

HTML & CSS:

BOAS PRÁTICAS DE DESENVOLVIMENTO

UM GUIA PRÁTICO

POR DANIEL HASAN DALIP E IZABELA A. ANDRADE

HTML & CSS: BOAS PRÁTICAS DE DESENVOLVIMENTO UM GUIA PRÁTICO

“Nada na vida deve ser temido, somente compreendido.”
Marie Curie

[GITHUB CLASSROOM - Um Guia Prático](#)

© 2025 por Daniel Hasan Dalip, Izabela Amélia
Marques De Andrade licenciado sob [CC BY 4.0](#)



“Bem-vindo ao nosso e-book de boas práticas em HTML e CSS!

Este guia foi criado para ajudar nossos alunos a desenvolverem códigos mais organizados, eficientes e fáceis de manter ao longo do curso de Programação Web. Aqui, reunimos diretrizes essenciais para escrever HTML e CSS de maneira clara, semântica e escalável, garantindo um desenvolvimento mais profissional.

A estrutura e a estilização de uma página web vão muito além de simplesmente "fazer funcionar". Um código bem estruturado facilita a colaboração, evita problemas futuros e melhora a acessibilidade e a performance do site.

Neste material, abordamos desde a organização do HTML e a nomeação de classes e IDs até o uso adequado de seletores e a otimização do CSS, sempre com exemplos práticos para reforçar o aprendizado.

Este e-book será um ótimo recurso para aprimorar suas habilidades e criar projetos mais bem estruturados.

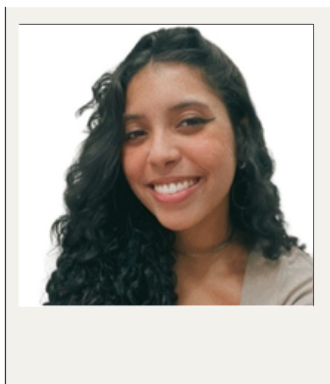
Fique à vontade para compartilhar este material com colegas e revisité-lo sempre que precisar. Vamos juntos escrever códigos melhores e mais eficientes! 🚀”

IDEALIZAÇÃO & ELABORAÇÃO



DANIEL HASAN DALIP

Daniel H. Dalip é professor no Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG) e é coordenador de Inovação e Empreendedorismo na instituição. Ele possui doutorado (UFMG/2015), mestrado (UFMG/2009) e bacharelado (Uni-BH/2006) em Ciência da Computação. Com ampla experiência docente, Daniel leciona disciplinas como Programação Web, Algoritmos, Recuperação de Informação e Aprendizado de Máquina.



IZABELA A. M. ANDRADE

Cursou ensino médio técnico em Informática no CEFET-MG (2019-2021), onde participou dos projetos Bytes & Elas e Enxurrada de Bits, promovendo a inclusão de mulheres e jovens de escolas públicas na computação. Tem grande interesse em desenvolvimento web, especialmente front-end. Atualmente, é estudante de Ciências Biomédicas na USP e atua como voluntária no curso de Introdução a HTML, CSS, JavaScript e UX.

INFORMAÇÕES DO CURSO

Instagram - [@expresso.dev.cefetmg](#)

[Inscrições e mais informações](#) 

[Dúvidas frequentes](#) 

WEB.EXPRESSODEV.COM.BR

ÍNDICE

1. HTML	7
◦ Estrutura e aspectos básicos	7
◦ Tags Semânticas	9
◦ Carregamento e organização de recursos	10
2. CSS: Boas Práticas de Estilização	12
◦ Separação de responsabilidades: CSS e HTML.....	12
◦ Separação entre estrutura e estilo	13
◦ Seletores e generalizações	14
◦ Uso de IDs e classes	15



1.HTML



1.1 ESTRUTURA & ASPECTOS BÁSICOS

VERSÃO DE UTILIZAÇÃO: DOCTYPE HTML

Ao analisar a estrutura padrão de um documento HTML, você provavelmente já notou a presença da tag **<!DOCTYPE>**, mas sabe qual a sua função?

