Revisão – Desenvolvimento Web I

Turma 201 • Teoria, Sintaxe e Fundamentos

Sumário

1. HTML – Estrutura e Organização

Teoria

HTML (*HyperText Markup Language*) define a **estrutura** de uma página web. O conteúdo é organizado em **elementos** (tags) que descrevem títulos, parágrafos, imagens, links, listas e seções semânticas.

> **Dica:** Prefira elementos semânticos como `<header>`, `<main>`, `<section>`, `<article>` e `<footer>` para melhorar acessibilidade e SEO.

Sintaxe

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <title>Minha Página</title>
  </head>
  <body>
    <h1>Título principal</h1>
    <p>Um parágrafo de exemplo.</p>
    <img src="foto.jpg" alt="Descrição da imagem" />
    <a href="contato.html">Ir para contato</a>
  </body>
</html>
```

Fundamentos

- **Atributos** adicionam informações às tags (ex.: `src`, `alt`, `href`, `id`, `class`).
- **Inline x Bloco:** inline (ex.: `<a>`, `<span>`) não quebra linha; bloco (ex.: `<p>`, `<div>`) ocupa a largura inteira.
- **Semântica:** escolha a tag que melhor representa o papel do conteúdo.

2. CSS – Estilos e Apresentação

Teoria

CSS (*Cascading Style Sheets*) controla a **aparência** do HTML: cores, tipografia, espaçamentos, bordas e posicionamento. A "cascata" define qual regra prevalece (especificidade, ordem e herança).

Sintaxe

```
seletor {
  propriedade: valor;
}

/* Exemplo */
p {
  color: #1f2937;       /* cor do texto */
  font-size: 16px;      /* tamanho da fonte */
  line-height: 1.6;     /* altura da linha */
}
```

Fundamentos

- **Seletores:** por tag (`p`), classe (`.card`), id (`#principal`), descendente (`nav a`).
- **Cores:** nome, hexadecimal (`#0ea5e9`), `rgb()`, `hsl()`.
- **Fontes:** use pilhas seguras (ex.: `font-family: system-ui, Arial, sans-serif;`).
- **Separação de responsabilidades:** HTML (estrutura) e CSS (estilo) em arquivos distintos.

3. Box Model – Modelo de Caixa

Teoria

Todo elemento é uma caixa composta por **content**, **padding**, **border** e **margin**. Entender o box model é essencial para controlar espaçamentos e alinhamentos.

Sintaxe

```
.caixa {
  margin: 16px;            /* espaço externo */
  border: 2px solid #0f172a;
  padding: 12px;           /* espaço interno */
  background: #f8fafc;
}
```

Fundamentos

- **margin** afasta elementos; **padding** afasta o conteúdo da borda.
- Use `box-sizing: border-box;` para facilitar cálculos de largura/altura.

```
*,
*::before,
*::after {
  box-sizing: border-box;
}
```

4. Flexbox – Layout Unidimensional

Teoria

O Flexbox organiza itens em **uma dimensão** (linha ou coluna), simplificando centralização, espaçamentos e distribuição. É ideal para navegações, barras e grupos de botões.

Sintaxe

```
.nav {
  display: flex;
  flex-direction: row;         /* row | column */
  justify-content: space-between;/* eixo principal */
  align-items: center;          /* eixo cruzado */
  gap: 12px;
}
```

Fundamentos

- `flex-direction` define a direção (linha/coluna).
- `justify-content` alinha no eixo principal; `align-items` no eixo cruzado.
- Itens podem crescer/encolher com `flex: grow shrink basis`.

```
.card { flex: 1 1 240px; }
```

5. CSS Grid – Layout Bidimensional

Teoria

O Grid organiza conteúdo em **linhas e colunas**. É excelente para layouts de página, galerias e áreas com múltiplas regiões.

Sintaxe

```
.container {
  display: grid;
  grid-template-columns: 1fr 2fr;
  gap: 16px;
}

.galeria {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(160px, 1fr));
  gap: 12px;
}
```

Fundamentos

- `fr` representa frações do espaço disponível.
- `repeat(auto-fit, minmax())` ajuda a tornar a grade fluida/responsiva.
- Combine Grid (estrutura geral) e Flex (componentes internos) quando fizer sentido.

6. Responsividade e Media Queries

Teoria

Responsividade adapta o layout a diferentes telas. Em **mobile first**, você estiliza primeiro para telas pequenas e expande com media queries para telas maiores.

Sintaxe

```
/* Base (mobile) */
.card { font-size: 1rem; }

/* Telas a partir de 768px */
@media (min-width: 768px) {
  .card { font-size: 1.0625rem; }
}

/* Telas a partir de 1024px */
@media (min-width: 1024px) {
  .card { font-size: 1.125rem; }
}
```

Fundamentos

- Use unidades relativas (`rem`, `%`) para tipografia e larguras.
- Garanta que imagens/iframes não "estourem" a largura: `img { max-width: 100%; height: auto; }`.

```
img, video, iframe {
  max-width: 100%;
  height: auto;
  display: block;
}
```

7. Boas Práticas

Teoria

Código limpo facilita evolução, correção e colaboração. Comentários pontuais, nomes claros e consistência de estilo são essenciais.

Fundamentos

- Organize arquivos: `index.html`, `style.css`, `assets/`.
- Padronize indentação (2 ou 4 espaços) e ordem de propriedades.
- Valide HTML/CSS quando possível; teste em telas e navegadores diferentes.

```
<!-- Comentário HTML curto e útil -->
/* Comentário CSS breve explicando a intenção do bloco */
```