

Flexbox e Grid

Labenu_



O que vamos ver hoje?

- Motivação
- Flexbox
- Grid
- Exercícios para praticar!



Motivação

Labenu_



Dispor Elementos na Tela ➡

- Uma tarefa muito comum no CSS é **dispor elementos na tela** de uma determinada maneira
- Por exemplo:
 - Centralizar elementos
 - Posicionar Header e Footer
 - Menu lateral
 - Card com informações
 - ...



Dispor Elementos na Tela ➡

- Antigamente, o CSS não fornecia muitas ferramentas para lidar com esse tipo de problema
- As formas que existiam para posicionamento não eram muito intuitivas:
 - display **block** e **inline**
 - position **relative** e **absolute**
 - float
 - table



Grid e Flexbox

- Para resolver problemas como esses, foram criados dois novos tipos de display: **grid** e **flex**
- Ambos são muito parecidos, e muitas vezes os dois resolvem os mesmos problemas
- São usados para **posicionar elementos (itens)** que estão **dentro** (são filhos diretos) de um **container**



Grid e Flexbox

- **Container:**

- Bloco (frequentemente uma div) que será a **base** do layout, ele agrupa os itens
- Determina os limites do layout

- **Itens (Filhos):**

- Elementos posicionados dentro do container
- Somente **filhos diretos** são afetados pelas propriedades do grid/flex



Grid e Flexbox



```
1 <div id="container">
2   <div id="item1">Filho</div>
3   <div id="item2">Filho</div>
4   <div id="item3">Filho</div>
5 </div>
```



Flexbox

Labenu_



Flexbox

- É uma ferramenta que dispõe os elementos em **uma única 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 a Flexbox:
 - Estrutura
 - Propriedades do **Container**
 - Propriedades dos **Itens**



Flexbox - Aviso

- Veremos várias propriedades de itens e containers:
 - **NÃO se preocupe em decorar todas!** Se não se lembrar o que cada uma faz, basta pesquisar no google ou testar no seu próprio código 😊
 - Apresentaremos as propriedades **mais usadas**
 - Para **relembrar** ou **saber mais**, recomendamos [esse site](#) (inglês) ou [esse outro](#) (português)



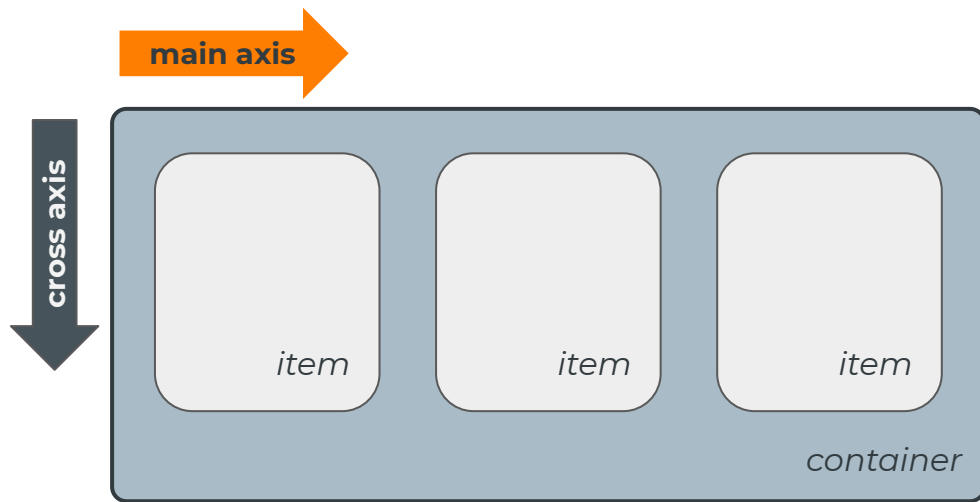
Estrutura

Labenu_



Flex - Estrutura

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



Propriedades do Container

Labenu_



Flex - Propriedades do Container

- Principais propriedades do container:
 - display: flex 
 - flex-direction 
 - justify-content 
 - align-items 





Flex - Propriedades do Container

- **display: flex**
 - Define que aquele container segue as regras do **flex** e permite utilizar as demais propriedades
 - Um elemento com **display: flex** se comporta como **block-level**
 - Afetará o posicionamento dos elementos que são **filhos diretos** do container

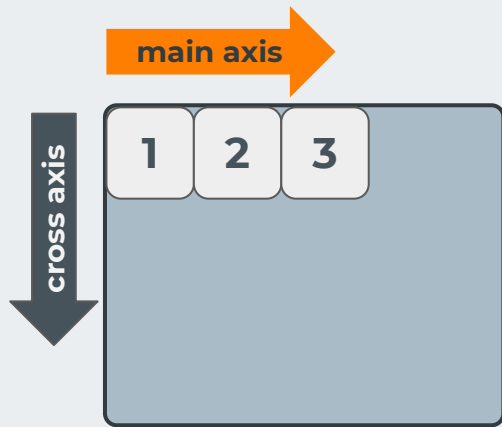


Flex - Propriedades do Container

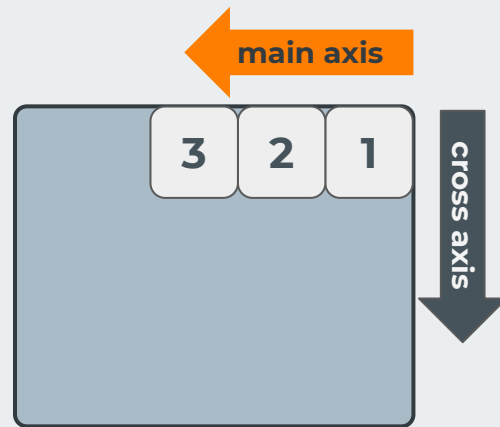
- **flex-direction**: permite indicar a **direção dos eixos** que, basicamente, determina como queremos dispor os itens (em uma **linha** ou **coluna**)
- Valores possíveis:
 - **row (padrão)** 
 - column 
 - row-reverse
 - column-reverse