DOCTYPE, que significa "Document Type Declaration", é uma declaração essencial que informa ao navegador qual versão do HTML está sendo utilizada na página. A versão mais recente e amplamente adotada é o HTML5. Para indicar que você está utilizando essa versão, basta incluir a seguinte linha no início do seu código:

```
<!DOCTYPE html>
```

Caso queira utilizar versões anteriores, como o HTML4 ou o XHTML, a declaração seria diferente. Abaixo estão alguns exemplos para essas versões mais antigas.

```
<!-- HTML4 -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

<!-- XHTML 1.0 -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```



Essa tag é essencial para garantir a compatibilidade da página entre diferentes navegadores, além de auxiliar na correta renderização do conteúdo, evitando problemas de formatação e comportamento inesperado.

1.HTML

1.1 ESTRUTURA & ASPECTOS BÁSICOS

ORGANIZAÇÃO: INDENTAÇÃO

Um dos aspectos que facilita muito o entendimento da estrutura de um arquivo HTML é a **indentação**. As tags HTML são organizadas em "seções" ou áreas, que podem representar tanto o conteúdo visível da página (como cabeçalhos e postagens) quanto a estrutura interna necessária para o navegador entender como exibir essa página.

Essas tags frequentemente são **aninhadas**, ou seja, uma tag estará dentro de outra.



Para tornar essa hierarquia clara e facilitar a leitura, utilizamos um padrão de indentação. Quando um elemento está dentro de outro, ele é "deslocado" para a direita, geralmente com um espaço ou um tab, de modo que fique visualmente claro que ele faz parte do elemento anterior.

Além disso, é importante lembrar que, em HTML, todas as tags devem ser abertas e fechadas corretamente. A indentação ajuda a identificar rapidamente quais tags estão relacionadas e como a estrutura do código se organiza.

EXEMPLO PRÁTICO:

Abaixo, apresentamos um exemplo comparativo: à esquerda, um código com indentação adequada e, à direita, um código onde não se seguiu o padrão de indentação. O primeiro exemplo é muito mais fácil de entender e visualizar a estrutura, enquanto o segundo pode se tornar confuso e difícil de manter.

CÓDIGO INDENTADO

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"></head>
    <title>Exemplo Indentado</title>
  </head>
  <body>
    <div>
      <p>Conteúdo</p>
    </div>
  </body>
</html>
```

CÓDIGO NÃO INDENTADO

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"></head>
<title>Exemplo Não Indentado</title>
</head>
<body>
<div>
<p>Conteúdo</p>
</div>
</body>
</html>
```


1.HTML

1.2 TAGS SEMÂNTICAS

TAGS SEMÂNTICAS: EVITANDO A “DIVITIDE”

Embora as tags `<div>` sejam úteis para estruturar e dividir seções de uma página, elas são **genéricas** e não transmitem significado sobre o conteúdo que contém.

Para tornar o código mais organizado e semanticamente claro, o ideal é utilizar **tags semânticas**. Essas tags são mais descritivas e ajudam tanto outros desenvolvedores que leem o código quanto o próprio navegador a compreender melhor a estrutura e o propósito de cada parte da página.

Além de melhorar a organização, o uso de tags semânticas também contribui para acessibilidade e SEO, tornando a página mais bem interpretada por leitores de tela e motores de busca.

Alguns exemplos de tags semânticas e seus usos:

- **<header>** → Representa o cabeçalho da página ou de uma seção
- **<footer>** → Estrutura o rodapé da página
- **<section>** → Define uma seção do conteúdo
- **<nav>** → Indica uma área de navegação, como menus e links
- **<article>** → Representa um artigo independente, como uma postagem de blog

UM EXEMPLO PRÁTICO:

Com tags semânticas

```
<!DOCTYPE html>
<html lang="pt">
<head>
  <meta charset="UTF-8">
  <title>Embriologia - Introdução</title>
</head>
<body>

  <header>
    <h1>Embriologia - Introdução</h1>
    <nav>
      <ul>
        <li><a href="#fertilization">Fertilização</a></li>
        <li><a href="#development">Desenvolvimento</a></li>
      </ul>
    </nav>
  </header>

  <section id="fertilization">
    <h2>Fertilização</h2>
    <p>A fertilização é o processo em que...</p>
  </section>

  <section id="development">
    <h2>Desenvolvimento Embrionário</h2>
    <p>O desenvolvimento embrionário começa...</p>
  </section>

  <footer>
    <p>Embriologia Básica</p>
  </footer>

</body>
</html>
```

Com divs genéricas

```
<!DOCTYPE html>
<html lang="pt">
<head>
  <meta charset="UTF-8">
  <title>Embriologia - Introdução</title>
</head>
<body>

  <div class="header">
    <h1>Embriologia - Introdução</h1>
    <div class="nav">
      <ul>
        <li><a href="#fertilization">Fertilização</a></li>
        <li><a href="#development">Desenvolvimento</a></li>
      </ul>
    </div>
  </div>

