



# INTENSIVÃO DE JAVASCRIPT

## Apostila Completa Aula 1

**Guia passo a passo: Projeto Aplicativo de Audiobook**



Parte 1

# Instalando as ferramentas



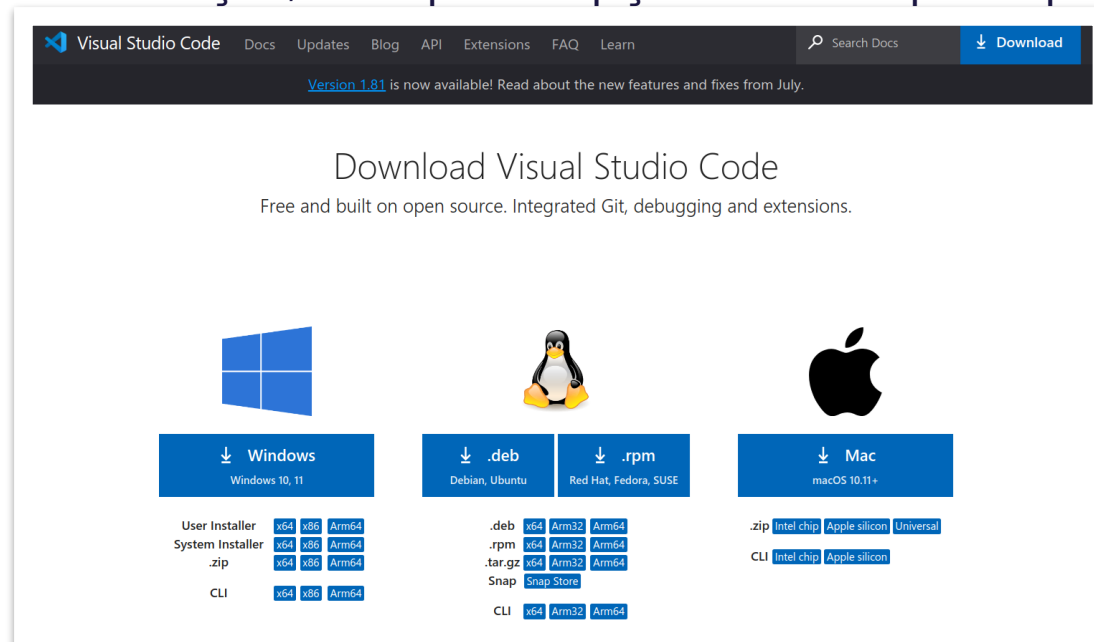


# Instalando as ferramentas – VS CODE

Caso você ainda não tenha o Visual Studio Code instalado, basta seguir os procedimentos abaixo. A instalação do VS Code é totalmente gratuita, e você pode usá-lo em seu computador sem precisar pagar nada. O link para fazer o download do programa é mostrado abaixo:

<https://code.visualstudio.com/>

- 1 – Baixe o arquivo de instalação correspondente ao seu sistema operacional (Windows, MacOS ou Linux).
- 2 – Execute o arquivo de instalação e siga as instruções na tela. Em geral, é só continuar clicando em "Próximo".
- 3 – No Windows, durante a instalação, marque a opção "Add to path" para adicionar o VS Code às suas variáveis de ambiente.



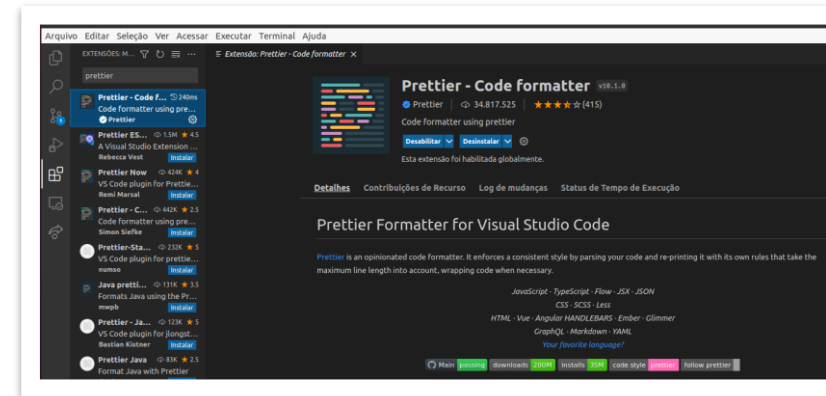


# Instalando as ferramentas – VS CODE - EXTENSÕES

As extensões no Visual Studio Code são pequenos programas que adicionam recursos extras ao editor de código. Elas são como "plugins" que podem ser instalados para personalizar e estender as funcionalidades do VS Code.

A extensão Prettier adiciona a capacidade de formatar automaticamente o código, tornando-o mais legível e organizado.

Para instalar a extensão Prettier no Visual Studio Code:



- Clique no ícone de extensões no menu lateral esquerdo (ou use o atalho "Ctrl+Shift+X").
- Na barra de pesquisa, digite "Prettier".
- A extensão "Prettier - Code Formatter" deve aparecer nos resultados da pesquisa. Clique no botão "Instalar" ao lado dela.
- Após a instalação, você pode configurar o Prettier como o formatador padrão para o seu projeto. Para fazer isso, abra as configurações do Visual Studio Code (menu "File" > "Preferences" > "Settings" ou use o atalho "Ctrl+,").
- Na barra de pesquisa das configurações, digite "Default Formatter" e selecione a opção "Editor: Default Formatter".
- No campo de valor, digite "esbenp.prettier-vscode" e pressione Enter para salvar as alterações.
- Agora, o Prettier está instalado e configurado como o formatador padrão no Visual Studio Code. Você pode usar o comando "Format Document" (menu "Edit" > "Format Document" ou atalho "Shift+Alt+F") para formatar o código.



# Instalando as ferramentas – VS CODE - EXTENSÕES

Já a extensão ES6 String HTML adiciona suporte para destacar a sintaxe do HTML dentro de strings de várias linhas no código JavaScript.

Para instalar a extensão ES6 String HTML no Visual Studio Code:

- Clique no ícone de extensões no menu lateral esquerdo (ou use o atalho "Ctrl+Shift+X").
- Na barra de pesquisa, digite "ES6 String HTML".
- A extensão "ES6 String HTML" deve aparecer nos resultados da pesquisa. Clique no botão "Instalar" ao lado dela.
- Após a instalação, a extensão irá adicionar suporte de realce de sintaxe para código HTML dentro de strings de várias linhas no ES6.



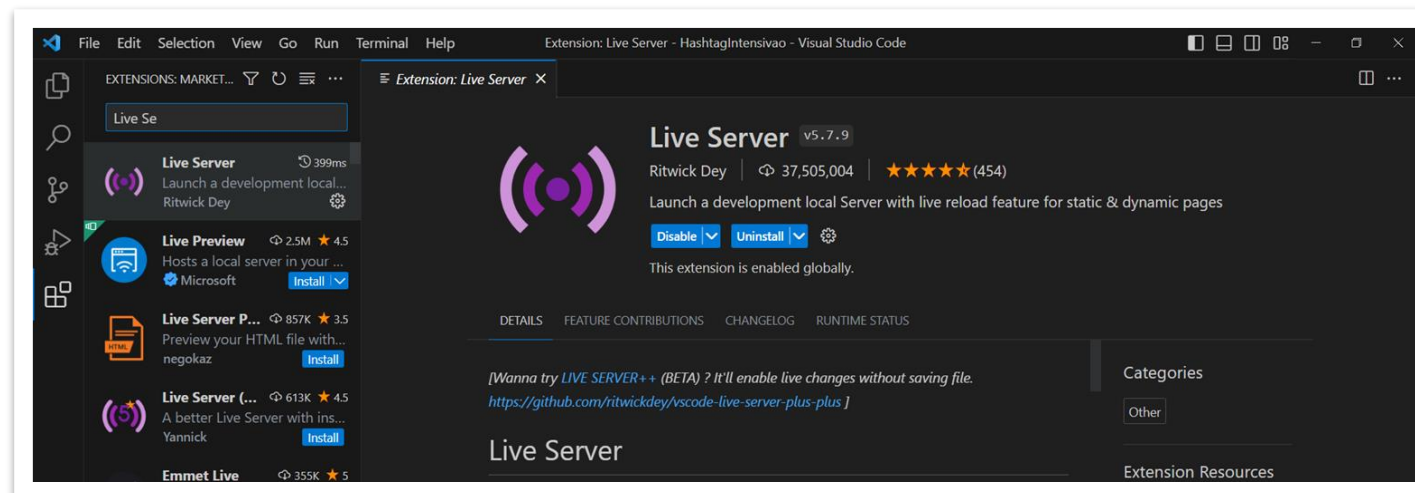


# Instalando as ferramentas – VS CODE - EXTENSÕES

O Live Server é uma extensão muito útil para desenvolvimento web, pois facilita a visualização e a atualização instantânea das alterações feitas no código HTML.

Vou te explicar como baixar a extensão "Live Server" no VS Code:

- Abra o Visual Studio Code (VS Code) e vá para a seção de extensões.
- Pesquise por "Live Server" e clique em "Instalar" ao lado da extensão desenvolvida por Ritwick Dey.
- Aguarde a instalação e clique em "Gerenciar" para acessar as configurações da extensão.
- Personalize as configurações, se desejar.
- Abra um arquivo HTML no VS Code, clique com o botão direito do mouse e selecione "Abrir com Live Server".
- O Live Server iniciará um servidor local e abrirá o arquivo HTML no navegador padrão.





# Instalando as ferramentas – Node.js

O Node.js é um ambiente de execução de código JavaScript do lado do servidor. Ele permite que você execute código JavaScript fora do navegador, o que significa que você pode criar aplicativos de servidor, scripts de linha de comando e muito mais usando JavaScript. Para instalar o Node.js, você pode seguir os seguintes passos:



- Acesse o site oficial do Node.js em <https://nodejs.org>.
- Na página inicial, você verá duas versões para download: LTS (Long Term Support) e Current. A versão LTS é recomendada para a maioria dos usuários, pois é mais estável e possui suporte a longo prazo. Selecione a versão LTS ou a versão mais recente, se preferir.
- Após selecionar a versão desejada, você será redirecionado para a página de download. Escolha o instalador adequado para o seu sistema operacional (Windows, macOS ou Linux) e clique no link para iniciar o download.
- Após o download ser concluído, execute o instalador e siga as instruções na tela para concluir a instalação.
- Após a instalação ser concluída, você pode verificar se o Node.js foi instalado corretamente abrindo o terminal ou prompt de comando e digitando o comando `node -v`. Se a versão do Node.js for exibida, significa que a instalação foi bem-sucedida.

Parte 2

# Apresentação do Projeto Aplicativo de Audiobook!



# Apresentação do Projeto Aplicativo de Audiobook!

Bem-vindos ao nosso projeto de **Aplicativo Audiobook**, onde exploraremos a implementação de uma plataforma interativa para desfrutar da obra-prima literária "Dom Casmurro" em formato de áudio. Alinhado à proposta de sites de audiobook, nosso projeto oferecerá aos usuários acesso a áudios previamente preparados. Inspirados pelo conceito de plataformas de audiobook online, nosso objetivo é proporcionar uma experiência envolvente, permitindo que os usuários explorem uma diversificada seleção de áudios, desfrutando das narrativas sem a necessidade de leitura convencional. Com o uso do JavaScript, destacaremos como a programação web pode criar uma plataforma dinâmica e acessível, facilitando a imersão na riqueza da obra de Machado de Assis.

Dentro deste PDF, você encontrará um guia apresentando o passo a passo para criar o projeto completo. O conteúdo está incrível e acredito que você vai curtir cada detalhe!



Parte 3

# Tríade - HTML, CSS E Javascript

# Tríade - HTML, CSS e Javascript

Bem-vindos à nossa aula sobre a criação de um audiobook do zero! Antes de mergulharmos no fascinante mundo da narração sonora, é crucial compreendermos a interação entre HTML, CSS e JavaScript – as ferramentas fundamentais que utilizaremos para dar vida ao nosso projeto. Essas três linguagens trabalham em conjunto para criar experiências web dinâmicas e envolventes.

- O **HTML (Hypertext Markup Language)** é a espinha dorsal do nosso conteúdo. Ele estrutura a informação, definindo os elementos que compõem a nossa página, como títulos, parágrafos, imagens e links. Imagine o HTML como o esqueleto do corpo, proporcionando a base necessária para a construção do conteúdo.
- O **CSS (Cascading Style Sheets)** entra em cena para adicionar estilo e estética à nossa página HTML. Enquanto o HTML fornece a estrutura, o CSS permite a personalização visual, aplicando cores, fontes, margens e outros estilos para criar uma experiência atraente e coesa. Analogamente, o CSS é como a roupa que vestimos sobre o esqueleto, tornando a aparência mais agradável e organizada.

# Tríade - HTML, CSS e Javascript

- **JavaScript** é a linguagem de programação que dá vida à nossa experiência de audiobook, tornando-a interativa e dinâmica. Essencial para manipular HTML e CSS, o JavaScript entra em cena para adicionar, remover ou modificar conteúdo, respondendo às ações do usuário de forma inteligente. Em nosso projeto, ele não apenas gerencia a interface, mas também é responsável pela reprodução de áudio. Ao responder a cliques em botões ou comandos específicos, o JavaScript se torna o maestro por trás da cortina, coordenando a experiência auditiva de forma envolvente.

Juntos, HTML, CSS e JavaScript formam a tríade essencial para a criação de páginas web interativas e atraentes. Ao longo desta aula, exploraremos como essas linguagens se complementam para dar vida ao nosso projeto de audiobook, proporcionando não apenas uma experiência auditiva memorável, mas também uma compreensão mais profunda de como essas ferramentas se entrelaçam para criar a magia da web moderna. Vamos começar a jornada da criação do nosso audiobook!



Parte 4

# 0 Javascript

# O Javascript

## O que é JavaScript?



JavaScript é uma linguagem de programação de alto nível, dinâmica e interpretada, que é usada principalmente para tornar as páginas da web interativas. Com JavaScript, os desenvolvedores podem controlar o comportamento de elementos da página da web, criar animações, processar e manipular dados, entre outras coisas. A linguagem é executada no navegador do cliente, o que significa que o código é executado no dispositivo do usuário, em vez do servidor web, tornando a interação do usuário com a página da web mais rápida e eficiente.

O JavaScript desempenha um papel crucial na dinâmica e interatividade do nosso projeto de audiobook. Em essência ao ser incorporada nas páginas web, permite aos desenvolvedores controlar e aprimorar a experiência do usuário de maneira personalizada e eficiente.

Originalmente concebido em 1995 por Brendan Eich, o JavaScript foi criado para trazer animações e alertas às páginas da web. Ao longo do tempo, sua popularidade cresceu, e com o suporte da Microsoft em 1996, tornou-se uma linguagem-chave na programação web.

# O Javascript

A evolução do JavaScript trouxe o ES6 (ECMAScript 2015), uma versão que introduziu recursos avançados, como declarações de variáveis mais flexíveis (let e const), classes, módulos e promessas. Esses aprimoramentos elevaram o JavaScript a uma linguagem mais poderosa e amigável.

Quando aplicamos o JavaScript ao nosso projeto de audiobook, ele se torna a força motriz por trás da interface interativa. Responsável por manipular HTML e CSS, o JavaScript permite a atualização dinâmica da interface do usuário. Ele opera no lado do cliente, garantindo uma resposta rápida e eficiente, sem depender constantemente do servidor.

No contexto específico do audiobook, o JavaScript assume o papel de maestro, controlando a reprodução de áudio, respondendo a interações do usuário, e adicionando camadas de dinamismo à experiência de audição. Com ele, podemos oferecer aos usuários a capacidade de controlar a reprodução, pular entre capítulos e interagir de maneira intuitiva com o conteúdo sonoro.

Em resumo, a implementação do JavaScript no nosso projeto de audiobook proporciona não apenas uma experiência interativa, mas também adiciona camadas de dinamismo e controle aos elementos sonoros, elevando a narrativa a uma experiência envolvente e personalizada para os usuários.



Parte 5

# Primeiros Passos



# Primeiros Passos


Para iniciar o nosso projeto, vamos criar uma nova pasta com o nome "**Intensivão**". Essa pasta será o local onde iremos armazenar todos os arquivos do nosso audiobook. Dentro dessa pasta, colocaremos os arquivos HTML, CSS e Javascript, além de alguns arquivos adicionais que serão utilizados, como os arquivos de áudio do livro "Dom Casmurro" e imagem .



