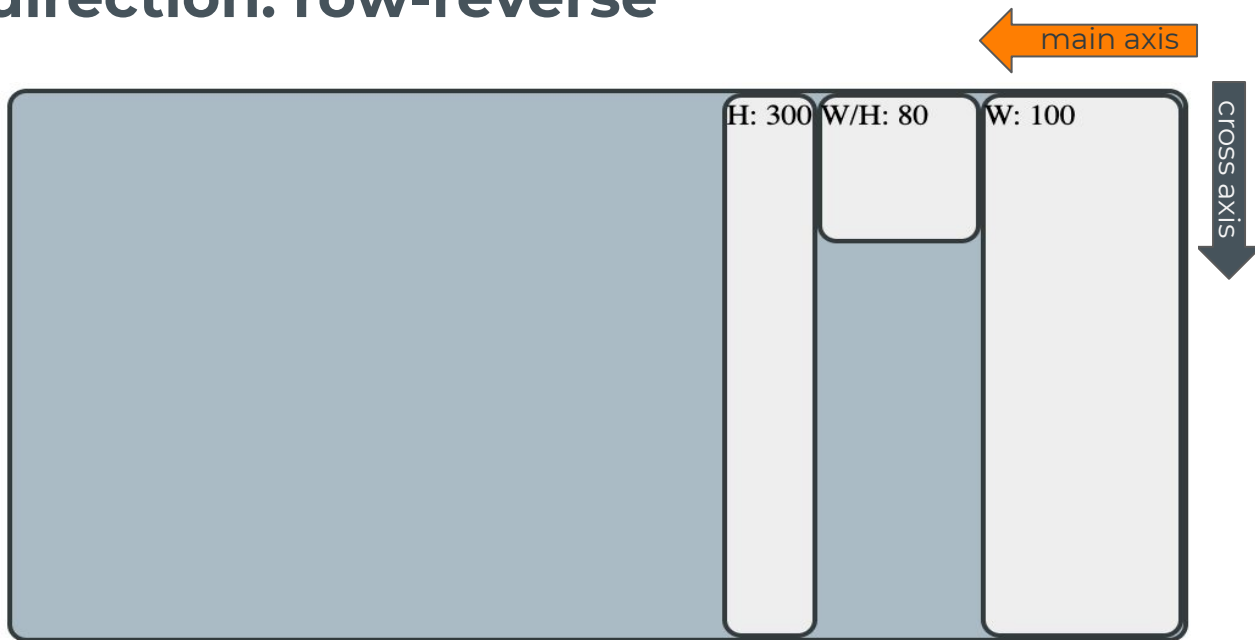
- **flex-direction: row**



Flexbox: Propriedades do Container



- **flex-direction: row-reverse**



Flexbox: Propriedades do Container



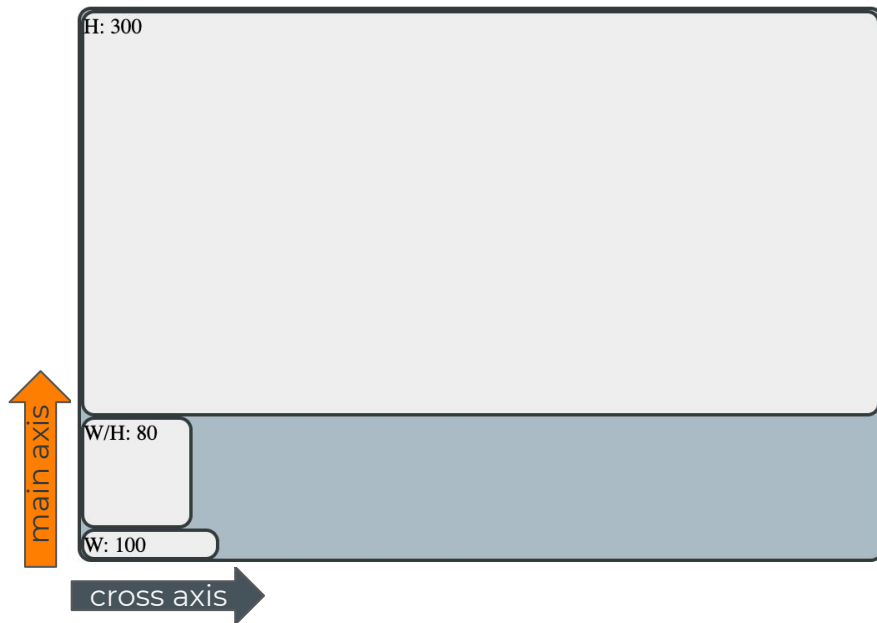
- **flex-direction: column**



Flexbox: Propriedades do Container



- **flex-direction: column-reverse**



Flexbox: Propriedades do Container



- **justify-content**

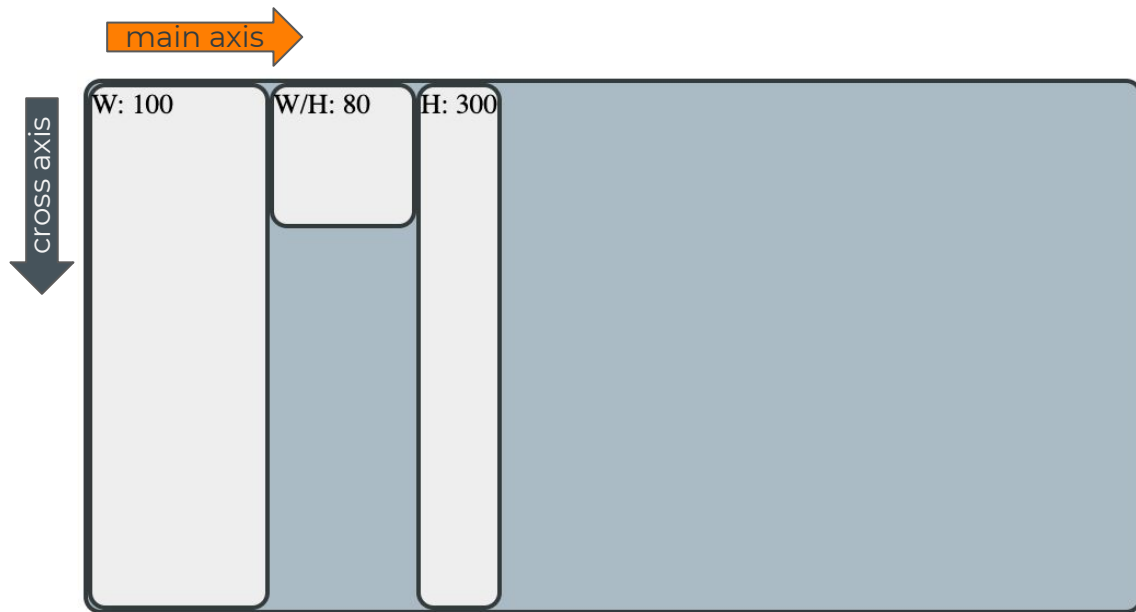
- Define a disposição e o alinhamento dos itens em relação ao main-axis
- Valores possíveis:
 - **flex-start (padrão)**
 - flex-end
 - center 🏆1
 - space-between 🥈2
 - space-around 🏆3
 - space-evenly 🏆3



