Apostila 2

CSS – Folhas de Estilo em Cascata

Seção 1: Introdução ao CSS

1.1. O que é CSS?

CSS é a sigla para *Cascading Style Sheets*, ou Folhas de Estilo em Cascata. Conforme estabelecido na Apostila 1, o CSS é a linguagem usada para descrever a *apresentação* de um documento HTML. Ele controla o layout, as cores, as fontes e a aparência visual geral do seu site.

O termo "Cascata" (Cascading) refere-se ao sistema de regras que os navegadores usam para determinar qual estilo aplicar a um elemento quando múltiplas regras CSS entram em conflito. A regra "mais específica" ou a última declarada geralmente vence.

Exercícios para testar o aprendizado

1. O que significa a sigla CSS e qual é o seu papel em relação ao HTML?
2. O que significa o termo "Cascata" (Cascading) no nome do CSS?

Seção 2: Sintaxe e Aplicação de CSS

2.1. A Regra Básica do CSS

A sintaxe de uma regra CSS é composta por três partes:

seletor { propriedade: valor; }

- **Seletor:** O elemento (ou elementos) HTML que você deseja estilizar (ex: h1, .minha-classe).
- **Propriedade:** O atributo de estilo que você deseja alterar (ex: color, font-size).
- **Valor:** A configuração que você deseja aplicar à propriedade (ex: blue, 16px).

Um bloco de declaração pode conter múltiplas regras, separadas por ponto e vírgula:

```
CSS
p {
  color: blue;
  font-size: 16px;
  line-height: 1.5;
}
```

2.2. Os Três Métodos de Adição de CSS

Existem três maneiras de aplicar CSS a um documento HTML:

1. **Inline (Em linha):** Usando o atributo style diretamente na tag HTML.
   ```
   HTML
   <p style="color: red;">Este parágrafo é vermelho.</p>
   ```

2. **Internal (Interno):** Usando a tag <style> dentro do <head> do documento HTML.
   ```
   HTML
   <head>
     <style>
       p {
         color: red;
       }
     </style>
   </head>
   ```

3. **External (Externo):** Criando um arquivo separado (ex: style.css) e vinculando-o no <head> do HTML usando a tag <link>.
   ```
   HTML
   <head>
     <link rel="stylesheet" href="style.css">
   </head>
   ```

Melhor Prática (Site "Bem Estruturado"):
Para um site "bem estruturado", o método Externo é a única prática profissional recomendada. Os métodos Inline e Interno misturam a apresentação (CSS) com a estrutura (HTML), violando o princípio da "Separação de Preocupações". O método externo permite que um único arquivo style.css estilize centenas de páginas, tornando a manutenção e a escalabilidade do site muito mais fáceis.

2.3. Seletores Fundamentais

Seletores são o que conectam seu CSS ao seu HTML. Os três tipos mais básicos são:

- **Seletor de Tag (Elemento):** Estiliza todas as instâncias de uma tag HTML.
  CSS
  ```
  /* Estiliza todos os parágrafos */
  p {
    color: gray;
  }
  ```

- **Seletor de Classe:** Estiliza todos os elementos que possuem um atributo class correspondente. As classes são reutilizáveis e são a forma mais comum de aplicar estilos. O seletor de classe começa com um ponto (.).
  CSS
  ```
  /* Estiliza <p class="destaque"> e <div class="destaque"> */
  ```

```
.destaque {
background-color: yellow;
}
```

- **Seletor de ID:** Estiliza um *único* elemento que possui um atributo id correspondente. Um ID deve ser único por página. O seletor de ID começa com uma cerquilha (#).
  CSS
  ```
  /* Estiliza <div id="menu-principal"> */
  #menu-principal {
    border-bottom: 1px solid black;
  }
  ```

Exercícios para testar o aprendizado

1. Descreva os três componentes de uma regra CSS (ex: h1 { color: blue; }).
2. Quais são os três métodos de adicionar CSS a um site? Qual deles é o mais recomendado para um site "bem estruturado" e por quê?
3. Qual é a diferença de uso entre um seletor de Classe (.) e um seletor de ID (#)?
4. **Prática:** Crie um arquivo style.css. Usando o método "Externo", vincule este arquivo ao seu index.html (usando a tag <link> no <head>). Escreva três regras no style.css: uma usando um seletor de tag (ex: body), uma usando um seletor de classe (crie uma classe no HTML) e uma usando um seletor de ID (crie um ID no HTML).