  <div class="section" id="fertilization">
    <h2>Fertilização</h2>
    <p>A fertilização é o processo em...</p>
  </div>

  <div class="section" id="development">
    <h2>Desenvolvimento Embrionário</h2>
    <p>O desenvolvimento embrionário começa...</p>
  </div>

  <div class="footer">
    <p>Fonte: Curso de Embriologia Básica</p>
  </div>

</body>
</html>
```

1.HTML

1.2 CARREGAMENTO & ORGANIZAÇÃO DE RECURSOS

MAPEAMENTO DE HYPERLINKS: OS LOCAIS ADEQUADOS PARA CADA TIPO

Um arquivo HTML, em sua essência, define para o navegador a estrutura e o conteúdo que será exibido na página. Para tornar essa apresentação mais estilizada e personalizada, utilizamos o **CSS** (Cascading Style Sheets). Embora seja possível incluir CSS diretamente no arquivo HTML, a melhor prática é manter os estilos em um arquivo separado (.css) e referenciá-lo por meio de um link.



ARQUIVOS .CSS: ONDE REFERENCIAR?

O ideal é que os arquivos CSS sejam vinculados dentro do `<head>`, junto com outras referências de estilo externas, como fontes do Google Fonts.



E por que não no `<body>`?

Isso acontece porque o `<head>` é processado antes do `<body>` e contém informações essenciais para a exibição inicial da página. Se o CSS for carregado tardiamente, há o risco de a página ser exibida sem estilos por um breve momento (efeito conhecido como FOUC - Flash of Unstyled Content).

Portanto, para garantir que a estilização seja aplicada corretamente desde o início, sempre referencie seus arquivos de estilo no `<head>`.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"></head>
    <link rel="stylesheet" href="./CSS/estilo_base.css">
    <link rel="stylesheet" href="./CSS/estilo_especifico.css">
    <link href="https://fonts.googleapis.com/css?family=Open+Sans[Quicksand&display=swap" rel="stylesheet">
  </head>
  <body>
    <!-- Conteúdo da página :> -->
  </body>
</html>
```

⚠ ORGANIZAÇÃO & CARREGAMENTO

Um ponto importante ao trabalhar com múltiplos arquivos CSS é a ordem de carregamento. Como o CSS segue um conceito de "cascata", as regras são aplicadas na sequência em que os arquivos são carregados. *Isso significa que um arquivo posterior pode sobrescrever estilos definidos em arquivos anteriores, caso haja regras conflitantes.*

Para evitar problemas, organize seus arquivos de estilo de forma lógica. Por exemplo, arquivos com estilos gerais devem ser carregados primeiro, enquanto arquivos específicos ou personalizados devem vir depois.

1.HTML

1.2 CARREGAMENTO & ORGANIZAÇÃO DE RECURSOS

REFERENCIAMENTO DE SCRIPTS & MELHORIAS DE DESEMPENHO

Além do HTML, que organiza o conteúdo, e do CSS, que define sua aparência, páginas web também podem incluir scripts para tornar a experiência mais dinâmica e interativa.

Esses scripts, geralmente escritos em JavaScript (.js), podem ser incorporados diretamente no HTML, mas, assim como o CSS, a melhor prática é mantê-los em arquivos separados e referenciá-los no código.

No entanto, diferentemente do CSS, a recomendação é que os scripts não sejam referenciados no <head>, mas sim ao final do <body>.

MAS POR QUÊ?

Isso acontece porque arquivos JavaScript podem ser mais complexos e levar mais tempo para serem processados.

Se fossem carregados no <head>, o navegador precisaria processá-los antes de exibir o conteúdo da página, o que poderia causar um atraso na renderização e resultar em uma tela em branco por alguns instantes.



Ao posicionar os scripts no final do <body>, garantimos que o HTML seja carregado primeiro, permitindo que o usuário visualize o conteúdo da página imediatamente, enquanto o JavaScript é processado em segundo plano.

