



# Apresentação de ferramentas e conceitos básicos

# A

## apresentação das linguagens

No desenvolvimento de software, o ambiente e as ferramentas utilizadas desempenham um papel crucial no sucesso do projeto.

Nesta primeira parte, vamos explorar as linguagens fundamentais para o desenvolvimento web, compreendendo suas características e aplicações práticas.

### HTML: A Base da Estrutura Web

HTML (HyperText Markup Language) é a linguagem de marcação responsável por definir a estrutura de uma página web. Imagine o HTML como o esqueleto de um edifício; ele dá forma e organização ao conteúdo, determinando onde cada elemento estará localizado. Através de um conjunto de tags, o HTML permite a criação de títulos, parágrafos, listas, links, imagens e muito mais.

Por exemplo, ao criar um site de notícias, o HTML será utilizado para definir as seções de manchetes, artigos e rodapés, organizando o conteúdo de maneira lógica e acessível. As tags são como os tijolos que constroem essa estrutura, permitindo ao navegador interpretar e exibir a página adequadamente.

## **CSS: Estilo e Apresentação**



Enquanto o HTML define a estrutura, o CSS (Cascading Style Sheets) cuida da aparência visual da página. Se compararmos a construção de uma página web à construção de um prédio, o CSS seria a pintura e a decoração, responsáveis por tornar o espaço atraente e funcional.

Com o CSS, é possível definir cores, fontes, espaçamentos, alinhamentos e outras características estilísticas. Por exemplo, em um site de e-commerce, o CSS será utilizado para estilizar os botões de compra, destacar promoções com cores vibrantes e garantir que o layout seja responsivo, adaptando-se a diferentes tamanhos de tela.

## **JavaScript: Interatividade e Funcionalidade**

JavaScript é a linguagem de programação que traz vida e interatividade às páginas web. Enquanto o HTML e o CSS são responsáveis pela estrutura e estilo, respectivamente, o JavaScript permite que os elementos na página respondam às ações dos usuários.

Imagine um site de música, como o Spotify, por exemplo. Quando o usuário clica em um botão para reproduzir uma música, é o JavaScript que detecta essa ação e inicia a reprodução. Além disso, ele permite a criação de menus interativos, animações, validação de formulários e muitas outras funcionalidades que tornam a experiência do usuário mais dinâmica e envolvente.

Por exemplo, em um formulário de cadastro, o JavaScript pode verificar se todos os campos obrigatórios foram preenchidos corretamente antes de permitir o envio, melhorando a usabilidade e a segurança do sistema.

## Exemplo Prático



Para ilustrar a aplicação das linguagens que abordamos, vamos criar uma estrutura básica de uma página web utilizando HTML, CSS e JavaScript.

### Estrutura HTML

Começaremos definindo a estrutura da página utilizando HTML. Imagine que queremos criar uma página simples com um cabeçalho, um parágrafo de texto e um botão que, ao ser clicado, exibe uma mensagem. O código HTML poderia ser:

```
<!DOCTYPE html>
```

```
<html lang="pt-BR">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>Página Exemplo</title>
```

```
</head>
```

```
<body>
```

```
  <header>
```

```
    <h1>Bem-vindo à Página Exemplo</h1>
```

```
  </header>
```

<main>



<p>Este é um exemplo prático de como utilizar HTML, CSS e JavaScript em uma página web.</p>

<button id="mensagemBotao">Clique aqui</button>

</main>

</body>

</html>

Neste exemplo, o HTML cria um cabeçalho com o título "Bem-vindo à Página Exemplo", um parágrafo descritivo e um botão com o texto "Clique aqui".