A direção dos itens sempre segue o **main axis**

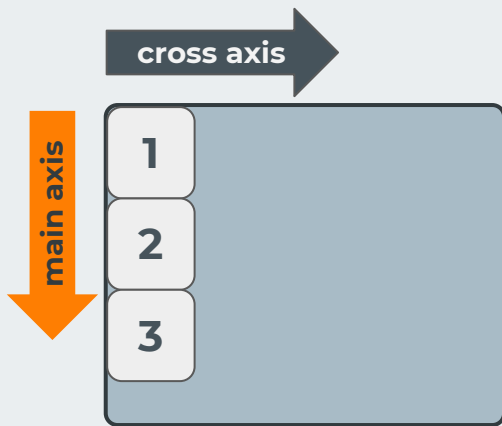




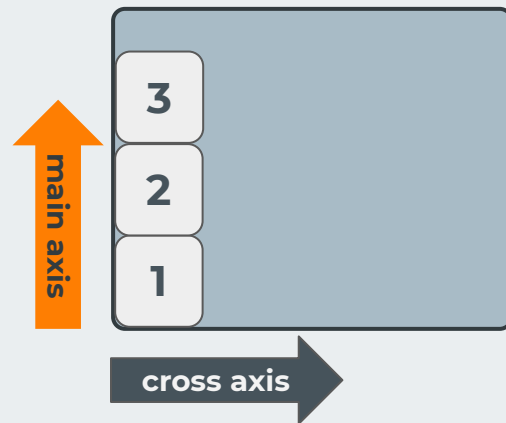
flex-direction: **row**



flex-direction: **row-reverse**






flex-direction: **column**



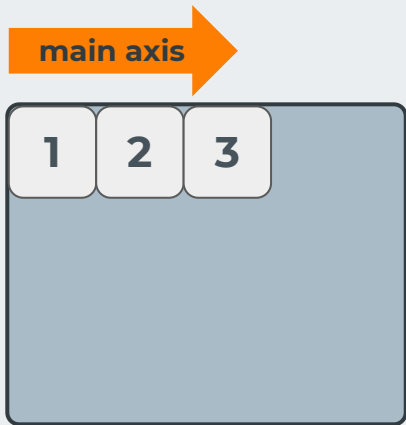
flex-direction: **column-reverse**



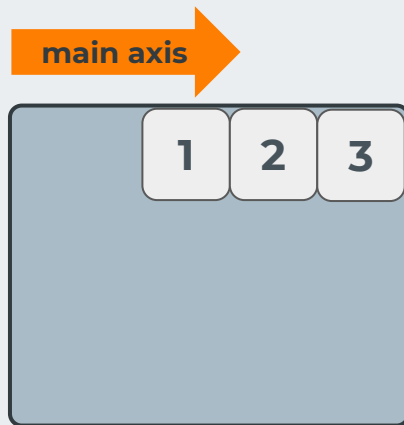
Flex - Propriedades do Container

- **justify-content**: define a posição e alinhamento dos itens em relação ao **main axis**
- Valores possíveis:
 - **flex-start (padrão)**
 - flex-end
 - center 
 - space-between 
 - space-around
 - space-evenly 





`justify-content: flex-start`



`justify-content: flex-end`



`justify-content: center`



`justify-content: space-between`



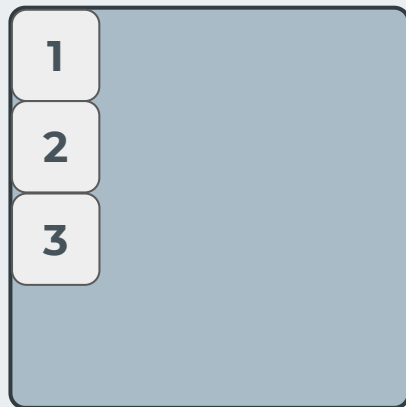
`justify-content: space-around`



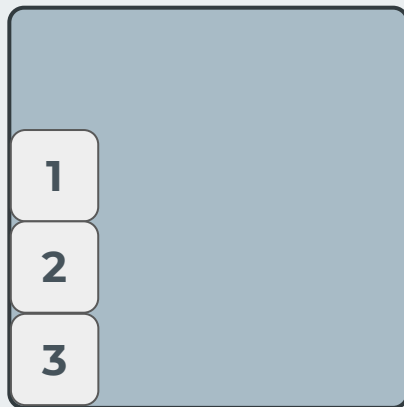
`justify-content: space-evenly`



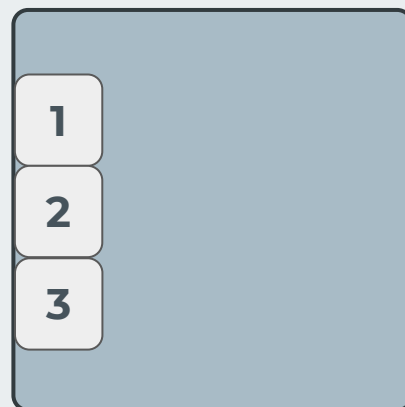
main axis

`justify-content: flex-start`

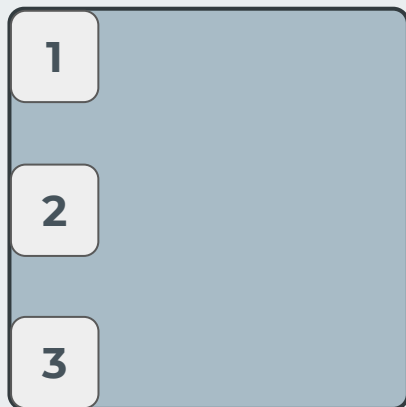
main axis

`justify-content: flex-end`

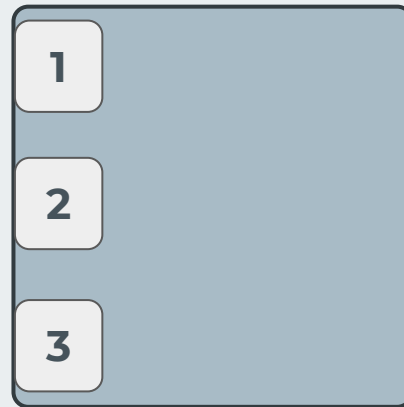
main axis

`justify-content: center`

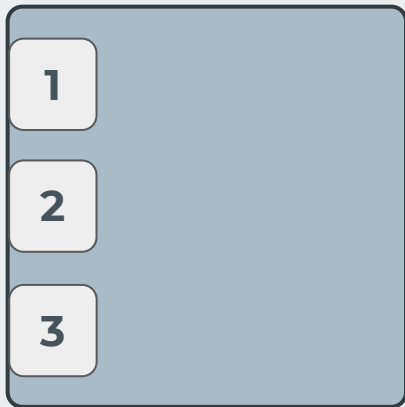
main axis

`justify-content: space-between`




main axis

`justify-content: space-around`

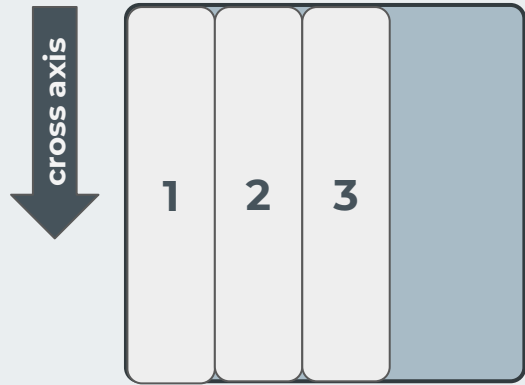
main axis

`justify-content: space-evenly`

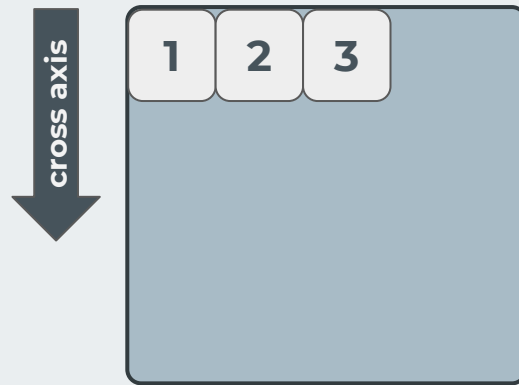
Flex - Propriedades do Container

- **align-items**: define a posição e alinhamento dos itens em relação ao **cross axis**
- Valores possíveis:
 - **stretch (padrão)**
 - flex-start 
 - flex-end 
 - center 