Vou te explicar passo a passo como criar uma pasta nova no Windows:

- Primeiro, abra o "Explorador de Arquivos" clicando no ícone da pasta amarela na barra de tarefas ou pressionando a tecla "Windows" + "E" no teclado.
- Navegue até o local onde você deseja criar a nova pasta. Por exemplo, você pode escolher criar a pasta na área de trabalho ou dentro de outra pasta existente.
- Com o local selecionado, clique com o botão direito do mouse em um espaço vazio da janela do Explorador de Arquivos. Um menu de opções será exibido.

# Primeiros Passos

- No menu de opções, passe o cursor sobre a opção "Novo" e, em seguida, clique em "Pasta". Uma nova pasta será criada com o nome "Nova pasta" destacado.
- Digite o nome desejado para a nova pasta. No nosso caso, digite "Intensivão" como o nome da pasta.
- Pressione a tecla "Enter" no teclado para confirmar o nome da pasta. A nova pasta será criada no local selecionado.



Nome	Status	Data de modificação	Tipo	Tamanho
 Intensivão		18/01/2024 18:33	Pasta de arquivos	

# Primeiros Passos

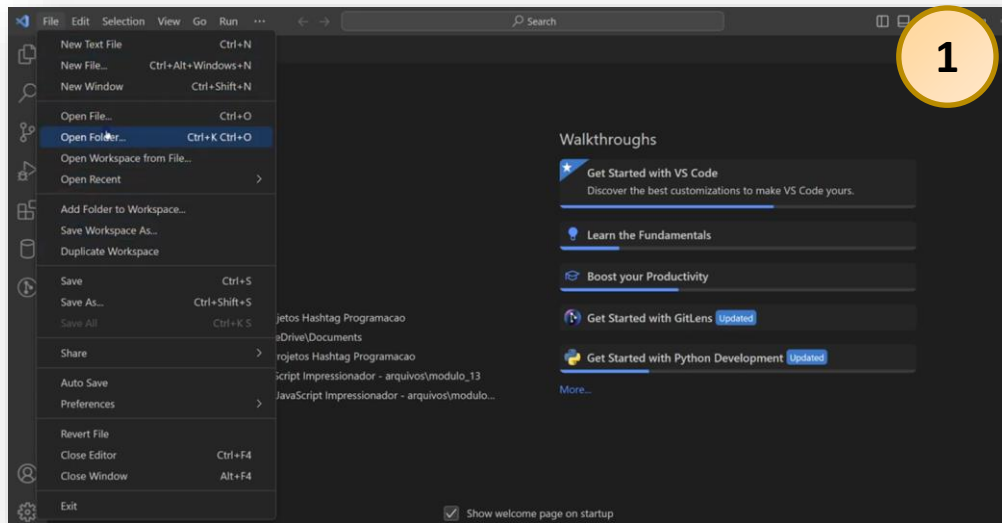
Agora vamos abrir a nossa pasta no VS Code, que será o programa que utilizaremos para realizarmos a construção do nosso audiobook. Para fazer isso, siga os passos abaixo:

- Certifique-se de ter o Visual Studio Code instalado no seu computador. Se ainda não tiver, você pode baixá-lo gratuitamente no site oficial do VS Code.
- Abra o Visual Studio Code clicando no ícone do programa na área de trabalho ou no menu Iniciar.
- No VS Code, clique em 'Arquivo' no menu superior e, em seguida, selecione 'Abrir Pasta'. Uma janela de seleção de pasta será exibida **(Imagem 1)**.
- Navegue até o local onde você criou a pasta 'Intensivão'. Por exemplo, se você a criou na área de trabalho, vá até a área de trabalho **(Imagem 2)**.
- Selecione a pasta 'Intensivão' e clique no botão 'Selecionar Pasta' na parte inferior direita da janela.
- A pasta 'Intensivão' será aberta no VS Code. No painel lateral esquerdo, você encontrará o local específico para criar os arquivos do seu projeto **(Imagem 3)**.

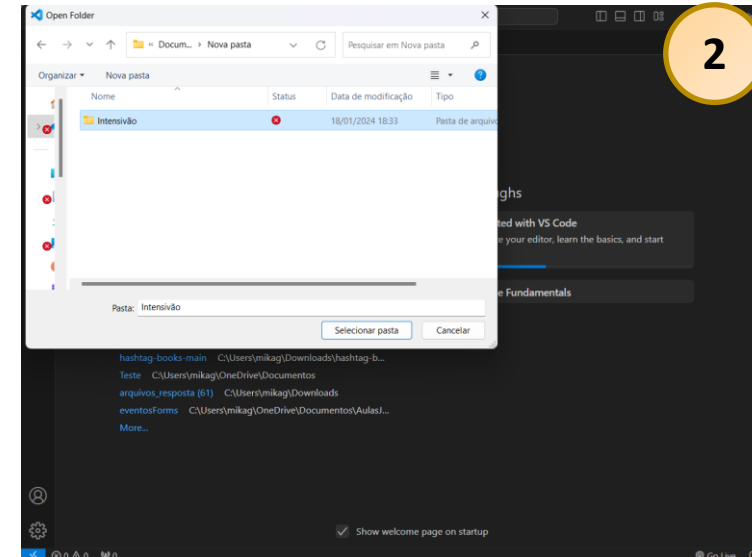
# Parte 5

## Primeiros Passos

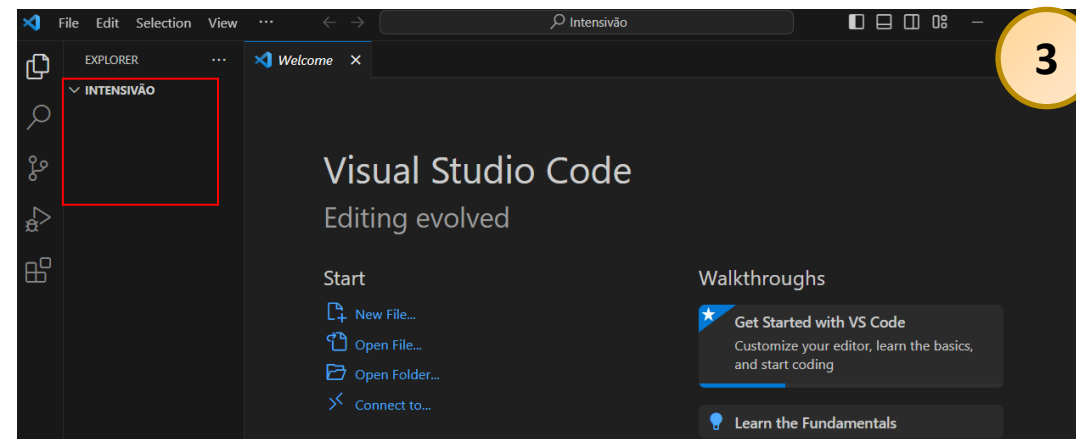
### - Abrir Folder/Pasta



### - Procurando pasta criada para projeto



### - Local dos Arquivos e Pastas do projeto



Parte 6

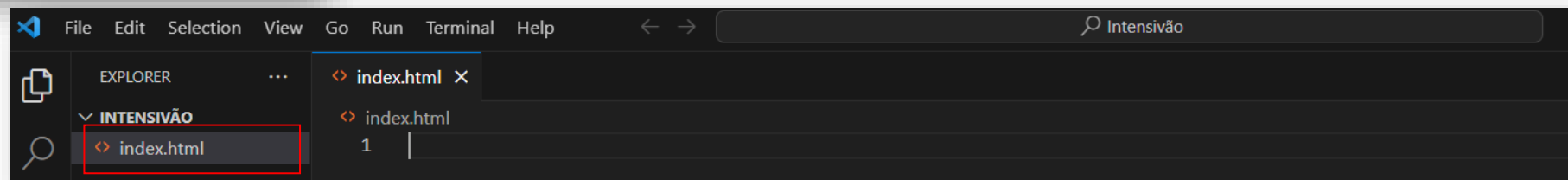
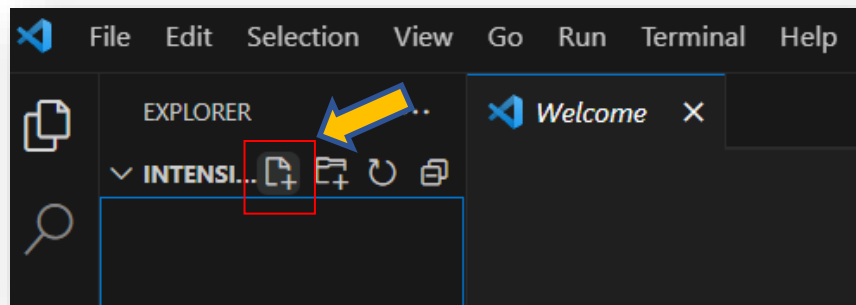
# Estruturas – HTML/CSS

# Estruturas – HTML / CSS

O arquivo "index.html" é um arquivo HTML que serve como a página inicial do seu site. Ele contém a estrutura básica do seu site, incluindo a marcação HTML, os estilos CSS e os scripts JavaScript. É a partir desse arquivo que você irá construir e desenvolver o conteúdo do seu site, adicionando elementos, estilos e funcionalidades.

Ao abrir o seu site no navegador, o arquivo "index.html" será o primeiro a ser carregado e exibido como a página inicial do seu site. É nele que você irá definir o layout, a estrutura e o conteúdo principal do seu site.

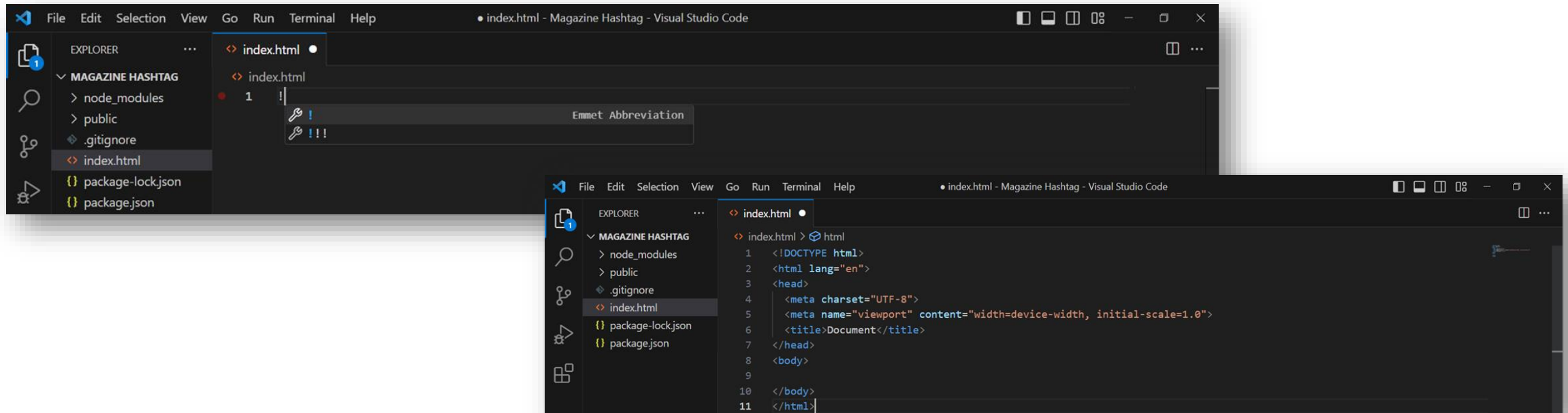
Vamos agora criar o arquivo "index.html". Para isso, clique no ícone "Novo Arquivo", escreva "index.html" e pronto! Dessa forma, você terá criado um novo arquivo.



# Estruturas – HTML / CSS

Agora você pode começar a editar o arquivo "index.html" no VS Code, adicionando o código HTML necessário para construir o audiobook.

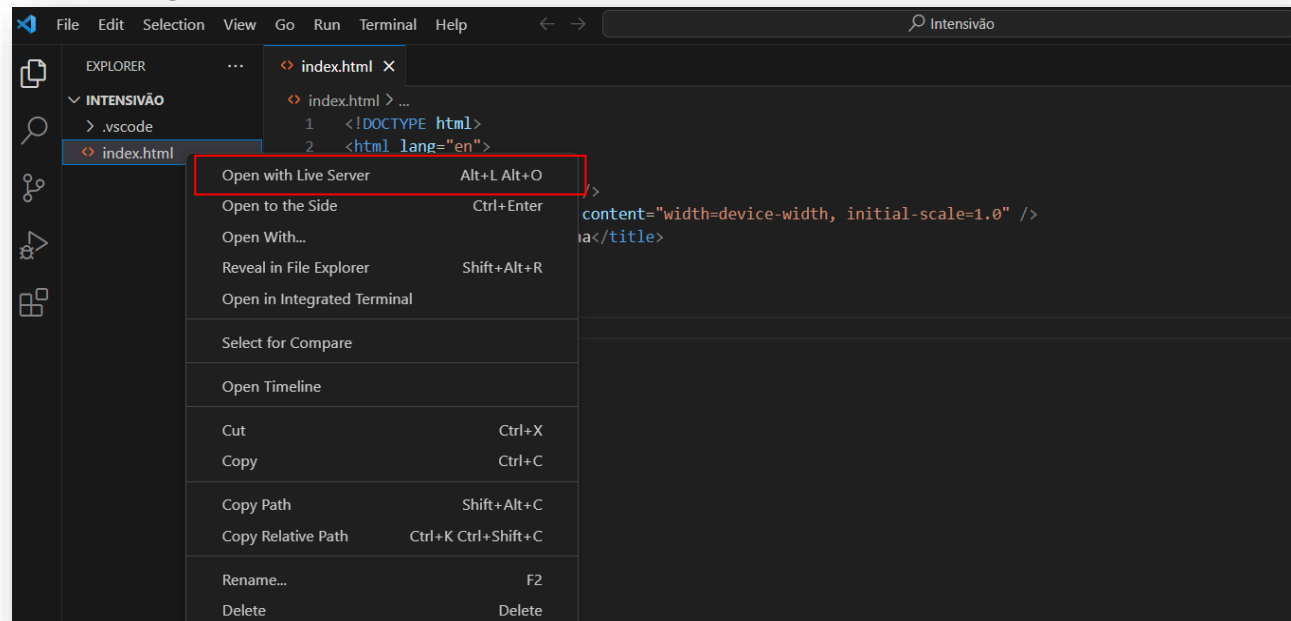
- Com o arquivo "index.html" aberto no VS Code, na linha em branco dentro do arquivo digite (!) + tab ou Enter.
- O VS Code irá expandir automaticamente o snippet ! para um modelo básico de arquivo HTML.
- O modelo básico de arquivo HTML incluirá a estrutura básica do HTML, incluindo as tags <html>, <head> e <body>.



# Estruturas – HTML / CSS

Para visualizar a sua primeira página web, você utilizará a extensão Live Server, uma ferramenta incrivelmente útil que torna o desenvolvimento web mais dinâmico e eficiente. Para aplicá-la a um arquivo **index.html**, basta seguir estes passos:

- Com o arquivo **index.html** aberto, clique com o botão direito no editor.
- Selecione a opção "Open with Live Server" no menu suspenso.
- O Live Server iniciará automaticamente, e o seu navegador padrão será aberto, exibindo a página **index.html**.
- Agora, sempre que você fizer alterações no arquivo **index.html** e salvar, o Live Server atualizará automaticamente o navegador.





# Estruturas - HTML / CSS

A estrutura básica do HTML gerada pelo VS Code inclui a declaração do tipo de documento (**<!DOCTYPE html>**), a **tag <html>** que envolve todo o conteúdo da página, a **tag <head>** que contém informações sobre a página, como o título exibido na aba do navegador (**tag <title>**), e a **tag <body>** onde você irá adicionar o conteúdo visível do seu site.

Agora você pode começar a adicionar o conteúdo do seu site dentro das **tags <body>...</body>**. Por exemplo, você pode adicionar cabeçalhos, parágrafos, imagens, links e outros elementos HTML.



Em HTML, uma **tag** é um elemento fundamental usado para estruturar e formatar o conteúdo de uma página da web. As **tags** são escritas **entre os símbolos < e >**, e podem conter atributos e/ou texto.

As **tags HTML** são essenciais para estruturar e organizar o conteúdo de uma página da web. Elas permitem que você crie seções, formate o texto, insira imagens, crie links e muito mais.