## **Estilizando com CSS**

Agora, vamos adicionar um estilo básico à nossa página para melhorar sua aparência. O CSS pode ser adicionado diretamente no cabeçalho do HTML ou em um arquivo separado. Para este exemplo, vamos adicionar o CSS no próprio HTML:

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;



background-color: #f4f4f4;

}

header {

background-color: #333;

color: white;

padding: 10px 0;

text-align: center;

}

main {

margin: 20px;

padding: 20px;

background-color: white;

border-radius: 8px;

}

button {

background-color: #007bff;

color: white;

padding: 10px 20px;

```
border: none;
```



```
border-radius: 5px;
```

```
cursor: pointer;
```

```
}
```

```
button:hover {
```

```
background-color: #0056b3;
```

```
}
```

```
</style>
```

Aqui, aplicamos um estilo limpo e moderno à página. O fundo da página é cinza claro, enquanto o cabeçalho tem um fundo escuro com texto branco. O botão é azul com um efeito de hover, mudando para uma tonalidade mais escura quando o usuário passa o cursor sobre ele.

### **Interatividade com JavaScript**

Finalmente, vamos adicionar interatividade ao botão utilizando JavaScript. Queremos que, ao clicar no botão, uma mensagem seja exibida para o usuário:

```
<script>
```

```
document.getElementById('mensagemBotao').addEventListener('click',  
function() {
```

```
    alert('Você clicou no botão!');
```

```
});
```



```
</script>
```

Neste código, utilizamos JavaScript para adicionar um evento de clique ao botão. Quando o botão é clicado, uma mensagem de alerta com o texto “Você clicou no botão!” aparece na tela.

## **Resultado**


Ao juntar todas essas partes, temos uma página web simples, mas funcional, que utiliza HTML para estruturar o conteúdo, CSS para estilizar e JavaScript para adicionar interatividade. Este exemplo prático demonstra como as três linguagens trabalham juntas para criar uma experiência completa para o usuário na web.

## **Ambiente de Desenvolvimento Web**

No desenvolvimento web, o ambiente de trabalho é onde tudo acontece: é onde você cria, testa e ajusta cada detalhe das suas páginas e aplicações. Nesta parte, vamos explorar as ferramentas que serão suas companheiras durante o curso, começando pelo Visual Studio Code (VS Code) e o navegador Google Chrome.

## **Visual Studio Code em Ação**

O Visual Studio Code é mais do que apenas um editor de texto; ele é uma ferramenta completa para desenvolvimento. Imagine que você está construindo uma página de portfólio pessoal. No VS Code, você começa

criando o arquivo `index.html`, que é o coração da sua página. Conforme vai escrevendo o código HTML, o VS Code destaca a sintaxe, sugere  autocompletar para as tags, o que facilita e acelera seu trabalho.

Agora, você quer que a sua página tenha um visual atraente. No VS Code, você cria um arquivo `style.css` e começa a definir as cores, fontes e layout da página. Aqui, o VS Code já está configurado para te ajudar com dicas de CSS e até mesmo com a aplicação automática de algumas correções de estilo.

Por fim, você adiciona interatividade com JavaScript. Quer que um botão na sua página mostre uma mensagem quando clicado? No VS Code, você cria um arquivo **`script.js`**, escreve o código para adicionar esse comportamento, e em seguida já pode ver o resultado em ação.

## **Testando no Google Chrome**

Depois de escrever o código, é hora de ver como tudo funciona no mundo real, ou seja, no navegador. **Google Chrome** será a sua principal ferramenta para isso. Você salva o seu trabalho no VS Code, abre o Chrome e carrega a página **`index.html`**.

Aqui começa a parte interessante: com o Chrome, você não apenas visualiza a página, mas também pode inspecionar cada elemento. Quer saber por que aquele botão não está alinhado corretamente? Use as ferramentas de desenvolvedor do Chrome para inspecionar o CSS aplicado ao botão e ajustar no VS Code.

Além disso, o Chrome permite que você teste como sua página se comporta em diferentes dispositivos, como smartphones e tablets. Você pode simular a visualização em um iPhone, por exemplo, para garantir que seu design é responsivo e funciona bem em telas menores.





