INSTITUTO FEDERAL
Espírito Santo
Campus de Alegre

Apostila — Material de apoio

Links, Âncoras e Versionamento com Git e GitHub

Da Teoria Fundamental à Prática de Mercado

Desenvolvimento Front-End I

Período: TADS EaD - Semana 6

Autor: Prof. Cleziel Franzoni da Costa

Instituto Federal do Espírito Santo — Campus de Alegre

2025

Sumário

Capítulo 1

Links, Âncoras e Versionamento com Git e GitHub

Introdução: A Importância Estratégica das Unidades

No universo do design e desenvolvimento web, a escolha de uma unidade de medida transcende a mera definição de tamanho. É uma decisão arquitetural que impacta diretamente a escalabilidade, a manutenibilidade, a responsividade e, crucialmente, a acessibilidade de uma interface. Compreender a distinção fundamental entre unidades **absolutas** e **relativas** é o primeiro passo para evoluir de layouts fixos e frágeis para sistemas de design fluidos e resilientes.

Este capítulo irá dissecar as principais unidades de medida disponíveis no CSS, não como uma lista de opções, mas como um conjunto de ferramentas estratégicas, cada qual com um propósito, vantagens e desvantagens bem definidos.

1.1 Unidades Absolutas: A Medida da Invariância

Unidades absolutas são aquelas cujo tamanho é fixo e não depende de nenhum outro elemento na página, como o tamanho da fonte raiz ou as dimensões da viewport. Elas representam uma medida física.

1.1.1 O Pixel (`px`): A Unidade Fundamental (e Mal Compreendida)

O pixel é a unidade absoluta mais comum. No entanto, é vital entender que um "CSS Pixel"não corresponde a um pixel de hardware.

- **Definição Técnica:** Um pixel em CSS é uma unidade de "ângulo visual", definida

como `1/96` de uma polegada. Em telas de alta densidade (Retina), 1 CSS pixel pode corresponder a 4 ou mais pixels de hardware. O navegador realiza a abstração para manter a consistência visual.

Quando utilizar `px`?

O uso de pixels é ideal para propriedades que não devem escalar em proporção ao conteúdo ou à tela.

- **Bordas:** `border: 1px solid black;`
- **Sombras:** `box-shadow: 0 4px 8px rgba(0,0,0,0.1);`
- **Elementos de UI finos:** Separadores ou ícones com tamanho canônico.

Onde evitar `px`?

Evite usar pixels para tipografia (`font-size`), espaçamentos principais (`margin`, `padding`) e larguras de containers de layout.

1.2 Unidades Relativas: A Medida do Contexto

Unidades relativas são a espinha dorsal do design responsivo e acessível. Seu valor é calculado com base em outra medida.

1.2.1 Unidades Relativas à Tipografia: `em` e `rem`

A Unidade `em`

Definição: `1em` é igual ao `font-size` computado do **elemento pai**.

O Problema do Aninhamento (Compounding): O valor do `em` pode se tornar imprevisível em elementos aninhados, pois seu valor é recalculado a cada nível.

```
<div style="font-size: 20px;">
  <div style="font-size: 0.8em;">
    <p style="font-size: 0.8em;">
      </p>
  </div>
</div>
```

Listing 1.1: Exemplo do efeito cascata da unidade 'em'

A Unidade `rem` (Root EM): A Escolha Profissional

Definição: `1rem` é igual ao `font-size` do **elemento raiz** (a tag `<html>`).

A Vantagem da Acessibilidade: O `rem` resolve o problema do aninhamento e respeita as configurações de tamanho de fonte do usuário no navegador, permitindo que toda a interface escale de forma proporcional.

1.2.2 Unidades Relativas à Viewport: `vw`, `vh`

Estas unidades são relativas às dimensões da área visível do navegador.

- `vw` (Viewport Width): `1vw` é igual a 1% da largura da viewport.
- `vh` (Viewport Height): `1vh` é igual a 1% da altura da viewport.

São ideais para elementos que devem ocupar toda a tela, como seções "Hero"(`height: 100vh;`).

1.2.3 Unidades Relativas ao Container Pai: `%`

A porcentagem é calculada com base em uma propriedade do elemento pai. Por exemplo, `width: 50%` define a largura de um elemento como 50% da largura de seu pai.

1.3 Guia de Decisão Rápido: Escolhendo a Unidade Certa

Para `font-size`, `margin`, `padding` e dimensões gerais:
Primeira escolha: `rem`. Garante escalabilidade e acessibilidade.

Para espaçamentos internos de um componente que devem ser proporcionais ao seu `font-`

Use: `em`.

