

1 Introdução

Neste material, estudaremos sobre a linguagem HTML5 e suas principais características.

2 Desenvolvimento

(Ambiente) O primeiro passo é obter um ambiente para desenvolvermos nossos testes. Há diversos ambientes on-line de boa qualidade que nos permitem desenvolver sem ter de instalar um ambiente local. Um deles é o Replit:

<https://replit.com/>

Um outro muito bom é o CodePen:

<https://codepen.io/>

Há também o Visual Studio Code for the Web:

<https://vscode.dev/>

Você pode testá-los e adotar aquele que achar mais interessante.

Há também a possibilidade de utilizar um ambiente local. Esta é a abordagem adotada neste material. O editor de texto/código que utilizamos é o Visual Studio Code:

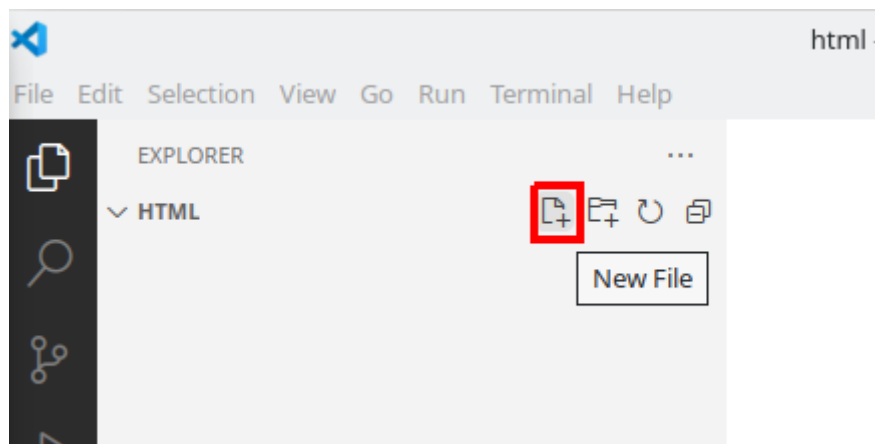
<https://code.visualstudio.com/>

(Uma pasta no seu sistema de arquivos) Comece criando uma pasta no seu sistema de arquivos. Se estiver usando o Windows, uma boa opção pode ser

C:\Users\seuusuario\Documents\html

Depois disso, abra o VS Code e clique em **File >> Open Folder**. Navegue até a pasta que acabou de criar para vincular o VS Code a ela.

(Primeira página HTML) No VS Code, clique no botão destacado a seguir para criar um novo arquivo:



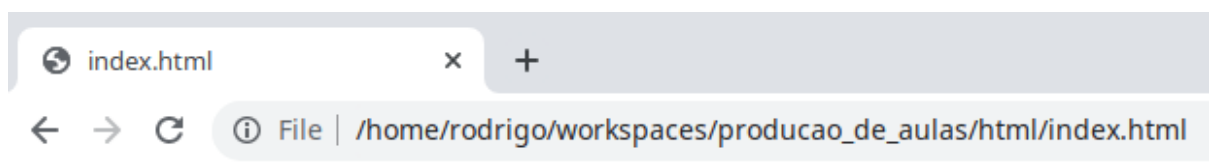
Seu nome pode ser algo como **index.html**.

Nota. O nome index.html tem significado histórico. Em geral, servidores web entregam o conteúdo descrito em arquivos com esse nome mediante acessos à raiz. Por exemplo, se o usuário acessar www.exemplo.com.br em seu navegador, a menos que configurado de forma diferente, o servidor entregará ao cliente o conteúdo do arquivo index.html, ou variações, como index.php, index.jsp etc.

(Elementos e tags HTML) A linguagem HTML define muitos **elementos**, cada qual com um propósito. Temos elementos próprios para a exibição de texto, imagens, vídeos, links etc. Elementos são caracterizados por **tags**. Em geral, um elemento HTML tem a sua **tag de abertura, seu conteúdo e sua tag de fechamento**. Veja um exemplo:

```
<h1>Primeiro teste</h1>
```

Abra o arquivo no seu navegador (um simples duplo clique no sistema de arquivos ou um clique com o direito >> abrir com) para ver o resultado. Ele deve ser mais ou menos assim:



Primeiro teste

Nota. Desde já, observe que a exibição de cada elemento, feita pelo navegador, varia em função de seu tipo. **Um elemento HTML do tipo h1, por exemplo, foi exibido com fonte “grande” e em negrito.** A forma como cada elemento é exibido, entretanto, pode variar de navegador para navegador. **O mais importante é o significado do elemento (h1 é um cabeçalho de “alta” importância) e não a sua exibição padrão.** De fato, veremos como mexer com a exibição dos elementos mais tarde, quando falarmos sobre CSS.