## **Exemplo Prático: Criando e Testando um Formulário de Contato**

Vamos colocar a teoria em prática com um exemplo simples: criar e testar um formulário de contato.

Crie o HTML no VS Code: Comece criando um arquivo contact.html. Adicione os campos de entrada para nome, email e mensagem, e um botão para enviar o formulário.

```
<form>
```

```
  <label for="name">Nome:</label>
```

```
  <input type="text" id="name" name="name">
```

```
  <label for="email">Email:</label>
```

```
  <input type="email" id="email" name="email">
```

```
  <label for="message">Mensagem:</label>
```

```
  <textarea id="message" name="message"></textarea>
```

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

```
</form>
```

Estilize com CSS: No VS Code, crie um arquivo style.css e adicione estilo ao seu formulário para deixá-lo mais atraente.



```
form {  
  
    max-width: 500px;  
  
    margin: auto;  
  
    padding: 20px;  
  
    border-radius: 5px;  
  
    background-color: #f4f4f4;  
  
}
```

Adicione Interatividade com JavaScript: Ainda no VS Code, crie um arquivo script.js para validar os campos de entrada antes de enviar o formulário.

```
document.querySelector('form').addEventListener('submit', function(e) {  
  
    const name = document.getElementById('name').value;  
  
    const email = document.getElementById('email').value;  
  
  
    if (name === "" || email === "") {  
  
        alert('Por favor, preencha todos os campos.');
```

```
e.preventDefault();
```



```
}
```

```
});
```

Teste no Google Chrome: Salve os arquivos, abra o **contact.html** no Chrome, e teste. Veja se o estilo está correto e se a validação funciona. Use as ferramentas de desenvolvedor para ajustar o layout e inspecionar o comportamento do JavaScript.

## W3C

No desenvolvimento web, é essencial que as páginas que você cria sejam acessíveis, funcionais e compatíveis com diferentes dispositivos. Para garantir que isso aconteça, seguimos os padrões estabelecidos pelo World Wide Web Consortium (W3C). Vamos ver na prática como essa organização impacta diretamente o desenvolvimento do seu projeto.

### O que é o W3C e por que é importante?

O **W3C** foi fundado em 1994 por **Tim Berners-Lee**, o mesmo criador da web, com o objetivo de estabelecer padrões para o desenvolvimento de páginas e aplicações web. Esses padrões são essenciais para garantir que o seu site funcione corretamente em diferentes navegadores e dispositivos, além de ser acessível a todos os usuários, independentemente de suas capacidades ou do equipamento que estejam usando.

## **Como o W3C Afeta o Seu Código**



Imagine que você está criando uma página web que será acessada tanto em um computador quanto em um smartphone. Se você seguir as diretrizes do W3C, como o uso correto de HTML, CSS e JavaScript, a sua página será exibida de forma consistente em ambos os dispositivos. Por exemplo, ao usar tags semânticas como **<header>**, **<nav>**, **<article>**, e **<footer>**, você garante que o conteúdo seja bem organizado e compreensível, não apenas para os navegadores, mas também para leitores de tela usados por pessoas com deficiência visual.

Vamos para um exemplo prático: ao criar uma página de portfólio, você define uma estrutura HTML clara, usa CSS para garantir que o layout seja responsivo (se adapte a diferentes tamanhos de tela), e adiciona interatividade com JavaScript. Seguindo os padrões do W3C, você sabe que essa página funcionará corretamente, seja acessada em um iPhone, um laptop ou um tablet Android.

## **Acessibilidade na Prática**

A acessibilidade é um dos focos principais do W3C. Isso significa que, ao desenvolver uma página, você precisa garantir que ela seja utilizável por todas as pessoas, incluindo aquelas com deficiências. Por exemplo, ao incluir descrições alternativas (alt) para imagens, você permite que usuários com deficiência visual possam entender o conteúdo através de leitores de tela.