Flexbox: Propriedades do Container



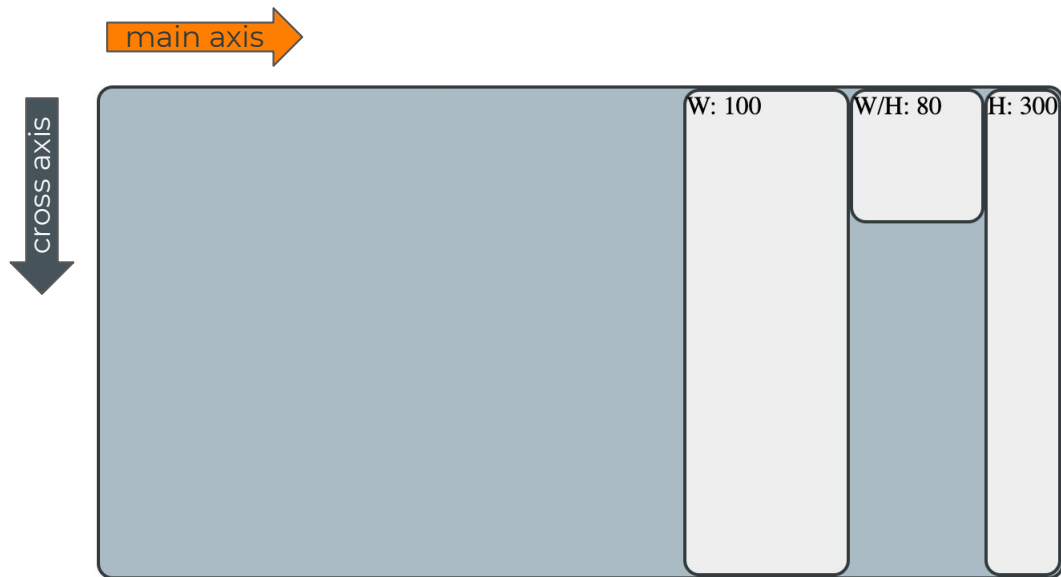
- **justify-content: flex-start**



Flexbox: Propriedades do Container



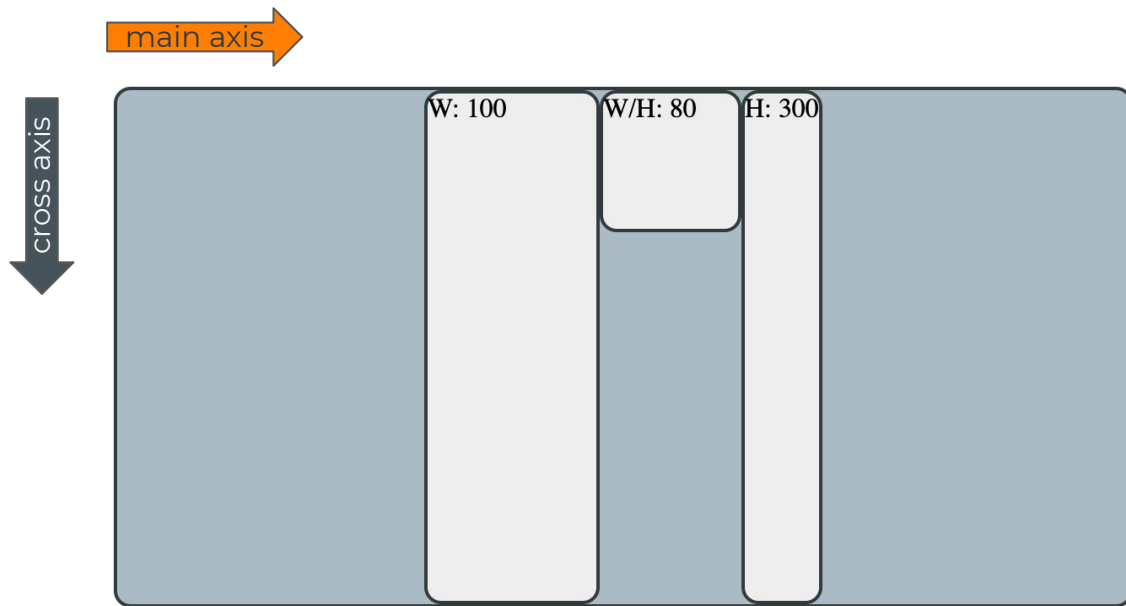
- **justify-content: flex-end**



Flexbox: Propriedades do Container



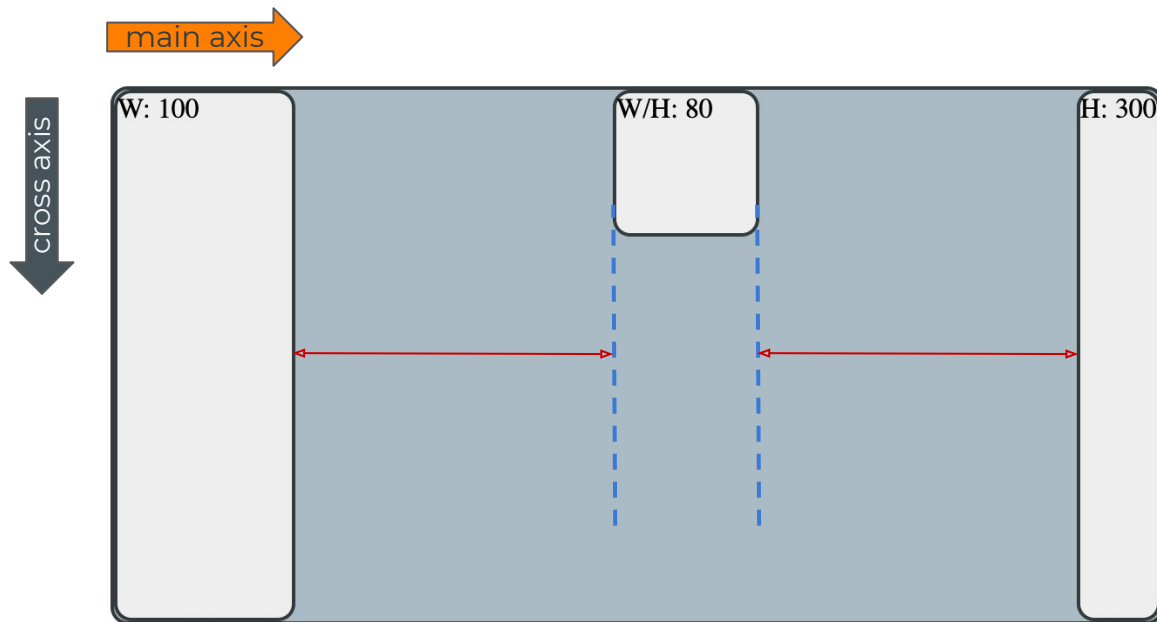
- **justify-content: center**



Flexbox: Propriedades do Container



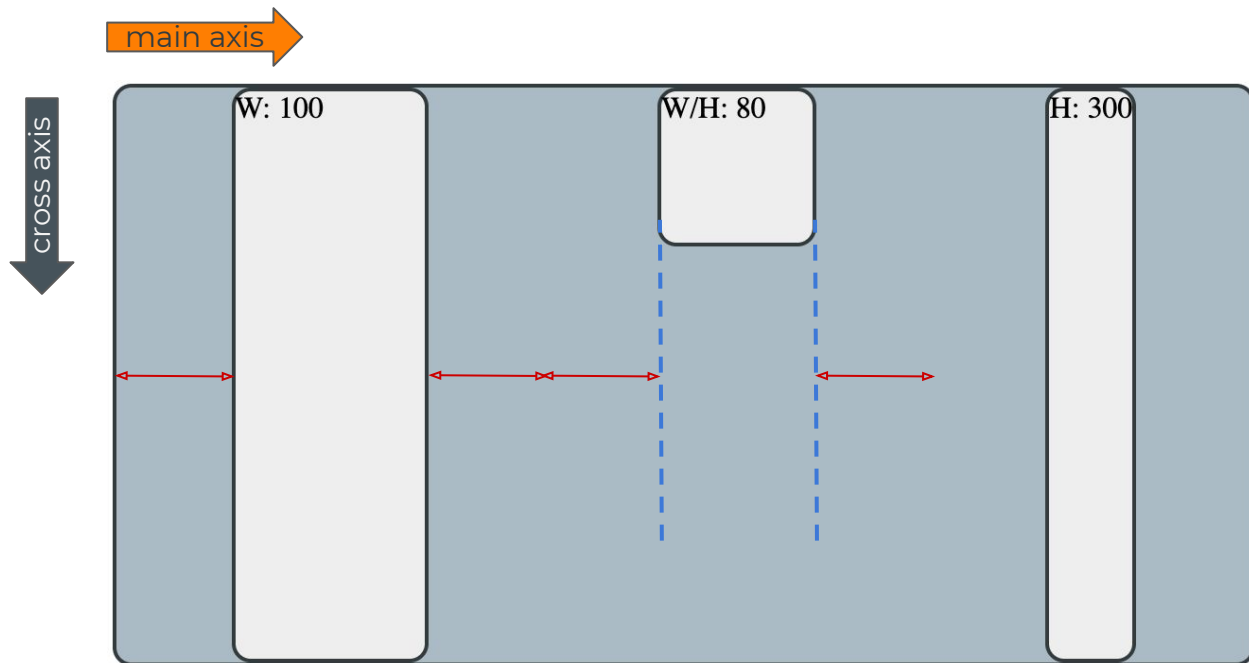
- **justify-content: space-between**



