📚 Material de Estudo: Introdução ao CSS em Desenvolvimento Web

🚀 O que vamos aprender?

O CSS (Cascading Style Sheets) é o que dá vida, cor e forma a qualquer página web. Nesta introdução, vamos aprender os fundamentos para estilizar nosso projeto de Gerenciamento de Tarefas, focando em **Seletores**, **Propriedades** e no crucial **Modelo de Caixa (Box Model)**.

I. Fundamentos e Configuração Inicial

1. Entendendo os Papéis: HTML e CSS

Linguagem	Função	Metáfora	Exemplo no Projeto
HTML	Define a **Estrutura** e o **Conteúdo**.	O esqueleto e os órgãos.	A tag `<form>` e o botão `<button>`.
CSS	Define a **Aparência** e o **Posicionamento**.	A pele, as cores e a roupa.	A cor de fundo (`background-color`) e as margens (`margin`).

Exportar para as Planilhas

2. Anatomia de uma Regra CSS

Toda regra CSS é uma instrução que diz ao navegador **quem** estilizar e **como** estilizar.

Componente	Exemplo	Descrição
Seletor	`h1` ou `.titulo`	Aponta para os elementos HTML a serem afetados.
Propriedade	`color`	O atributo que você deseja mudar (ex: cor, tamanho, fonte).
Valor	`blue` ou `16px`	O valor que você está definindo para a propriedade.

Exportar para as Planilhas

CSS

```
/* Anatomia da Regra */
h1 {
    color: #007bff; /* Propriedade: color; Valor: #007bff */
    font-size: 2em;
}
```

3. O Método Padrão de Inserção: Arquivo Externo

A melhor prática é manter seu CSS em um arquivo separado (`style.css`), que deve ser linkado ao seu HTML, mantendo o código organizado.

No seu `index.html` (dentro da tag `<head>`):

HTML

```
<link rel="stylesheet" href="style.css">
```

-

II. Seletores: O "Quem" do CSS

Os seletores são ferramentas para mirar elementos com precisão.

1. Seletor de Tipo (Tag)

Mira **todas** as ocorrências de uma tag específica.

CSS

```
/* Aplica-se a todo o documento */
body {
    font-family: Arial, sans-serif;
    background-color: #e9ecef;
}
```

2. Seletor de Classe (`.`) - O Mais Usado!

Mira elementos que possuem um atributo `class`. Permite que você **reutilize** o mesmo estilo em vários locais.

- **HTML:** `<form class="task-form">`
- **CSS:** Estilizando o bloco principal do formulário.

CSS

```
.task-form {
    background-color: #ffffff;
    padding: 25px;
    border-radius: 8px; /* Cantos arredondados */
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
}
```

3. Seletor de ID (`#`)

Mira o elemento que possui um atributo `id`. Deve ser usado **apenas uma vez** por página.

- **HTML:** `<ul id="task-list">`
- **CSS:**

```
CSS
#task-list {
```

```
    list-style: none; /* Remove os marcadores de lista */
    padding: 0;
}
```

III. O Modelo de Caixa (Box Model)

Este é o conceito mais fundamental para o layout: **Todo elemento HTML é uma caixa**. O Modelo de Caixa define como essa caixa se comporta em termos de tamanho e espaçamento.

1. Os Quatro Componentes (De dentro para fora)

Componente	Propriedade CSS	Função
Conteúdo	`width`, `height`	Ocupado pelo texto ou imagem.
Padding	`padding`	**Espaçamento interno** (entre o conteúdo e a borda).
Border	`border`	Linha que envolve o `padding`.
Margin	`margin`	**Espaçamento externo** (entre a caixa e os vizinhos).

Exportar para as Planilhas

2. Aplicando o Box Model e Centralização

A. Centralização de Blocos (`margin: 0 auto;`)

Para centralizar um elemento de **bloco** (`<div>`, `<form>`) na horizontal, ele precisa de uma largura definida e margens automáticas.

CSS

```
.container {
    max-width: 900px; /* Largura máxima definida */
    margin: 0 auto;   /* Margem superior/inferior 0, Margem lateral automática */
```

```
}
```

> **Lembrete:** `margin: 0 auto;` centraliza o **bloco**. Para centralizar **texto**, use `text-align: center;`.

B. `box-sizing: border-box;` (A Solução Moderna)

Por padrão, `padding` e `border` aumentam o tamanho total do elemento. Usamos `border-box` para que eles sejam incluídos **dentro** da `width` definida, facilitando o cálculo de layout.

