



# Pensando fora da caixa

Módulo 1 - Aula 3 - HARD



Fonte: Pixels



Todos os direitos reservados  
©2022 Resilia Educação

# Layout

# Problema: Layout

---

Queremos inovar um pouco na **disposição dos elementos** na página: Em uma determinada parte da página gostaríamos de disponibilizar os conteúdos em uma matriz!

Como poderíamos replicar esse formato só com o que vimos até agora?

Conteúdo 1	Conteúdo 2
Conteúdo 3	Conteúdo 4





# Dívs

# HTML: Div

---

A tag **div** é utilizada para criar uma **divisão genérica** de conteúdo. Assim podemos colocar parte do **conteúdo** dentro de uma **"caixa"** e **formatá-lo** através dela.

Ex:

```
<div class=centralizado>  
    
</div>
```

A pyramid of 15 CRT monitors is arranged on a dark surface. The monitors are stacked in four rows: the top row has 1 monitor, the second row has 2, the third row has 3, and the bottom row has 4. Each monitor displays a different image, mostly abstract or architectural scenes. In the center of the pyramid, the word "Display" is written in a large, bold, yellow sans-serif font.

# Display



# Display

---

A regra **display** é muito importante na hora de definir **como desenhar** o elemento: os elementos em volta e os elementos de dentro dele.

Os principais tipos de display são:

- none
- inline
- inline-block
- block
- flex

# Display - none



# Display - inline

O display **inline** aplica todos os estilos no elemento, mas **ignora mudanças de tamanho**.

O inline foi feito para se adaptar ao tamanho da fonte do contexto atual.

```
.inline {  
  display: inline; /* the default for span */  
  width: 100px;  
  height: 100px;  
  padding: 5px;  
  border: 1px solid blue;  
  background-color: yellow;  
}
```

## display: inline

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. **Aliquam** **venenatis** gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.

Fonte: Material autoral

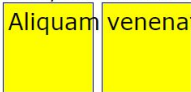
# Display - inline-block

O display **inline-block** aplica todos os estilos no elemento e **aceita mudanças de tamanho**. Com o inline-block conseguimos colocar elementos de tamanhos distintos em uma mesma linha, lado a lado.

```
.inline-block {  
  display: inline-block;  
  width: 100px;  
  height: 100px;  
  padding: 5px;  
  border: 1px solid blue;  
  background-color: yellow;  
}
```

## display: inline-block

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. Aliquam venenatis gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.



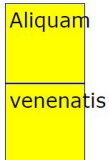
# Display - block

O display **block** “bloqueia” toda a linha, como se separasse um **bloco da página somente para si**. É algo equivalente a apertar “enter” antes de um texto e depois dele: ele fica **isolado na sua própria linha**.

```
.block {  
  display: block;  
  width: 100px;  
  height: 100px;  
  padding: 5px;  
  border: 1px solid blue;  
  background-color: yellow;  
}
```

## display: block

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat.



gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.

Fonte: Material autoral

# CSS: arquivos

---

Podemos **segregar** os arquivos **CSS** da página **HTML**. Dessa forma, no caso da página index, teremos **index.html** e **index.css**.

Para fins de **organização** devemos colocar os arquivos na **mesma pasta** e **importar** o **CSS** no **HTML** pelo **caminho**.

```
<head>
  <link rel="stylesheet" href="./index.css">
</head>
```



# Flexbox



# Flexbox: flex-container

---

O display **flex** transforma o elemento em um **flex-container**.

O container possui uma determinada orientação e se comporta como uma **lista**.

# Flexbox: flex-items

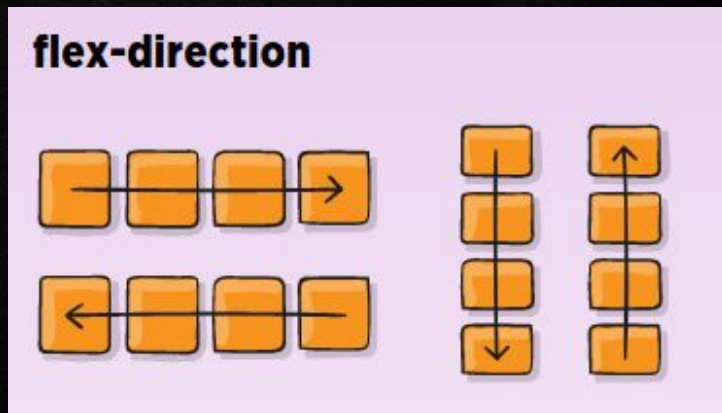
---

Qualquer elemento adicionado **dentro** de um **flex-container** é chamado de **flex-item**

# Flexbox: flex-direction

A **flex-direction** indica o **eixo principal** de um **flex-container**. É de acordo com ela que o navegador **organiza** os próximos **elementos** colocados **dentro** do **flex-container**.