Suponha que você está criando uma página de notícias. Ao seguir as diretrizes do W3C, você garante que cada imagem tenha uma descrição que explique seu conteúdo. Assim, se alguém com deficiência visual acessar a sua página, o leitor de tela descreverá a imagem com base no

texto alternativo que você forneceu. Isso torna a web mais inclusiva e acessível a todos.



## **Testando a Interoperabilidade**


Outro ponto crucial que o W3C garante é a **interoperabilidade**, ou seja, a capacidade do seu site de funcionar bem em diferentes navegadores e dispositivos. Para garantir que o seu site siga esses padrões, você deve testá-lo em diferentes navegadores, como Chrome, Firefox, Safari e Edge.

Por exemplo, ao finalizar o desenvolvimento de um formulário de contato, você deve abrir o site em vários navegadores para verificar se o layout permanece consistente e se todas as funcionalidades, como a validação dos campos, funcionam corretamente. Se você seguiu os padrões do W3C, o comportamento do site será praticamente o mesmo em todos os navegadores, garantindo uma experiência de usuário consistente.

## **Motores de renderização**

No desenvolvimento de software para a web, um dos elementos mais críticos para garantir que o conteúdo que você cria seja exibido corretamente no navegador é o **motor de renderização**. Esses motores são responsáveis por pegar o código que você escreve—HTML, CSS e JavaScript—e transformá-lo em uma página web que os usuários possam ver e interagir.

## **O Que São Motores de Renderização?**


Os motores de renderização, também conhecidos como rendering engines ou motores de layout, são componentes essenciais dos navegadores. E  são os responsáveis por processar o código-fonte das páginas web e renderizá-lo na tela do usuário. Cada navegador tem seu próprio motor de renderização: o Google Chrome usa o **Blink**, o Firefox usa o **Gecko**, e o Safari usa o **WebKit**.

Para entender melhor, pense no motor de renderização como um chef de cozinha. Esse chef recebe os ingredientes—neste caso, o HTML, CSS e JavaScript—e, a partir deles, prepara um prato final, que é a página web que aparece no navegador. Se o HTML define a estrutura da página, o CSS define o estilo, e o JavaScript adiciona interatividade, o motor de renderização combina todos esses elementos para criar uma experiência visual e funcional completa para o usuário.

### **Como Funciona o Processo de Renderização?**

Vamos a um exemplo prático: imagine que você está desenvolvendo uma página de portfólio pessoal. Você começa criando a estrutura básica com HTML, definindo seções para o seu nome, uma lista de projetos e uma foto de perfil. Em seguida, você estiliza a página com CSS, definindo cores, fontes e o layout da página. Por fim, você adiciona interatividade com JavaScript, como animações ao passar o mouse sobre um projeto ou uma barra de progresso que mostra suas habilidades.

Quando você abre essa página no navegador, o motor de renderização entra em ação. Primeiro, ele interpreta o HTML para entender a estrutura da página. Depois, aplica o CSS para estilizar os elementos conforme definido. Finalmente, processa o JavaScript para adicionar a interatividade e as funcionalidades dinâmicas.

Por exemplo, ao definir uma cor de fundo e uma fonte no CSS, o motor de renderização garante que essas escolhas sejam aplicadas corretamente a todos os elementos da página. Se você adiciona uma animação em JavaScript que faz com que uma imagem mude de tamanho quando o usuário passa o mouse sobre ela, o motor de renderização executa essa função e apresenta a animação no navegador em tempo real. 

## **Importância dos Motores de Renderização**

Sem os motores de renderização, o código que você escreve seria apenas texto, e as páginas web não poderiam ser exibidas visualmente. Eles são essenciais para que o navegador possa interpretar o código de forma correta e transformar as instruções em um layout visual que o usuário possa ver e usar.