Neste exemplo,

<h1> é a tag de abertura

Primeiro teste é o conteúdo do elemento

</h1> é a tag de fechamento

Este conjunto define um elemento HTML do tipo h1 que tem como finalidade representar o título principal de uma seção.

(Listas e aninhamento) Elementos HTML podem ser aninhados uns aos outros. Isso quer dizer que um elemento HTML pode ser definido entre as tags de abertura e fechamento de outro. Para ilustrar, vamos criar uma **lista não ordenada** de produtos.

Nota. **ul** significa unordered list e **li** significa list item.

Digamos que desejamos criar uma lista de produtos. Começamos ajustando o cabeçalho da página e definindo um elemento do tipo **ul**:

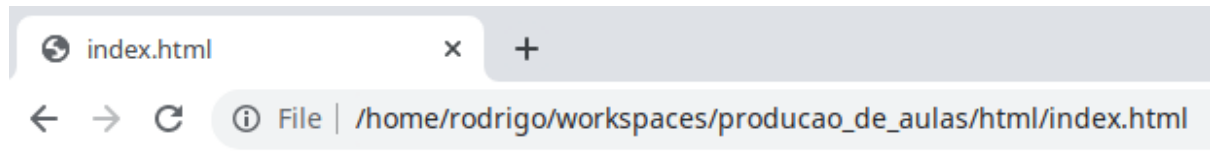
```
<h1>Lista de produtos</h1>
<ul>

</ul>
```

Para definir os itens da lista, definimos elementos do tipo **li** aninhados dentro do elemento **ul**:

```
<h1>Lista de produtos</h1>
<ul>
  <li>Arroz</li>
  <li>Feijão</li>
  <li>Laranja</li>
</ul>
```

Atualize o navegador para ver o resultado:



Lista de produtos

- Arroz
- Feijão
- Laranja

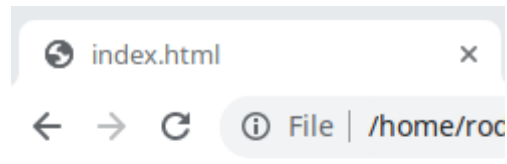
(Comentários em arquivos HTML) Os comentários em arquivos HTML são feitos da seguinte forma:

```
<!-- isso é um comentário -->  
<h1>Lista de produtos</h1>  
...
```

(Sublistas) Suponha que desejamos comprar diferentes tipos de frutas. A sua descrição poderia ser uma sublista da lista principal. Para tal, escrevemos um novo elemento ul aninhado dentro de um elemento li da lista principal:

```
<h1>Lista de produtos</h1>  
<ul>  
  <li>Arroz</li>  
  <li>Feijão</li>  
  <li>  
    Frutas  
    <ul>  
      <li>Laranja</li>  
      <li>Pêssego</li>  
      <li>Maçã</li>  
    </ul>  
  </li>  
</ul>
```

Atualize o navegador e veja o resultado:



Lista de produtos

- Arroz
- Feijão
- Frutas
 - Laranja
 - Pêssego
 - Maçã

(Estrutura completa de uma página HTML: html, head e body) Estudamos alguns exemplos de elementos HTML sem nos preocuparmos com a estrutura básica que todo documento HTML deve definir:

- O primeiro elemento, na “raiz”, é do tipo html
- O primeiro filho do elemento html é do tipo head e ele abriga metadados sobre a página, não visíveis para o usuário
- O segundo filho do elemento html é do tipo body e ele abriga o conteúdo visível para o usuário.