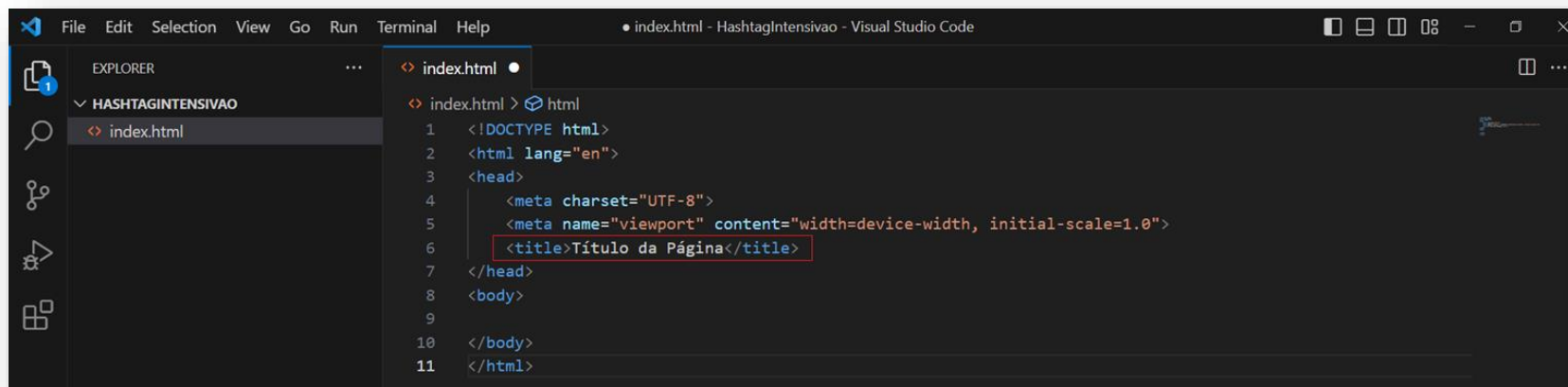
Ao escrever código HTML, é importante usar as **tags** corretas e fechá-las adequadamente para garantir que a página seja exibida corretamente nos navegadores.

Compreender e utilizar as **tags** corretamente é fundamental para criar páginas da web bem estruturadas e funcionais.

# Estruturas – HTML / CSS

A **tag <title>** é usada dentro da **seção <head>** do seu documento HTML para definir o título da página. O título é exibido na barra de título do navegador e também pode ser exibido em outros lugares, como nos resultados de pesquisa.

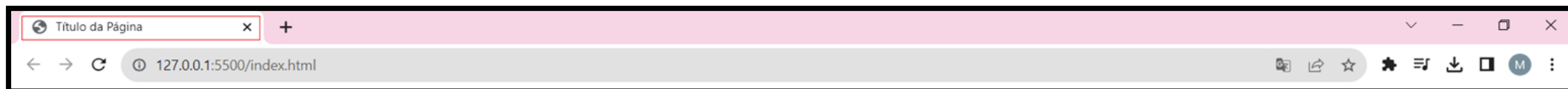
É importante notar que o título da página deve ser relevante e descritivo para que os usuários possam entender o conteúdo da página antes mesmo de abri-la. Além disso, os mecanismos de busca também levam em consideração o título da página ao exibir os resultados de pesquisa.



```
File Edit Selection View Go Run Terminal Help
index.html - HashtagIntensivao - Visual Studio Code

EXPLORER
HASHTAGINTENSIVAO
index.html

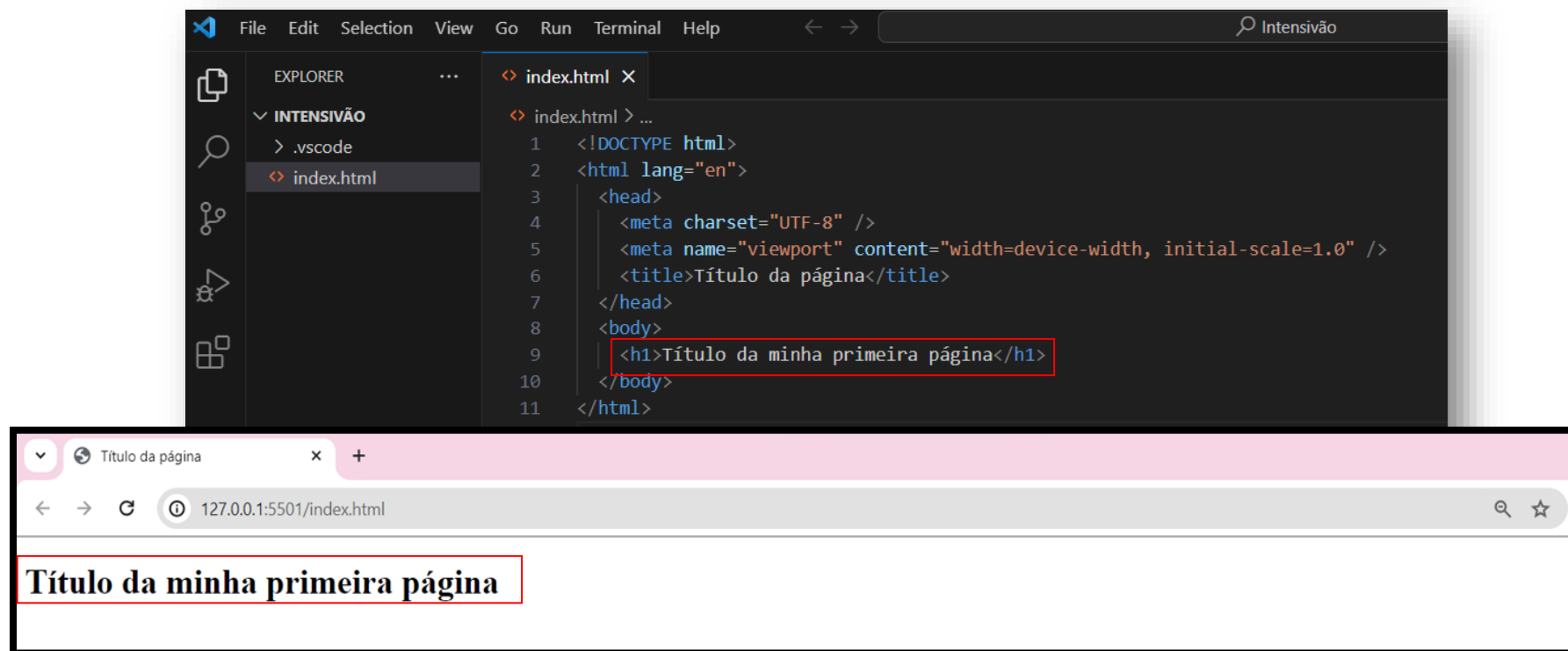
index.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Título da Página</title>
7 </head>
8 <body>
9
10 </body>
11 </html>
```



# Estruturas – HTML / CSS

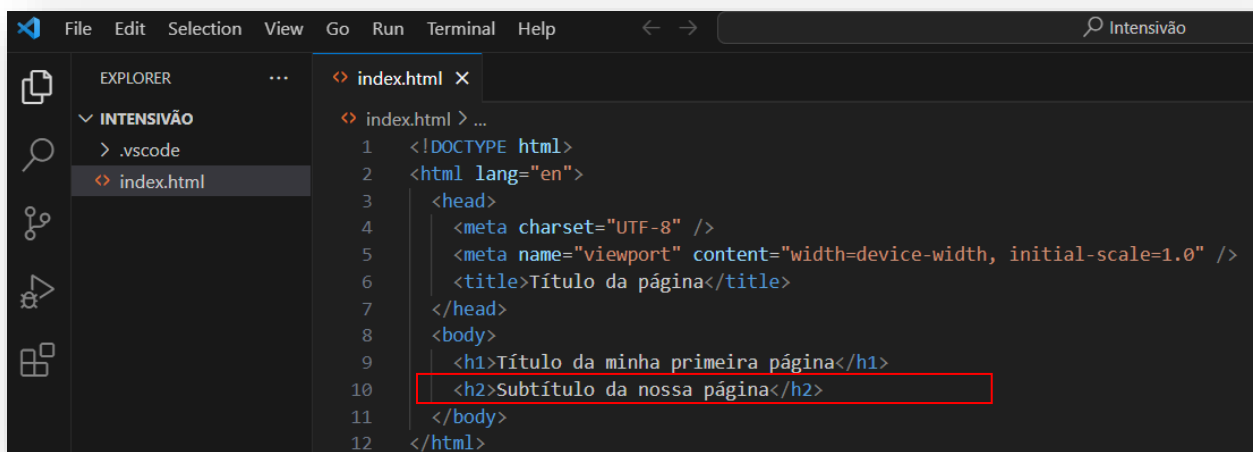
A tag **<h1>** é uma tag HTML que representa o título principal de uma seção ou página em um documento. O "h" em **<h1>** indica "header" (cabeçalho), e o número "1" indica o nível de importância do título. Portanto, **<h1>** é usado para o título mais importante e geralmente é o primeiro título em uma hierarquia de títulos HTML.

O objetivo principal da tag **<h1>** é destacar visualmente o texto dentro dela como o título mais significativo. O texto envolto pela tag **<h1>** geralmente é exibido com uma formatação maior e mais audaciosa do que o restante do conteúdo na página.



# Estruturas – HTML / CSS

Além do **<h1>**, há também tags **<h2>**, **<h3>**, **<h4>**, **<h5>**, e **<h6>**, cada uma representando títulos de importância decrescente. Essa hierarquia é valiosa para estruturar a página e indicar a relação entre diferentes seções.



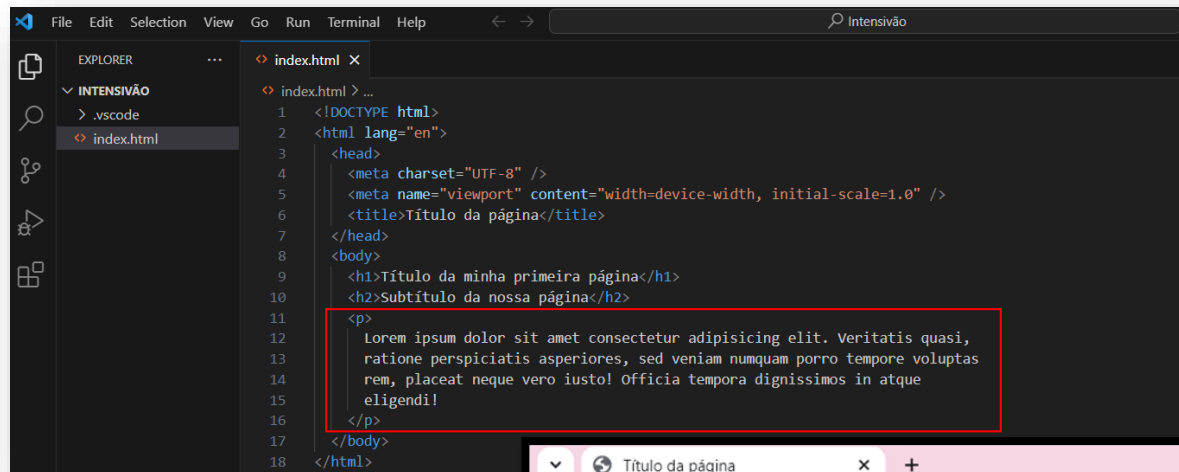
```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>Título da página</title>
7   </head>
8   <body>
9     <h1>Título da minha primeira página</h1>
10    <h2>Subtítulo da nossa página</h2>
11  </body>
12 </html>
```



# Estruturas – HTML / CSS

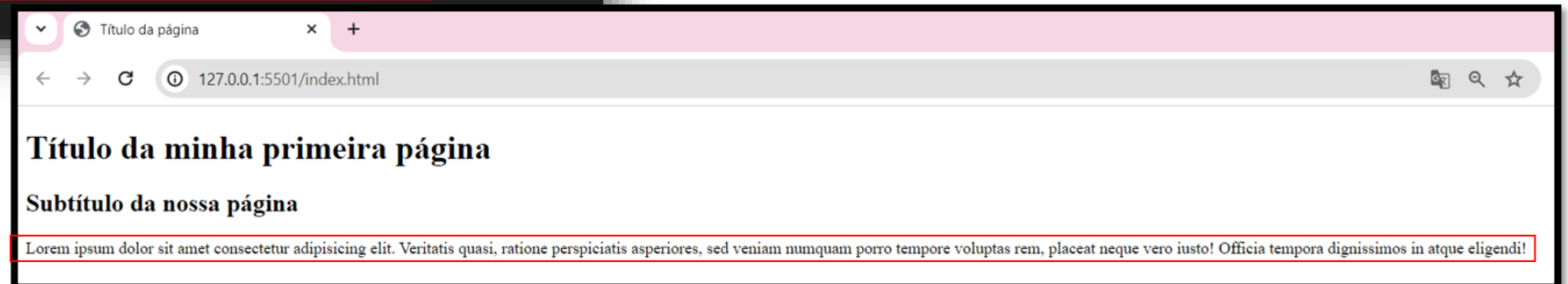
A **tag <p>** é uma tag HTML usada para definir um parágrafo de texto em um documento HTML. A **tag <p>** é usada para separar e estruturar o conteúdo de texto em parágrafos distintos.

A **tag <p>** é útil para organizar e estruturar o conteúdo de texto em um documento HTML. Ela também é importante para a acessibilidade, pois ajuda a tornar o conteúdo mais legível e compreensível para os usuários



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6   <title>Título da página</title>
7 </head>
8 <body>
9   <h1>Título da minha primeira página</h1>
10  <h2>Subtítulo da nossa página</h2>
11  <p>
12    Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis quasi,
13    ratione perspiciatis asperiores, sed veniam numquam porro tempore voluptas
14    rem, placeat neque vero iusto! Officia tempora dignissimos in atque
15    eligendi!
16  </p>
17 </body>
18 </html>
```

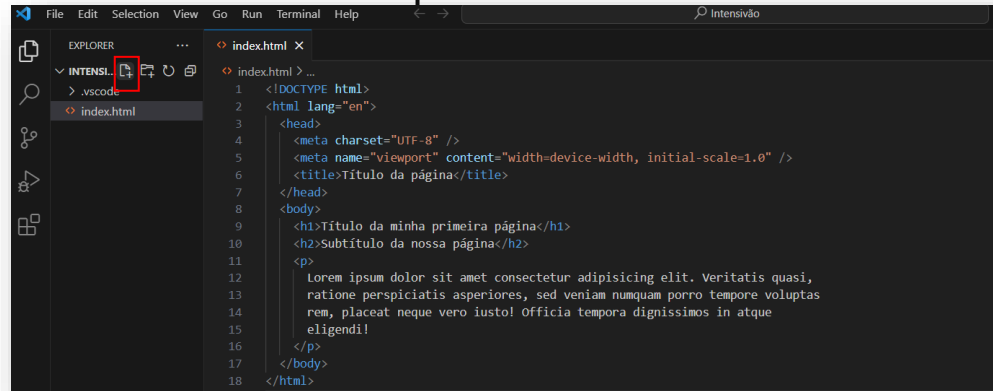
Se você escrever "lorem" dentro de uma tag <p>, estará utilizando um texto de preenchimento padrão conhecido como "Lorem Ipsum". Esse texto é frequentemente utilizado na indústria de design e tipografia para preencher espaços temporários e visualizar o layout de uma página antes que o conteúdo real seja inserido. O "Lorem Ipsum" não possui significado real em nenhum idioma conhecido, mas suas palavras e estrutura de frase se assemelham a um texto legível, proporcionando uma representação visual do layout do conteúdo.



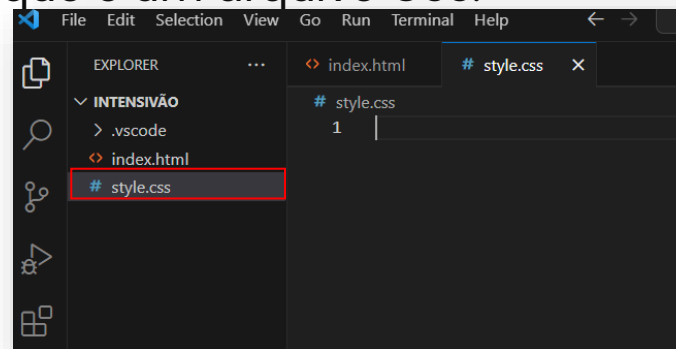
# Estruturas – HTML / CSS

Nesse momento iremos criar nosso arquivo **style.css**. O arquivo style.css é um arquivo que contém as regras de **estilo em CSS** (Cascading Style Sheets) para um documento HTML. Ele é usado para adicionar estilos e personalizar a aparência dos elementos HTML em uma página da web.

No VS Code clique no ícone adicionar novo arquivo:



- Crie um **arquivo style.css** na mesma pasta onde o documento HTML está localizado. Certifique-se de usar a extensão .css para indicar que é um arquivo CSS.



# Estruturas – HTML / CSS

Você pode adicionar as regras de estilo no arquivo style.css para estilizar os elementos HTML. Por exemplo, para definir a cor do texto do elemento de título (**<h1>**) como azul e o subtítulo (**h2**) como vermelho, você pode adicionar a seguinte regra CSS no arquivo style.css:

```
# style.css > ...
1  h1 {
2    color: blue;
3  }
4
5  h2 {
6    color: red;
7  }
```

No documento HTML, adicione a seguinte linha dentro da **tag <head>** para vincular o arquivo style.css ao documento HTML:

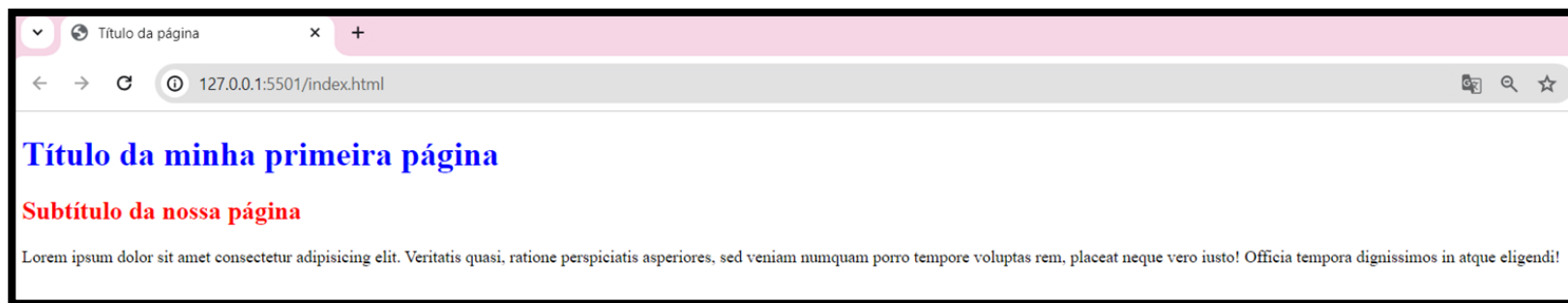
```
< index.html • # style.css
< index.html > html > head
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Título da página</title>
7    lin
8  </hea link
9  <body link:atom
10 <p> link:css Emmet Abbreviation
11 <di link:favicon
12 </bod link:im
13 </htm link:import
```

```
< index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8" />
5    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6    <title>Título da página</title>
7    <link rel="stylesheet" href="./style.css" />
8  </head>
9  <body>
10 <h1>Título da minha primeira página</h1>
11 <h2>Subtítulo da nossa página</h2>
12 <p>
13   Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis quasi,
14   ratione perspiciatis asperiores, sed veniam numquam porro tempore voluptas
15   rem, placeat neque vero iusto! Officia tempora dignissimos in atque
16   eligendi!
17 </p>
18 </body>
19 </html>
```

# Estruturas – HTML / CSS

Essa linha de código adiciona um link para o arquivo `style.css` na página HTML, permitindo que as regras de estilo sejam aplicadas. O **`href="/style.css"`** é um atributo da **tag <link>** que especifica o local do arquivo CSS que será vinculado ao documento HTML.

Depois de adicionar as regras de estilo no arquivo `style.css`, salve-o. As regras de estilo serão aplicadas automaticamente aos elementos HTML correspondentes quando a página for carregada no navegador.



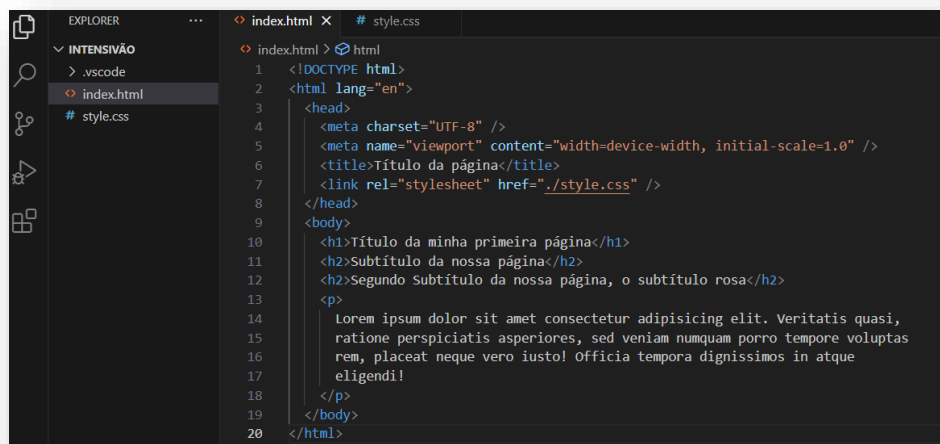
Ao criar conteúdo em HTML, você pode usar tags diversas vezes para representar diferentes elementos na sua página web. Essa prática é fundamental para estruturar e organizar o conteúdo de maneira clara e semântica. Cada tag, como **<h1>**, **<h2>**, **<p>**, entre outras, desempenha um papel específico na definição do tipo de conteúdo e na formatação visual.



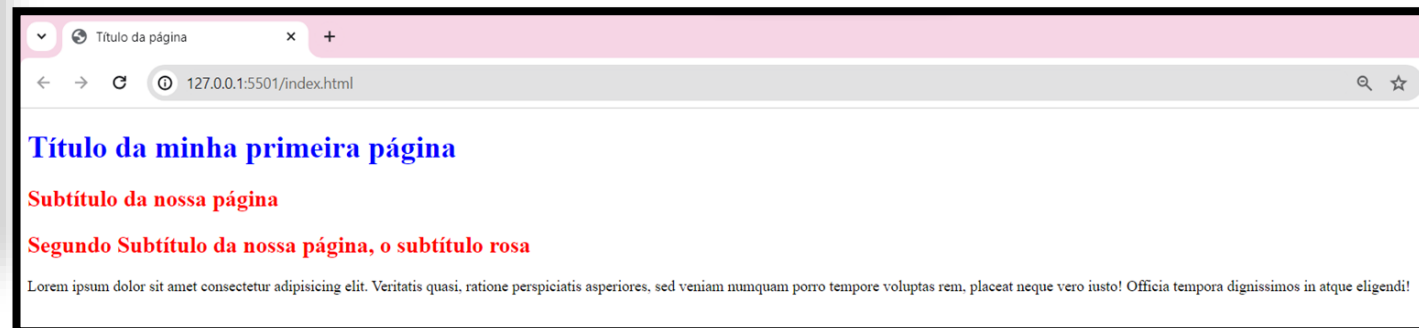
# Estruturas – HTML / CSS

Por exemplo, ao usar a tag **<h1>** repetidamente, você pode criar vários títulos de diferentes seções. Cada instância da tag **<p>** pode representar parágrafos distintos. O mesmo princípio se aplica a diversas outras tags HTML, permitindo a criação de uma estrutura lógica e bem organizada para o seu conteúdo.

Ao utilizar tags múltiplas, você molda a aparência e a funcionalidade da sua página, proporcionando aos usuários uma experiência de navegação consistente e compreensível. Essa abordagem é essencial para o desenvolvimento web eficaz, pois permite a representação clara e estruturada de informações variadas em um ambiente online.



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6   <title>Título da página</title>
7   <link rel="stylesheet" href="/style.css" />
8 </head>
9 <body>
10  <h1>Título da minha primeira página</h1>
11  <h2>Subtítulo da nossa página</h2>
12  <h2>Segundo Subtítulo da nossa página, o subtítulo rosa</h2>
13  <p>
14    Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis quasi,
15    ratione perspiciatis asperiores, sed veniam numquam porro tempore voluptas
16    rem, placeat neque vero iusto! Officia tempora dignissimos in atque
17    eligendi!
18  </p>
19 </body>
20 </html>
```



# Estruturas – HTML / CSS

É importante lembrar que você pode adicionar várias regras de estilo no arquivo `style.css` para estilizar diferentes elementos HTML. Você pode utilizar seletores CSS para segmentar elementos específicos, como classes, IDs, tags HTML, etc., e aplicar estilos específicos a esses elementos.

Além disso, você pode utilizar várias propriedades CSS para definir diferentes estilos, como cor de texto, cor de fundo, tamanho da fonte, margens, preenchimento, alinhamento, bordas e muitos outros.

O atributo **class** em HTML é utilizado para designar uma ou mais classes a um elemento específico. Classes são identificadores que podem ser estilizados no CSS, permitindo a aplicação consistente de estilos a diferentes partes da sua página.

Ao atribuir uma classe a um elemento HTML, você está essencialmente dizendo que este elemento faz parte de um grupo específico. Por exemplo, você pode ter uma classe chamada "destaque" que define um estilo de fundo amarelo para todos os elementos que compartilham essa classe.

No CSS, você define como essas classes devem ser estilizadas. Ao invés de aplicar estilos diretamente a uma tag HTML específica (como `<p>` ou `<div>`), você pode criar classes no CSS e aplicá-las a qualquer elemento que precise desses estilos. Isso torna o código mais modular e fácil de manter, já que você pode ajustar os estilos de toda uma classe alterando apenas uma definição no CSS.

# Estruturas – HTML / CSS

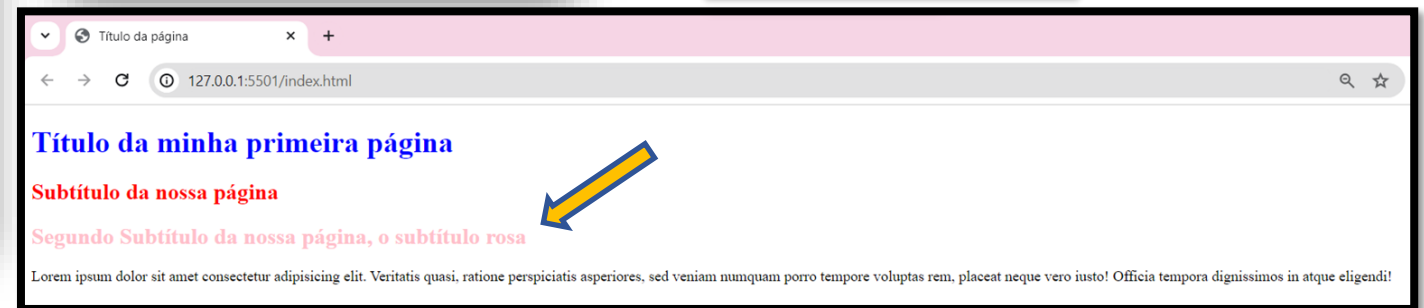
Vamos criar uma classe para o segundo elemento **<h2>**, chamada "texto-rosa". Para aplicarmos um estilo diferente do primeiro **<h2>**, iremos utilizar o arquivo "style.css" e atribuir a propriedade **color** ao seletor de classe ".texto-rosa".

Em outras palavras, queremos dar um estilo especial ao segundo **<h2>** chamando-o de "texto-rosa". Vamos definir as características visuais específicas para essa classe no arquivo de estilo, diferenciando-a do primeiro **<h2>**.

```
index.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>Título da página</title>
7     <link rel="stylesheet" href="./style.css" />
8   </head>
9   <body>
10    <h1>Título da minha primeira página</h1>
11    <h2>Subtítulo da nossa página</h2>
12    <h2 class="texto-rosa">
13      Segundo Subtítulo da nossa página, o subtítulo rosa
14    </h2>
15    <p>
16      Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis quasi,
17      ratione perspiciatis asperiores, sed veniam numquam porro tempore voluptas
18      rem, placeat neque vero iusto! Officia tempora dignissimos in atque
19      eligendi!
20    </p>
21  </body>
22 </html>
```

```
# style.css > ...
1 h1 {
2   color: blue;
3 }
4
5 h2 {
6   <element class="texto-rosa">
7   Selector Specificity: (0, 1, 0)
8   .texto-rosa {
9     color: pink;
10  }
11 }
```

```
# style.css > ...
1 h1 {
2   color: blue;
3 }
4
5 h2 {
6   color: red;
7 }
8
9 .texto-rosa {
10  color: pink;
11 }
```

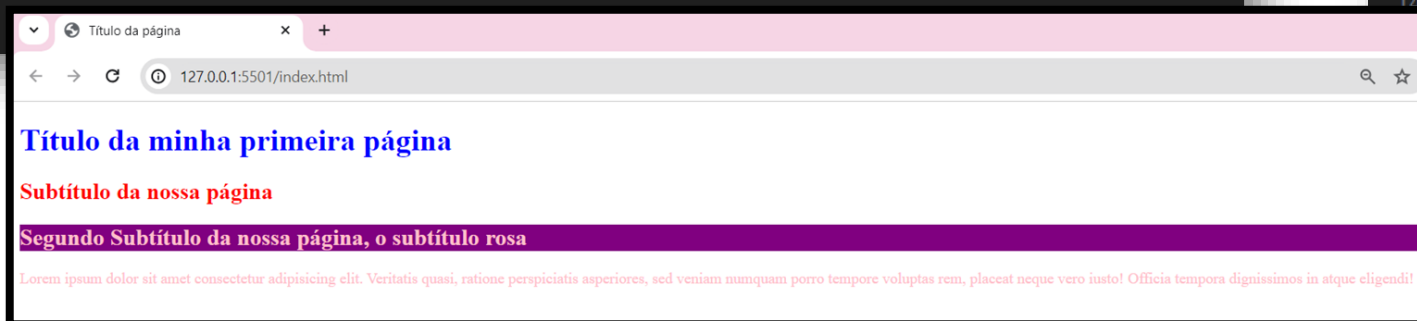


# Estruturas – HTML / CSS

**id** e **class** são atributos em HTML para identificação e agrupamento de elementos, permitindo estilização e manipulação via CSS e JavaScript. **id** é exclusivo por página, identificando singularmente um elemento, enquanto **class** é reutilizável, aplicada a vários elementos similares, promovendo uma abordagem modular no desenvolvimento web. Use **id** para singularidade e **class** para reusabilidade. No CSS, **id** é referenciado por (**#**), e **class** por (**.**).

```
<h1 id="titulo">Título da minha primeira página</h1>
<h2 id="subtitulo-1">Subtítulo da nossa página</h2>
<h2 id="subtitulo-2" class="texto-rosa">
  Segundo Subtítulo da nossa página, o subtítulo rosa
</h2>
<p class="texto-rosa">
  Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis quasi,
  ratione perspiciatis asperiores, sed veniam numquam porro tempore voluptas
  rem, placeat neque vero iusto! Officia tempora dignissimos in atque
  eligendi!
</p>
```

```
# style.css > ...
1  h1 {
2    color: blue;
3  }
4
5  h2 {
6    color: red;
7  }
8
9  .texto-rosa {
10   color: pink;
11 }
12
13 #subtitulo-2 {
14   background-color: purple;
15 }
```



Parte 7

# Iniciando Projeto

# Iniciando Projeto

Agora que instalamos e tivemos o primeiro contato com o VS Code, compreendemos as estruturas HTML e CSS. Vamos iniciar nosso projeto Audiobook e o primeiro passo é baixar o arquivo CSS, juntamente com as pastas de images e books.

Esses arquivos são essenciais para a construção do nosso Audiobook, contendo os áudios dos capítulos do livro "Dom Casmurro", a imagem do livro e toda a estilização do projeto.

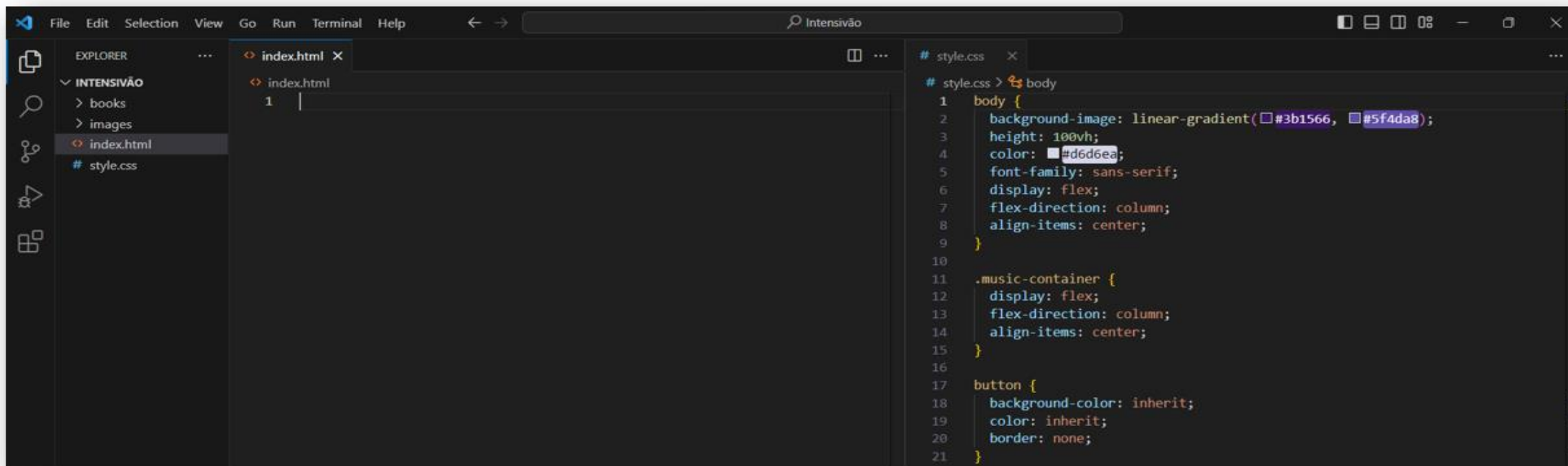
Lembrando que há uma aula opcional em que o professor Daniel apresenta detalhadamente como estruturar o arquivo CSS utilizado.

