








Estudo de CSS

O que é CSS?

CSS (**Cascading Style Sheets**) é a linguagem usada para definir o estilo de páginas web. Com ela, conseguimos alterar cores, fontes, espaçamento, layout e muito mais.

Ele trabalha junto com o **HTML**: o HTML cria a estrutura, e o CSS é responsável pela aparência.

Para que serve o CSS?

-  Mudar **cores** de textos e fundos
-  Alterar **fontes** e tamanhos
-  Controlar **margens, padding e espaçamentos**
-  Criar **layouts responsivos** (desktop, tablet, celular)
-  Adicionar **animações, sombras e efeitos visuais**

Exemplo básico de CSS

Vamos aplicar cor, fonte e centralizar um texto com CSS:

```
/* Arquivo style.css */
h1 {
  color: blue;
  text-align: center;
  font-family: Arial, sans-serif;
```

```
}

/* Aplicando ao HTML */
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
  <h1>Olá com CSS!</h1>
</body>
</html>
```

T Formas de aplicar CSS

- **Inline CSS:** estilo direto na tag HTML

```
<h1 style="color: red;">Título Vermelho</h1>
```

- **Internal CSS:** dentro da tag <style> no HTML

```
<style>
  h1 { color: green; }
</style>
```

- **External CSS:** arquivo .css separado (o mais usado)

</> CSS Syntax (Sintaxe do CSS)

A sintaxe do CSS é composta por três partes principais: **seletor** + **propriedade** + **valor**.

- ✦ **Seletor:** indica qual elemento HTML será estilizado.
- ✦ **Propriedade:** o que você quer modificar (cor, fonte, tamanho...).
- ✦ **Valor:** define como essa propriedade será aplicada.

```
/* Sintaxe Geral */
seletor {
  propriedade: valor;
}

/* Exemplo real */
h1 {
  color: blue;
  font-size: 24px;
```


```
text-align: center;
}
```

✓ Neste exemplo, o **seletor** é h1, a **propriedade** é color e o **valor** é blue.

 Estrutura detalhada:

 **Seletor:** Escolhe o(s) elemento(s) HTML a ser estilizado.

 **Propriedade:** Define qual característica será alterada.

 **Valor:** Define como a propriedade será aplicada.

Como adicionar CSS

Quando um navegador carrega uma página HTML, ele pode aplicar estilos usando CSS. Existem **três formas principais** de inserir CSS em um documento:

- **External CSS** – arquivo CSS separado.
- **Internal CSS** – dentro da tag <style> no arquivo HTML.
- **Inline CSS** – direto na linha do elemento HTML.

1 External CSS

O CSS fica em um arquivo separado (ex: style.css) e é conectado ao HTML com a tag <link>. É a forma mais profissional e usada na prática.

```
/* style.css */
h1 {
  color: blue;
  text-align: center;
}

/* HTML */
<head>
<link rel="stylesheet" href="style.css">
</head>
```

2 Internal CSS

O CSS fica dentro da tag <style>, diretamente no arquivo HTML. Bom para páginas simples.

```
<head>
  <style>
```

```
h1 {  
  color: green;  
  text-align: center;  
}  
</style>  
</head>
```


3 Inline CSS

O estilo é aplicado diretamente na tag HTML usando o atributo `style`. Não é recomendado para projetos grandes.

```
<h1 style="color: red; text-align: center;">  
  Título Vermelho  
</h1>
```

✅ **Dica:** Para projetos profissionais, use **External CSS** — ele deixa o código organizado e facilita a manutenção.

CSS Backgrounds

A propriedade **background** no CSS permite definir a cor de fundo,  imagens de fundo, repetição, posição e muito mais para elementos HTML.

 **background-color** – Define a cor de fundo

 **background-image** – Adiciona uma imagem de fundo

 **background-repeat** – Controla se a imagem repete ou não

 **background-size** – Ajusta o tamanho da imagem

 **background-position** – Define onde a imagem será posicionada

 **background-attachment** – Define se a imagem fica fixa ou rola com a página

Exemplo 1: Background Color

```
h1 {  
  background-color: lightblue;  
}
```

Exemplo 2: Background Image

```
body {  
    background-image: url('background.png');  
    background-repeat: no-repeat;  
    background-position: center;  
    background-size: cover;  
}
```

⚡ Exemplo 3: Shorthand (resumido)

Você pode combinar várias propriedades em uma só linha:

```
/* background: color image repeat position / size */  
body {  
    background: #1e293b url('fundo.jpg') no-repeat center / cover;  
}
```

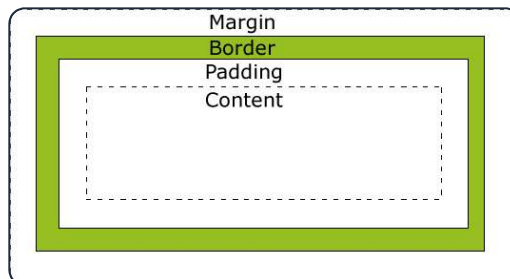
✅ **Dica:** Use `background-size: cover;` para imagens que ocupam toda a tela sem distorcer.