Outro ponto importante é que diferentes navegadores podem interpretar o código de maneiras ligeiramente diferentes devido aos seus respectivos motores de renderização. Por isso, é fundamental testar seu código em vários navegadores para garantir que a experiência do usuário seja consistente em todos eles.

## **Exemplo Prático: Desenvolvendo e Testando uma Galeria de Fotos**

Para entender como os motores de renderização funcionam na prática, vamos desenvolver uma página simples que exibe uma galeria de fotos com legendas e interatividade.

### **Passo 1: Criando a Estrutura HTML**

Primeiro, vamos criar a estrutura básica da galeria usando HTML. Aqui, definimos uma grade com várias imagens e suas respectivas legendas.



```
<!DOCTYPE html>
```

```
<html lang="pt-BR">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>Galeria de Fotos</title>
```

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

```
</head>
```

```
<body>
```

```
  <div class="galeria">
```

```
    <figure>
```

```
      
```

```
      <figcaption>Foto 1</figcaption>
```

```
    </figure>
```

```
    <figure>
```

```
      
```

```
      <figcaption>Foto 2</figcaption>
```

```
    </figure>
```



<!-- Mais figuras -->



</div>

<script src="scripts.js"></script>

</body>

</html>

Neste código, usamos a tag <figure> para agrupar cada imagem (<img>) com sua legenda (<figcaption>). Essa estrutura semântica é importante para acessibilidade e organização do conteúdo.

## **Passo 2:** Estilizando com CSS

Agora, aplicamos o CSS para criar um layout em grade e estilizar a galeria.

```
body {
```

```
    font-family: Arial, sans-serif;
```

```
    background-color: #f4f4f4;
```

```
    margin: 0;
```

```
    padding: 20px;
```

```
}
```

```
.galeria {
```

```
    display: grid;
```

```
grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
```



```
gap: 10px;
```

```
}
```

```
figure {
```

```
background-color: white;
```

```
padding: 10px;
```

```
border-radius: 8px;
```

```
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
```

```
transition: transform 0.3s;
```

```
}
```

```
figure:hover {
```

```
transform: scale(1.05);
```

```
}
```

```
img {
```

```
width: 100%;
```

```
border-radius: 5px;
```

```
}
```

Aqui, definimos uma grade com colunas automáticas usando `grid-template-columns`, para que as imagens se ajustem ao tamanho da tela. Também

adicionamos um efeito de hover em cada figura, para que ela aumente ligeiramente quando o usuário passa o mouse.



### **Passo 3:** Adicionando Interatividade com JavaScript

Por fim, adicionamos um script JavaScript para permitir que o usuário clique em uma foto para ampliá-la.

```
document.querySelectorAll('.galeria img').forEach(image => {  
  
  image.addEventListener('click', function() {  
  
    const src = this.src;  
  
    const modal = document.createElement('div');  
  
    modal.classList.add('modal');  
  
    modal.innerHTML = `<span class="fechar">&times;</span>`;  
  
    document.body.appendChild(modal);  
  
    modal.querySelector('.fechar').addEventListener('click', function() {  
  
      modal.remove();  
  
    });  
  
  });  
  
});
```



Este código adiciona um evento de clique a cada imagem da galeria. Quando o usuário clica em uma imagem, um modal é criado e exibido com a imagem ampliada. Ao clicar no botão “fechar”, o modal é removido.

## **Testando em Diferentes Navegadores**

Agora, é hora de ver como a página funciona em diferentes navegadores. Primeiro, você abre a página no Google Chrome, onde o motor Blink processa o HTML, aplica o CSS e executa o JavaScript para exibir a galeria de fotos com todas as funcionalidades. Em seguida, você testa no Firefox (Gecko) e no Safari (WebKit) para verificar se o layout e a interatividade são consistentes.

Durante o teste, pode ser que você note pequenas diferenças na forma como a animação de hover ou o modal são exibidos entre os navegadores. Isso é normal devido às variações entre os motores de renderização. Se necessário, você pode ajustar o CSS ou o JavaScript para garantir uma experiência uniforme em todos os navegadores.