- row
- row-reverse
- column
- column-reverse



Fonte: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

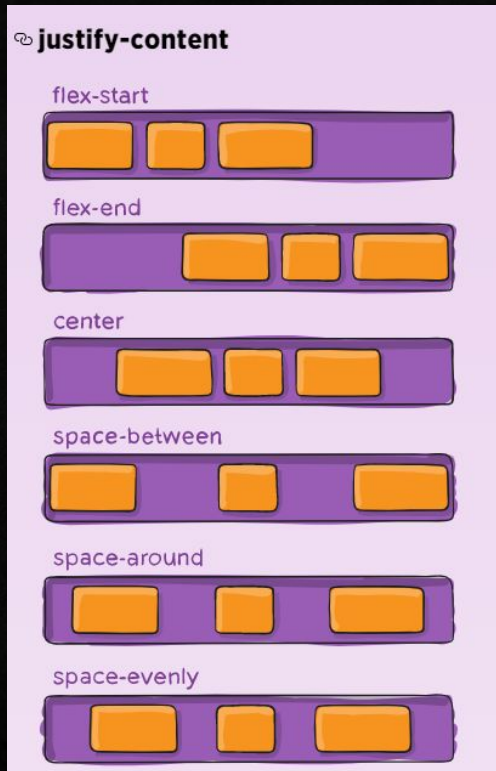


# Flexbox: justify-content

---

A propriedade **justify-content** de um **flex-container** define o comportamento dos **flex-items** no **eixo principal**.

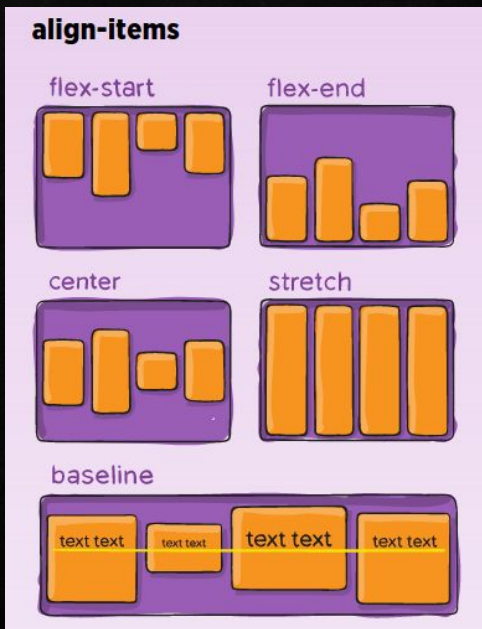
# Flexbox: justify-content



Fonte: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

# Flexbox: align-items

A propriedade **align-items** de um **flex-container** define o comportamento dos **flex-items** no **eixo secundário**.



Fonte: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

# Atividade: Matriz

---

A partir dos conteúdos vistos sobre **divs**, **flexbox** e a **propriedade display**:

- Crie uma nova pasta com os arquivos **index.html** e **index.css**
- Importe o arquivo **css** na página **html** para que seja possível sua utilização
- Implemente o código de tal forma que a página apresente a imagem abaixo quando aberta no navegador





# A cara da web

Ao longo dos anos o **tráfego** gerado por **dispositivos móveis** na web tem aumentado consideravelmente. Por conta disso, precisamos nos preocupar com a **experiência** em **diferentes dispositivos**.



Fonte: <https://www.authorityhacker.com/mobile-affiliate/>

A dark, narrow tunnel with walls covered in graffiti. In the center of the tunnel, a bright, glowing light trail forms a rectangular shape with a smaller, curved loop inside. The light trail has a red and white glow. The text "Como fazemos isso?" is overlaid in the center of the image.

**Como fazemos isso?**



# Media queries!

# Media queries

---

**Media queries** são mecanismos de especificar **diferentes** propriedades de **CSS** para **diferentes** tipos de **dispositivos**.

Elas declaram **regras** que são **aplicadas** apenas em **determinados cenários**.



# Media query: sintaxe

No **CSS**, utilizamos o **@media** para **declarar** uma media query.

Passamos como **parâmetro** para ela o que deve ser **verificado** quanto ao **navegador**.

Ex:

```
@media (max-width: 600px) {  
  h1 {  
    color: purple;  
  }  
}
```