Após o download, adicione esses arquivos à pasta previamente criada chamada "Intensivão". Remova o arquivo "style.css" que foi criado anteriormente (usado para entender a estrutura de um arquivo CSS) e apague o conteúdo do arquivo "index.html", pois iniciaremos novamente a sua estrutura.



# Iniciando Projeto

Neste ponto, a estrutura do seu projeto deve consistir em um arquivo "index.html" vazio, o arquivo "style.css" já com a estilização fornecida pelo professor Daniel (o qual você baixou), e as pastas "books" e "images" localizadas dentro da pasta "Intensivão".

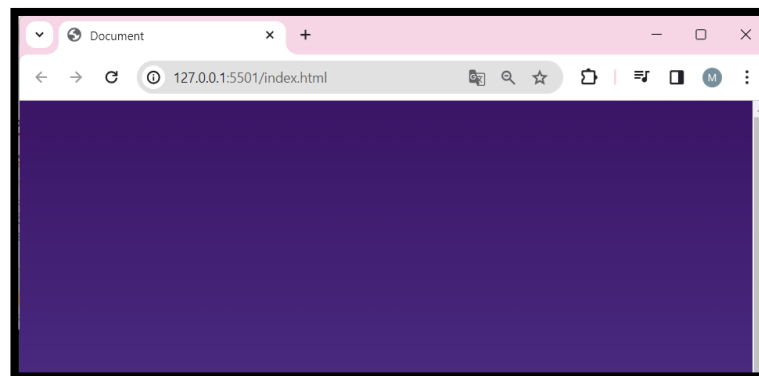


# Iniciando Projeto

Dentro do arquivo "index.html", vamos começar a estrutura básica do HTML usando o snippet ! + "Enter". Em seguida, adicionaremos o link para o nosso arquivo "style.css" utilizando o snippet "link:css" e fornecendo o caminho e nome correto do nosso arquivo, que é "style.css".

```
<> index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Document</title>
7      <link rel="stylesheet" href="./style.css" />
8    </head>
9    <body></body>
10 </html>
```

Com isso, já podemos acionar a extensão Live Server e verificar que, embora não haja nenhum conteúdo dentro da nossa página, já conseguimos visualizar a estilização aplicada.





# Iniciando Projeto

Adicionaremos as primeiras tags html dentro do corpo da nossa página (tag `<body>`) que será a **tag `<div>`** e dentro dela adicionaremos a **tag `<img>`**.

```
<> index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Document</title>
7      <link rel="stylesheet" href="./style.css" />
8    </head>
9    <body>
10     <div>
11       <img src="" alt="">
12     </div>
13   </body>
14 </html>
```

A **tag `<div>`** é uma das tags mais conhecidas e utilizadas no HTML. Ela é um elemento de divisão que permite agrupar e organizar o conteúdo de uma página. A **`<div>`** não possui um significado especial, mas é amplamente utilizada para criar blocos de conteúdo e aplicar estilos CSS a esses blocos.

A sintaxe da **tag `<div>`** é simples: você abre a tag com **`<div>`** e fecha com **`</div>`**. Todo o conteúdo que estiver entre essas tags será considerado parte da divisão. Por padrão, a **`<div>`** é um elemento de bloco, o que significa que ela gera uma quebra de linha automática

# Iniciando Projeto

A tag **<img>** em HTML é usada para incorporar imagens em uma página web. Duas propriedades-chave são:

- **src (source):** Especifica o caminho ou URL da imagem. Pode ser um caminho relativo no seu projeto ou uma URL completa.
- **alt (alternative text):** Fornece um texto alternativo para a imagem. É exibido se a imagem não carregar ou para usuários com leitores de tela. Essencial para acessibilidade e SEO, descrevendo o conteúdo ou propósito da imagem.

```
index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Document</title>
7      <link rel="stylesheet" href="./style.css" />
8    </head>
9    <body>
10     <div>
11       
12     </div>
13   </body>
14 </html>
```



# Iniciando Projeto

Agora, vamos incluir botões em nosso audiobook. Cada botão, representado pela tag **<button>**, terá uma função específica. Teremos um botão para avançar para o próximo capítulo, um botão para voltar ao capítulo anterior e um botão para iniciar e pausar o áudio do capítulo.

```
<div>
  
  <button id="anterior"></button>
  <button id="play-pause"></button>
  <button id="proximo"></button>
</div>
```

- **Botão "anterior" (acesso a um capítulo):**
  - Permite retroceder para o início do capítulo atual ou para o capítulo anterior.
- **Botão "play-pause" (reprodução/pausa no capítulo):**
  - Inicia ou pausa a reprodução do audiobook no capítulo atual.
- **Botão "proximo" (avanço para o próximo capítulo):**
  - Avança para o próximo capítulo, facilitando a navegação entre partes da obra.

# Iniciando Projeto

Utilizaremos uma biblioteca de ícones para estilizar o nossos botões. Essa é uma maneira eficiente e fácil de incorporar ícones em seus projetos web, sem a necessidade de criar ou carregar imagens separadas.

## BIBLIOTECAS

Uma biblioteca em JavaScript é um conjunto de código pré-escrito e disponibilizado para uso em projetos de desenvolvimento web. Essas bibliotecas contêm funções, classes e métodos que podem ser utilizados para realizar tarefas específicas de forma mais fácil e eficiente. Elas são criadas para resolver problemas comuns e fornecer funcionalidades adicionais que podem auxiliar no desenvolvimento de um projeto.

As bibliotecas em JavaScript são desenvolvidas para serem reutilizáveis, ou seja, podem ser utilizadas em diferentes projetos, economizando tempo e esforço na implementação de funcionalidades comuns. Elas são criadas por desenvolvedores e disponibilizadas para a comunidade, permitindo que outros programadores as utilizem em seus projetos.



# Iniciando Projeto

## BIBLIOTECA DE ÍCONES

Essas bibliotecas contêm uma variedade de ícones que representam diferentes conceitos, objetos ou ações, como setas, símbolos de redes sociais, ícones de menu, entre outros.

Essas bibliotecas de ícones são muito úteis para desenvolvedores, pois permitem que eles utilizem ícones prontos em seus projetos, economizando tempo e esforço no design e criação de ícones personalizados. Além disso, as bibliotecas de ícones costumam fornecer uma ampla variedade de estilos, tamanhos e formatos de ícones, permitindo que os desenvolvedores escolham os que melhor se adequam ao visual e às necessidades de seus projetos.



### Bootstrap Icons

**Bootstrap Icons** é uma biblioteca de ícones, fornece uma coleção de ícones vetorizados em formato SVG. Os ícones do Bootstrap Icons são projetados para serem escaláveis e acessíveis. Eles são fornecidos no formato SVG, o que significa que podem ser redimensionados sem perder qualidade ou nitidez. Além disso, o formato SVG oferece suporte a recursos de acessibilidade, tornando os ícones mais amigáveis para pessoas com deficiência visual.

# Iniciando Projeto

## Funcionalidade de uma Biblioteca de Ícones:



- **Consistência Visual:**

- Bibliotecas de ícones, como a do Bootstrap, garantem que os ícones tenham uma aparência visual consistente, mantendo o design do seu projeto coeso.

- **Facilidade de Implementação:**

- Ao usar uma biblioteca de ícones, você evita a necessidade de criar ou procurar imagens separadas. Os ícones já estão prontos para uso e podem ser facilmente adicionados ao seu código.

- **Responsividade:**

- Os ícones fornecidos pelo Bootstrap são vetores escaláveis, o que significa que eles se adaptam bem a diferentes tamanhos de tela sem perder qualidade.

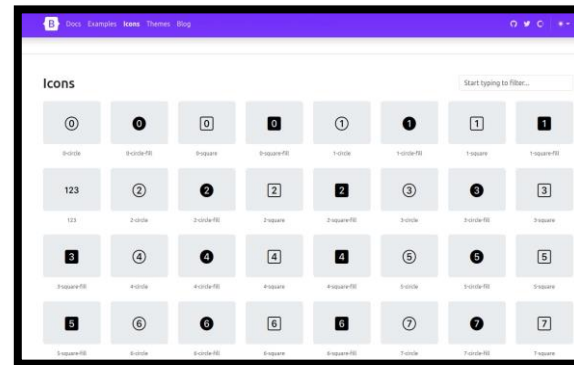
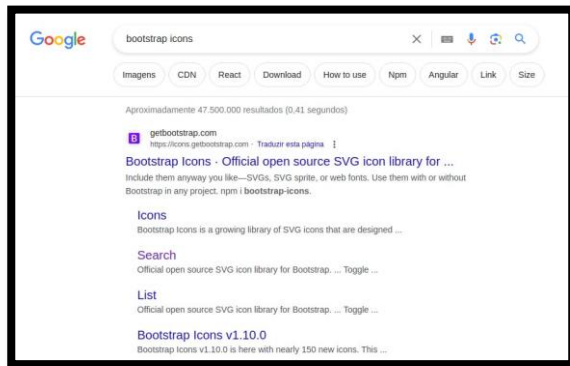
- **Atualizações Simples:**

- Se a biblioteca de ícones for atualizada para incluir novos ícones ou melhorias, você pode facilmente atualizar a versão do Bootstrap em seu projeto para aproveitar essas atualizações.

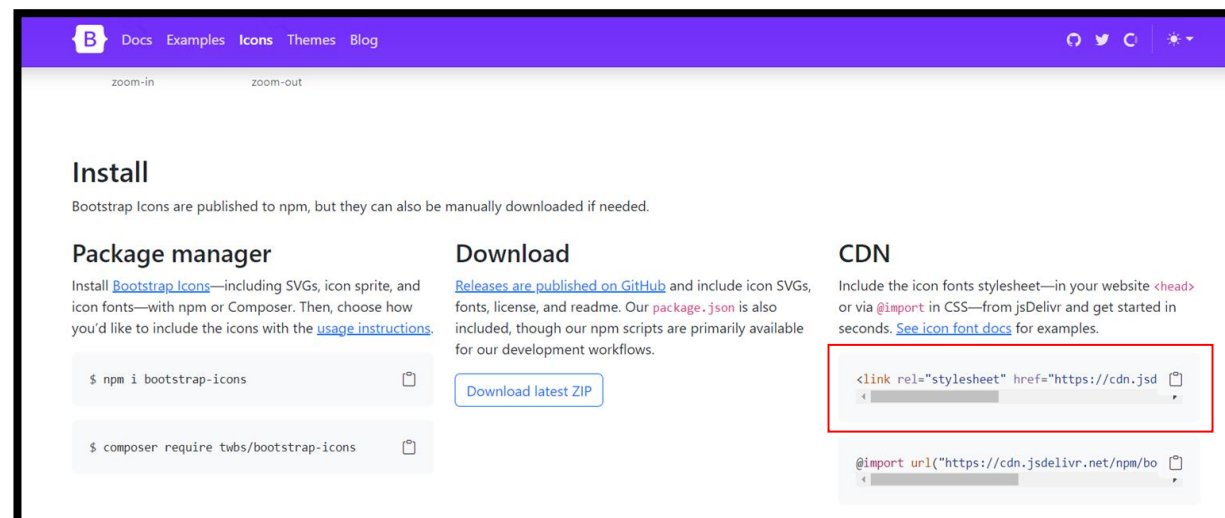
# Parte 7

## Iniciando Projeto

**Passo 1:** Pesquise no google, e entre no site oficial [Bootstrap Icons](https://icons.getbootstrap.com/):



Vá para o final do site, e lá você encontrará como utilizar o bootstrap icons dentro do seu projeto:



# Iniciando Projeto

## Passo 2: Inclua o Bootstrap Icons via CDN

Adicione o link do Bootstrap Icons no cabeçalho do seu documento HTML, logo abaixo do link do Bootstrap. Certifique-se de utilizar a versão mais recente do Bootstrap e Bootstrap Icons.

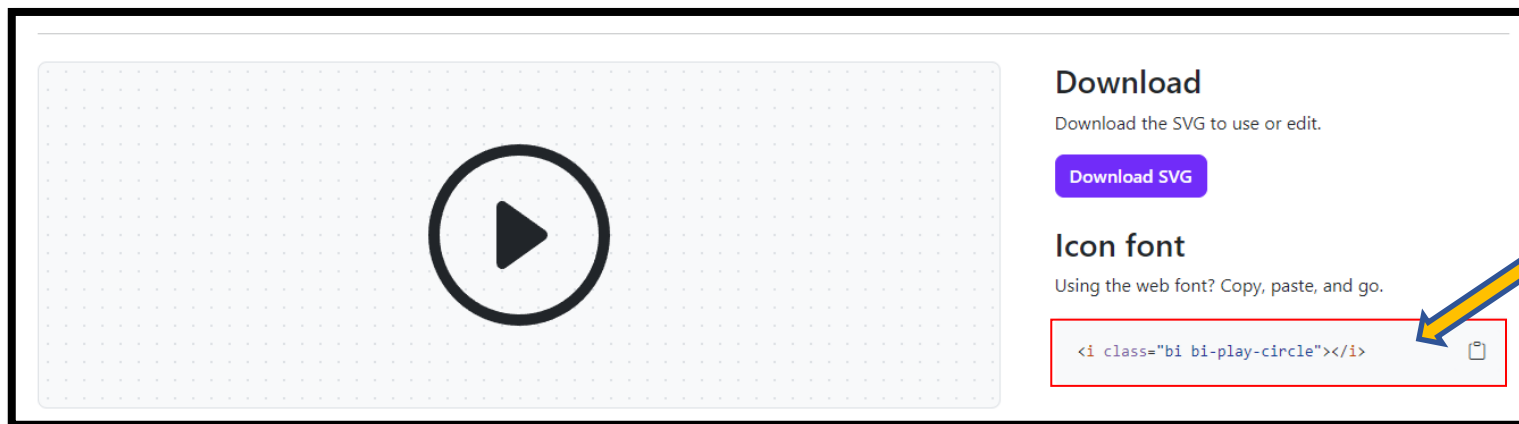
```
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Document</title>
  <link rel="stylesheet" href="./style.css" />
  <link
    rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.min.css"
  />
</head>
```

Após seguir os passos mencionados, o seu projeto estará pronto para utilizar o ícone escolhido da biblioteca Bootstrap Icons.



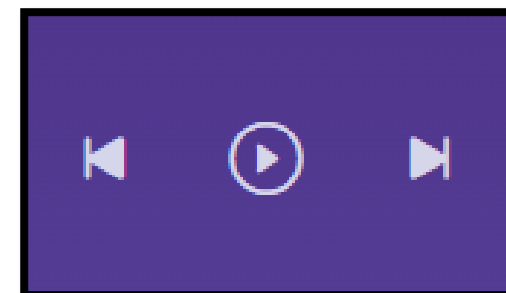
# Iniciando Projeto

Agora vamos escolher um ícone da biblioteca que melhor se adapte às necessidades do projeto. O primeiro será através da pesquisa de um ícone que represente "play", e as setas que representadas por "skip" do audiobook.



Vamos adicionar o código do ícone para adicionarmos ao nosso elementos de botões.

```
<div>
  
  <button id="anterior"><i class="bi bi-skip-start-fill"></i></button>
  <button id="play-pause"><i class="bi bi-play-circle"></i></button>
  <button id="proximo"><i class="bi bi-skip-end-fill"></i></button>
</div>
```



# Iniciando Projeto

Neste momento, vamos adicionar o estilo que foi criado anteriormente e está contido no arquivo que você baixou. No entanto, sinta-se à vontade para aplicar uma estilização personalizada, ou se preferir, assista à nossa aula adicional. Nessa aula, o professor Daniel nos guia passo a passo na criação do arquivo de estilização do CSS, o `style.css`.

No código abaixo, incluímos a classe `'music-container'` dentro da tag **<div>**. Além disso, criamos um novo elemento **<div>** que agrupará os botões. Este novo elemento recebeu um ID chamado `'button-container'`, que será utilizado para facilitar a interação com esses elementos através do JavaScript.

```
<body>
  <div class="music-container">
    

    <div id="button-container">
      <button id="anterior"><i class="bi bi-skip-start-fill"></i></button>
      <button id="play-pause"><i class="bi bi-play-circle"></i></button>
      <button id="proximo"><i class="bi bi-skip-end-fill"></i></button>
    </div>
  </div>
</body>
```



# Iniciando Projeto

Vamos adicionar o elemento mais importante para a criação do nosso audiobook, que é o de áudio. A tag **<audio>** é uma tag HTML usada para incorporar arquivos de áudio diretamente em uma página web. Essa tag oferece suporte nativo a reprodução de áudio sem a necessidade de plugins adicionais, proporcionando uma experiência de usuário mais integrada.