## **IDEs, Plugins e outros**

No desenvolvimento de software, especialmente para a web, o ambiente de desenvolvimento desempenha um papel crucial. Escolher as ferramentas certas pode fazer toda a diferença na sua produtividade e na qualidade do código que você entrega. Nesta seção, vamos explorar como utilizar uma IDE (Ambiente de Desenvolvimento Integrado) e plugins para otimizar seu trabalho como desenvolvedor.

## O Papel das IDEs



Uma **IDE** (Integrated Development Environment) é uma ferramenta que reúne em um único local todos os recursos necessários para escrever, testar e depurar seu código. O **Visual Studio Code (VS Code)** é uma das IDEs mais populares entre os desenvolvedores, devido à sua simplicidade, eficiência e extensa biblioteca de plugins que podem ser instalados para estender suas funcionalidades.

Imagine que você está trabalhando em um projeto web. No VS Code, você pode abrir múltiplos arquivos ao mesmo tempo, escrever código em diferentes linguagens (como HTML, CSS e JavaScript), e usar atalhos para formatar o código automaticamente. Além disso, a IDE oferece recursos como destaque de sintaxe, autocompletar, e depuração, que ajudam a evitar erros e a escrever código mais rapidamente.

### Utilizando Plugins para Aumentar a Produtividade

Uma das grandes vantagens do VS Code é a possibilidade de instalar plugins que aumentam a produtividade. Vamos focar em dois plugins essenciais para o desenvolvimento web: **Live Server** e **Auto Rename Tag**.

**Live Server:** Este plugin cria um servidor local na sua máquina e atualiza automaticamente a página no navegador toda vez que você salva uma alteração no código. Imagine que você está desenvolvendo uma landing page e precisa ajustar constantemente o layout e o conteúdo. Com o Live Server, você não precisa ficar atualizando manualmente o navegador cada vez que faz uma mudança. Por exemplo, ao alterar o texto em uma tag `<h1>` e salvar o arquivo, o Live Server imediatamente reflete essa alteração no

navegador. Isso não só economiza tempo, mas também torna o processo de desenvolvimento mais fluido.



**Auto Rename Tag:** Ao trabalhar com HTML, você frequentemente precisa renomear tags de abertura e fechamento. O Auto Rename Tag faz isso automaticamente para você. Suponha que você tenha várias tags `<div>` e decida que uma delas deve ser alterada para `<section>`. Com esse plugin, ao modificar a tag de abertura, a tag de fechamento é atualizada automaticamente. Isso é especialmente útil em projetos maiores, onde pode ser fácil perder de vista quais tags precisam ser alteradas.

## **Exemplo Prático: Configurando e Usando Plugins no VS Code**

Vamos explorar como configurar e usar os plugins Live Server e Auto Rename Tag no VS Code, com uma ilustração prática de cada passo.


### **Passo 1:** Instalação do VS Code

Primeiro, você precisa instalar o Visual Studio Code. Acesse o site oficial ([code.visualstudio.com](https://code.visualstudio.com)), escolha a versão correspondente ao seu sistema operacional (Windows, macOS ou Linux), e siga as instruções para instalação. Assim que o VS Code estiver instalado, você pode começar a configurá-lo com os plugins necessários.

### **Passo 2:** Instalando o Live Server

**Abrindo a Seção de Extensões:** No VS Code, localize o ícone de extensões na barra lateral esquerda. Ele se parece com um quadrado com um canto destacado. Clique nele para abrir a seção de extensões.

**Procurando o Live Server:** Na barra de pesquisa que aparece no topo da seção de extensões, digite “Live Server” e pressione Enter. O primeiro resultado geralmente será o plugin correto. Clique em “Install” para adicionar o Live Server ao seu ambiente de desenvolvimento.