Para definir a largura de colunas dentro de um layout:
Use: `%` ou, preferencialmente, unidades de Flexbox (`flex-basis`) e Grid (`fr`).

Para elementos que devem ocupar a tela inteira ou uma fração dela:
Use: `vh` e `vw`.

Para valores que NUNCA devem mudar, como bordas finas ou sombras:
Use: `px`.

Conclusão: Da Rigidez à Fluidez

A jornada de um desenvolvedor front-end pode ser medida por sua transição gradual do uso excessivo de `px` para a adoção consciente e estratégica de unidades relativas como `rem`. Esta mudança não é apenas uma preferência estilística; é uma evolução na forma de pensar sobre a web.

Interfaces modernas não são documentos fixos, mas sistemas de design fluidos que devem se adaptar a uma infinidade de dispositivos e, mais importante, às necessidades individuais de cada usuário. Dominar o sistema de medidas do CSS é dominar a linguagem fundamental da flexibilidade e da inclusão na web.

Introdução: A Transição de Codificador para Engenheiro de Software

Nesta sétima semana, iniciamos uma transição fundamental no seu desenvolvimento profissional: a passagem da simples codificação para a **engenharia de software**. A diferença reside no método, na estratégia e na profundidade. Um engenheiro não apenas implementa funcionalidades; ele projeta soluções robustas, manuteníveis e centradas no usuário.

Os três módulos a seguir — **Arquitetura CSS Responsiva**, **Controle de Versão Profissional** e **Navegação com Foco em Acessibilidade** — foram estruturados para cultivar essa mentalidade. Cada seção é um mergulho técnico que combina teoria, implementação prática comentada e a justificativa por trás de cada decisão. Este não é um manual de "como fazer", mas um tratado de "por que e como fazer da maneira correta".

1.4 Módulo 1: Masterclass em Arquitetura CSS Responsiva com Mobile First

A metodologia Mobile First é o padrão da indústria para a criação de interfaces flexíveis. Sua premissa é o **Aprimoramento Progressivo (Progressive Enhancement)**: projetamos e construímos a experiência base para o ambiente mais restrito (dispositivos móveis) e, em seguida, adicionamos camadas de complexidade e recursos à medida que o espaço de tela e a capacidade do dispositivo permitem. Esta abordagem resulta em código mais limpo, performático e fácil de manter.

1.4.1 Estudo de Caso: A Construção de um Componente de Produto

Vamos analisar a construção de um card de produto, um componente onipresente no e-commerce. O objetivo é que ele seja legível e funcional em uma tela pequena (vertical) e se transforme em um layout mais rico e informativo em telas maiores (horizontal).

1.4.1.1 Estrutura HTML (O Esqueleto Semântico)

A estrutura utiliza a tag `<article>` por ser um conteúdo autossuficiente e distribuível. A nomenclatura das classes segue um padrão similar ao BEM (Block__Element–Modifier), que ajuda na legibilidade e escopo do CSS.

```
<article class="product-card">
  <img class="product-card__image" src="produto.jpg" alt="Descricao acessivel e
      detalhada do produto">

  <div class="product-card__content">
    <h3 class="product-card__title">Nome do Produto Incrivel</h3>
    <p class="product-card__description">Uma breve descricao que destaca os
       principais beneficios e caracteristicas do produto para o usuario.</p>
    <span class="product-card__price">R$ 99,90</span>
    <button class="product-card__button">Comprar</button>
  </div>
</article>
```

Listing 1.2: Estrutura HTML do Componente

1.4.1.2 Arquitetura CSS (A Implementação Mobile First Comentada)

O CSS é estruturado em duas seções claras: os estilos base (Mobile) e os aprimoramentos para telas maiores (Desktop).

```
/* ========================================================================
   1. FUNDACAO (MOBILE-FIRST) - Estilos que se aplicam a TODOS os dispositivos
   ========================================================================
      */

/* A pseudo-classe :root eh usada para declarar variaveis CSS globais.
   Isso centraliza valores que se repetem, facilitando a manutencao. */
:root {
  --primary-color: #007bff;
  --text-color: #333;
  --surface-color: #fff;
  --spacing-unit: 1rem; /* 1rem = 16px por padrao */
```

```
}

.product-card {
  /* Flexbox eh a base do layout. 'column' empilha os itens verticalmente. */
  display: flex;
  flex-direction: column;
  background-color: var(--surface-color);
  border-radius: 8px;
  box-shadow: 0 4px 8px rgba(0,0,0,0.1);
  overflow: hidden;
  max-width: 400px;
  margin: 0 auto;
}

/* ... (demais estilos base) ... */


/* ========================================================================
   2. APRIMORAMENTO PROGRESSIVO (PROGRESSIVE ENHANCEMENT) PARA DESKTOPS
   ======================================================================== */

/* A media query com 'min-width' eh o coracao do Mobile First.
   Este bloco SOH SERA aplicado por navegadores com viewport >= 768px. */
@media (min-width: 768px) {
  .product-card {
    /* A principal mudanca: o layout passa de coluna para linha. */
    flex-direction: row;
    max-width: 700px;
  }

  .product-card__image {
    /* A imagem agora tem uma largura fixa de 40% do card. */
    width: 40%;
  }

  .product-card__content {
    /* O conteudo ocupa os 60% restantes. */
    width: 60%;
    justify-content: center;
  }
}
```