📦 CSS Box Model — Margin, Padding e Border

O **Box Model** é a base do layout no CSS. Todo elemento HTML é uma “caixa” composta por:

- **Content** — O conteúdo (texto, imagem...).
- **Padding** — Espaço interno entre o conteúdo e a borda.
- **Border** — A borda ao redor do elemento.
- **Margin** — Espaço externo entre o elemento e outros ao redor.

📊 Esquema do Box Model:



🔧 Exemplo de código:

```
div {  
  width: 200px;  
  padding: 20px; /* Espaço interno */  
  border: 3px solid #4b5563; /* Borda */  
  margin: 30px; /* Espaço externo */  
}
```

◆ Padding (espaçamento interno)

Cria espaço entre o conteúdo e a borda.

```
padding: 20px; /* Aplica nos 4 lados */  
padding: 10px 20px; /* Top/Bottom | Left/Right */  
padding: 5px 10px 15px 20px; /* Top | Right | Bottom | Left */
```

◆ Margin (espaçamento externo)

Cria espaço entre o elemento e outros ao redor.

```
margin: 20px; /* Todos os lados */  
margin: auto; /* Centraliza o elemento */  
margin: 10px 20px; /* Top/Bottom | Left/Right */
```

◆ Border (borda)

Adiciona bordas ao elemento.

```
border: 2px solid black; /* Largura | Estilo | Cor */  
border-radius: 10px; /* Borda arredondada */  
border-bottom: 3px dashed red; /* Borda inferior pontilhada */
```

✅ **Dica:** para ver o Box Model facilmente, use o inspetor do navegador (F12) e selecione um elemento.

Display: block, inline e inline-block

O **display** define como um elemento é mostrado na tela. Os mais usados são:

- **block:** ocupa toda a largura disponível e quebra linha (ex: div, h1, p).
- **inline:** não quebra linha e só ocupa o tamanho do conteúdo (ex: span, a).
- **inline-block:** fica na mesma linha, mas permite definir altura e largura.

Flexbox (display: flex)

O **Flexbox** é um modo de layout no CSS que facilita organizar, alinhar e distribuir elementos dentro de um div (container). Ele é ativado com `display: flex;`.

📌 Conceitos básicos:

- **Flex container:** o elemento pai que utiliza `display: flex;`.
- **Flex items:** os elementos filhos dentro do container.
- **Eixo principal (main axis):** pode ser horizontal ou vertical.
- **Alinhamentos fáceis:** centralizar, espaçar, organizar em linha ou coluna.

💻 Exemplo de Flexbox:

```
div.container {  
  display: flex;           /* Ativa o Flexbox */  
  gap: 10px;              /* Espaço entre os itens */  
  justify-content: center; /* Alinha itens no eixo principal */  
  align-items: center;     /* Alinha itens no eixo cruzado */  
}  
  
div.item {  
  background: #38bdf8;  
  padding: 20px;  
  border-radius: 8px;  
}
```

Visualização:

1 2 3

🔄 Direção dos itens:

```
flex-direction: row;           /* Padrão: itens em linha */  
flex-direction: column;       /* Itens em coluna */  
flex-direction: row-reverse; /* Linha invertida */  
flex-direction: column-reverse; /* Coluna invertida */
```

🎯 Alinhamento horizontal (justify-content):

```
justify-content: flex-start; /* Esquerda */  
justify-content: center;    /* Centro */  
justify-content: space-between; /* Espaço entre os itens */  
justify-content: space-around; /* Espaço ao redor */
```

Alinhamento vertical (align-items):

```
align-items: flex-start; /* Topo */
align-items: center;     /* Centro vertical */
align-items: flex-end;   /* Base */
```

✅ **Dica:** Flexbox é perfeito para centralizar elementos no meio da tela:

```
.center {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh; /* tela inteira */
}
```

Layout com CSS Grid

O **CSS Grid** é um sistema poderoso de layout bidimensional — ou seja, permite organizar elementos em **linhas e colunas** com facilidade.