Usando o Live Server: Após a instalação, você verá a opção “Go Live” no canto inferior direito do VS Code. Ao clicar nessa opção, o Live Ser  iniciará um servidor local e abrirá seu arquivo HTML no navegador. Agora, toda vez que você salvar uma alteração no código, a página será atualizada automaticamente no navegador.

**Exemplo:** Imagine que você está desenvolvendo uma página de boas-vindas com um título `<h1>`. Você decide alterar o texto de “Bem-vindo ao Meu Site” para “Bem-vindo ao Meu Portfólio”. Ao salvar a alteração no VS Code, o Live Server automaticamente reflete essa mudança no navegador, sem que você precise recarregar a página manualmente.

### **Passo 3:** Instalando o Auto Rename Tag

Procurando o Auto Rename Tag: Da mesma forma, na seção de extensões, pesquise por “Auto Rename Tag”. Clique em “Install” para adicionar essa extensão ao VS Code.

Usando o Auto Rename Tag: Com o plugin instalado, abra um arquivo HTML e edite uma tag. Ao modificar a tag de abertura, o Auto Rename Tag automaticamente atualiza a tag de fechamento correspondente.

Exemplo: Suponha que você tenha uma tag `<div>` que precisa ser alterada para `<section>`. Ao mudar a tag de abertura para `<section>`, o Auto Rename Tag automaticamente ajusta a tag de fechamento, economizando tempo e evitando possíveis erros, especialmente em documentos HTML mais complexos.

## **Resultado**

Ao seguir esses passos, você terá configurado um ambiente de desenvolvimento mais eficiente e prático. O **Live Server** agiliza o processo de visualização de mudanças em tempo real, enquanto o **Auto Rename Tag**

facilita a manipulação de tags HTML, minimizando o risco de erros e economizando tempo.



## Conteúdo Bônus

Para aprofundar o conhecimento sobre a configuração do ambiente de desenvolvimento com o Visual Studio Code e suas extensões, recomendo o seguinte vídeo disponível gratuitamente na plataforma YouTube:

**Título:** Configurando o Visual Studio Code | Configurações iniciais + Extensões

**Canal:** Manual do Dev

**Plataforma:** YouTube

**Descrição:** Nesse vídeo, o canal **Manual do Dev** apresenta as configurações iniciais recomendadas para o Visual Studio Code, além das extensões essenciais para começar a usar a IDE de forma eficiente. Este conteúdo é altamente relevante para complementar a nossa aula, fornecendo um guia prático para otimizar seu ambiente de desenvolvimento.

Para assistir, basta buscar pelo título do vídeo diretamente no YouTube.

## Referência Bibliográfica

ASCÊNCIO, Ana Fernanda Gomes; CAMPOS, Edilene Aparecida Veneruchi de. **Fundamentos da programação de computadores**. 3. ed. São Paulo: Pearson, 2012.

BRAGA, Paulo Henrique. **Teste de software**. São Paulo: Pearson, 2016.



GALLOTTI, Guilherme Moreira Alves (Org.). **Arquitetura de software**. São Paulo: Pearson, 2017.



GALLOTTI, Guilherme Moreira Alves. **Qualidade de software**. São Paulo: Pearson, 2015.

MEDEIROS, Edson. **Desenvolvendo software com UML 2.0 definitivo**. São Paulo: Pearson, 2004.

PFLEEGER, Shari Lawrence. **Engenharia de software: teoria e prática**. 2. ed. São Paulo: Pearson, 2003.

SOMMERVILLE, Ian. **Engenharia de software**. 10. ed. São Paulo: Pearson, 2019.

SOMMERVILLE, I. **Engenharia de software**. 10.ed. Pearson: 2019.

## **Prática Integradora Desenvolvimento de Software**

### **Atividade Prática 1 – Criação e Estilização de uma Página Web Simples com HTML, CSS e JavaScript**

#### **Objetivos:**

- Compreender a estrutura básica de uma página web utilizando HTML.
- Aplicar estilos visuais com CSS para melhorar a apresentação da página.
- Adicionar interatividade à página com JavaScript.
- Utilizar ferramentas de desenvolvimento, como Visual Studio Code e Google Chrome, para criar, testar e depurar o código.