Um exemplo básico:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"></head>
    <link rel="stylesheet" href="/CSS/estilo_base.css">
    <link rel="stylesheet" href="/CSS/estilo_especifico.css">
    <link href="https://fonts.googleapis.com/css?family=Open+Sans|Quicksand&display=swap" rel="stylesheet">
  </head>
  <body>
    <!-- Conteúdo da página :) -->
    <div>
      <p>Minha página!</p>
    </div>

    <!-- Ao fim do body, referência do script -->
    <script src="/JS/script.js"></script>
  </body>
</html>
```

2. CSS: BOAS PRÁTICAS DE ESTILIZAÇÃO

2.1 SEPARAÇÃO DE RESPONSABILIDADES: CSS E HTML

MÁ PRÁTICA: USO DE TAGS PARA ESTILIZAR

Usar `
`, `` e outras tags para mudar a aparência do seu site pode parecer prático, mas deixa o código bagunçado e difícil de manter.

Há uma separação de responsabilidade: ***O HTML serve para estruturar o conteúdo, enquanto o CSS cuida da aparência.***

EXEMPLO PRÁTICO

Vamos supor que queremos deixar um espaçamento maior abaixo do primeiro para o segundo parágrafo de um texto:

Primeira tentativa:

```
<section>
  <p>Meu primeiro paragrafo<p>
  <br><br><br>
  <p>Meu segundo paragrafo mais distante</p>
</section>
```



Imagina se quiséssemos mudar o espaçamento? Ou ajustá-lo conforme o tamanho da tela?

Segunda tentativa:

```
<section>
  <p>Meu primeiro paragrafo<p>
  <p>Meu segundo paragrafo mais distante</p>
</section>
```

```
section p:first-child{
  margin-bottom: 50px;
}

@media (max-width: 480px) {
  section p:first-child {
    margin-bottom: 10px;
  }
}
```

Manter HTML e CSS separados deixa seu código limpo, flexível e muito mais escalável. Seu "eu" programador do futuro agradece!

CSS: ONDE COLOCAR?

Embora seja possível usar a tag `<style>` no `<head>` ou até estilizar elementos com CSS inline, essas práticas tornam o código menos escalável e difícil de manter. Isso porque cada regra fica isolada, sem possibilidade de reaproveitamento.



O ideal é manter o CSS em um arquivo externo e referenciá-lo no `<head>`, garantindo organização e flexibilidade no desenvolvimento.

2. CSS: BOAS PRÁTICAS DE ESTILIZAÇÃO

2.2 SEPARAÇÃO ENTRE ESTRUTURA & ESTILO

CSS INLINE X ARQUIVO EXTERNO: UM EXEMPLO PRÁTICO

CSS INLINE (MÁ PRÁTICA):

Usado dessa forma, ele não pode ser reutilizado para outras porções de código ou outros arquivos html.

```
<!DOCTYPE html>
<html lang="pt">
<head>
  <meta charset="UTF-8">
  <title>Exemplo de Estilização</title>
</head>
<body>

  <h1><strong style="color: blue;">Título em azul com estilização inline</strong></h1>
  <p><em style="font-size: 18px;">Texto em itálico com tamanho de fonte específico</em></p>
  <p style="color: red;">Texto com estilo inline</p>

</body>
</html>
```



Esse código mostrou ainda uma outra má prática que informamons na seção X: usou-se **strong** e **em** para estilizar. Conforme informado nessa seção, se quisermos um parágrafo enfatizado, ideal é alterar o seu estilo pelo **CSS**.

SEPARAÇÃO HTML E CSS: O MAIS ADEQUADO

A estilização é feita separadamente em um arquivo .CSS que é referenciado no html. Esse modelo permite uma estilização altamente escalável entre diferentes porções de código utilizando classes, além de permitir reutilização do CSS entre diferentes arquivos html.

```
<!DOCTYPE html>
<html lang="pt">
<head>
  <meta charset="UTF-8">
  <title>Exemplo de Estilização</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

  <h1 class="titulo">Título em azul</h1>
  <p class="texto-it">Texto em itálico</p>
  <p class="texto-vermelho">Texto com cor vermelha</p>

</body>
</html>
```

```
/* Estilo para título */
.titulo {
  color: blue;
}

/* Estilo para texto em itálico */
.texto-it {
  font-size: 18px;
  font-style: italic;
}