## O atributo src:

- Especifica o caminho do arquivo de áudio que será reproduzido. Pode ser um arquivo local ou uma URL.

```
<div class="music-container">  
    
  <audio src="./books/dom-casmurro/1.mp3"></audio>  
  <div id="button-container">
```

Parte 8

# Integrando Inteligência: Aplicando Javascript ao Audiobook

# Integrando Inteligência: Aplicando Javascript ao Audiobook

Ao seguir esses dois passos, você estará preparando a base para implementar funcionalidades inteligentes ao seu audiobook utilizando Javascript. Essa abordagem facilitará a interação do usuário com o audiobook, proporcionando uma experiência mais dinâmica e envolvente.


## Passo 1: Criação do Arquivo Javascript:

- Para começar, crie um arquivo na raiz do seu projeto chamado **script.js**. Este arquivo será responsável por conter o código JavaScript que implementará as funcionalidades inteligentes para o audiobook.

## Passo 2: Vínculo entre HTML e script.js:

- Para garantir que as funcionalidades do script.js se apliquem à estrutura do audiobook que construímos, é essencial criar uma ligação entre o arquivo HTML e o script.js. Certifique-se de incluir o seguinte código no final da tag <body> do seu arquivo HTML:

```
24     <script src="script.js"></script>  
25 </body>
```



A tag **<script>** é uma tag HTML usada para incorporar ou referenciar código Javascript em uma página web. Essa tag permite a execução de scripts do lado do cliente, proporcionando interatividade e dinamismo às páginas.

# Integrando Inteligência: Aplicando Javascript ao Audiobook

Para iniciarmos com a construção do nosso código Javascript precisamos entender sobre o porquê selecionar elementos HTML no JavaScript é crucial para tornar páginas web dinâmicas e interativas.

Essa prática permite a manipulação dinâmica do conteúdo, atualizações em tempo real, validação de formulários, gerenciamento de eventos, dinamismo na estilização, manipulação de conteúdo e integração com dados externos. Essa interação entre o JavaScript e os elementos HTML é fundamental para criar uma experiência de usuário mais envolvente e funcional, além de possibilitar o desenvolvimento de aplicativos web completos e responsivos.

Para realizarmos essa tarefa vamos utilizar a estrutura de código:

```
JS script.js  
1 document.getElementById("play-pause");
```

# Integrando Inteligência: Aplicando Javascript ao Audiobook

## 1. document:

### O que é:

- Pense no **document** como uma representação do documento HTML da sua página. É como se fosse uma grande pasta que contém todas as informações sobre o que está na sua página, como textos, imagens, botões, etc.

### Para que serve:

- Quando você quer fazer alguma coisa na sua página (como mudar um texto ou mostrar/ocultar algo), você precisa dizer ao computador onde encontrar essa coisa. É aí que entra o **document**. Ele ajuda você a dizer ao computador qual parte da sua página você quer mexer.

## 2. Por que usamos o ponto ( . ) – No Contexto do Documento (document):

- Quando você vê o ponto . depois do **document** (como em **document.getElementById()**), é como se estivesse dizendo: "Ei, **document**, use o seu poder de busca para encontrar algo!"

### Selecionar Elementos:

- O ponto . é usado para selecionar elementos específicos no HTML usando métodos como **getElementById**, **getElementsByClassName**, ou **getElementsByTagName**.

# Integrando Inteligência: Aplicando Javascript ao Audiobook

## 3. `.getElementById()`:

### O que é:

- Imagine que seu documento HTML é uma cidade, e cada coisa na cidade tem um endereço único. O `.getElementById()` é como um mapa que ajuda você a encontrar uma coisa específica na cidade.

### Para que serve:

- Quando você usa `.getElementById()`, você está pedindo ao computador para encontrar algo na sua página usando o ID único desse algo. Um ID é como um número da casa em uma rua. É único, o que significa que não há dois números iguais.

Em Javascript, as **aspas simples** (') e as **aspas duplas** (") são usadas para criar strings, que são sequências de caracteres. A escolha entre aspas simples e duplas geralmente é uma questão de preferência do desenvolvedor, mas existem algumas considerações práticas.

O **ponto e vírgula** (;) é usado como um delimitador de instrução. Ele indica o final de uma instrução ou expressão.



# Integrando Inteligência: Aplicando Javascript ao Audiobook

Para finalizarmos, vamos armazenar o resultado de **document.getElementById("play-pause")** em uma variável chamada "botaoPlayPause", que será do tipo **const**.

Ao armazenar o elemento em uma variável, você evita que o JavaScript precise procurar o elemento no DOM toda vez que você quiser interagir com ele. Isso pode melhorar o desempenho, especialmente se você estiver acessando o mesmo elemento várias vezes em seu código.

```
JS script.js > ...  
1  const botaoPlayPause = document.getElementById("play-pause");
```

## O que é uma Variável:

- Em programação, uma variável é um espaço de armazenamento nomeado que contém um valor. Elas são usadas para armazenar e manipular dados em um programa. Em JavaScript, você declara uma variável usando palavras-chave como **var**, **let**, ou **const**, seguidas pelo nome da variável.

## Tipo const em JavaScript:

- **const** é uma palavra-chave em JavaScript usada para declarar uma variável cujo valor não pode ser reatribuído. Isso significa que, após atribuir um valor a uma variável usando **const**, você não pode mais alterar esse valor.

# Integrando Inteligência: Aplicando Javascript ao Audiobook

Agora vamos adicionar um id ao elemento <audio>, para conseguir armazená-lo dentro de uma variável.

```
<div class="music-container">  
    
  <audio id="audio-capitulo" src="./books/dom-casmurro/1.mp3"></audio>  
  <div id="button-container">
```

E realizar o passo a passo novamente, para armazenar chamando o `document.getElementById("id_do_elemento")`.

```
JS script.js > ...  
1  const botaoPlayPause = document.getElementById("play-pause");  
2  const audio = document.getElementById("audio-capitulo");
```

# Integrando Inteligência: Aplicando Javascript ao Audiobook

Em JavaScript, uma função é um bloco de código nomeado que pode ser definido e chamado para realizar uma tarefa específica. Funções ajudam a organizar o código, promovem a reutilização e tornam o código mais modular. Aqui estão os principais elementos de uma função em JavaScript:

```
function nomeDaFuncao(parametro1, parametro2, ...) {  
    // Código a ser executado  
    return resultado; // Opcional: a função pode retornar um valor  
}
```

- **function:** Palavra-chave que inicia a definição da função.
- **nomeDaFuncao:** Nome da função, que é utilizado para chamá-la posteriormente.
- **parametro1, parametro2, ...:** Parâmetros são valores que a função espera receber. Eles são opcionais.
- **{ ... }:** Corpo da função, onde o código a ser executado está contido.
- **return resultado;:** A palavra-chave **return** é usada para retornar um valor da função. Isso é opcional e depende do propósito da função.

# Integrando Inteligência: Aplicando Javascript ao Audiobook

Vamos dar um passo importante na construção do nosso audiobook! Agora, vamos criar a primeira ação específica para o nosso projeto. O próximo passo consiste em desenvolver uma função que será responsável por iniciar a reprodução do áudio do capítulo.

```
function tocarFaixa() {  
    audio.play();  
}
```

Essa função tem um propósito simples: ela usa a propriedade `play()` de um elemento de áudio para iniciar a reprodução da faixa de áudio associada a esse elemento.

- `tocarFaixa()`: Este é o nome da função.
- `audio`: Este é o nome da variável que representa um elemento de áudio no código.
- `.play()`: Isso é chamado de método e é usado para começar a reprodução do áudio.

Portanto, quando esta função é chamada, ela basicamente inicia a reprodução da faixa de áudio associada ao elemento de áudio representado pela variável `audio`.

# Integrando Inteligência: Aplicando Javascript ao Audiobook

Para chamar ou executar uma função em JavaScript, você utiliza o nome da função seguido pelos parênteses (). Isso é conhecido como invocar a função.

```
// Chamada da função tocarFaixa()  
tocarFaixa();
```

Vamos utilizar uma ferramenta muito importante para o desenvolvedor, O DevTools, ou Ferramentas de Desenvolvedor, que é um conjunto de utilitários que são incorporados nos navegadores da web para ajudar os desenvolvedores a inspecionar, depurar e otimizar o código de uma página da web. Ele fornece várias ferramentas que facilitam o desenvolvimento, testes e diagnóstico de problemas no código.

## Como Acionar o DevTools (Chrome):

1. Botão Direito + "Inspecionar" em qualquer lugar na página.
2. Ou pressione Ctrl + Shift + I (Windows/Linux) ou Cmd + Opt + I (Mac).
3. Ou pressione o atalho F12 ou Fn + F12.

# Integrando Inteligência: Aplicando Javascript ao Audiobook

## Funcionalidades Principais do DevTools:

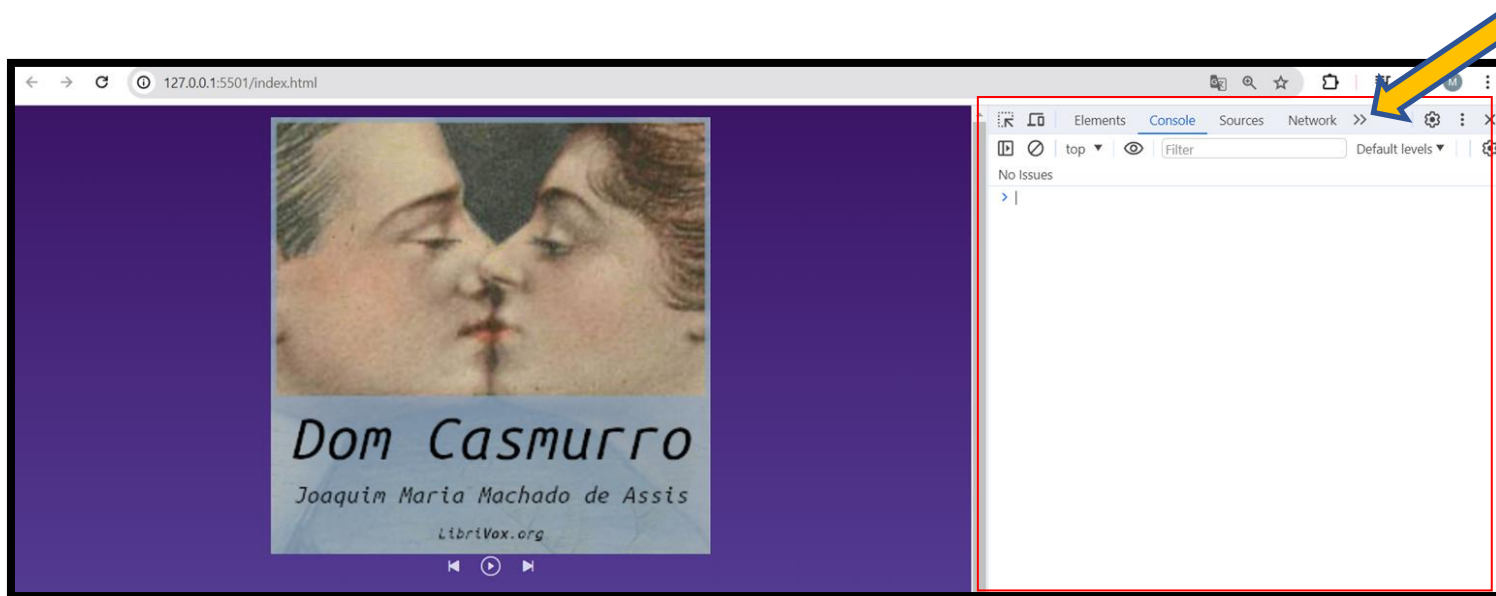
**Elementos (ou Inspecionar):** Permite examinar e modificar o HTML e o CSS de uma página interativamente.

- **Console:** Fornece um console interativo para a execução de comandos JavaScript, exibição de erros e mensagens de depuração.
- **Network (ou Rede):** Monitora as solicitações HTTP feitas pela página, permitindo análise de desempenho e diagnóstico de problemas de rede.
- **Sources (ou Origens):** Permite a edição, depuração e definição de pontos de interrupção no código JavaScript.
- **Application (ou Aplicação):** Oferece informações sobre armazenamento local, cookies, cache e outros recursos relacionados à aplicação.
- **Performance (ou Desempenho):** Ajuda a analisar e otimizar o desempenho da página.
- **Memory (ou Memória):** Monitora o uso de memória e identifica vazamentos de memória no código JavaScript.

# Integrando Inteligência: Aplicando Javascript ao Audiobook

- **Security (ou Segurança):** Fornece informações sobre a segurança da conexão da página.
- **Audits (ou Auditorias):** Oferece auditorias automáticas para melhorar o desempenho, acessibilidade e as práticas de SEO da página.

Usar o DevTools é uma prática essencial para desenvolvedores web, pois proporciona uma visão detalhada e interativa do comportamento de uma página da web, ajudando a aprimorar a qualidade e o desempenho do código.



# Integrando Inteligência: Aplicando Javascript ao Audiobook

A aba "Console" no DevTools é uma ferramenta poderosa e versátil que oferece várias funcionalidades úteis para os desenvolvedores. Aqui estão algumas razões pelas quais você pode querer utilizar a aba "Console":

## Depuração de Código JavaScript:

- Permite a execução interativa de comandos JavaScript. Você pode testar expressões, verificar variáveis, e corrigir bugs diretamente no console.

## • Exibição de Mensagens e Erros:

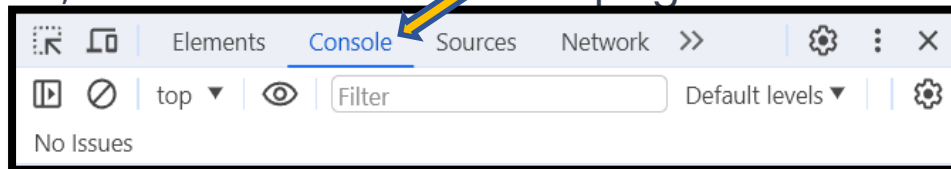
- Mostra mensagens de depuração, informações e erros gerados pelo seu código JavaScript. É uma ferramenta valiosa para identificar e corrigir problemas no código.

## • Saída de Logs:

- Você pode usar **console.log()**, **console.warn()**, **console.error()**, entre outros, para exibir informações e mensagens de log diretamente no console. Isso é útil para entender o fluxo de execução do seu código.

## • Execução de Comandos Úteis:

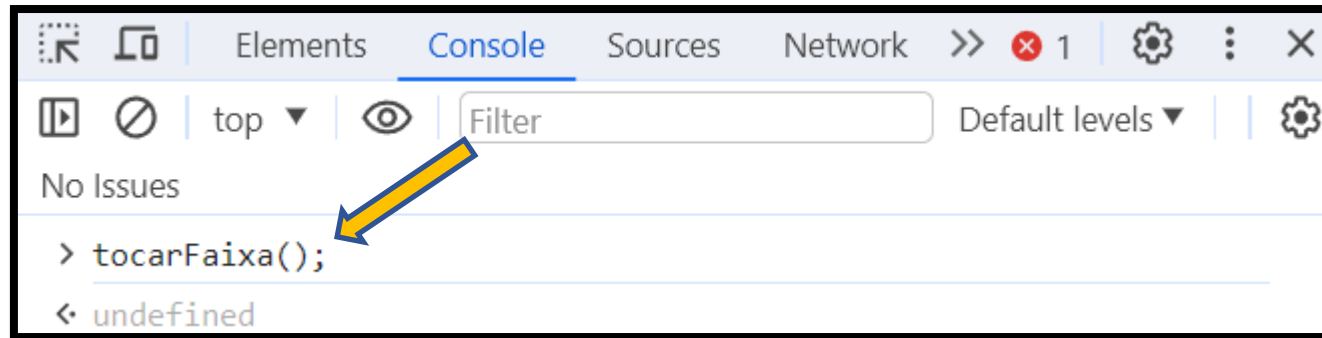
- Além de JavaScript, você pode executar comandos úteis diretamente no console, como manipulações de DOM, acesso a elementos da página.





# Integrando Inteligência: Aplicando Javascript ao Audiobook

Agora, faremos a chamada da função na aba "Console", pressionando ENTER. Vamos observar a mágica acontecer, com o programa reproduzindo o Capítulo 1 do nosso audiobook.



O próximo passo é adicionarmos um evento ao nosso botão para controlar a ação da nossa função "tocarFaixa()".