Seção 3: O Modelo de Caixa (CSS Box Model)

3.1. O Conceito Fundamental de Layout

O conceito mais importante no layout CSS é o **Box Model** (Modelo de Caixa). Todo elemento HTML na sua página é, para o CSS, uma caixa retangular. O Box Model define como essa caixa é composta e como ela interage com outras caixas.

3.2. Análise dos Componentes

Uma caixa é composta por quatro camadas, de dentro para fora:

1. **Content (Conteúdo):** A área onde o seu conteúdo (texto, imagens) é exibido. Suas dimensões são definidas pelas propriedades width e height.
2. **Padding (Preenchimento):** O espaço interno entre o conteúdo e a borda. O background-color do elemento é visível nesta área.
3. **Border (Borda):** A linha que envolve o conteúdo e o padding.
4. **Margin (Margem):** O espaço externo entre a borda desta caixa e as caixas de outros elementos. A margem é sempre transparente e é usada para "empurrar" outros elementos.

3.3. O Padrão Opcional vs. Alternativo (box-sizing)

Por padrão, quando você define width: 100px;, o CSS aplica essa largura *apenas* ao **Content**. Se você adicionar padding: 10px; e border: 2px;, a largura total visível da caixa se torna:

$100px \text{ (content)} + 20px \text{ (padding esq/dir)} + 4px \text{ (border esq/dir)} = 124px$

Isso é o modelo padrão (box-sizing: content-box) e é extremamente confuso para iniciantes, pois seus elementos se tornam maiores do que o width especificado.

A prática moderna universal é usar o modelo alternativo, border-box. Com box-sizing: border-box;, a propriedade width define a largura *total* da caixa, incluindo padding e border. O navegador ajusta o content internamente para que tudo caiba.

Melhor Prática: Um site "bem estruturado" moderno aplica esta regra a todos os elementos para um layout previsível:

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

Exercícios para testar o aprendizado

1. Quais são as quatro camadas do Box Model, de dentro para fora?
2. Qual é a diferença entre padding e margin?
3. O que a propriedade box-sizing: border-box; faz? Por que ela é considerada uma "melhor prática" moderna?
4. **Prática:** No seu style.css, adicione border, padding e margin a um dos seus parágrafos. Experimente mudar o valor de box-sizing entre content-box (padrão) e border-box e veja como isso afeta o tamanho total do elemento no navegador (use as "Ferramentas de Desenvolvedor" do navegador para inspecionar).

Seção 4: Estilização de Conteúdo

4.1. Propriedades Essenciais de Texto

Abaixo estão as propriedades básicas para estilizar o texto dentro de suas caixas:

- font-family: Define a fonte. É uma boa prática fornecer uma "pilha de fontes" (font stack), terminando com um nome genérico (ex: font-family: Arial, Verdana, sans-serif;).
- font-size: Define o tamanho do texto (ex: font-size: 16px;).
- font-weight: Define o peso (negrito) da fonte. Pode ser uma palavra-chave (normal, bold) ou um número (400 para normal, 700 para negrito).
- color: Define a cor do texto.

4.2. Propriedades Essenciais de Fundo

Estas propriedades estilizam o fundo de uma caixa (incluindo seu padding):

- background-color: Define uma cor de fundo sólida.
  CSS
  ```
  div {
    background-color: #f0f0f0; /* Um cinza claro */
  }
  ```

- background-image: Aplica uma imagem de fundo, usando a função url().
  CSS
  ```
  body {
    background-image: url('imagens/fundo.jpg');
  }
  ```

- background: Você pode combinar ambos. A cor será exibida por baixo da imagem.
  CSS
  ```
  div {
    background: #f0f0f0 url('icone.png');
  }
  ```

Exercícios para testar o aprendizado

1. Liste 4 propriedades CSS usadas para estilizar texto.
2. O que é uma "pilha de fontes" (font stack) e por que é uma boa prática usá-la na propriedade font-family?
3. Qual propriedade CSS você usaria para definir uma imagem de fundo em um elemento?
4. **Prática:** No seu style.css, estilize seu <h1> e seus <p>. Mude o font-family, font-size e color. Em seguida, mude o background-color do seu <header> e do seu <footer>.