## **Materiais, Métodos e Ferramentas:**



- Computador com acesso à internet.
- Visual Studio Code (VS Code) instalado.
- Navegador Google Chrome (ou outro navegador para testes).
- Editor de texto (VS Code).
- Conhecimento básico em HTML, CSS e JavaScript.

## **Atividade Prática**

Primeiramente, leia atentamente o texto a seguir:

Nesta atividade, você criará uma página web simples que utiliza HTML para a estrutura, CSS para o estilo e JavaScript para adicionar interatividade. Você também usará o Visual Studio Code para editar o código e o Google Chrome para testar e depurar a página. Abaixo estão os passos detalhados que você deve seguir.

Agora, vamos praticar!

## **PASSO A PASSO DETALHADO DA ATIVIDADE**

### **1. Criação da Estrutura HTML**

- Abra o Visual Studio Code e crie um novo projeto.
- No projeto, crie um arquivo chamado index.html.
- A estrutura HTML deve conter um cabeçalho, um parágrafo e um botão com o ID mensagemBotao.



## **2. Estilização com CSS**

- No mesmo projeto, crie um arquivo chamado style.css.
- O CSS deve aplicar estilos para a página, incluindo um cabeçalho escuro, corpo com fundo branco e um botão estilizado que muda de cor ao passar o cursor.

## **3. Adicionar Interatividade com JavaScript**

- Ainda no projeto, crie um arquivo chamado script.js.
- O JavaScript deve adicionar um evento de clique ao botão que exibe um alerta com a mensagem “Você clicou no botão!”.

## **4. Testando no Google Chrome**

- Salve todos os arquivos.
- No Visual Studio Code, clique no ícone “Go Live” no canto inferior direito para abrir o index.html no Google Chrome.
- Verifique se a página está exibindo o conteúdo conforme o esperado e se o botão exibe a mensagem ao ser clicado.

## **5. Usando as Ferramentas de Desenvolvedor**

- No Google Chrome, abra as ferramentas de desenvolvedor (pressionando F12 ou Ctrl+Shift+I).
- Inspecione os elementos HTML, ajuste o CSS e verifique o funcionamento do JavaScript.
- Faça os ajustes necessários no código e salve as alterações.

- O teste no navegador deve mostrar a página corretamente estilizada e a interatividade do botão funcionando como esperado.



**Gabarito esperado para corrigir a atividade prática:**

```
<!DOCTYPE html>
```

```
<html lang="pt-BR">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Página Exemplo</title>
```

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

```
</head>
```

```
<body>
```

```
<header>
```

```
<h1>Bem-vindo à Página Exemplo</h1>
```

```
</header>
```

```
<main>
```

```
<p>Este é um exemplo prático de como utilizar HTML, CSS e  
JavaScript em uma página web.</p>
```

```
<button id="mensagemBotao">Clique aqui</button>
```



```
</main>
```

```
<script src="script.js"></script>
```

```
</body>
```

```
</html>
```

```
body {
```

```
    font-family: Arial, sans-serif;
```

```
    margin: 0;
```

```
    padding: 0;
```

```
    background-color: #f4f4f4;
```

```
}
```

```
header {
```

```
    background-color: #333;
```

```
    color: white;
```

```
    padding: 10px 0;
```

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

```
}
```

```
main {
```

```
    margin: 20px;
```

padding: 20px;



background-color: white;

border-radius: 8px;

}

button {

background-color: #007bff;

color: white;

padding: 10px 20px;

border: none;

border-radius: 5px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

document.getElementById('mensagemBotao').addEventListener('click',  
function() {

alert('Você clicou no botão!');

});

**Ir para exercício**