## O que são Eventos:

- Eventos em JavaScript são ações ou ocorrências que acontecem no navegador, geralmente desencadeadas pelo usuário ou pelo próprio navegador. Eles são fundamentais para criar interatividade em páginas web.
- Eventos são como "coisas" que acontecem em uma página web. Elas podem ser ações do usuário (como clicar em um botão), interações do navegador (como a conclusão do carregamento de uma página), ou até mesmo mudanças no conteúdo da página.

# Integrando Inteligência: Aplicando Javascript ao Audiobook

```
JS script.js > ...  
1  const botaoPlayPause = document.getElementById("play-pause");  
2  const audio = document.getElementById("audio-capitulo");  
3  
4  function tocarFaixa() {  
5    |   audio.play();  
6  }  
7  
8  botaoPlayPause.addEventListener("click", tocarFaixa);
```

Essa linha de código está dizendo ao computador para observar o botão chamado "botaoPlayPause", e quando alguém clicar nele, execute a função **tocarFaixa**.

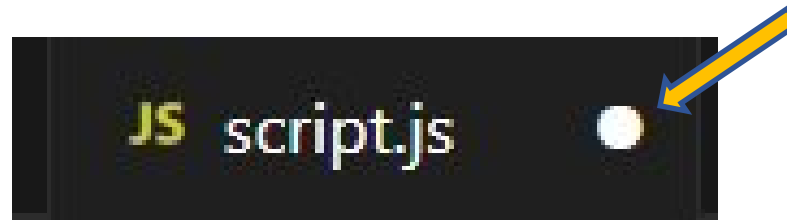
## **.addEventListener("click", tocarFaixa):**

- Adiciona um "ouvinte" ao botão, esperando pela ação de clique ("**click**"). Em outras palavras, estamos dizendo ao computador para "escutar" quando alguém clica no botão.

# Integrando Inteligência: Aplicando Javascript ao Audiobook

💡 Não se esqueça de que você pode explorar vários tipos de eventos na internet. Basta ir até um mecanismo de busca e procurar por eventos com JavaScript. Você encontrará sites e documentações que oferecem uma variedade de eventos para você ver e aprender, ampliando suas possibilidades de aplicação.

💡 Quando você faz alguma alteração no VS Code, os arquivos exibem um ícone de uma bolinha branca. Esse ícone indica que há alterações que ainda não foram salvas. Para salvar essas mudanças, você pode usar o atalho CTRL + S. Isso significa pressionar simultaneamente a tecla CTRL no seu teclado e, enquanto a mantém pressionada, pressionar a tecla S. Essa ação permite que as modificações que você fez sejam salvas, garantindo que o seu trabalho atualizado seja armazenado.



# Integrando Inteligência: Aplicando Javascript ao Audiobook

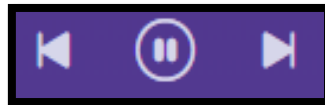
Agora vamos adicionar o ícone para o próximo botão que será da funcionalidade de pausar a faixa que está tocando, utilizando o site oficial do [Bootstrap Icons](#).



Vamos simplificar os botões removendo a tag `<i>` do Bootstrap Icons e adicionando as classes diretamente aos elementos `<button>` para facilitar a implementação de novas funcionalidades.

```
<div id="button-container">
  <button id="anterior" class="bi bi-skip-start-fill"></button>
  <button id="play-pause" class="bi bi-play-circle"></button>
  <button id="proximo" class="bi bi-skip-end-fill"></button>
</div>
```

Essa mudança elimina a necessidade da tag `<i>` e adiciona as classes diretamente aos botões. Dessa forma, os estilos e ícones do Bootstrap podem ser aplicados diretamente aos botões, tornando mais simples a implementação de novas funcionalidades sem a necessidade de modificar a estrutura dos botões.



# Integrando Inteligência: Aplicando Javascript ao Audiobook

Precisamos criar a lógica para que ocorra a troca dos ícones de play e pause, e vamos utilizar Javascript para isso utilizando os seguintes conceitos:

**classList** é uma propriedade de objetos DOM em JavaScript que fornece acesso às classes de um elemento HTML. Ela contém métodos para adicionar, remover e verificar a presença de classes em um elemento.

## Métodos Principais:

**.add("classe1", "classe2", ...):** Adiciona uma ou mais classes ao elemento.

**.remove("classe1", "classe2", ...):** Remove uma ou mais classes do elemento.

```
function tocarFaixa() {  
    audio.play();  
    botaoPlayPause.classList.remove("bi-play-circle");  
    botaoPlayPause.classList.add("bi-pause-circle");  
}
```

# Integrando Inteligência: Aplicando Javascript ao Audiobook

**botaoPlayPause.classList.remove("bi-play-circle"):** Remove a classe "bi-play-circle" do botão, caso esteja presente.

**botaoPlayPause.classList.add("bi-pause-circle"):** Adiciona a classe "bi-pause-circle" ao botão, indicando visualmente que a faixa está sendo reproduzida e agora o botão representa uma opção de pausa.

Essas manipulações de classe permitem a alteração dinâmica da aparência do botão conforme o estado da reprodução do áudio, proporcionando uma experiência visual clara para o usuário.

Nesse momento, vamos criar a segunda funcionalidade que será pausar o áudio do capítulo. Vamos construir essa função da mesma forma que fizemos com a função **tocarFaixa()**. O nome da nova função será **pausarFaixa()**, e inverteremos apenas as classes dos botões.

```
function pausarFaixa() {  
    audio.pause();  
    botaoPlayPause.classList.add("bi-play-circle");  
    botaoPlayPause.classList.remove("bi-pause-circle");  
}
```

# Integrando Inteligência: Aplicando Javascript ao Audiobook

Ao explorarmos a aplicação das funções **tocarFaixa()** e **pausarFaixa()** em um elemento de botão, surge uma questão intrigante: como podemos designar funcionalidades a um único botão para desempenhar tanto a ação de reprodução quanto a de pausa em um áudio?

Essa dúvida é crucial, pois, à primeira vista, poderia parecer desafiador conciliar duas operações distintas em um único elemento. No entanto, é exatamente aqui que a flexibilidade e poder das estruturas de programação entram em cena.

Ao associarmos a função **tocarFaixa()** ao evento de clique em um botão, conseguimos iniciar a reprodução do áudio quando o botão é pressionado. Mas como podemos estender essa lógica para incluir a pausa? Este é um ponto intrigante, pois precisamos considerar como diferenciar entre as ações de tocar e pausar no mesmo botão.

Essa situação nos leva a explorar ainda mais a capacidade das estruturas condicionais em JavaScript. Podemos criar uma verificação que, ao ser acionada pelo clique no botão, avalia se o áudio está atualmente em reprodução ou pausado. Com base nessa avaliação, podemos então determinar se a função a ser executada será **tocarFaixa()** ou **pausarFaixa()**.

# Integrando Inteligência: Aplicando Javascript ao Audiobook

## ESTRUTURA CONDICIONAL:

A estrutura condicional é um conceito fundamental na programação, proporcionando a capacidade de tomar decisões dinâmicas com base em condições específicas. Em JavaScript, linguagem de programação amplamente utilizada para desenvolvimento web, as estruturas condicionais mais comuns são **if**, **else if** e **else**.

O bloco **if** é utilizado para executar um conjunto de instruções quando uma condição é avaliada como verdadeira. Por exemplo, podemos verificar se uma pessoa é maior de idade antes de permitir o acesso a determinadas funcionalidades.

Já o **else if** entra em cena quando há a necessidade de avaliar várias condições em sequência. Cada bloco **else if** é verificado apenas se as condições anteriores foram consideradas falsas. Isso é especialmente útil para situações em que diferentes ações precisam ser tomadas com base em diferentes cenários.

O bloco **else** é utilizado para definir um conjunto de instruções a ser executado quando nenhuma das condições anteriores for considerada verdadeira. Isso permite que um código mais abrangente seja desenvolvido, abordando todos os possíveis resultados.



# Integrando Inteligência: Aplicando Javascript ao Audiobook

Vamos criar uma função que terá o propósito de decidir se deve tocar ou pausar a faixa de áudio com base no estado atual.

## **function tocarOuPausarFaixa():**

- Declaração de uma função chamada **tocarOuPausarFaixa**. Esta função será chamada quando o evento associado ao botão for acionado.

## **if (taTocando === true) { pausarFaixa(); } else { tocarFaixa(); }:**

- Estrutura condicional (**if...else**) que verifica se a variável **taTocando** é igual a **true**.
- Se for verdadeiro (**taTocando === true**), significa que a faixa está sendo reproduzida, então a função **pausarFaixa()** é chamada para pausar a faixa.
- Se for falso (**else**), ou seja, a faixa não está sendo reproduzida, a função **tocarFaixa()** é chamada para começar a reprodução.

```
function tocarOuPausarFaixa() {  
  if (taTocando === true) {  
    pausarFaixa();  
  } else {  
    tocarFaixa();  
  }  
}
```

Essencialmente, essa função realiza uma ação de alternância entre tocar e pausar a faixa de áudio. Se a faixa estiver tocando, ela será pausada, e se estiver pausada, será tocada. A variável **taTocando** é usada como uma espécie de sinalizador para indicar o estado atual da reprodução da faixa. Essa abordagem é comum ao criar controles de reprodução toggle (alternância) em interfaces de áudio.

# Integrando Inteligência: Aplicando Javascript ao Audiobook

Vamos adicionar a declaração da variável `taTocando`, ela será do tipo "let" com o valor booleano de `false`.

## let:

- A palavra-chave **let** é usada em JavaScript para declarar variáveis. Quando declaramos uma variável com **let**, estamos basicamente reservando um espaço na memória para armazenar informações que podem ser alteradas ao longo do tempo.

## Booleano:

- É um tipo de dado em programação que pode ter apenas dois valores: **true** ou **false**. É frequentemente utilizado para representar estados binários, como ligado/desligado, verdadeiro/falso, ativo/inativo.

Agora, juntando esses conceitos, ao adicionar a declaração da variável **taTocando** com **let** e inicializá-la com o valor booleano **false**, estamos criando uma variável que pode ser usada para controlar o estado de reprodução da faixa de áudio. Por padrão, ela começa com o valor **false**, indicando que a faixa não está sendo tocada. Conforme o código evolui, esse valor pode ser atualizado para **true** quando a faixa estiver em reprodução e para **false** quando estiver pausada.

```
JS script.js > ...  
1  const botaoPlayPause = document.getElementById("play-pause");  
2  const audio = document.getElementById("audio-capitulo");  
3  let taTocando = false;
```

# Integrando Inteligência: Aplicando Javascript ao Audiobook

Estamos usando a variável **taTocando** para controlar o estado da reprodução da faixa de áudio. Para fazer isso, dentro das funções **tocarFaixa()** e **pausarFaixa()**, iremos atualizar o valor dessa variável para refletir se a faixa está sendo tocada ou pausada.

## **tocarFaixa():**

- Dentro desta função, vamos definir **taTocando** como **true**, indicando que a faixa está sendo tocada.

## **pausarFaixa():**

- Dentro desta função, vamos definir **taTocando** como **false**, indicando que a faixa está sendo pausada.

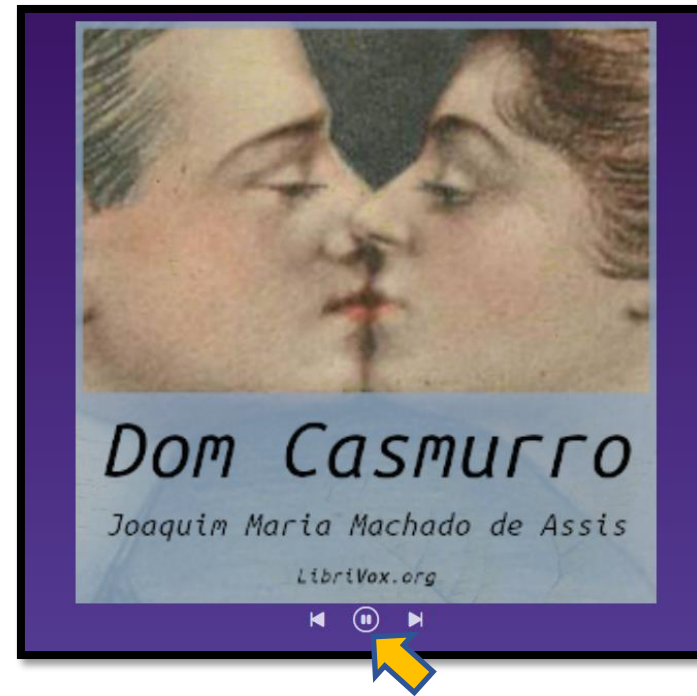
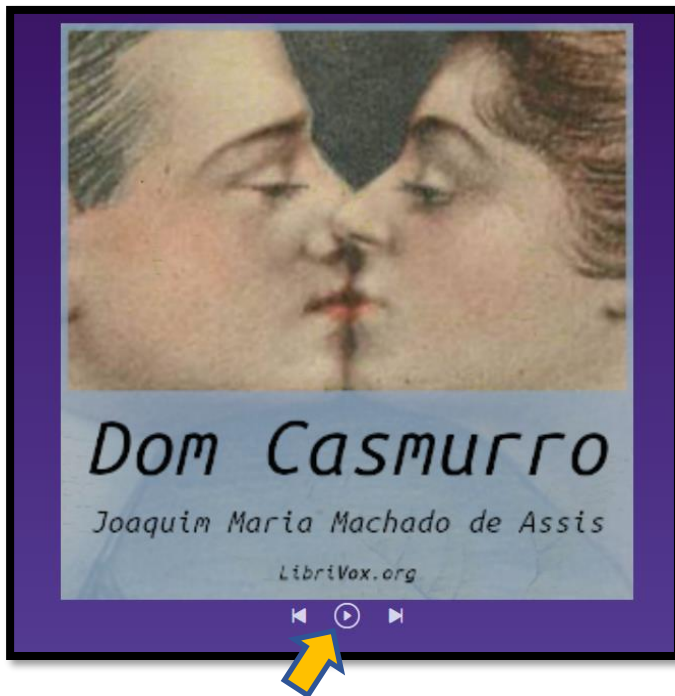
Essas atualizações garantem que a variável **taTocando** reflète o estado atual da reprodução da faixa. Quando tocamos a faixa, **taTocando** se torna **true**, e quando pausamos, **taTocando** se torna **false**. Essa abordagem permite que o código saiba em que estado a faixa se encontra e tome decisões com base nesse estado.

```
function tocarFaixa() {  
  audio.play();  
  botaoPlayPause.classList.remove("bi-play-circle");  
  botaoPlayPause.classList.add("bi-pause-circle");  
  taTocando = true;  
}  
  
function pausarFaixa() {  
  audio.pause();  
  botaoPlayPause.classList.add("bi-play-circle");  
  botaoPlayPause.classList.remove("bi-pause-circle");  
  taTocando = false;  
}
```

# Integrando Inteligência: Aplicando Javascript ao Audiobook

Por fim, vamos utilizar a função "tocarOuPausarFaixa()" como parâmetro do método `addEventListener` que está associado ao elemento "botaoPlayPause".

```
botaoPlayPause.addEventListener("click", tocarOuPausarFaixa);
```




# Integrando Inteligência: Aplicando Javascript ao Audiobook

Com a lógica de play e pause no lugar, o próximo passo será iniciar a narrativa no primeiro capítulo, e depois iremos explorar as funcionalidades de navegação, permitindo que os usuários retrocedam ou avancem entre os capítulos com facilidade. Essa adição não só incrementa a experiência de audição, mas também confere aos ouvintes o controle sobre sua própria jornada narrativa, capacitando-os a explorar e revisitando momentos específicos.

Vamos criar uma variável para representar os capítulos do nosso Audiobook. Como queremos que a jornada comece no capítulo um, vamos atribuir o valor 1 a essa variável.

```
1  const botaoPlayPause = document.getElementById("play-pause");
2  const audio = document.getElementById("audio-capitulo");
3  let taTocando = false;
4  let capitulo = 1;
```

E vamos adicionar também uma variável que armazenará o botão se avançar para o próximo capítulo:

```
JS script.js >  tocarFaixa
1  const botaoPlayPause = document.getElementById("play-pause");
2  const botaoProximoCapitulo = document.getElementById("proximo");
3  const audio = document.getElementById("audio-capitulo");
4  let taTocando = false;
5  let capitulo = 1;
```

# Integrando Inteligência: Aplicando Javascript ao Audiobook

Vamos adicionar uma funcionalidade empolgante ao nosso audiobook chamada **proximoCapitulo()**. Ao acionar essa função, não só pulamos para o próximo capítulo na sequência, mas também nos certificamos de que o áudio correspondente a esse capítulo seja carregado e esteja pronto para tocar.

