FMX Academy

INFORMAÇÕES		
MÓDULO	AULA	INSTRUTOR
03 HTML e CSS para Desenvolvedores Web	04 - Layouts com CSS + Layouts Responsivos	Pedro Henrique Amadio

CONTEÚDO

1. Na aula anterior...

- Vimos o que é CSS e como aplicá-lo (inline, interno e externo).
- Exploramos seletores (por tag, classe, ID, atributos, descendentes).
- Entendemos a cascata, herança e especificidade.
- Aprendemos propriedades básicas: color, background, margin, padding, border, font.
- Estilizamos partes do portfólio e formulários.

- **Pontos de destaque**
  - **Cascata e ordem:**

    ```
    p { color: blue; }
    p { color: red; }
    ```

    Resultado: o texto do <p> ficará vermelho, pois a última regra vence.

  - **Herança:**

    ```
    body { font-family: Arial; color: green; }
    p { font-size: 18px; }
    ```

    Resultado: os parágrafos terão fonte Arial, cor verde (herdada) e tamanho 18px.

  - Especificidade (placar de pontos):
    - Inline (style="...") -> **1000 pontos**
    - ID (#id) -> **100 pontos**
    - Classe (.classe), atributo ([type="..."]), pseudoclasse (:hover) -> **10 pontos**
    - Elemento (p, div, h1), pseudoelemento (::before) -> **1 ponto**
    - Universal (*) ou combinadores (>, +, ~, espaço) -> **0 pontos**

    Exemplo:
    - CSS

      ```
      p { color: blue; }              /* (1) */
      .texto { color: green; }        /* (10) */
      #principal { color: red; }      /* (100) */
      ```

- HTML:
  Olá mundo
  Resultado: o texto ficará VERMELHO, pois #principal (100) vence sobre .texto (10) e p (1).

2. Box Model

Cada elemento em HTML é como se fosse uma “caixa”:

- Content: o conteúdo (pode se uma imagem, texto etc.)
- Padding: spaço entre o conteúdo e a borda
- Border: a borda em volta do elemento
- Margin: espaço externo, separando o elemento de outros.

- **Exemplo:**
  Imagine uma caixa com width: 200px;

```
.card {
  width: 200px;
  padding: 10px;
  border: 2px solid black;
  margin: 20px;
}
```

  - **Content:** 200px
  - **Padding:** +20px (10 de cada lado)
  - **Border:** +4px (2 de cada lado)
  - **Margin:** +40px (20 de cada lado, não soma na largura interna, mas afasta dos outros elementos)

  -> O espaço **real ocupado** pela caixa na tela será **200 + 20 + 4 = 224px de largura**, mais a margem (que é externa).

- **Box Sizing**
  Por padrão, o CSS usa content-box (só o conteúdo conta para o width).
  Mas geralmente usamos border-box, que facilita muito:

```
* {
  box-sizing: border-box;
}
```

  Com isso, width: 200px; já inclui **content + padding + border**, simplificando o cálculo.

3. Display

O **display** é uma das propriedades mais importantes do CSS, porque define **como um elemento será exibido na página**.
Ou seja, ele controla se o elemento ocupa uma linha inteira, se fica ao lado de outros, ou se participa de um sistema de layout mais avançado (como Flexbox e Grid).

- **Por padrão:**
  - A maioria das tags de bloco, como , , <h1>, vêm com display: block.

- Tags de linha, como , <a>, , vêm com display: inline.

- **Tipos principais de display**
  1. **block**
     - O elemento ocupa toda a largura disponível.
     - Sempre começa em uma nova linha.
     - Ex.: , <p>, <h1>.
  2. **inline**
     - O elemento ocupa apenas o espaço necessário ao seu conteúdo.
     - Não quebra a linha.
     - Ex.: , <a>.
  3. **inline-block**
     - Funciona como inline (fica na mesma linha), mas permite definir width, height, margin e padding.
  4. **flex**
     - Ativa o sistema de **Flexbox**, ideal para organizar elementos em linha ou coluna de forma responsiva.
  5. **grid**
     - Ativa o sistema de **CSS Grid**, que organiza os elementos em linhas e colunas.

- **Exemplos:**

```
<div style="display: block; background: lightblue;">Bloco 1</div>
<div style="display: block; background: lightgreen;">Bloco 2</div>
<span style="display: inline; background: lightpink;">Inline 1</span>
<span style="display: inline; background: lightyellow;">Inline 2</span>
```

O primeiro exemplo mostra duas caixas **uma abaixo da outra** (block).
O segundo mostra duas caixas **lado a lado** (inline).

4. Position

Controla como os elementos são posicionados no fluxo da página.

- static -> padrão.
- relative -> desloca em relação à posição normal.
- absolute -> posicionado em relação ao ancestral posicionado.
- fixed -> fixo na tela, mesmo rolando.
- sticky -> alterna entre relative e fixed conforme a rolagem.