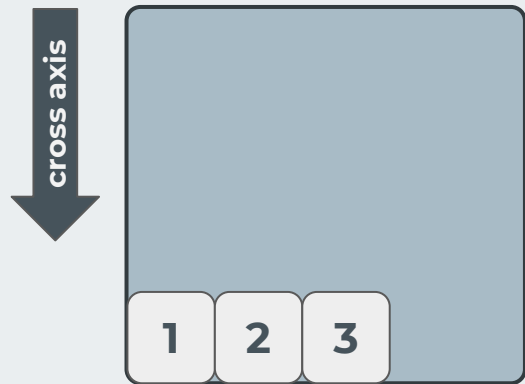




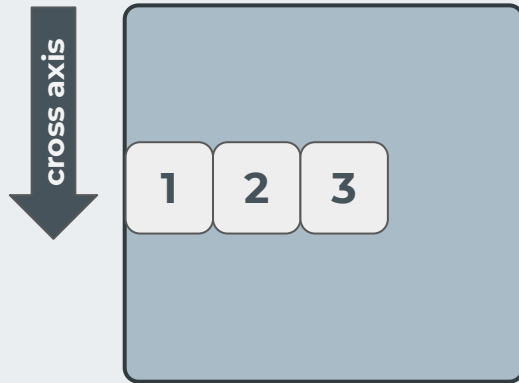
align-items: **stretch**



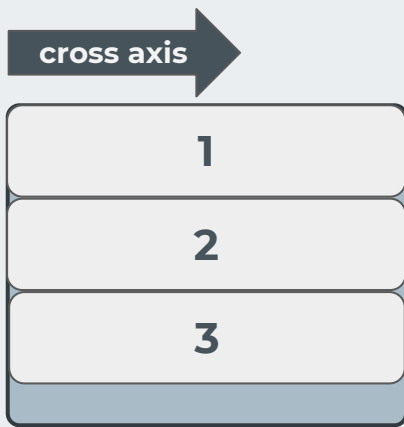
align-items: **flex-start**



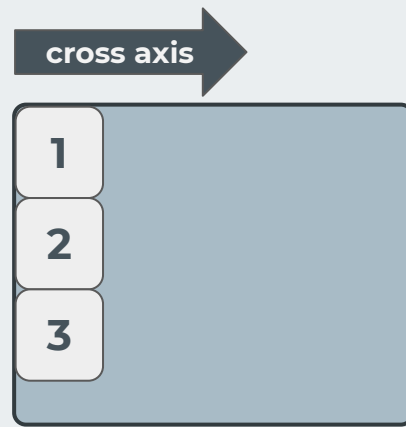
align-items: **flex-end**



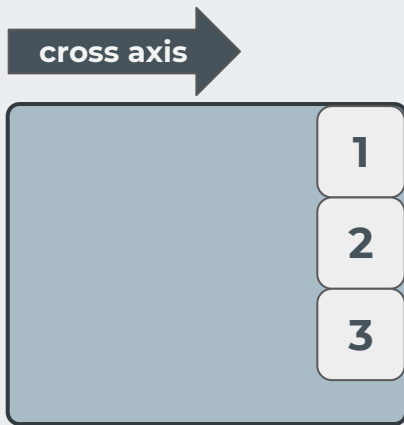
align-items: **center**



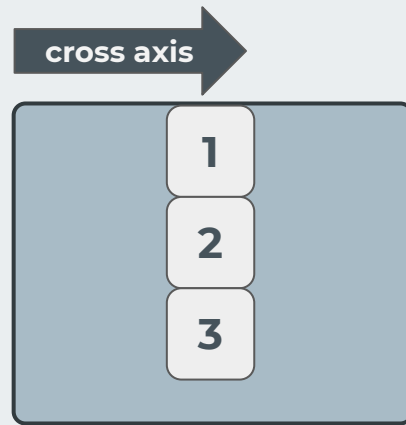
`align-items: stretch`



`align-items: flex-start`



`align-items: flex-end`



`align-items: center`

Flex - Propriedades do Container

- **flex-wrap**: Define se os itens devem quebrar ou não a linha. Por padrão eles não quebram linha, isso faz com que os flex itens sejam compactados além do limite do conteúdo.
- Valores possíveis:
 - **flex-wrap: nowrap (padrão)**
 - flex-wrap: wrap;
 - flex-wrap: wrap-reverse;



flex-wrap





Propriedades dos Itens

Labenu_



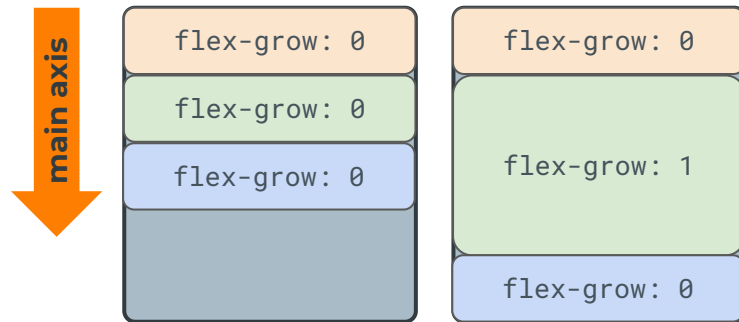
Flex - Propriedades dos Itens

- Principais propriedades dos itens:
 - flex-grow 
 - align-self 
- As propriedades dos itens servem para fazer com que **um item específico saia do padrão** imposto pelo container



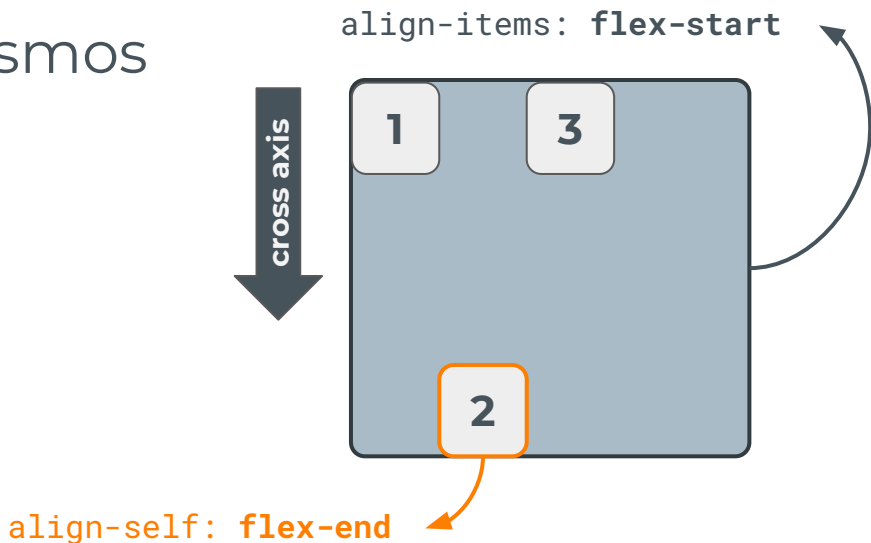
Flex - Propriedades dos Itens

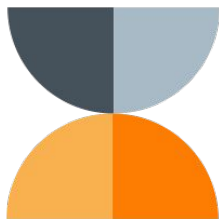
- **flex-grow**: define a habilidade dos elementos de **crescer** para ocupar o **espaço vazio do main axis**
- Quanto maior o número atribuído à essa propriedade, mais o item **cresce**
- O valor **padrão é 0**