/* Estilo para texto vermelho */
.texto-vermelho {
  color: red;
}
```

2. CSS: BOAS PRÁTICAS DE ESTILIZAÇÃO

2.3 SELETORES & GENERALIZAÇÕES

TAGS COMO SELETORES PORQUE NÃO UTILIZAR?

A estilização através do CSS ocorre por meio de regras que aplicam estilos às propriedades de um ou mais elementos definidos pelos seletores. Esses seletores podem ser IDs, classes ou até mesmo tags HTML. No entanto, o uso de tags como seletores deve ser feito com cautela.

Isso porque, ao utilizar uma tag como seletor, **a regra será aplicada automaticamente a todas as ocorrências daquela tag na página**. Isso pode gerar conflitos com outras estilizações mais específicas e dificultar a manutenção do código. Além disso, se o mesmo CSS for reaproveitado em diferentes páginas, o uso excessivo de seletores por tag pode **comprometer a escalabilidade**, já que diferentes páginas podem ter estilos distintos para as mesmas tags.

Por isso, a recomendação principal é priorizar o uso de classes, ids (e seletores combinem tags, classes e/ou ids), conforme será detalhado no item 2.3.

OUTRO PONTO DE ATENÇÃO: REPETIÇÕES DE CÓDIGO CSS

Para otimizar o código e facilitar sua manutenção, é essencial evitar a repetição de regras CSS. Quando diferentes elementos compartilham estilos semelhantes, em vez de duplicar regras, podemos agrupá-los em um mesmo seletor e definir separadamente apenas as propriedades que precisam ser diferenciadas. Isso torna o código mais limpo, eficiente e fácil de manter.

EXEMPLO PRÁTICO:

HTML base:

```
<p class="mensagem">Bem-vindo à Padaria!</p>
<p class="aviso">Promoção por tempo limitado!</p>
```

CSS base:

Com repetição

```
.mensagem {
  font-size: 18px;
  color: □ #333;
}


.aviso {
  font-size: 18px;
  color: ■ #ff0000;
}
```

Sem repetição

```
.mensagem, .aviso {
  font-size: 18px;
}

.mensagem {
  color: □ #333;
}

.aviso {
  color: ■ #ff0000;
}
```



Observe que a propriedade `font-size`, que possui o mesmo valor para ambos os seletores, foi agrupada em uma única regra, evitando repetições desnecessárias.

Outra abordagem seria criar uma classe geral, como `texto`, e aplicá-la a ambos os parágrafos. Isso tornaria a estilização ainda mais organizada e reutilizável, facilitando futuras manutenções.

2. CSS: BOAS PRÁTICAS DE ESTILIZAÇÃO

2.3 USO EXCESSIVO DE IDS



ESPECIFICIDADE: USANDO IDS E DEMAIS SELETORES

Ao separar diferentes porções do código para estilização, podemos utilizar duas categorias principais de identificadores:

- **IDs:** para elementos únicos.
- **Classes e demais seletores combinados:** para grupos de elementos semelhantes.



É recomendável utilizar classes ou seletores tags e classes, sempre que possível para facilitar a reutilização de um estilo ao longo de várias páginas

Isso ocorre porque, no desenvolvimento, a ideia é criar códigos que sejam reutilizáveis. Assim, ao adicionarmos mais páginas, não seria necessário muitos ajustes no estilo, apenas usar a mesma padronização de classes e tags.

Dessa forma, também é recomendável realizar uma convenção de nomenclatura das classes e não criar classes muito específicos. Pense no caso extremo: de criar uma classe para cada elemento e, assim, ficaria similar ao uso de ids.

IDS E CLASSES: NOMENCLATURA ADEQUADA

Embora seja possível nomear seus IDs e classes da maneira que achar mais conveniente, existe um padrão amplamente utilizado que torna o código mais legível e organizado. Por convenção, o ideal é que os nomes sejam **descritivos**, ou seja, representem claramente o conteúdo ou a função do elemento, e **curtos**, sempre que possível.

Além disso, para garantir a consistência e facilitar a leitura, recomenda-se que os nomes sejam escritos conforme o padrão **kebab-case**: em **letras minúsculas**, utilizando o caractere "-" (hífen) para **separar palavras**, ao invés de espaços ou underscores (sublinhados).

Exemplo prático:

Classes com nomes pouco descritivos.

```
<div class="s">
  <h2 class="tt">Bolo de Chocolate</h2>
  <p class="ds">Bolo fofinho com cobertura.</p>
</div>
```

```
<div class="produto">
  <h2 class="titulo-produto">Bolo de Chocolate</h2>
  <p class="descricao-produto">Bolo fofinho com cobertura.</p>
</div>
```

Classes com nomes descritivos e seguindo as convenções de nomenclatura.



Procure usar sempre nomes mais descritivos. Por exemplo, ao invés de "quantidade" use "quantidade-produtos" ao invés de "titulo", "descricao", use "titulo-cartao", "descricao-cartao". Ah, e não use acentos ;-)