- **Exemplo:**

```
.caixa {
  position: absolute;
  top: 50px;
  left: 100px;
}
```

5. Flexbox

Alinhamento e distribuição dos elementos em linha ou coluna.

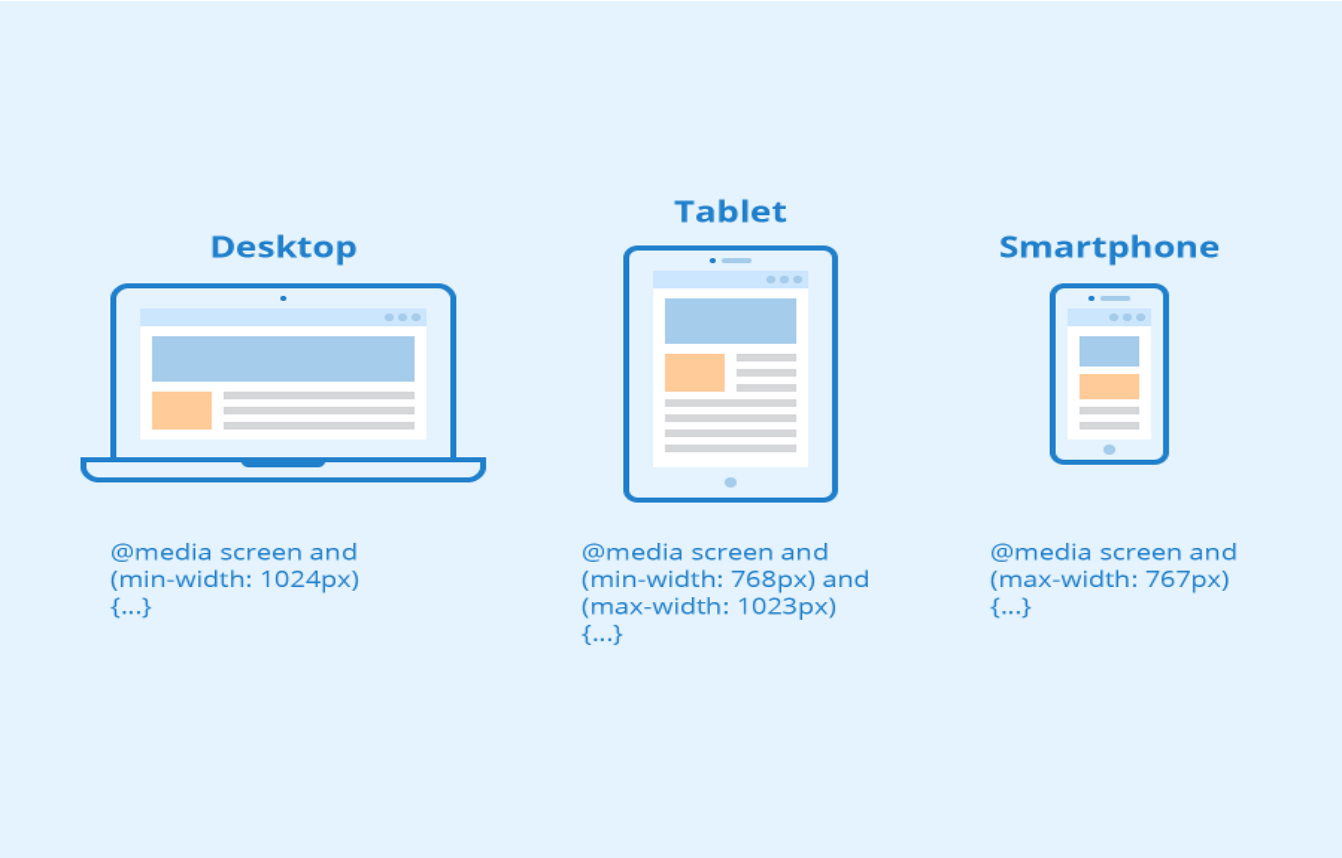
- **Propriedades principais:**
  - display: flex **-> ativa o container flexível.**
  - justify-content **-> alinhamento horizontal.**
  - align-items **-> alinhamento vertical.**
  - flex-direction **-> define direção (row ou column).**

- **Exemplo:**

```
.container {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

6. Media Queries (responsividade)

Permite aplicar estilos diferentes dependendo do tamanho da tela.



Media Queries – Author: Seobility – License: CC BY-SA 4.0

- **Conceito “mobile-first”**
  - Estilos padrão para telas pequenas, e media queries para telas maiores.
  - Desenvolvimento com foco principal em mobile, por isso “mobile-first”

- **Exemplo:**

```
/* Mobile First */

/* Padrão (mobile) */
.container { flex-direction: column; }

/* Desktop */
@media (min-width: 768px) {
  .container { flex-direction: row; }
}