Flex - Propriedades dos Itens

- **align-self**: determina a disposição do elemento em relação ao **cross axis**, sobrescrevendo o **align-items**
- **Valores possíveis**: mesmos que os do align-items
 - stretch
 - flex-start
 - flex-end
 - center





Pausa para relaxar 🧘

10 min

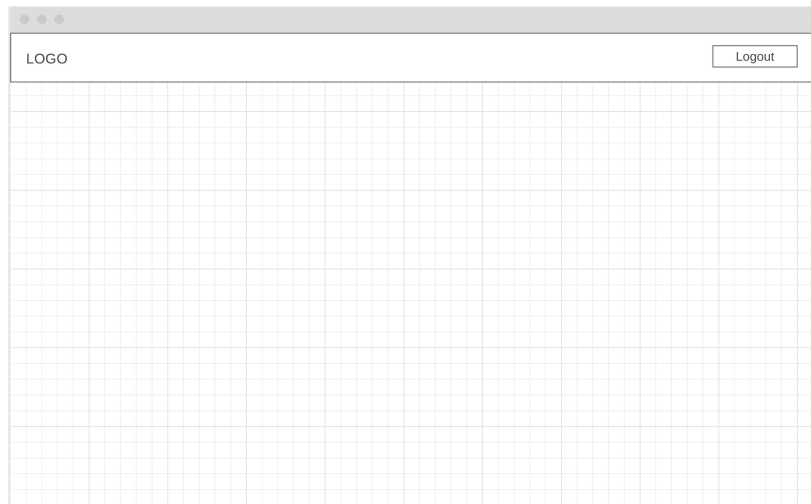
- Aplicamos propriedades de flex em um **container** para distribuir seus itens no eixo horizontal (row) ou vertical (column)
- Propriedades podem ser aplicadas aos **itens** para fugir do padrão imposto pelo container
- Não se preocupe em decorar tudo, **teste e pesquise!**





Exercício 1

- Implemente um Header seguindo o layout ao lado utilizando flexbox
- Garanta que o logo e o botão estejam centralizados verticalmente





Exercício 2

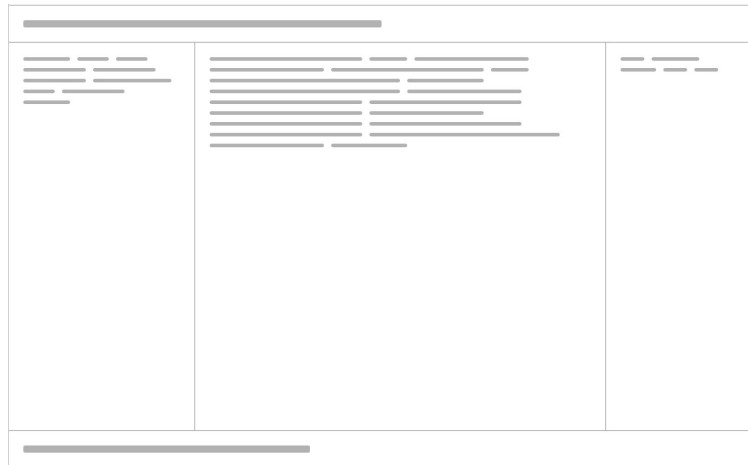
- Utilizando o Header do exercício anterior, implemente a página ao lado que possui:
 - Header
 - Conteúdo
 - Footer





Exercício 3

- Utilizando o site do exercício anterior, implemente na sua página:
 - Menus laterais na esquerda e na direita do conteúdo principal



Pausa para relaxar 🐱💤

5 min



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

- Veremos os seguintes tópicos sobre o Grid:
 - Estrutura
 - Propriedades do **Container**
 - Propriedades dos **Itens**



Grid - Aviso

- Veremos várias propriedades de itens e containers:
 - **NÃO se preocupe em decorar todas!** Se não se lembrar o que cada uma faz, basta pesquisar no google ou testar no seu próprio código 😊
 - Apresentaremos as propriedades **mais usadas**
 - Para **relembrar** ou **saber mais**, recomendamos [esse site](#) (inglês) ou [esse outro](#) (português)



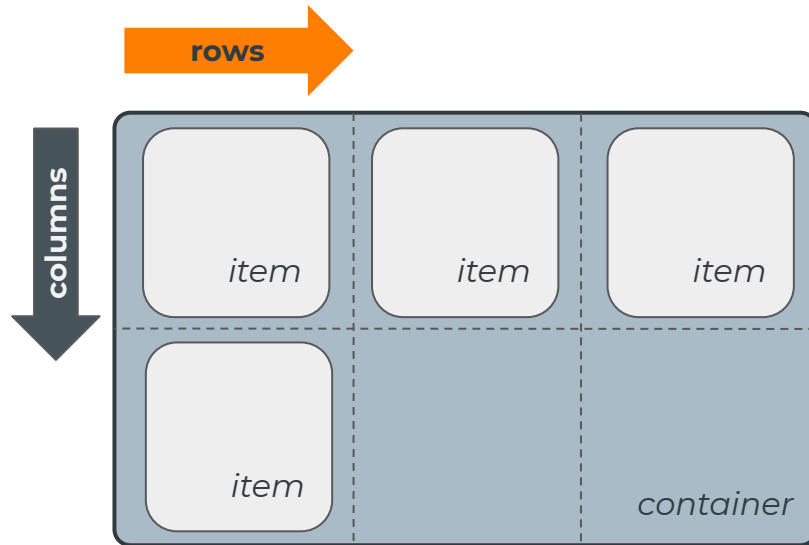
Estrutura

Labenu_



Grid - Estrutura

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





Propriedades do Container

Labenu_



Grid - Propriedades do Container

- Principais propriedades do container:
 - display: grid
 - grid-template-rows e grid-template-columns 
 - row-gap e column-gap 
 - justify-items
 - align-items
- As propriedades do container servem principalmente para determinar o layout da **malha em si**



Grid - Propriedades do Container

- **display: grid**
 - Define que aquele container segue as regras do grid e permite utilizar as demais propriedades
 - Um elemento com **display: grid** se comporta como **block-level**
 - Afetará o posicionamento dos elementos que são **filhos diretos** do container



Grid - Propriedades do Container

- **grid-template-rows** e **grid-template-columns**:
definem o template das linhas e colunas
 - Determina a quantidade e tamanho de cada
 - **1fr**: indica uma fração do espaço vazio
 - Primeiro distribui o **espaço fixo** (por exemplo, em pixels) e depois distribui o **espaço vazio**



```
1 display: grid;  
2 grid-template-rows: 2fr 1fr 1fr;  
3 grid-template-columns: 1fr 1fr 100px;
```