Seção 5: Layout Moderno com CSS

Agora que sabemos como estilizar caixas, precisamos aprender a posicioná-las. As técnicas modernas para isso são o Flexbox e o Grid.

A heurística principal para decidir entre eles é a dimensionalidade:

- **Flexbox (1D):** Para layout em *uma dimensão* (uma linha **OU** uma coluna).
- **Grid (2D):** Para layout em *duas dimensões* (linhas **E** colunas, como uma grade).

5.1. Layout Unidimensional (1D) com Flexbox

Flexbox é um modelo de layout projetado para organizar, alinhar e distribuir espaço entre itens em um contêiner, mesmo quando seus tamanhos são desconhecidos ou dinâmicos. Ele opera em um único eixo (principal) de cada vez (linha ou coluna).

Para usá-lo, você aplica display: flex; ao elemento *pai* (o contêiner).

Propriedades Principais do Contêiner (Pai):

- display: flex;: Ativa o Flexbox.
- flex-direction:: Define o eixo principal.
  - row (padrão): Itens se organizam horizontalmente.
  - column: Itens se organizam verticalmente.
- justify-content:: Alinha os itens ao longo do *eixo principal*.
  - flex-start (padrão): Agrupa no início.
  - flex-end: Agrupa no final.

    - center: Agrupa no centro.
    - space-between: Espaça uniformemente, com o primeiro item no início e o último no final.
    - space-around: Espaça uniformemente, com "meio espaço" nas bordas.
- align-items:: Alinha os itens ao longo do *eixo transversal* (o eixo perpendicular).
    - stretch (padrão): Estica os itens para preencher o contêiner.
    - flex-start: Alinha no topo (se flex-direction: row).
    - flex-end: Alinha na base.
    - center: Alinha no centro.

Uso Ideal: Perfeito para componentes, como alinhar os links em um menu de navegação (<nav>) ou centralizar um botão.

CSS

```
.menu-nav {
  display: flex;
  justify-content: center; /* Centraliza os links horizontalmente */
  align-items: center;     /* Centraliza verticalmente (se tiverem alturas diferentes) */
  gap: 20px;               /* Adiciona 20px de espaço entre os links */
}
```

5.2. Layout Bidimensional (2D) com Grid

CSS Grid é um sistema de layout bidimensional, o que o torna a ferramenta mais poderosa e adequada para o layout *geral* da página (organização de header, main, sidebar, footer).

Para usá-lo, você aplica display: grid; ao elemento *pai* (o contêiner).

Propriedades Principais do Contêiner (Pai):

- display: grid;: Ativa o Grid.
- grid-template-columns:: Define o número e o tamanho das colunas. A unidade fr (fração) é muito útil aqui, pois distribui o espaço fracionário disponível.
    - Ex: grid-template-columns: 200px 1fr 1fr; (Cria 3 colunas: a primeira com 200px, a segunda e a terceira dividem o espaço restante igualmente).
    - Ex: grid-template-columns: repeat(3, 1fr); (Cria 3 colunas de larguras iguais).
- grid-template-rows:: Define a altura das linhas (funciona como o grid-template-columns).
- gap: (ou grid-gap): Define o espaçamento (calha) entre as linhas e colunas.

- Ex: gap: 20px; (Cria 20px de espaço entre todas as células do grid).

Uso Ideal: Perfeito para o layout principal da página.

CSS

```css
/* Layout de um blog com 2 colunas */
.container-principal {
  display: grid;
  grid-template-columns: 2fr 1fr; /* Coluna de conteúdo 2x maior que a sidebar */
  gap: 30px;
}
/* Os filhos diretos (.artigo-principal e.sidebar) se encaixarão automaticamente */
```

5.3. Análise Comparativa: Quando usar Flexbox vs. Grid

- **Use Grid** para o layout macro da página (a estrutura 2D).
- **Use Flexbox** para alinhamento de componentes micro (o conteúdo 1D *dentro* de uma célula do Grid).