Flexbox: Propriedades do Container



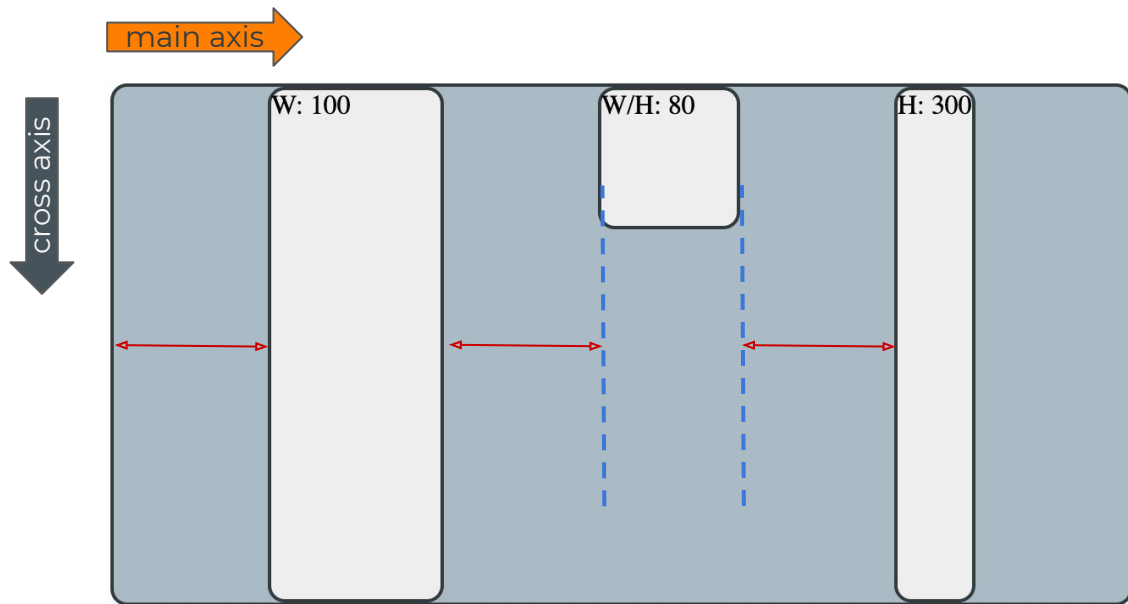
- **justify-content: space-around**



Flexbox: Propriedades do Container



- **justify-content: space-evenly**



Flexbox: Propriedades do Container



- **align-items**

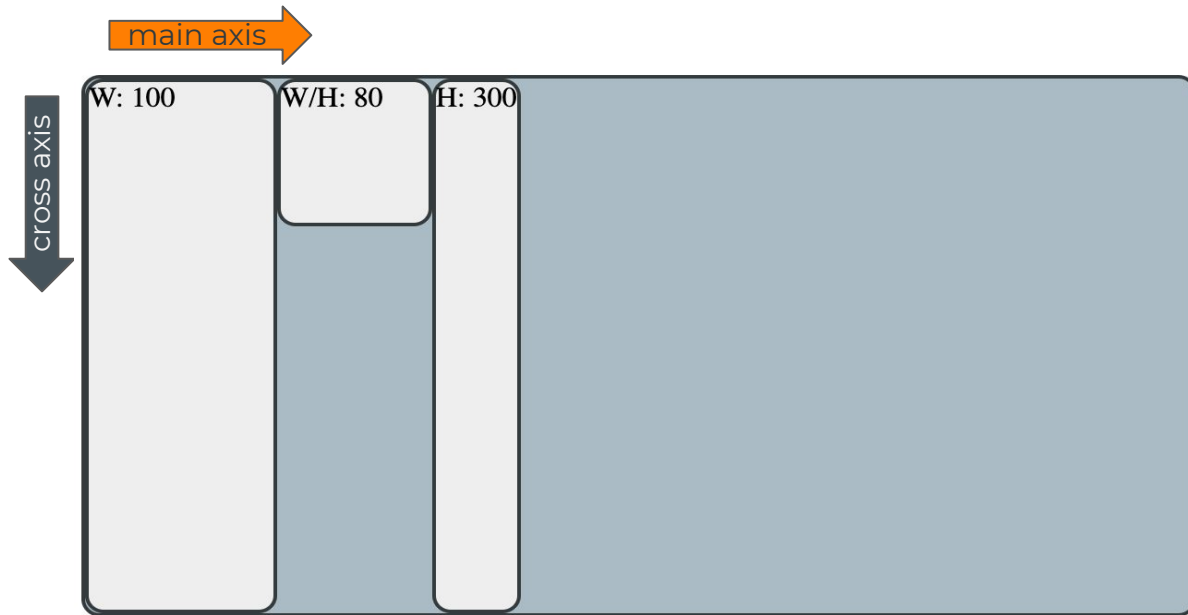
- Define a disposição e o alinhamento dos itens em relação ao cross-axis
- Valores possíveis:
 - **stretch (padrão)**
 - flex-start 🏅 3
 - flex-end 🥈 2
 - center 🥇 1
 - baseline