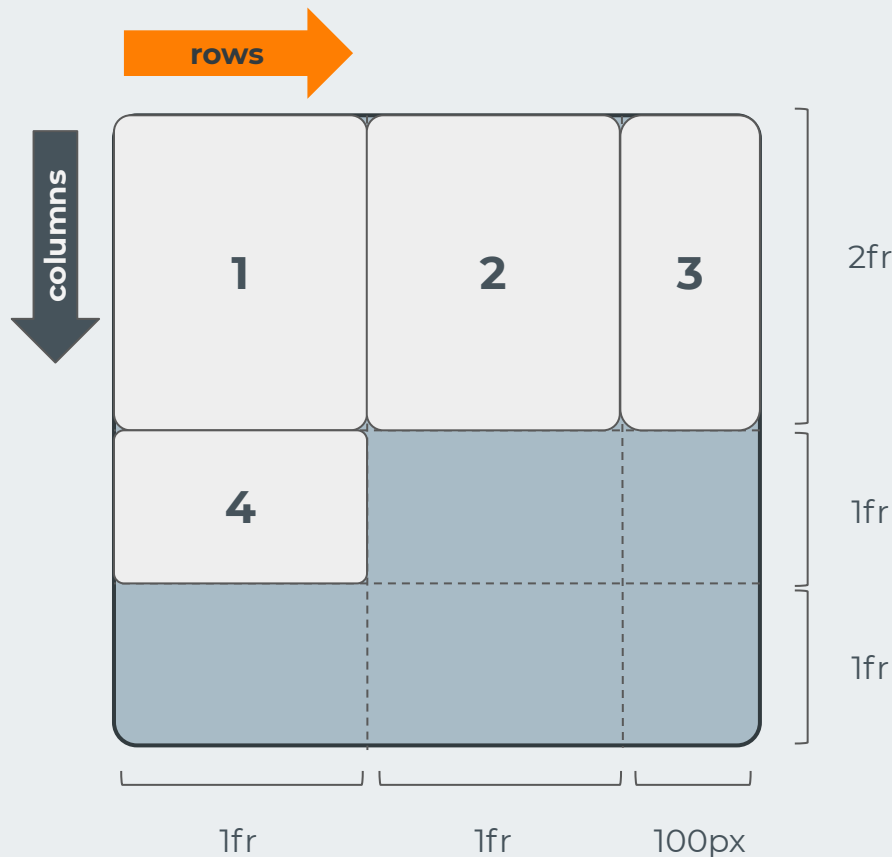
Supondo um container de **500px**:

- **Linhas**

- $500/4 = 125\text{px}$ (cada fr)
- Linha 1: 250px
- Linhas 2 e 3: 125px

- **Colunas**

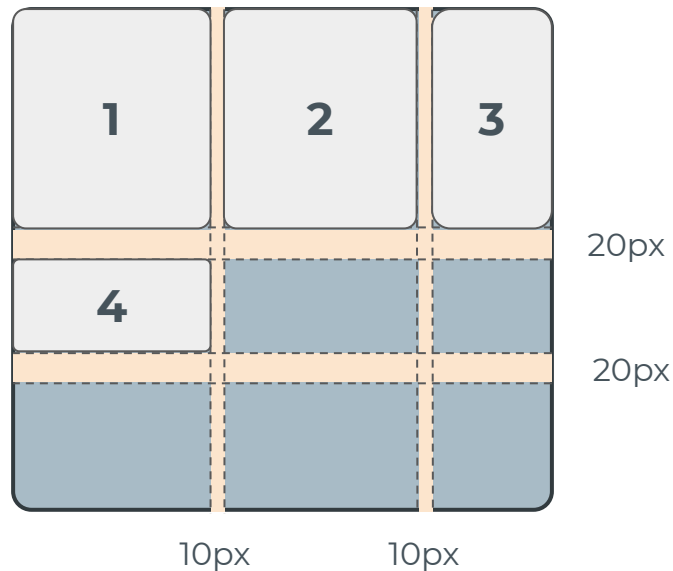
- $500 - 100 = 400\text{px}$
- $400/2 = 200\text{px}$ (cada fr)
- Colunas 1 e 2: 200px
- Coluna 3: 100px (fixo)




Grid - Propriedades do Container 🏁

- **row-gap** e **column-gap**: determina o espaço **entre** as linhas e colunas, recebe um valor numérico

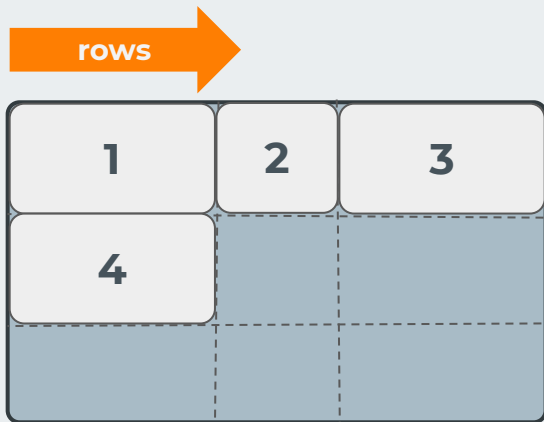
```
1 display: grid;  
2 grid-template-rows: 2fr 2fr 1fr;  
3 grid-template-columns: 1fr 1fr 100px;  
4 row-gap: 20px;  
5 column-gap: 10px;
```



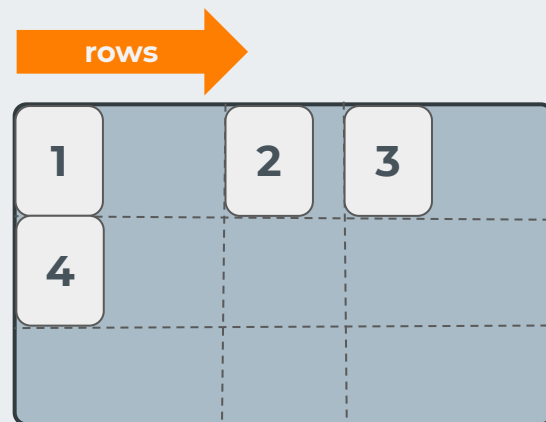
Grid - Propriedades do Container

- **justify-items**: determina a posição dos elementos no **eixo das linhas** dentro de cada uma das células
- Valores Possíveis:
 - **stretch (padrão)** 
 - start
 - end
 - center