Isso significa que, ao navegar pelos capítulos da nossa história, o código automaticamente ajusta as configurações do áudio, proporcionando uma experiência fluida e contínua. Dessa forma, a função **proximoCapitulo()** não apenas avança a narrativa, mas também sincroniza o áudio para que os ouvintes possam aproveitar sem interrupções. É uma adição essencial para tornar a exploração dos capítulos do nosso audiobook ainda mais envolvente.

```
function proximoCapitulo() {  
  capitulo += 1;  
  audio.src = "books/dom-casmurro/" + capitulo + ".mp3";  
  tocarFaixa();  
}
```

# Integrando Inteligência: Aplicando Javascript ao Audiobook


Na função **proximoCapitulo()**, a primeira linha incrementa o número do capítulo (**capitulo**) em 1, avançando para o próximo na sequência. Na terceira linha, **audio.src** é atualizado para o novo caminho do arquivo de áudio correspondente ao próximo capítulo.

A linha **audio.src = "books/dom-casmurro/" + capitulo + ".mp3";** é crucial para carregar o áudio correto. Aqui, **"books/dom-casmurro/"** é o diretório onde os arquivos de áudio estão localizados, e **capitulo + ".mp3"** forma o nome do arquivo com base no número do capítulo. Essa concatenação dinâmica permite que o código se ajuste automaticamente ao capítulo atual, carregando o áudio apropriado.

Em resumo, ao acionar **proximoCapitulo()**, não apenas avançamos para o próximo capítulo, mas também garantimos que o áudio associado a esse capítulo seja carregado e preparado para ser reproduzido. Essa dinâmica facilita a navegação entre capítulos em nosso audiobook.

Agora, vamos conectar o botão de avançar para o próximo capítulo com o evento de clique.

```
botaoPlayPause.addEventListener("click", tocarOuPausarFaixa);  
botaoProximoCapitulo.addEventListener("click", proximoCapitulo);
```



# Integrando Inteligência: Aplicando Javascript ao Audiobook

Aqui, foi introduzida uma estrutura condicional (if-else) para verificar se o incremento do capítulo ultrapassaria o limite máximo (**quantidadeCapitulos**). Se o capítulo atual for menor que o limite, incrementamos normalmente. Caso contrário, resetamos o capítulo para 1, garantindo que a sequência seja cíclica.

A utilização do if-else é interessante para manter a lógica da navegação entre capítulos consistente e controlada. Ele assegura que, ao chegar ao último capítulo, ao invés de continuar indefinidamente, a navegação retorne ao início da história.

A próxima funcionalidade que adicionaremos ao nosso projeto de Audiobook permitirá a navegação para o capítulo anterior. Vamos progredir na construção, seguindo a abordagem passo a passo que utilizamos na função **proximoCapitulo()**.

```
function proximoCapitulo() {  
  if (capitulo < quantidadeCapitulos) {  
    capitulo += 1;  
  } else {  
    capitulo = 1;  
  }  
  
  audio.src = "books/dom-casmurro/" + capitulo + ".mp3";  
  tocarFaixa();  
}
```

```
function capituloAnterior() {  
  capitulo -= 1;  
  
  audio.src = "books/dom-casmurro/" + capitulo + ".mp3";  
  tocarFaixa();  
}
```



# Integrando Inteligência: Aplicando Javascript ao Audiobook

Agora, é hora de ajustar a função **proximoCapitulo()**. Atualmente, não estamos controlando o número de faixas, ou seja, o projeto tem 30 faixas, representando 30 capítulos. Na implementação atual, não conseguimos parar no capítulo 30 e voltar para o capítulo 1; em vez disso, a navegação continuaria indefinidamente para 31, 32 e assim por diante, resultando em um erro quando não houver áudio correspondente.

Para evitar esse problema, precisamos adicionar um controle que verifique se atingimos o último capítulo. Se sim, ao acionar a função, ela deve nos levar de volta ao primeiro capítulo, criando uma experiência de navegação cíclica e evitando possíveis erros de áudio. Vamos realizar esse ajuste para garantir que nossa aplicação funcione de maneira mais suave e controlada.

Precisamos de uma variável que armazene o valor do número de capítulos.

```
const quantidadeCapitulos = 30;  
let taTocando = false;  
let capitulo = 1;
```

# Integrando Inteligência: Aplicando Javascript ao Audiobook

Na função **capituloAnterior()**, a primeira linha decrementa o número do capítulo (**capitulo**) em 1, retrocedendo para o anterior na sequência.

Da mesma forma que fizemos na função **proximoCapitulo()**, será necessário aplicar uma lógica para controlar a quantidade de capítulos ao retroceder. Ao voltarmos do primeiro capítulo, a ideia é ir para o capítulo 30, que representa a última faixa do livro. Essa lógica garantirá uma navegação fluida, evitando erros e proporcionando uma experiência mais consistente ao ouvinte.

```
function capituloAnterior() {  
  if (capitulo === 1) {  
    capitulo = quantidadeCapitulos;  
  } else {  
    capitulo -= 1;  
  }  
  
  audio.src = "books/dom-casmurro/" + capitulo + ".mp3";  
  tocarFaixa();  
}
```

Nessa estrutura condicional (if-else), verificamos se o capítulo é igual a 1. Se for verdadeiro (ou seja, estamos no primeiro capítulo), ajustamos o capítulo para ser igual à **quantidadeCapitulos** (representando o último capítulo do livro). Caso contrário (se não estivermos no primeiro capítulo), simplesmente decrementamos o capítulo em 1.

Essa lógica permite a navegação entre capítulos, indo do primeiro para o último e vice-versa.

# Integrando Inteligência: Aplicando Javascript ao Audiobook

Na função **capituloAnterior()**, a primeira linha decrementa o número do capítulo (**capitulo**) em 1, retrocedendo para o anterior na sequência.

Da mesma forma que fizemos na função **proximoCapitulo()**, será necessário aplicar uma lógica para controlar a quantidade de capítulos ao retroceder. Ao voltarmos do primeiro capítulo, a ideia é ir para o capítulo 30, que representa a última faixa do livro. Essa lógica garantirá uma navegação fluida, evitando erros e proporcionando uma experiência mais consistente ao ouvinte.

Agora, vamos adicionar uma variável que armazenará o botão de avançar para o próximo capítulo (**imagem 1**). Também associaremos o botão de retroceder para o capítulo anterior ao evento de clique (**imagem 2**).

```
1 const audio = document.getElementById("audio-capitulo");  
2 const botaoPlayPause = document.getElementById("play-pause");  
3 const botaoProximoCapitulo = document.getElementById("proximo");  
4 const botaoCapituloAnterior = document.getElementById("anterior");
```

1

```
botaoPlayPause.addEventListener("click", tocarOuPausarFaixa);  
botaoProximoCapitulo.addEventListener("click", proximoCapitulo);  
botaoCapituloAnterior.addEventListener("click", capituloAnterior);
```

2

# Integrando Inteligência: Aplicando Javascript ao Audiobook

Agora precisamos criar uma lógica para quando a reprodução do áudio atual chegar ao fim, a função **proximoCapitulo** seja chamada, iniciando automaticamente a reprodução do próximo capítulo. Essa abordagem é útil para criar uma experiência contínua e automatizada ao ouvir um audiobook, onde a transição para o próximo capítulo ocorre sem a necessidade de intervenção do usuário.

```
botaoPlayPause.addEventListener("click", tocarOuPausarFaixa);  
botaoProximoCapitulo.addEventListener("click", proximoCapitulo);  
botaoCapituloAnterior.addEventListener("click", capituloAnterior);  
audio.addEventListener("ended", proximoCapitulo);
```

O trecho **audio.addEventListener('ended', proximoCapitulo);** refere-se a um evento que é acionado quando o áudio chega ao fim, ou seja, quando a reprodução da faixa de áudio atual é concluída.

- **'ended'**: é o tipo de evento que estamos ouvindo. Neste caso, estamos interessados no evento 'ended', que ocorre quando o áudio termina de ser reproduzido.

Parte 9

# Aprimorando Detalhes do Audiobook

# Aprimorando Detalhes do Audiobook

Chegamos aos ajustes finais do nosso projeto. Vamos incluir o nome do autor, o título do capítulo e a capacidade de alterar o nome dos capítulos. No arquivo **index.html**, adicionaremos os elementos que representarão esses dados no Audiobook.

```
<body>
  <div class="music-container">
    
    <audio id="audio-capitulo" src="./books/dom-casmurro/1.mp3"></audio>

    <div id="track-info">
      <div id="capitulo">Capítulo 1</div>
      <div id="nome-autor">Machado de Assis</div>
    </div>

    <div id="button-container">
      <button id="anterior" class="bi bi-skip-start-fill"></button>
      <button id="play-pause" class="bi bi-play-circle"></button>
      <button id="proximo" class="bi bi-skip-end-fill"></button>
    </div>
  </div>
```

# Aprimorando Detalhes do Audiobook

Criaremos uma variável para armazenar o elemento responsável por guardar o nome do Capítulo.

```
const nomeCapitulo = document.getElementById("capitulo");
```

Vamos adicionar essa linha de código dentro das funções `proximoCapitulo()` e `capituloAnterior()`:

```
nomeCapitulo.innerText = "Capítulo " + capitulo;
```

Essa linha de código está associada à manipulação do conteúdo de um elemento HTML na página, mais especificamente, o elemento que armazena o título do capítulo. Aqui está uma explicação passo a passo:

**nomeCapitulo:** Isso representa um elemento HTML que foi previamente selecionado no código, geralmente usando **document.getElementById()** ou métodos semelhantes. Esse elemento serve como o local onde desejamos exibir o título do capítulo.

**.innerText: innerText** é uma propriedade que permite modificar o texto interno de um elemento HTML. Ao usá-la, podemos alterar dinamicamente o conteúdo textual dentro do elemento.

**"Capítulo " + capitulo:** Aqui, estamos concatenando a string fixa "Capítulo " com a variável **capitulo**, que representa o número do capítulo atual. A concatenação (+) combina essas partes para criar uma nova string que representa o título formatado do capítulo.

# Aprimorando Detalhes do Audiobook

Nesse momento as funções `proximoCapitulo()` e `capituloAnterior()` terá essa estrutura:

```
function proximoCapitulo() {  
  if (capitulo < quantidadeCapitulos) {  
    capitulo += 1;  
  } else {  
    capitulo = 1;  
  }  
  audio.src = "books/dom-casmurro/" + capitulo + ".mp3";  
  nomeCapitulo.innerText = "Capítulo " + capitulo;  
  tocarFaixa();  
}  
  
function capituloAnterior() {  
  if (capitulo === 1) {  
    capitulo = quantidadeCapitulos;  
  } else {  
    capitulo -= 1;  
  }  
  audio.src = "books/dom-casmurro/" + capitulo + ".mp3";  
  nomeCapitulo.innerText = "Capítulo " + capitulo;  
  tocarFaixa();  
}
```



# Aprimorando Detalhes do Audiobook

Para completarmos o nosso projeto, vamos aplicar a estilização do gabarito, ou você pode assistir a aula adicional e opcional.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.2/font/bootstrap-icons.css">
9   <link rel="stylesheet" href="./style.css">
10 </head>
11 <body>
12   <div class="music-container">
13     <h4 id="audio-name"></h4>
14     </img>
15     <audio id='audio-capitulo' src="/books/dom-casmurro/1.mp3"></audio>
16     <div id="below-cover" class="full-width">
17       <div id="track-info">
18         <div id="capitulo">Capítulo 1</div>
19         <div id="nome-autor">Machado de Assis</div>
20       </div>
21     </div>
22     <div id="button-container" class="full-width">
23       <button id="anterior" class="button button-navigate bi bi-skip-start-fill"></button>
24       <button id="play-pause" class="button button-biggest bi bi-play-circle-fill"></button>
25       <button id="proximo" class="button button-navigate bi bi-skip-end-fill"></button>
26     </div>
27   </div>
28   <script src="./script.js"></script>
29 </body>
30 </html>
```

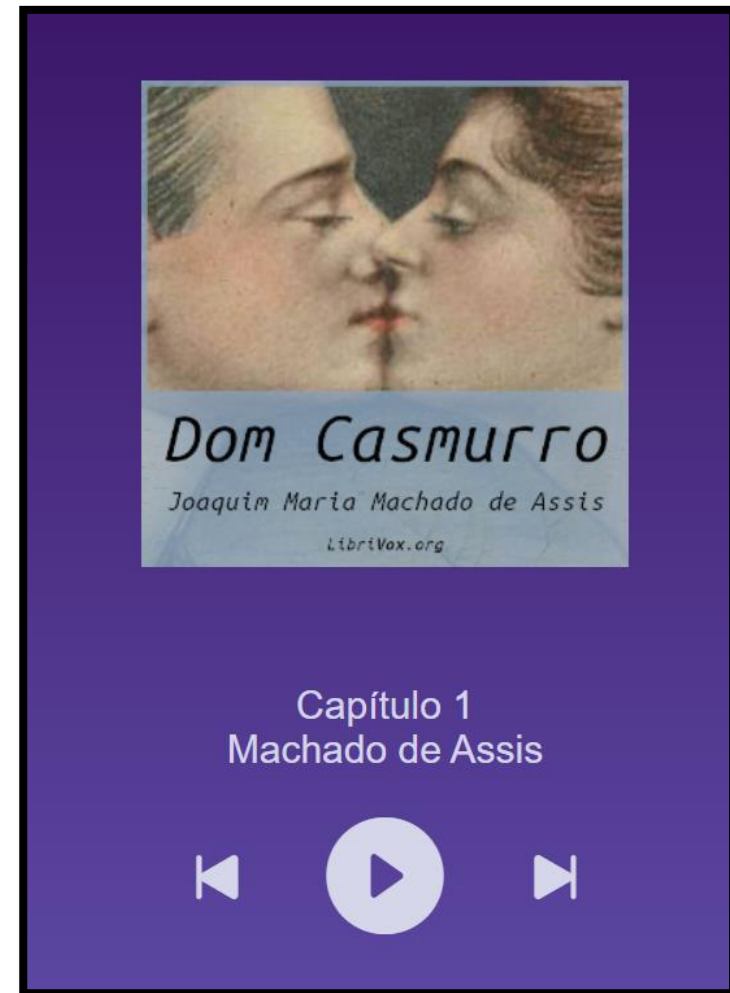
**Arquivo  
index.html**

# Aprimorando Detalhes do Audiobook

E vamos atualizar as classes dentro das funções:

```
function tocarFaixa() {  
  botaoPlayPause.classList.remove("bi-play-circle-fill");  
  botaoPlayPause.classList.add("bi-pause-circle-fill");  
  audio.play();  
  taTocando = true;  
}  
  
function pausarFaixa() {  
  botaoPlayPause.classList.add("bi-play-circle-fill");  
  botaoPlayPause.classList.remove("bi-pause-circle-fill");  
  audio.pause();  
  taTocando = false;  
}
```

**Arquivo  
script.js**

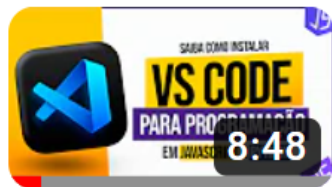


Parte Bônus

Vídeos

Complementares

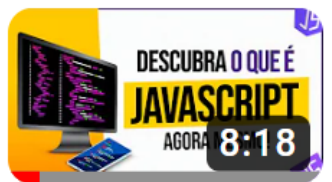
# Vídeos Complementares



## Instalação do VS Code para Programação em JavaScript e Python

Hashtag Programação

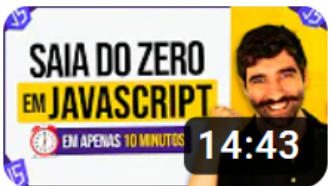
[https://www.youtube.com/watch?v=iLraM\\_NZYfA&list=PLpdAy0tYrnKyLd8e6e3suYCYO3LKzCVzi&index=2](https://www.youtube.com/watch?v=iLraM_NZYfA&list=PLpdAy0tYrnKyLd8e6e3suYCYO3LKzCVzi&index=2)



## O que é JavaScript?

Hashtag Programação

<https://www.youtube.com/watch?v=oWSxNI8Pg9k&list=PLpdAy0tYrnKyLd8e6e3suYCYO3LKzCVzi>



## Aprenda JavaScript em 10 min [Iniciantes]

Hashtag Programação

<https://www.youtube.com/watch?v=MombnkR3oD4&list=PLpdAy0tYrnKyLd8e6e3suYCYO3LKzCVzi&index=18>

Ainda não segue a gente no Instagram e nem é inscrito no nosso canal do Youtube? Então corre lá!



@hashtagtreinamentos



[youtube.com/hashtag-treinamentos](https://youtube.com/hashtag-treinamentos)