/* Orientação em Paisagem */
@media all and (orientation: landscape) {
  /* CSS code for output in landscape mode */
}
```

-> Em telas pequenas: coluna. Em telas largas: linha.

7. Grid

Organiza elementos em linhas e colunas.

- Exemplo css:

```
.grid {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  gap: 10px;
}
```

-> Cria 3 colunas iguais. 1fr significa 1 fração do espaço disponível.

- Exemplo prático:

```
<div class="grid">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>
```

-> Cada item ocupa uma coluna.
Forma abreviada: repeat(3, 1fr) → equivalente a 1fr 1fr 1fr.

- **Funções**
  - **repeat():**
    Evita repetição manual de valores:

    grid-template-columns: repeat(4, 1fr);
    -> Cria 4 colunas iguais.

    - Também pode combinar valores:

grid-template-columns: repeat(2, 100px 1fr);
-> Gera colunas 100px, 1fr, 100px, 1fr.

- auto-fill e auto-fit
  Usados com repeat() para responsividade.

  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  -> Cria quantas colunas couberem na largura da tela.

- auto-fill: mantém colunas vazias.
- auto-fit: colapsa colunas vazias, ajustando espaço.

- **minmax(min, max)**
  Define tamanho mínimo e máximo:

  grid-template-columns: repeat(3, minmax(200px, 1fr));
  -> Cada coluna tem no mínimo 200px e cresce até ocupar espaço disponível.

- **fr (fractional unit)**
  Divide espaço proporcionalmente:

  grid-template-columns: 2fr 1fr;
  -> A primeira coluna ocupa o dobro da largura da segunda.

- **fit-content(valor)**
  Cresce até certo limite.

  grid-template-columns: fit-content(300px) 1fr;
  -> A primeira coluna cresce até 300px, depois para.

8. Prática Guiada

- Continuação da estilização do portfólio criado nas Aulas 01, 02 e 03.
- Refinar o visual usando Box Model, Display, Flexbox/Grid e Media Queries, sem criar páginas novas.

- Ações:
  1) Transformar seções (Sobre, Hobbies, Contato/Formulário) em 'cards'.
  2) Ajustar tipografia e espaçamentos (margin/padding) de forma consistente.
  3) Tornar os cards responsivos (coluna no mobile, 2–3 por linha em telas maiores).

- **Estrutura sugerida (HTML existente):**
  - **HTML (modelo):**

```
<main id="principal">
  <section class="card">
    <h2>Sobre</h2>
    <p>Bio resumida...</p>
  </section>
  <section class="card">
    <h2>Hobbies</h2>
    <ul>...</ul>
  </section>
  <section class="card">
    <h2>Contato</h2>
    <form>...</form>
  </section>
</main>
```

  - **CSS (exemplo com Flexbox):**

```
:root {
  --borda: #e5e7eb; --texto: #0f172a; --bg-card: #ffffff;
}
body { font-family: system-ui, -apple-system, Segoe UI, Roboto, Arial,
sans-serif; color: var(--texto); }
#principal {
  max-width: 1024px; margin: 24px auto; padding: 0 16px;
  display: flex; flex-wrap: wrap; gap: 16px;
}
.card {
  flex: 1 1 280px; /* base 280px e cresce */
  background: var(--bg-card); border: 1px solid var(--borda);
  border-radius: 12px; padding: 16px;
  box-shadow: 0 1px 3px rgba(0,0,0,.06);
}
.card h2 { margin-bottom: 8px; }

/* Responsividade simples: 1 coluna no mobile */
@media (max-width: 640px) {
  #principal { display: grid; grid-template-columns: 1fr; }
}
```

```
/* 2-3 colunas em telas maiores usando Grid */
@media (min-width: 768px) {
  #principal { display: grid; grid-template-columns: repeat(2,
minmax(280px, 1fr)); gap: 20px; }
}
@media (min-width: 1024px) {
  #principal { grid-template-columns: repeat(3, minmax(280px, 1fr)); }
}
```

9. Exercícios

1) Verdadeiro ou Falso (V/F)

( ) O Box Model é composto por margin, border, padding e content.
( ) position: fixed mantém o elemento no mesmo lugar mesmo com rolagem.
( ) flexbox é usado apenas para tipografia.
( ) @media (min-width: 768px) aplica estilos em telas maiores que 768px.
( ) As media queries permitem aplicar estilos diferentes conforme a largura da tela.
( ) A propriedade display: flex transforma o elemento em um container flexível.

2) Qual das opções abaixo cria 2 colunas iguais em Grid?

a) grid-template-rows: 1fr 1fr;
b) grid-template-columns: 1fr 1fr;
c) grid-template-columns: 100px;
d) flex-direction: row

10. Para Casa (Portfólio - Continuidade)

- Criar a seção 'Projetos' no portfólio.
- Organizar os projetos em cards com Flexbox.
- Tornar a seção responsiva com media queries (coluna no mobile, linha no desktop).