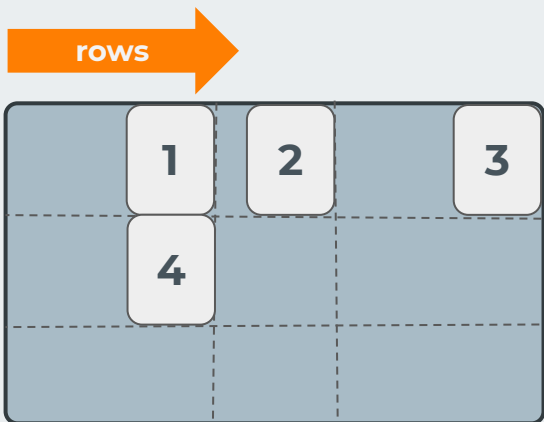




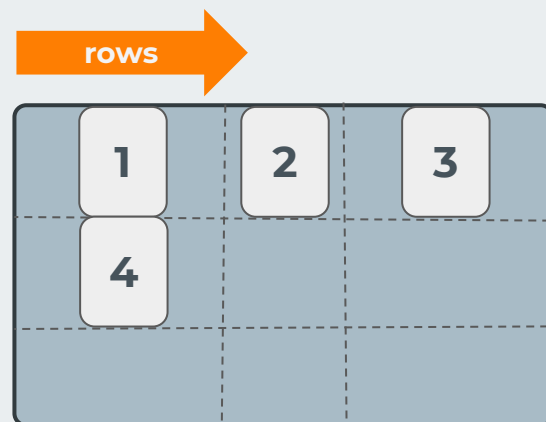
`justify-items: stretch`



`justify-items: start`




`justify-items: end`



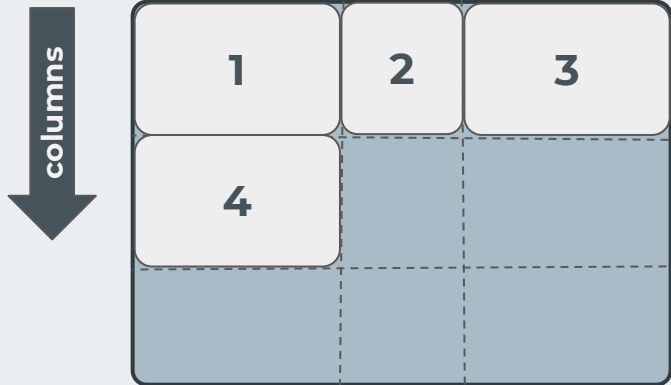
`justify-items: center`



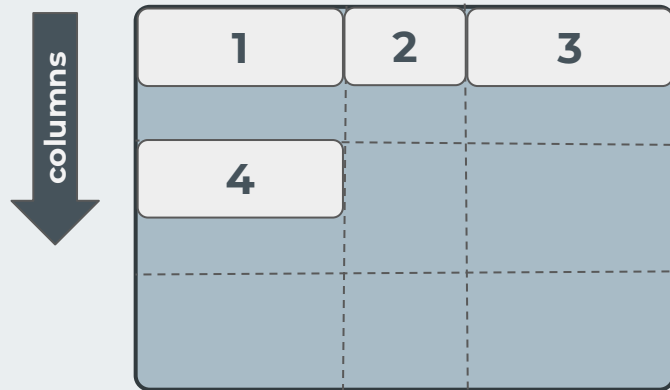
Grid - Propriedades do Container

- **align-items**: determina a posição dos elementos no **eixo das colunas** dentro de cada uma das células
- Valores Possíveis:
 - **stretch (padrão)** 
 - start
 - end
 - center

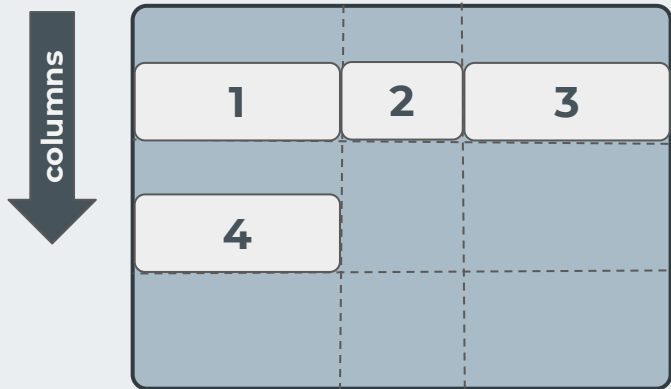




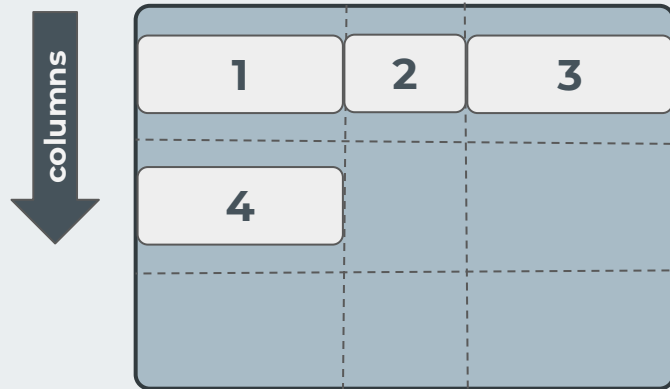
align-items: **stretch**



align-items: **start**



align-items: **end**



align-items: **center**




Propriedades dos Itens

Labenu_



Grid - Propriedades dos Itens

- Principais propriedades dos itens:
 - grid-row e grid-column 
 - justify-self
 - align-self
- Essas propriedades dos itens servem **posicionar** os itens dentro da malha ou para **alterar o alinhamento de um item específico**