É comum usar ambos: um display: grid; no <body> para criar as áreas principais, e um display: flex; no <nav> (que está *dentro* de uma célula do Grid) para organizar os links do menu.

Exercícios para testar o aprendizado

1. Qual é a diferença principal de uso entre Flexbox (1D) e Grid (2D)?
2. Para ativar o Flexbox ou o Grid, qual propriedade display você aplica ao contêiner *pai*?
3. No Flexbox, qual propriedade alinha itens no eixo principal (horizontal, por padrão)?
4. No Grid, qual propriedade define o número e o tamanho das colunas?
5. **Prática:** Use display: flex; no seu <nav> (criado na Seção 5 do HTML) para alinhar os links do menu horizontalmente. Use display: grid; no seu <main> para criar um layout de duas colunas (para o <article> e o <aside>).

Seção 6: Design Responsivo

Design Responsivo é a prática de criar um site que se adapta e tem uma boa aparência em *todos* os tamanhos de tela, de celulares a tablets e desktops. Isso é um requisito não-negociável no desenvolvimento moderno. Os pilares são o "Mobile-First", unidades flexíveis e Media Queries.

6.1. O Conceito Estratégico "Mobile-First"

"Mobile-First" é uma estratégia de desenvolvimento onde você projeta e escreve o CSS para as telas *menores* (celulares) *primeiro*. Por padrão, seu site será de coluna única e otimizado para toque.

Depois, você usa "Aprimoramento Progressivo" (*Progressive Enhancement*): usando Media Queries, você *adiciona* complexidade e layouts mais amplos (como colunas de grid) à medida que o espaço da tela aumenta (ex: para tablets e desktops).

Esta abordagem é superior à "Desktop-First" (começar pelo desktop e tentar "encolher" para o celular) porque resulta em CSS mais simples, focado no essencial, e garante melhor desempenho em dispositivos móveis, que hoje representam a maioria do tráfego web.

6.2. Unidades Flexíveis e Relativas

Para um site ser responsivo, você deve evitar unidades fixas como px (pixels) para layout e fontes. Em vez disso, use unidades relativas:

- % (Porcentagem): Relativa ao tamanho do elemento pai. Ótimo para larguras (ex: width: 100%;).
- rem (Root EM): Relativa ao tamanho da fonte do elemento raiz (<html>). Esta é a *melhor* unidade para font-size, padding e margin, pois ela se ajusta se o usuário aumentar ou diminuir o tamanho da fonte padrão do navegador por razões de acessibilidade.
- vw (Viewport Width) e vh (Viewport Height): Relativas ao tamanho da janela (viewport). 1vw é 1% da largura da janela; 100vh é 100% da altura da janela.

6.3. Media Queries

Media Queries são a ferramenta CSS que torna a responsividade possível. Elas permitem aplicar regras de CSS *condicionalmente*, com base nas características do dispositivo, como a largura da tela.

A sintaxe básica é: @media tipo_midia and (regra) {... }.

Exemplo (Abordagem Desktop-First / max-width):
Esta abordagem (menos recomendada) aplica estilos de desktop por padrão e, em seguida, aplica estilos de celular quando a tela for menor que 600px.40

CSS

```
/* Estilos de Desktop (Padrão) */
.container {
  width: 900px;
}


/* Aplica quando a tela for 600px OU MENOR */
@media screen and (max-width: 600px) {
.container {
    width: 100%; /* Muda para largura total no celular */
  }
}
```

Exemplo (Abordagem Mobile-First / min-width):
Esta é a abordagem preferida e "bem estruturada". Os estilos de celular são o padrão (fora de qualquer media query). Os estilos de desktop são adicionados quando a tela for maior que 600px.