Flexbox: Propriedades do Container



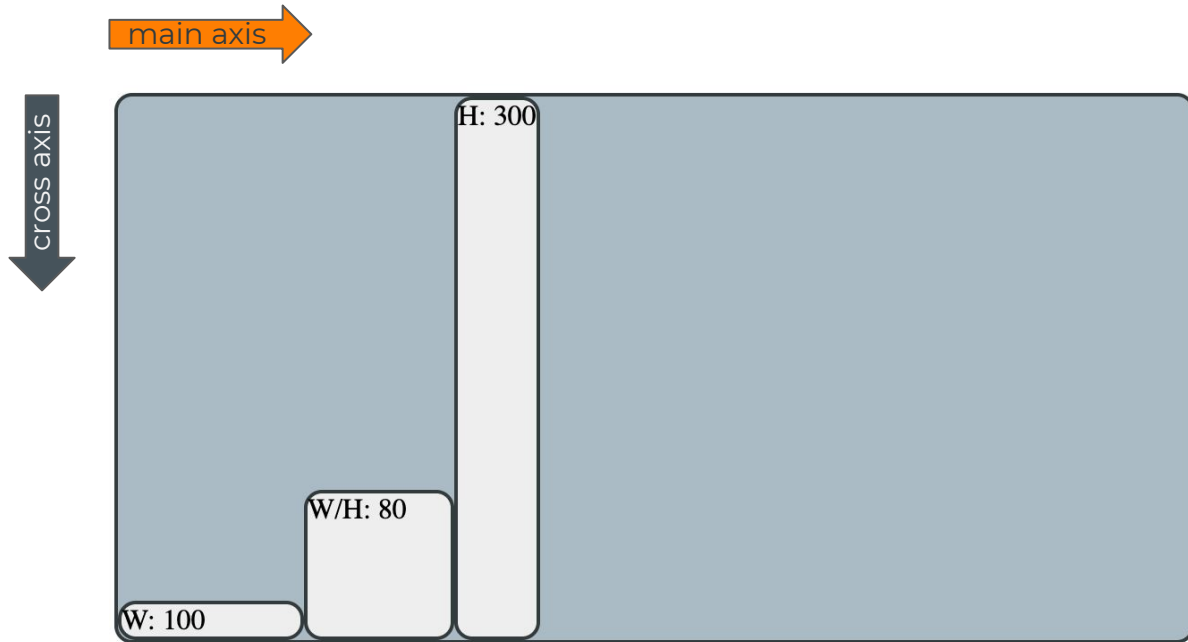
- **align-items: stretch**



Flexbox: Propriedades do Container



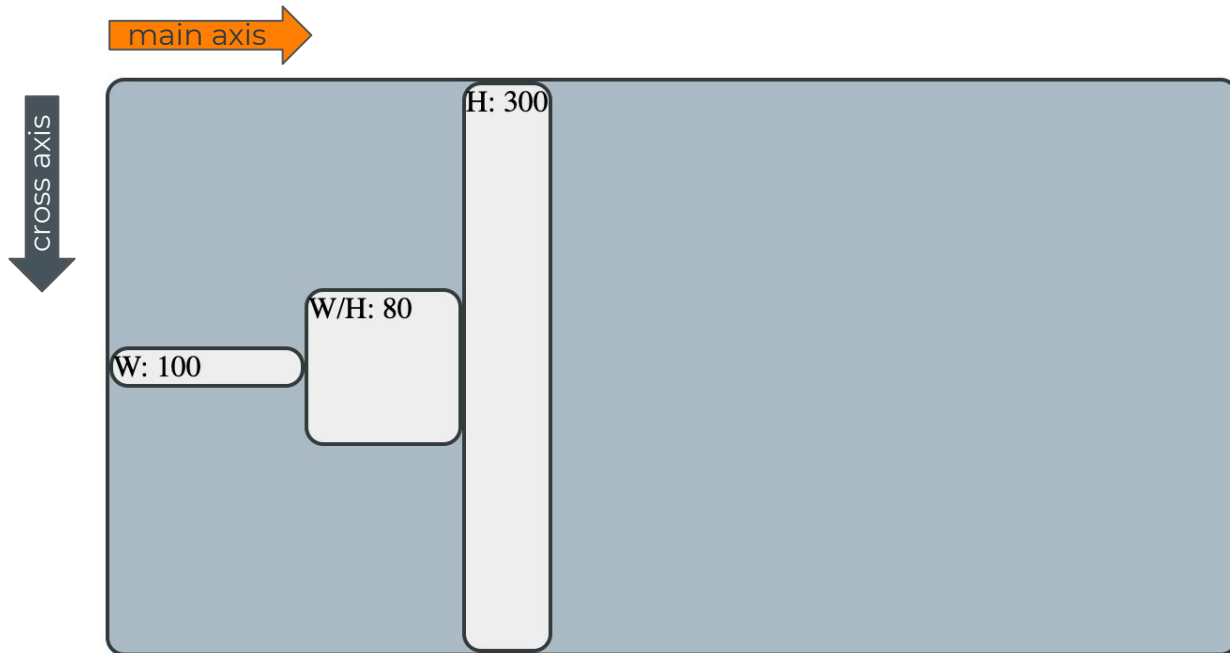
- **align-items: flex-end**



Flexbox: Propriedades do Container



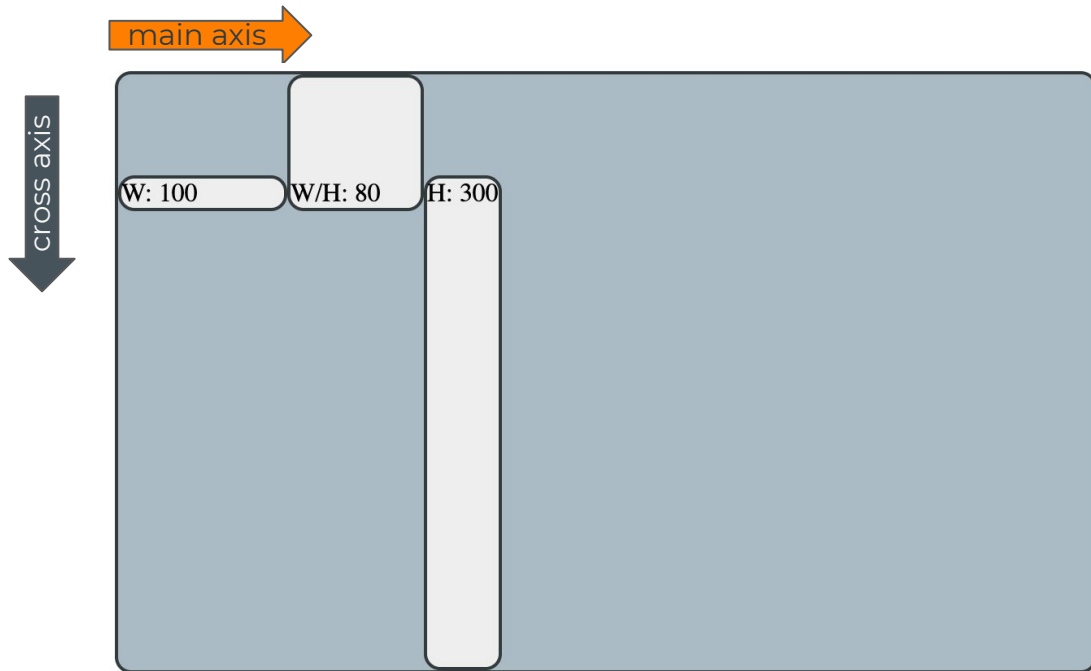
- **align-items: center**



Flexbox: Propriedades do Container





- **align-items: baseline**



Flexbox: Propriedades do Item

- **Propriedades do item**

- order
- flex-grow 
- flex-shrink
- flex-basis
- align-self 



Flexbox: Propriedades do Item

- **order**
 - Indica a ordem dos itens dentro do container, em relação ao main-axis
 - Quanto maior o valor, mais para o final do main-axis
 - Padrão é 0



Flexbox: Propriedades do Item

- **flex-grow**

- Define a habilidade dos elementos de crescer para ocupar o espaço vazio do main-axis