CSS

```
.task-form input[type="text"],
.task-form select,
.task-form textarea {
    width: 100%;
    /* ESSENCIAL para que 100% inclua o padding e border */
    box-sizing: border-box;
    padding: 10px;
}
```

IV. Refinamento Visual e Interatividade

1. Estilizando por Status (Classes Dinâmicas)

Nosso JavaScript adiciona classes como `.status-pendente` e `.status-concluido` aos itens da lista. Isso permite que o CSS aplique cores visuais distintas para cada estado.

CSS

```
/* Estilo base para todos os itens */
.task-item {
    border-left: 6px solid #ccc;
}

/* Status Pendente: Alerta */
```

```css
.status-pendente {
    border-left-color: #ffc107;
}

/* Status Concluído: Sucesso */
.status-concluido {
    border-left-color: #28a745;
    opacity: 0.85; /* Deixa levemente transparente */
}

/* Estilo aninhado: aplica-se ao nome da tarefa DENTRO de uma tarefa concluída */
.status-concluido .task-name {
    text-decoration: line-through; /* Riscado */
    color: #6c757d;
}
```

2. Feedback Visual com Pseudo-Classes

A **pseudo-classe `:hover`** aplica estilos *somente* quando o usuário passa o mouse sobre o elemento, adicionando interatividade.

CSS

```css
.btn-submit {
    background-color: #28a745;
    transition: background-color 0.3s; /* Transição suave */
}

/* O estilo abaixo SÓ será aplicado quando o mouse estiver em cima */
.btn-submit:hover {
    background-color: #218838; /* Escurece a cor no hover */
}
```

V. Regras de Prioridade: Cascata e Especificidade

O que acontece quando você define uma cor para o `h1` na linha 10 e outra cor diferente na linha 50?

1. A Cascata (Ordem)

Em caso de empate na Especificidade, a última regra definida no arquivo CSS é a que "vence" (a que está **mais abaixo**).

2. A Especificidade (Peso)

A Especificidade é a pontuação do seletor. O seletor **mais específico** sempre vence, independentemente de sua posição na Cascata.

Seletor	Exemplo	Peso	Ordem de Precedência
Tipo (Tag)	`p`, `input`	Baixo (1 ponto)	Perde para Classes e IDs.
Classe	`.titulo`, `.btn`	Médio (10 pontos)	Vence Tags.
ID	`#main-container`	Alto (100 pontos)	Vence Classes e Tags.

Exportar para as Planilhas

> **Regra de Ouro:** Prefira sempre usar **Classes** para estilos. IDs devem ser reservados para elementos únicos e FlexBox/Grid será o foco da próxima aula!

📝 Lista de Exercícios Práticos

Use o código-fonte fornecido pelo professor para aplicar as modificações abaixo e fixar o aprendizado.

Nível Básico (Propriedades e Box Model)

1. **Mudar a Tipografia:** Altere a `font-family` do `body` para uma fonte sem ser `Arial` (ex: `Verdana` ou `Georgia`).
2. **Espaçamento Interno:** Aumente o `padding` nos campos de `select` para torná-los mais altos e visíveis.
3. **Bordas:** Adicione uma `border` tracejada (`dashed`) de cor clara em volta do contêiner principal (`.container`).

Nível Intermediário (Seletores e Reutilização)

4. **Estilo para Labels:** Use o **seletor de Tipo** (`label`) para estilizar *TODOS* os rótulos do formulário, aumentando o `font-size` em 10% (ex: `1.1em`).
5. **Criando um Botão de Ação:** Crie uma nova classe chamada `.btn-action` e estilize-a com uma `background-color` azul-escura (`#0056b3`). Adicione esta classe ao botão de submissão do formulário.
6. **Foco:** Crie um pseudo-seletor **`:focus`** para os inputs de texto. Quando o usuário clica para digitar, a `border-color` do input deve mudar para `#007bff`.

Nível Avançado (Especificidade e Interatividade)

7. **Desafio da Especificidade:** Tente aplicar a cor vermelha (`red`) no nome de uma tarefa (`.task-name`) usando, ao mesmo tempo, duas regras:
    - **Regra 1:** Use apenas o seletor de Classe (`.task-item`). Defina a cor do texto para `green`.
    - **Regra 2:** Use a combinação de seletores (`.task-item .task-name`). Defina a cor do texto para `red`.
    - **Qual cor prevalece? Por quê?**
8. **Efeito Flutuante:** Para cada item da lista (`.task-item`), crie um efeito `:hover` que faça a caixa "subir" levemente. Você pode fazer isso mudando o `box-shadow` ou usando a propriedade `transform: translateY(-2px);` (opcional).