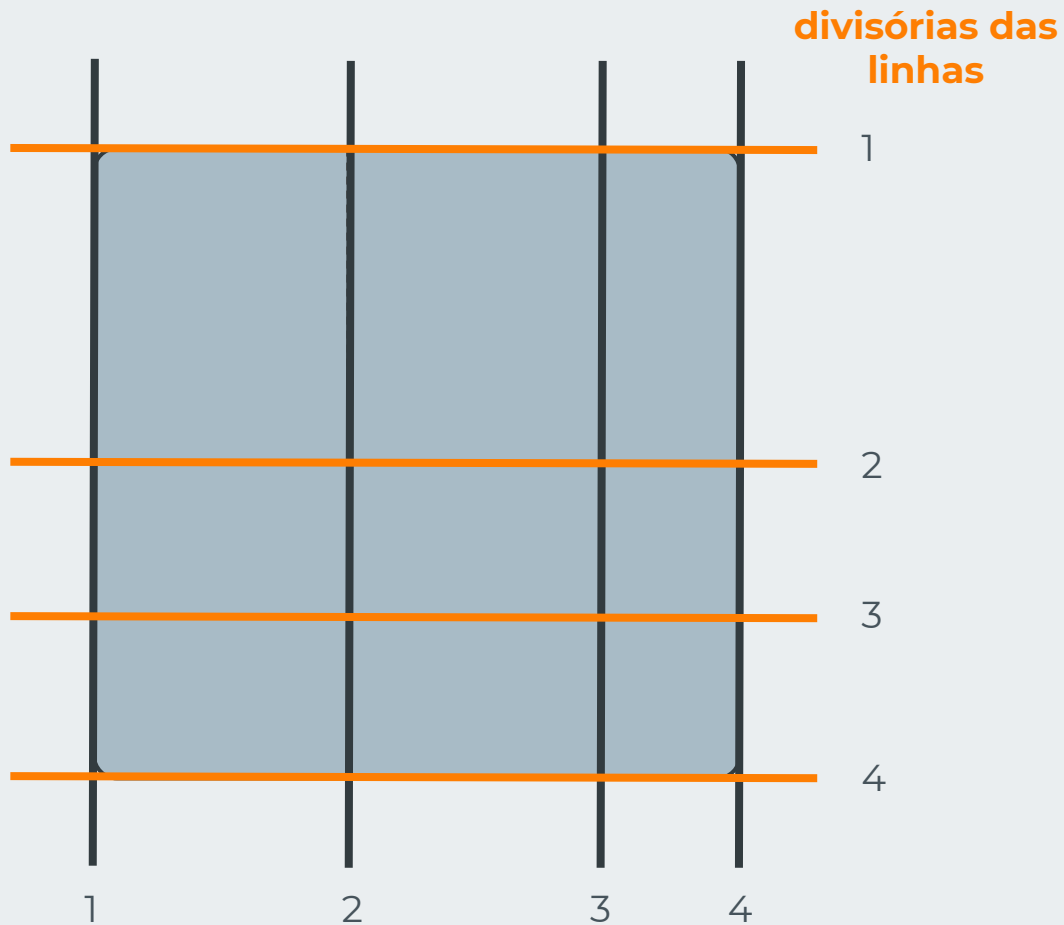


Grid - Propriedades dos Itens

- **grid-row e grid-column**: definem em quais divisórias do grid um determinado item começa e termina
 - Divisórias são as **divisões da malha**
 - **Linhas** são numeradas da esquerda para a direita
 - **Colunas** são numeradas de cima para baixo
 - Numeração começa por 1



**divisórias das
colunas**

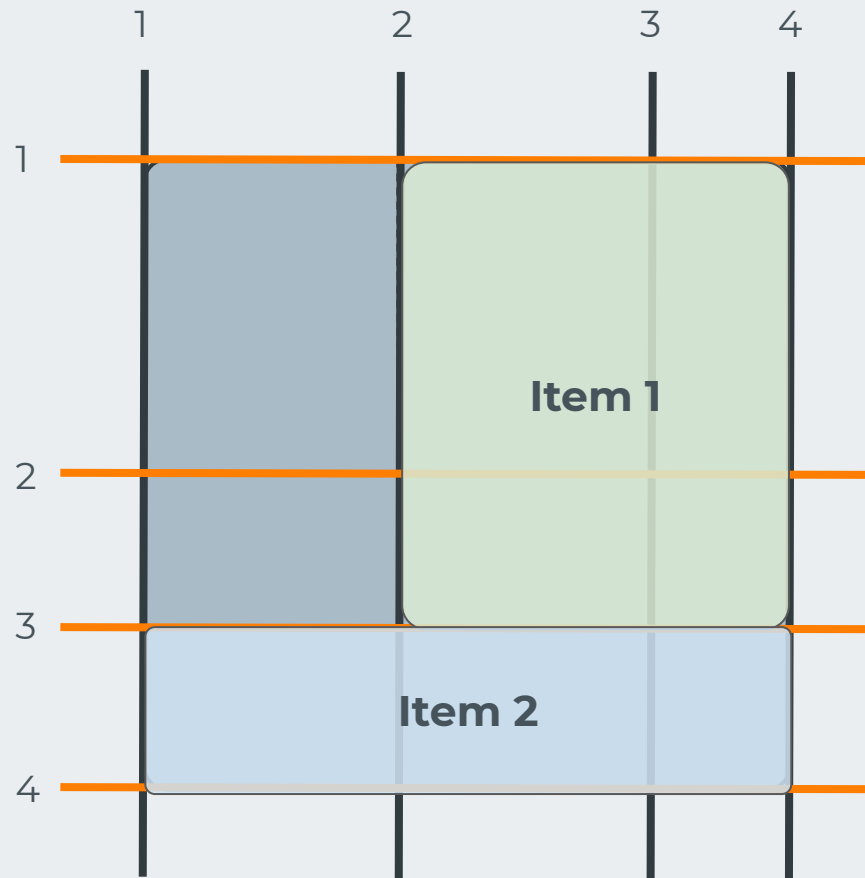


Grid - Propriedades dos Itens

- **grid-row e grid-column**: 2 possibilidades de sintaxe
 - Número da divisória de início e número da divisória de final separados por uma barra
 - `grid-column: 2 / 4`
 - Número da divisória de início e valor span com um número que determina quantas células serão ocupadas
 - `grid-column: 2 / span 3`

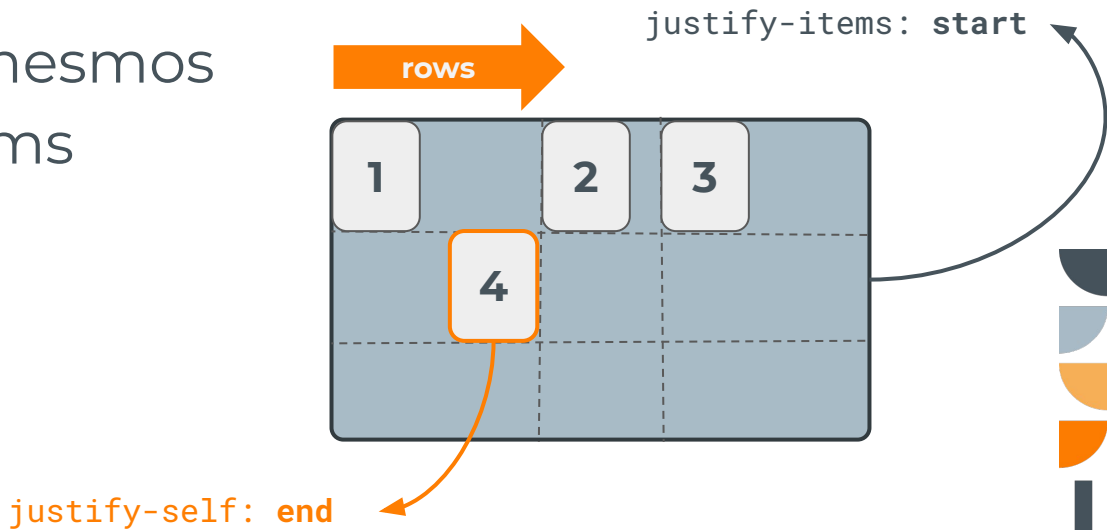


```
1 #item1 {  
2   grid-row: 1 / 3;  
3   grid-column: 2 / 4;  
4 }  
5 #item2 {  
6   grid-row: 3 / span 1;  
7   grid-column: 1 / span  
8   3;  
8 }
```