- Quanto maior o número, mais ele irá crescer
- Padrão é 0



Flexbox: Propriedades do Item

- **flex-shrink**

- Define a habilidade dos elementos de diminuir para ocupar o espaço vazio do main-axis
- Quanto maior o número, mais ele irá diminuir
- Padrão é 1



Flexbox: Propriedades do Item

- **flex-basis**

- Define o tamanho do elemento no main-axis antes do espaço vazio antes de ser distribuído
- Pode assumir os tamanhos com todas as unidades (px, pt, %, etc.)
- Se o valor for auto(padrão), o elemento assume o tamanho em width/height



Flexbox: Propriedades do Item

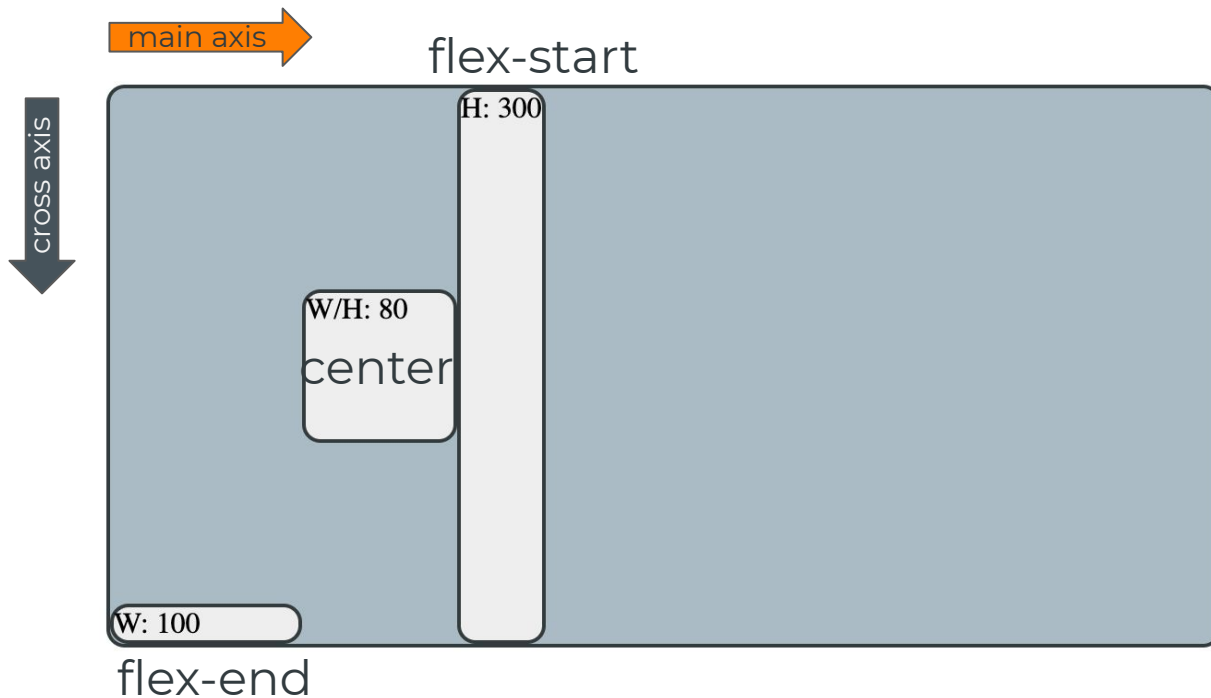
- **align-self**

- Determina a disposição do elemento em relação ao cross-axis
- Valores possíveis:
 - **auto(padrão)**
 - stretch
 - flex-start
 - flex-end
 - center
 - baseline