CSS

```
/* Estilos de Celular (Padrão) */
.container {
  width: 100%; /* Largura total por padrão */
}

/* Aplica quando a tela for 600px OU MAIOR */
@media screen and (min-width: 600px) {
```

```
.container {
    width: 900px; /* Muda para largura fixa no desktop */
  }
}
```

Exercícios para testar o aprendizado

1. O que é a estratégia "Mobile-First"? Por que ela é preferível à "Desktop-First"?
2. Por que a unidade rem é geralmente preferível ao px para font-size e padding?
3. Qual é a sintaxe de uma Media Query que aplica estilos *apenas* quando a tela é *maior* que 800 pixels?
4. **Prática:** Usando a abordagem Mobile-First, faça seu layout ser de coluna única por padrão. Em seguida, adicione uma Media Query (ex: @media screen and (min-width: 700px)) que aplica o layout de Grid (criado na Seção 5) apenas em telas maiores.

Seção 7: Animações Leves em CSS

O CSS permite adicionar animações leves que melhoram a experiência do usuário sem a necessidade de JavaScript.

7.1. Transições Suaves (transition)

A propriedade transition é usada para animar suavemente a mudança de uma propriedade CSS quando seu estado muda (como em :hover, quando o mouse passa por cima).

A sintaxe abreviada é: transition: <propriedade> <duração> <timing-function> <delay>;.

Exemplo: Botão com Hover Suave 42
Sem transição, a cor de um botão mudaria instantaneamente no hover. Com transition, ela muda suavemente.

CSS

```css
.botao {
  background-color: blue;
  color: white;
  padding: 10px 20px;
  /* Aplica uma transição de 0.3s a TODAS as propriedades que mudarem */
  transition: all 0.3s ease;
}

.botao:hover {
  background-color: darkblue;
  color: yellow;
}
```

7.2. Transformações 2D (transform)

A propriedade transform permite modificar a geometria de um elemento: girá-lo, escalá-lo ou movê-lo, sem afetar o layout de outros elementos.

Funções comuns incluem:

- rotate(45deg): Gira o elemento (ex: 45 graus).
- scale(1.1): Aumenta o tamanho do elemento (ex: em 10%).
- translate(50px, 10px): Move o elemento (ex: 50px para a direita, 10px para baixo).

Transições e transformações são frequentemente usadas juntas para criar efeitos de hover elegantes.

Exemplo: Animação de Zoom e Rotação no Hover

CSS

```css
.imagem-link {
  transition: transform 0.4s ease;
}

.imagem-link:hover {
  /* No hover, aumenta a imagem em 5% e a gira levemente */
  transform: scale(1.05) rotate(3deg);
}
```

7.3. Animações por Keyframes (@keyframes)

Para animações mais complexas que não dependem de um estado (como :hover) e que podem ter múltiplos estágios, usamos @keyframes. O processo tem duas partes:

1. **Definir a Animação:** Use a regra @keyframes para criar uma sequência de animação e nomeá-la. Você define os "pontos-chave" (keyframes) usando from/to ou porcentagens (0%, 50%, 100%).
2. **Aplicar a Animação:** Use a propriedade animation no elemento que você deseja animar.

7.4. Exemplos de Animações

Aqui estão os exemplos de animações solicitados.

Exemplo 1: "Fade In" (Opacidade)
Esta animação faz um elemento aparecer suavemente (de invisível para visível).

CSS

```css
/* 1. Definição do @keyframes */
@keyframes fadeIn {
  from {
    opacity: 0; /* Começa 100% transparente */
  }
  to {
    opacity: 1; /* Termina 100% visível */
  }
}
```

```css
/* 2. Aplicação da animação */
.elemento-fade-in {
  animation: fadeIn 2s; /* (nome) (duração) */
}
```

Exemplo 2: Mudança de Cor (Pulsante)
Esta animação muda a cor de fundo continuamente através de múltiplos estágios.

CSS

```css
/* 1. Definição do @keyframes */
@keyframes mudaCor {
  0% {
    background-color: #FF6B6B; /* Vermelho */
  }
  50% {
    background-color: #4ECDC4; /* Verde-água */
  }
  100% {
    background-color: #556270; /* Cinza-azulado */
  }
}

/* 2. Aplicação da animação */
.elemento-muda-cor {
  animation: mudaCor 5s infinite alternate; /* (nome) (duração) (contagem) (direção) */
}
```

- 5s: A animação leva 5 segundos para ir de 0% a 100%.
- infinite: A animação se repetirá para sempre.
- alternate: A animação irá do início ao fim (0% a 100%) e depois do fim ao início (100% a 0%), criando um efeito de pulsação suave.