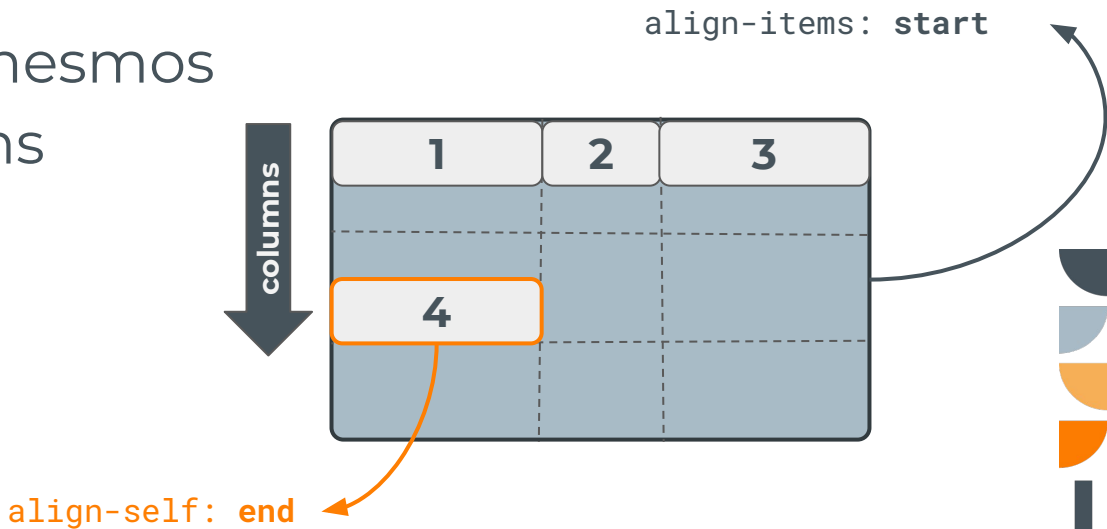
Grid - Propriedades dos Itens

- **justify-self**: determina a disposição do elemento em relação às **linhas**, sobrescrevendo o **justify-items**
- **Valores possíveis**: mesmos que os do justify-items
 - stretch
 - start
 - end
 - center



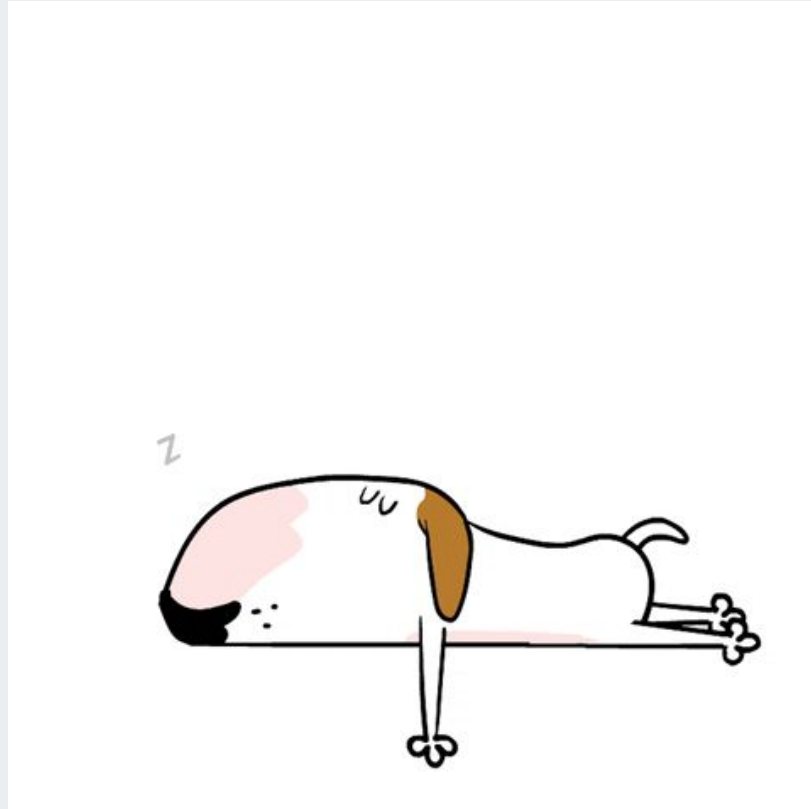
Grid - Propriedades dos Itens

- **align-self**: determina a disposição do elemento em relação às **colunas**, sobrescrevendo o **align-items**
- **Valores possíveis**: mesmos que os do align-items
 - stretch
 - start
 - end
 - center



Pausa para relaxar 🐾💤

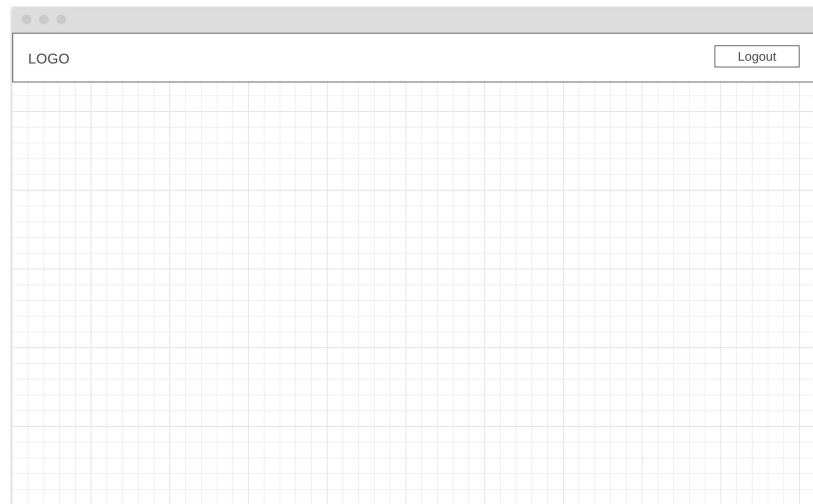
5 min





Exercício 4

- Implemente um Header seguindo o layout ao lado utilizando grid
- Garanta que o logo e o botão estão centralizados verticalmente





Exercício 5

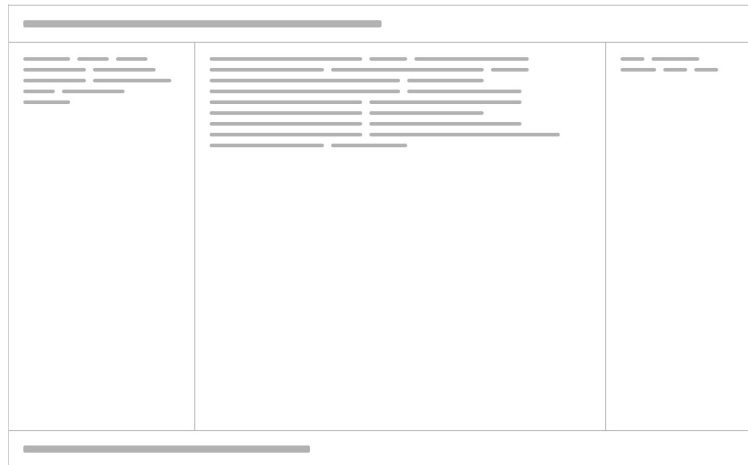
- Utilizando o Header do exercício anterior, implemente a página ao lado que possui:
 - Header
 - Conteúdo
 - Footer





Exercício 6

- Utilizando o site do exercício anterior, implemente na sua página:
 - Menus laterais na esquerda e na direita do conteúdo principal



Resumo

Labenu_



Resumo

- Aplicamos propriedades de **flex** em **containers** para distribuir seus **itens** no eixo horizontal (row) ou vertical (column)
- Aplicamos as propriedades de **grid** em **containers** para distribuir seus **itens** nos dois eixos (linhas e colunas)
- Propriedades podem ser aplicadas aos **itens** para fugir do padrão imposto pelo container
- Não se preocupe em decorar tudo, **teste e pesquise!**



Dúvidas? 🧐

Labenu_





Obrigado(a)!