Listing 1.3: Arquitetura CSS Mobile First do Componente

1.4.2 Análise Detalhada da Estratégia

- **Performance:** Navegadores em dispositivos móveis baixam e interpretam um CSS menor e mais simples. Os estilos dentro da `@media` são baixados, mas não aplicados, resultando em uma renderização inicial mais rápida.
- **Manutenibilidade:** A lógica do CSS flui de forma natural do simples para o complexo. É muito mais intuitivo adicionar novas regras para telas maiores do que sobrescrever regras complexas de desktop para simplificá-las no mobile.

1.5 Módulo 2: Fluxo de Trabalho de Versionamento com Git e VS Code

O controle de versão com Git é o pilar da colaboração e segurança no desenvolvimento de software. A interface gráfica (GUI) do VS Code abstrai os comandos do terminal, permitindo um fluxo de trabalho visual e intuitivo.

1.5.1 O Mapeamento: Ações na Interface vs. Comandos no Terminal

Inicializar Repositório

Ação no VS Code: Clicar em "Initialize Repository".
Comando Equivalente: `git init`

Adicionar à Staging Area

Ação no VS Code: Clicar no ícone `+` ao lado de um arquivo.
Comando Equivalente: `git add <arquivo>`

Realizar um Commit

Ação no VS Code: Escrever a mensagem e clicar no checkmark ().
Comando Equivalente: `git commit -m "mensagem"`

Publicar no GitHub

Ação no VS Code: Clicar em "Publish to GitHub".
Comandos Equivalentes: `git remote add origin <URL>` seguido de `git push -u origin main`

1.6 Módulo 3: Engenharia de UX e Acessibilidade (a11y) em Navegação

Acessibilidade não é um recurso extra, mas um aspecto central da qualidade de software. Uma navegação bem projetada deve ser funcional para todos.

1.6.1 Aprofundamento em JavaScript para Controle de Navegação

O método `scrollIntoView()` é uma API do navegador que oferece controle granular sobre a rolagem.

```
// Seleciona todos os links <a> cujo atributo 'href' COMECA COM '#'.
document.querySelectorAll('a[href^="#"]').forEach(anchor => {

  // Adiciona um "ouvinte" de eventos de clique a cada link de ancora.
  anchor.addEventListener('click', function (e) {

    // Previne o comportamento padrao do navegador (o "salto" imediato).
    e.preventDefault();

    const targetId = this.getAttribute('href');
    const targetElement = document.querySelector(targetId);

    // Verifica se o elemento de destino realmente existe na pagina.
    if (targetElement) {

      // A API 'scrollIntoView' eh chamada no elemento de destino.
      targetElement.scrollIntoView({
        behavior: 'smooth', // Rolagem suave
        block: 'start' // Alinha o topo do elemento ao topo da viewport
      });
    }
  });
});
```

Listing 1.4: Controle de rolagem suave com JavaScript

Conclusão: Síntese da Engenharia de Front-End

Ao concluir esta semana, você não apenas aprendeu três tópicos isolados. Você integrou três facetas de uma única disciplina: a **Engenharia de Front-End**.

1. **A Arquitetura (Mobile First)** demonstrou como planejar e estruturar o código CSS para ser resiliente, performático e adaptável.
2. **O Fluxo de Trabalho (Git/VS Code)** forneceu as ferramentas e processos para gerenciar a complexidade do código, garantir a segurança do seu trabalho e colaborar efetivamente.
3. **A Experiência do Usuário (Âncoras e a11y)** reforçou que o código que escrevemos tem um propósito final: servir a um ser humano.

Estes três pilares formam um ciclo virtuoso: uma boa arquitetura facilita o gerenciamento no Git. Um fluxo de trabalho Git limpo permite implementar funcionalidades de UX complexas com segurança. E uma UX bem definida guia as decisões de arquitetura. Dominar este ciclo é o que o diferenciará no mercado de trabalho.