Exercícios para testar o aprendizado

1. Qual é a diferença entre transition e animation (@keyframes)? Quando você usaria cada um?
2. O que a propriedade transform permite fazer? Dê dois exemplos de funções transform.
3. O que os seletores from (ou 0%) e to (ou 100%) fazem dentro de uma regra @keyframes?
4. **Prática:** Adicione uma transition suave ao efeito :hover dos links do seu menu (criado na Seção 5). Crie uma animação @keyframes (como um "fade in" ou um "pulsar") e aplique-a ao seu <h1> usando a propriedade animation.

Seção 8: Conclusão – Montando seu Primeiro Site

Ao combinar os conceitos de ambas as apostilas, agora você está pronto para construir seu primeiro site simples, responsivo e bem estruturado.

O processo é:

1. Criar um index.html com estrutura semântica (Apostila 1, Seção 5).
2. Criar um style.css e linká-lo ao HTML (Apostila 2, Seção 2.2).
3. Estilizar o site usando a abordagem Mobile-First (Apostila 2, Seção 6), aplicando box-sizing, Flexbox, Grid e Transições.

Abaixo está um projeto final simples que une tudo.

Arquivo index.html (A Estrutura Semântica):

HTML

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Meu Primeiro Site</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <header class="site-header">
    <h1>Meu Site</h1>
  </header>

  <nav class="site-nav">
    <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#">Sobre</a></li>
      <li><a href="#">Contato</a></li>
    </ul>
  </nav>

  <main class="site-main">
    <article>
      <h2>Artigo Principal</h2>
      <p>Este é o conteúdo principal. Em telas de celular, ele ocupa 100% da largura. Em
```

```
desktops, ele fica ao lado da barra lateral.</p>
    </article>

    <aside>
      <h3>Barra Lateral</h3>
      <p>Conteúdo secundário aqui.</p>
    </aside>
  </main>

  <footer class="site-footer">
    <p>© 2024</p>
  </footer>

</body>
</html>
```

Arquivo style.css (O Estilo Mobile-First e Responsivo):

CSS

```
/* Configuração Global e Estilos Mobile-First (Padrão) */
* {
  box-sizing: border-box; /* Melhor prática do Box Model */
  margin: 0;
  padding: 0;
}

body {
  font-family: Arial, sans-serif;
  line-height: 1.6;
  font-size: 1rem; /* Usa 'rem' para acessibilidade */
}

.site-header,.site-footer {
  background-color: #333;
  color: white;
  text-align: center;
  padding: 1rem;
}

.site-nav ul {
  list-style: none;
```

```
  display: flex; /* Usa Flexbox para o menu */
  justify-content: center;
  padding: 0.5rem;
  background-color: #444;
}

.site-nav a {
  color: white;
  text-decoration: none;
  padding: 0.5rem 1rem;
  /* Animação Leve (Transição) */
  transition: background-color 0.3s ease;
}

.site-nav a:hover {
  background-color: #555;
}

.site-main {
  padding: 1rem;
  /* Por padrão (mobile), article e aside são 100% da largura */
}

.site-main article,.site-main aside {
  padding: 1rem;
  background-color: #f4f4f4;
  margin-bottom: 1rem;
}

/* --- Media Query para Responsividade --- */
/* Estilos aplicados QUANDO a tela for 700px OU MAIOR */

@media screen and (min-width: 700px) {
.site-main {
    display: grid; /* Usa Grid para o layout de desktop */
    grid-template-columns: 2fr 1fr; /* Coluna principal 2x maior que a sidebar */
    gap: 1.5rem;
    padding: 1.5rem;
  }
}
```

Ao salvar esses dois arquivos e abrir o index.html em um navegador, o resultado será um site simples, porém perfeitamente estruturado, que é responsivo e inclui uma animação leve.