- **Grid Container:** o elemento pai que recebe `display: grid;`
- **Grid Items:** os elementos (filhos) que ficam dentro do grid
- **Linhas & Colunas:** definidas com `grid-template-rows` e `grid-template-columns`
- **Gap:** espaçamento entre os elementos da grade

Exemplo básico de Grid:

```
/* Container (pai) */
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr; /* 3 colunas iguais */
  gap: 10px; /* Espaço entre os itens */
}

/* Itens (filhos) */
.item {
  background-color: #38bdf8;
  padding: 20px;
  border-radius: 8px;
  text-align: center;
}
```

 Exemplo visual:

1

2

3

4

5

6

Colunas com tamanhos diferentes

```
grid-template-columns: 200px 1fr 2fr;  
/* 1ª coluna fixa (200px)  
2ª flexível (ocupa o que sobrar)  
3ª coluna ocupa o dobro da 2ª */
```

Linhas (grid-template-rows)

```
grid-template-rows: 100px 100px auto;  
/* Primeiras duas linhas com altura fixa, a última ocupa o resto */
```

Item ocupando mais colunas (grid-column)

```
.item1 {  
  grid-column: span 2; /* Esse item ocupa 2 colunas */  
}
```

✅ **Dica:** O Grid é perfeito para criar layouts de páginas, dashboards e galerias de forma organizada e responsiva.

Estilizando Formulários com CSS

Formulários são essenciais em qualquer site. Com CSS, podemos deixar inputs, botões e labels mais bonitos e fáceis de usar.

Exemplo básico:

```
<form>  
  <label for="nome">Nome:</label>  
  <input type="text" id="nome" placeholder="Digite seu nome">  
  
  <label for="email">E-mail:</label>  
  <input type="email" id="email" placeholder="Digite seu e-mail">
```

```
<button type="submit">Enviar</button>
</form>
```

```
form {
  display: flex;
  flex-direction: column;
  gap: 12px;
  max-width: 300px;
}

input {
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 6px;
  transition: 0.3s;
}

/* Efeito ao clicar no input (focus) */
input:focus {
  border-color: #3b82f6; /* azul */
  outline: none;
  box-shadow: 0 0 4px #60a5fa;
}

button {
  background-color: #3b82f6;
  color: white;
  padding: 10px;
  border: none;
  border-radius: 6px;
  cursor: pointer;
}

button:hover {
  background-color: #2563eb;
}
```

Visual do formulário:

Nome:

E-mail:

Enviar

Outros tipos de input:

```
input[type="checkbox"] { accent-color: #3b82f6; }      /* Muda cor do checkbox */  
input[type="radio"] { accent-color: #f97316; }     /* Radio laranja */  
input[type="range"] { width: 100%; }              /* Slider */
```

✅ **Dica:** Você pode combinar `:focus`, `:hover`, `:valid` e `:invalid` para criar formulários interativos e acessíveis.

[Próximo Módulo: JavaScript](#)[Baixar PDF - Módulo CSS](#)

<> Teste você mesmo:

HTML:

```
1  
2 v  <div class="container">  
3     <span class="badge">CSS</span>  
4     <h1>Olá com CSS!</h1>  
5     <p>Edite o CSS no editor para ver mudanças ao vivo.</p>  
6     <button class="btn">Botão estiloso</button>  
7 </div>  
8
```

CSS:

```
1  
2 v  :root {  
3     --brand: #2563eb;
```

```
4      --bg: #f8fafc;  
5      --text: #0f172a;  
6  }  
7  
8  v  body {  
9      margin: 0;  
10     font-family: Arial, Helvetica, sans-serif;  
11     background: var(--bg);  
12     color: var(--text);  
13  }  
14  
15  v  .container {  
16     max-width: 640px;  
17     margin: 2rem auto;  
18     padding: 1.25rem;  
19     background: #fff;  
20     border-radius: 12px;  
21     box-shadow: 0 6px 18px rgba(0, 0, 0, 0.08);  
22  }
```

Pré-visualização:

CSS

Olá com CSS!

Edite o CSS no editor para ver mudanças ao vivo.

Botão estiloso



Acesse nosso repositório no GitHub e deixe sua contribuição

© 2025 Adriano Souza Fosneca Desenvolvedor Full-Stack Jr. Todos os direitos reservados.