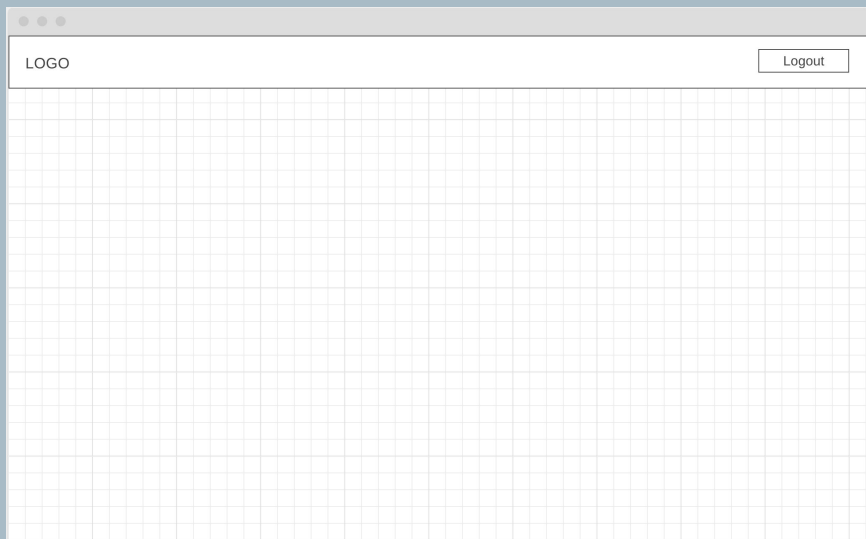
Flexbox: Propriedades do Item

- align-self



Exercício 1 - *Flex Box*

- Implemente um Header seguindo esse layout. Garanta que o Logo e o botão estão centralizados verticalmente.



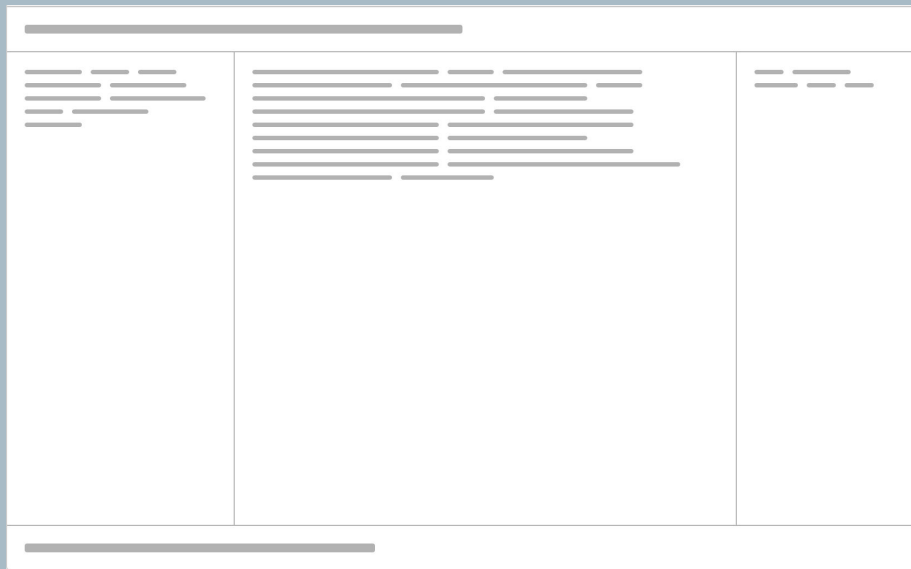
Exercício 2 - *Flex Box*

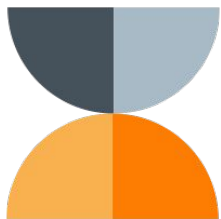
- Usando o último exercício, implemente o seguinte layout:



Exercício 3 - *Flex Box*

- Usando o último exercício, implemente o seguinte layout:





Pausa para relaxar 🧘



Grid

Labenu_



Grid

- **Grid** é outra poderosa ferramenta que facilita
 - O **posicionamento** dos elementos
 - O **alinhamento** deles
- Ele funciona como uma distribuição em **duas direções**, ou seja, ou os elementos estarão distribuídos na **horizontal** e na **vertical**.



Grid

- **Boilerplate:** vamos utilizar este arquivo para ver o funcionamento da Grid



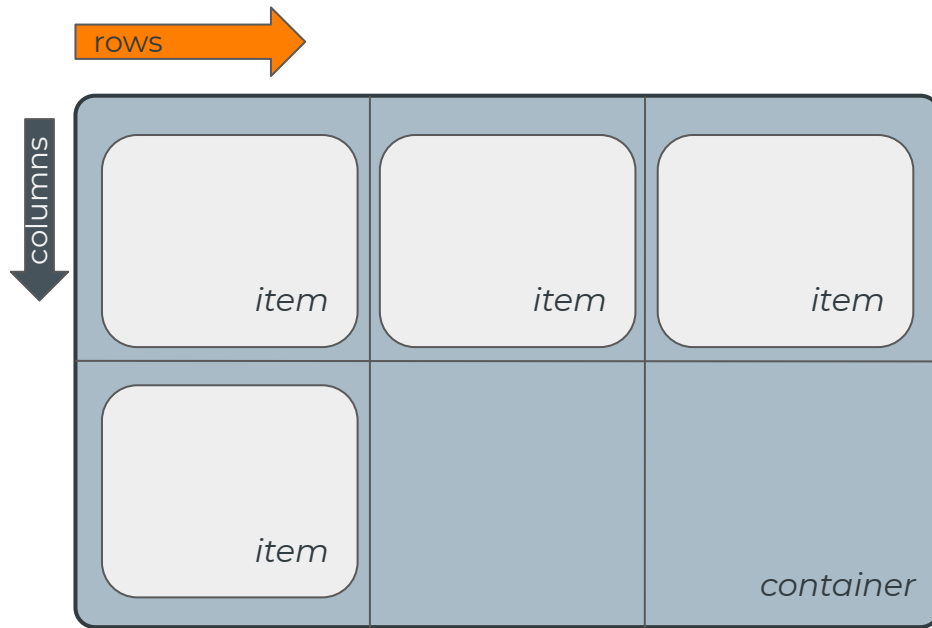
Grid

- Veremos os seguintes tópicos sobre o Grid:
 - A estrutura
 - Propriedades do container
 - Propriedades dos itens





Grid

- Possui o **eixo das linhas** (rows) e o **eixo das colunas** (columns)



Grid: Propriedades do Container

- **Propriedades do container**

- display: grid
- grid-template-rows/columns 
- grid-row/column-gap 
- justify-items
- align-items



Grid: Propriedades do Container

- **display: grid**

- Um elemento container, com display: grid se comporta como um block-level
- Afeta o comportamento dos elementos diretamente filhos



Grid: Propriedades do Container

- `display: grid`



Vamos ver na prática! 



Grid: Propriedades do Container

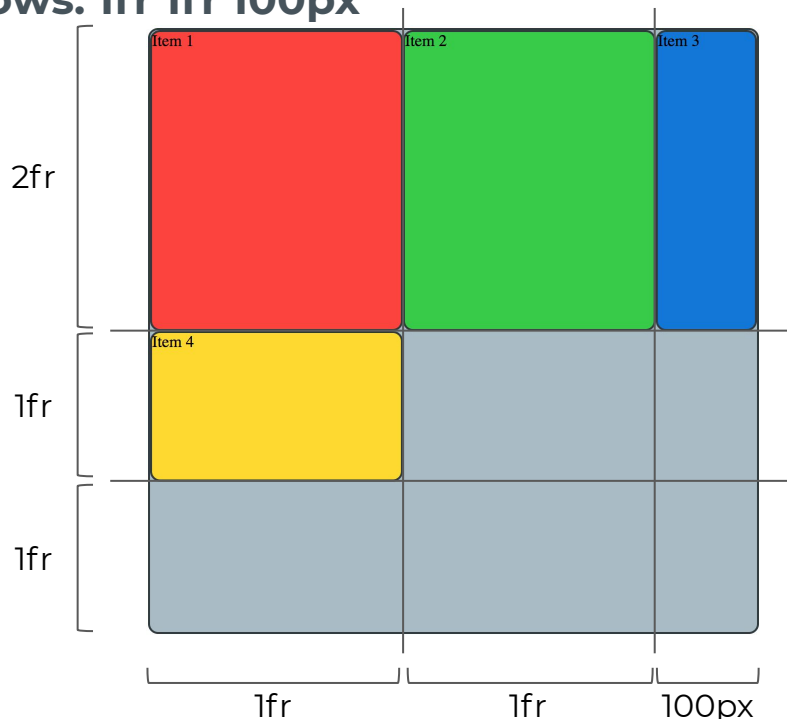
- **grid-template-rows/columns**
 - Determina o template de linhas ou colunas, ou seja, diz a quantidade e tamanho delas
 - Recebe série de tamanhos separados por espaço
 - 1fr: Indica uma fração de espaço vazio

Vamos ver na prática! 



Grid: Propriedades do Container

- **grid-template-columns: 2fr 1fr 1fr**
grid-template-rows: 1fr 1fr 100px



Grid: Propriedades do Container

- **grid-row/column-gap**
 - Determina o espaço entre as linhas ou colunas
 - Recebe um valor numérico de tamanho
 - Novidade: o nome mudou para row/column-gap

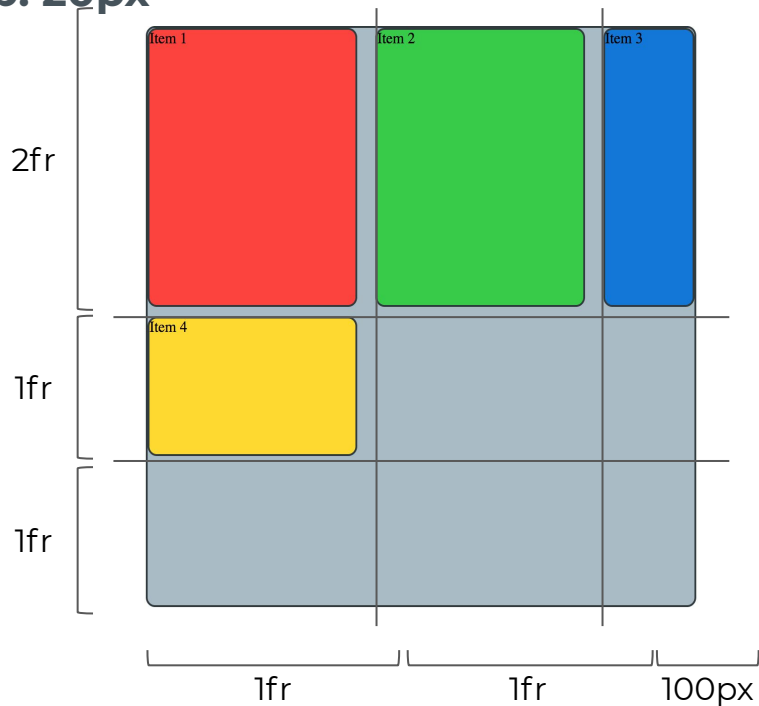
Vamos ver na prática! 



Grid: Propriedades do Container



- **grid-row-gap: 10px**
grid-column-gap: 20px



Grid: Propriedades do Container

- **justify-items**

- Determina a posição dos elementos no eixo das linhas
- Valores possíveis:
 - **stretch (padrão)**
 - start
 - end
 - center

Vamos ver na prática! 



Grid: Propriedades do Container

- **align-items**

- Determina a posição dos elementos no eixo das colunas
- Valores possíveis:
 - **stretch (padrão)**
 - start
 - end
 - center

Vamos ver na prática! 



Grid: Propriedades do Item

- **Propriedades do Item**
 - grid-row/column-start/end
 - justify/align-self



Grid: Propriedades do Item

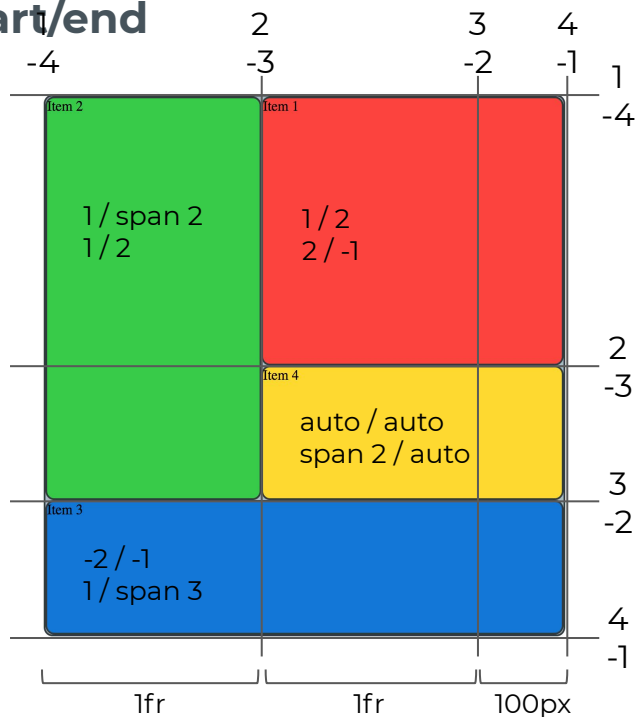
- **grid-row/column-start/end**
 - Determina em qual linha ou coluna o item começa ou acaba
 - Recebe o número da divisória em questão (começa do 1)
 - Pode receber o valor span seguido de um número, que determina um número de células a serem ocupadas

Vamos ver na prática! 



Grid: Propriedades do Item

- **grid-row/column-start/end**



Legenda:
row-start / row-end
col-start / col-end

Vamos ver na prática!



Grid: Propriedades do Item

- **justify/align-self**

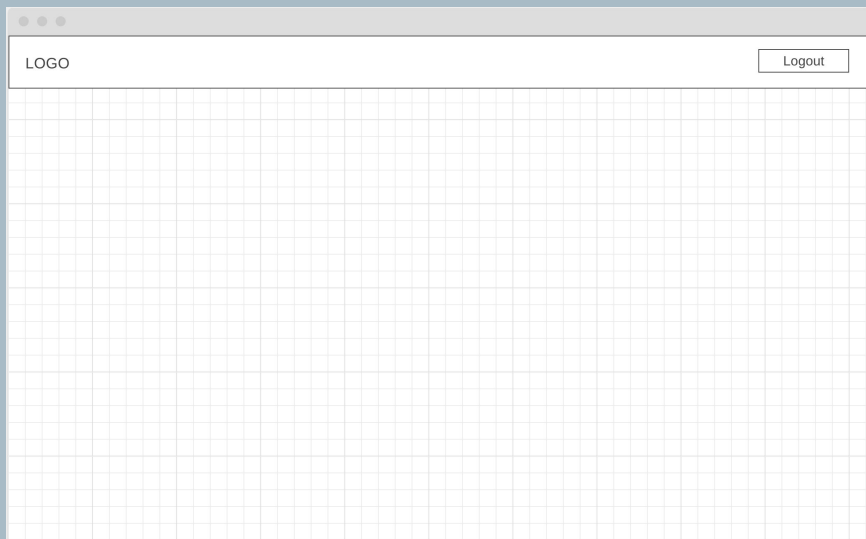
- Comportamento igual às propriedades justify-items e align-items do container
- Diz respeito somente ao item e sobrepõe as do container
- Valor padrão é auto, que segue as propriedades do container

Vamos ver na prática! 



Exercício 4 - *Grid*

- Implemente um Header seguindo esse layout. Garanta que o Logo e o botão estão centralizados verticalmente.



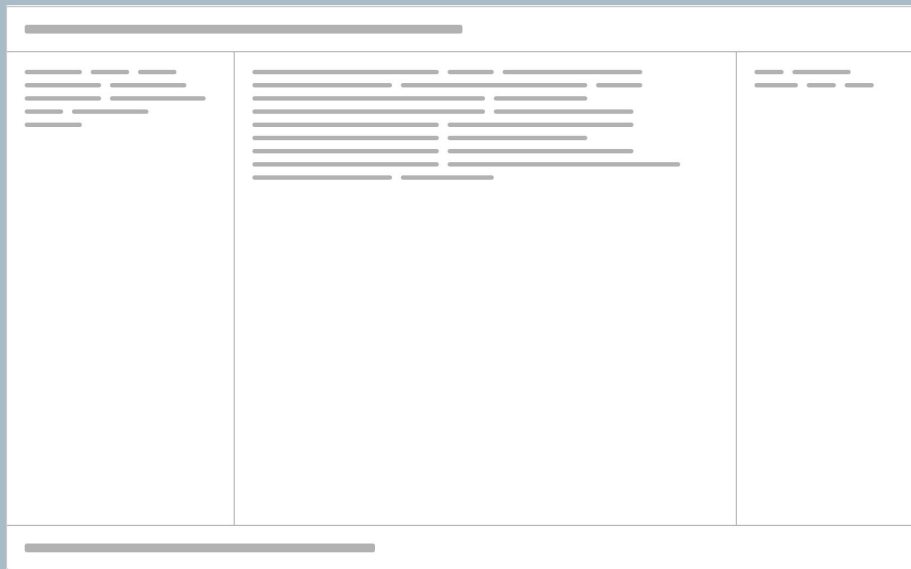
Exercício 5 - *Grid*

- Usando o último exercício, implemente o seguinte layout:



Exercício 6 - *Grid*

- Usando o último exercício, implemente o seguinte layout:



Flexbox VS Grid

Labenu_



Flexbox VS Grid

- Usamos **flexbox** quando:
 - O layout for em uma só direção
 - Elementos flexíveis (tamanho e posições)
 - **Conteúdo primeiro** - layout se adapta ao conteúdo
- Usamos **grid** quando:
 - O layout for em duas dimensões
 - Tamanho e posições mais determinadas
 - **Layout primeiro** - conteúdo se adapta ao layout

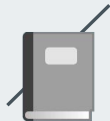


Resumo

Labenu_



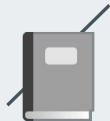
Resumo



- Flexbox
 - Estrutura
 - Possui um eixo principal e um eixo transversal
 - Propriedades do Container
 - display: flex
 - flex-direction
 - justify-content
 - align-items



Resumo



- Flexbox
 - Propriedades do Item
 - order
 - flex-grow
 - flex-shrink
 - flex-basis
 - align-self



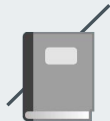
Resumo



- Grid
 - Estrutura
 - Dividida em linhas (horizontal) e colunas (vertical)
 - Propriedades do Container
 - display: grid
 - grid-template-rows/columns
 - grid-row/column-gap
 - justify-items
 - align-items



Resumo



- Grid
 - Propriedades do Item
 - grid-row/column-start/end
 - justify/align-self



Dúvidas?

Labenu_





Obrigado!