Crie um novo arquivo chamado **estrutura.html**. Veja como definimos os três elementos citados:

```
<html>
  <head>

</head>
  <body>

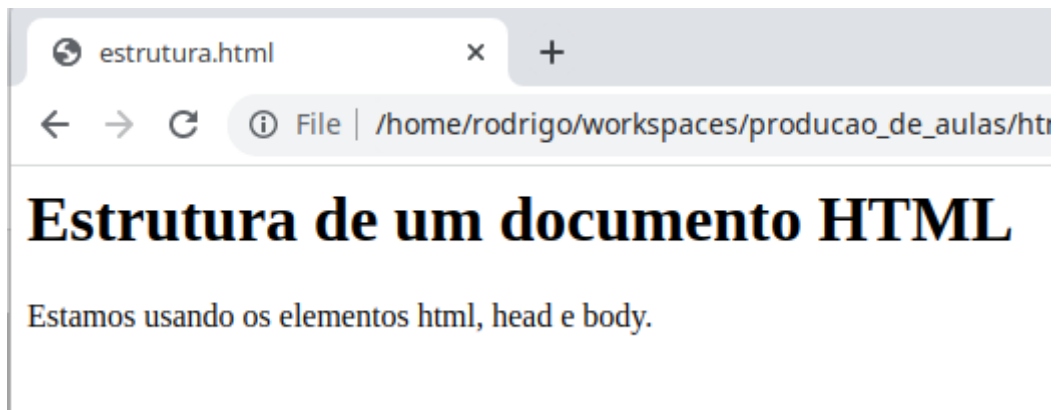
</body>
</html>
```

Nossa página poderia ter, por exemplo, um cabeçalho e um texto qualquer. Os elementos próprios para isso são definidos em seu body:

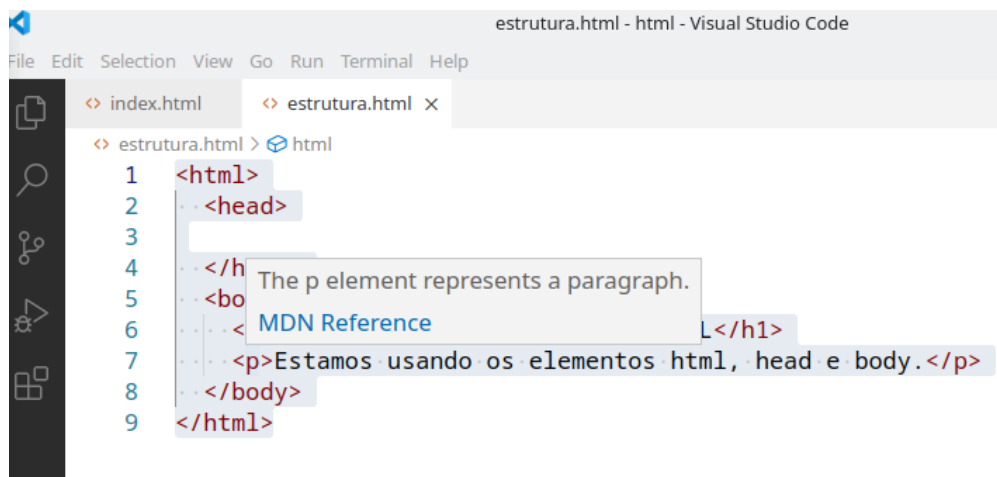
```
<html>
  <head>

</head>
<body>
  <h1>Estrutura de um documento HTML</h1>
  <p>Estamos usando os elementos html, head e body.</p>
</body>
</html>
```

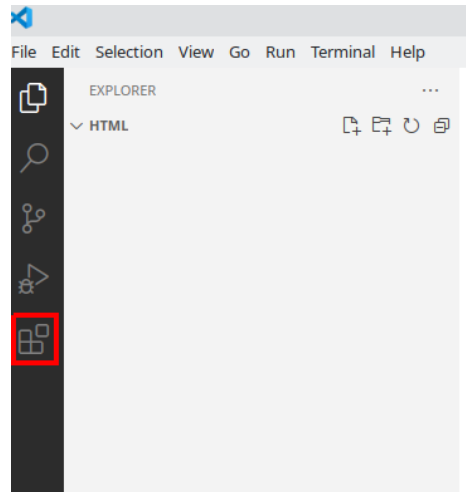
Abra o novo arquivo no navegador para ver o resultado:



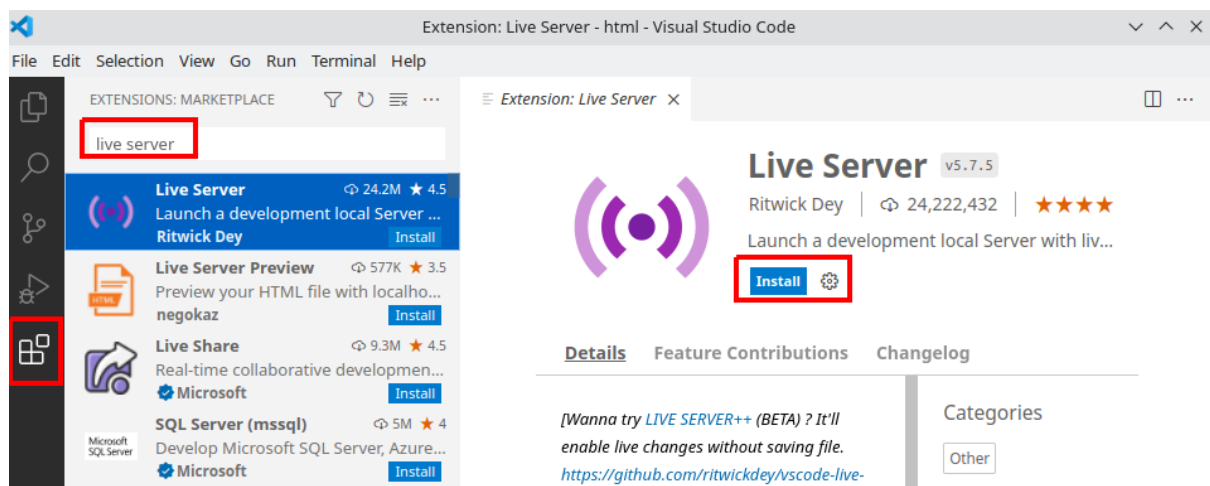
Nota. Passe o mouse sobre a tag de abertura de um elemento HTML no VS Code para obter uma descrição sobre ele. No exemplo a seguir, o ponteiro do mouse está sobre a tag de abertura do elemento p:



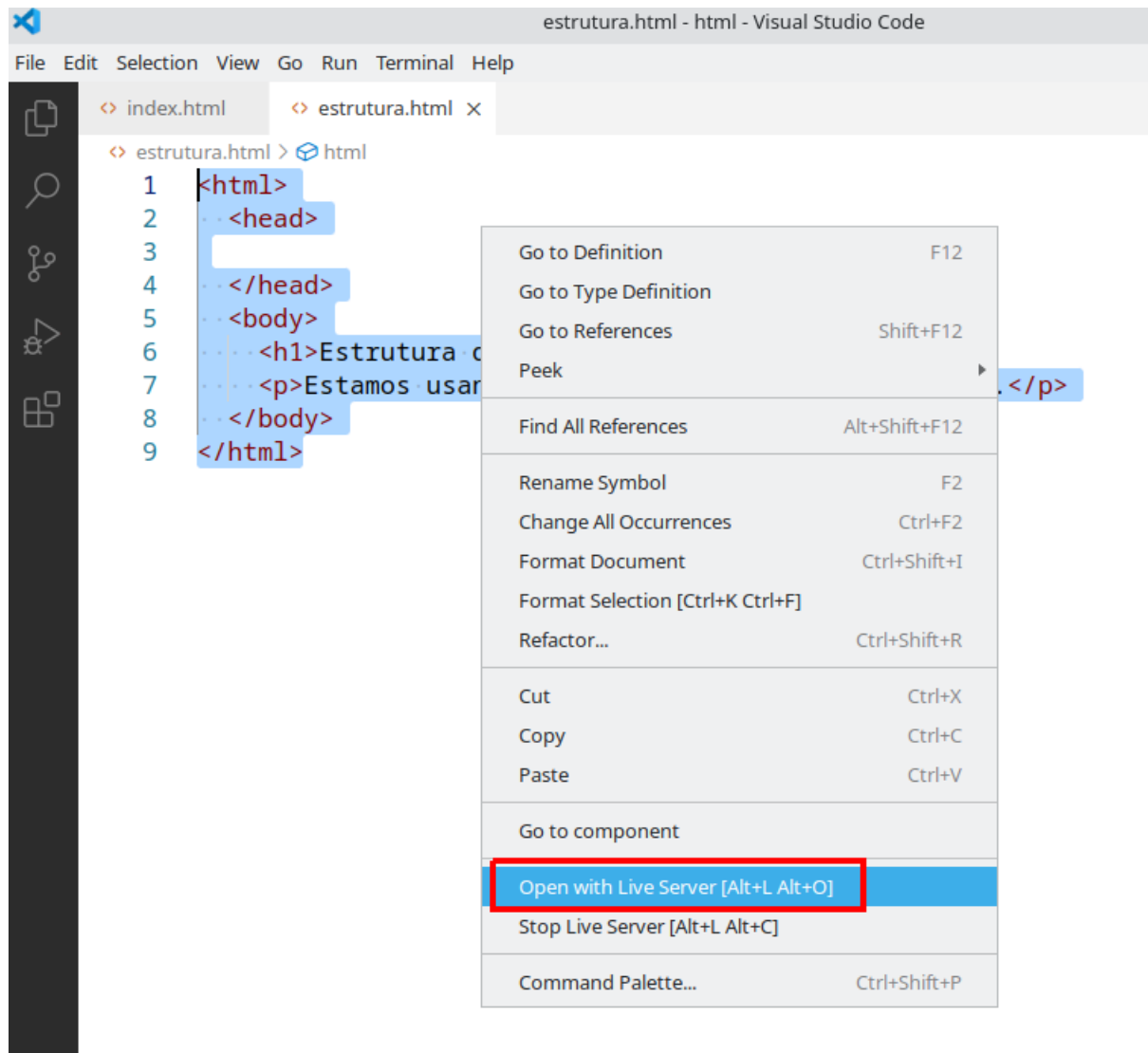
(Extensão Live Server para o VS Code) A extensão Live Server do VS Code é muito simples e muito útil. Ela permite que executemos um servidor responsável por “servir” a aplicação e atualizar a sua interface gráfica sempre que um dos arquivos que a compõem for atualizado. Para fazer a sua instalação, clique em **Extensions** no VS Code:



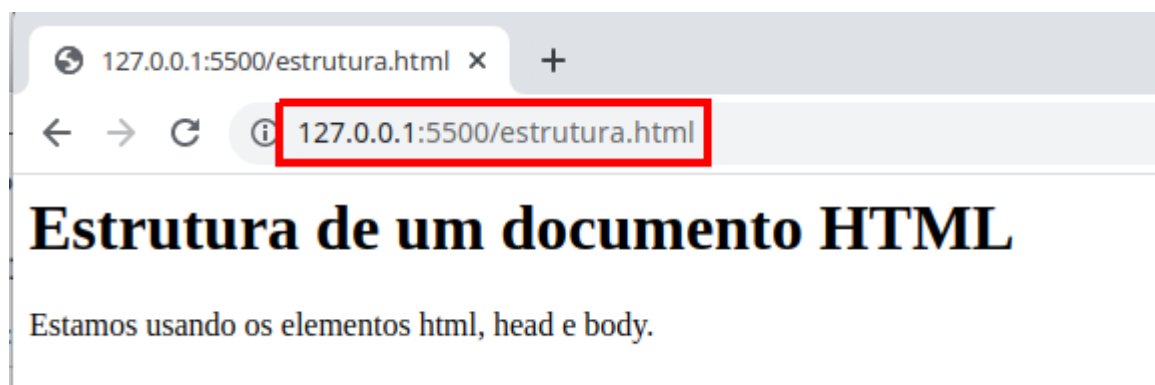
A seguir, busque por **Live Server**. Na tela resultante, a extensão desejada deve ser logo a primeira, já que ela é muito comum e possui milhões de downloads. Clique em **Install**:



Feita a sua instalação, clique com o direito em qualquer região do código e escolha a opção **Open with Live Server**:



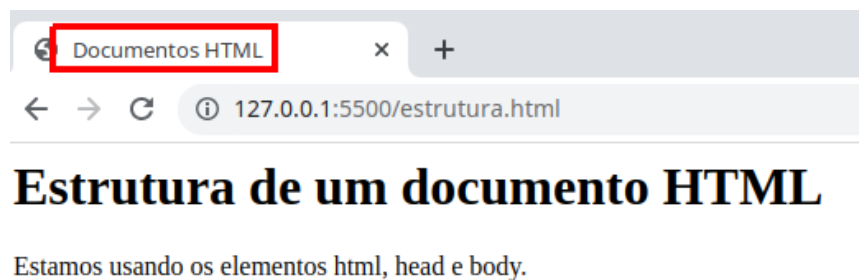
Uma aba de seu navegador deve ser aberta automaticamente. A partir deste momento, temos um servidor web em execução. Ele se encarrega de atualizar a página no navegador a cada alteração feita no código. Veja:



(Metadados definidos no elemento head) Quais informações podemos definir no elemento head? Um exemplo é o título da página:

```
<html>
  <head>
    <title>Documentos HTML</title>
  </head>
  <body>
    <h1>Estrutura de um documento HTML</h1>
    <p>Estamos usando os elementos html, head e body.</p>
  </body>
</html>
```

Após salvar o arquivo, o resultado esperado é esse:



(Encoding) Todo conteúdo digital é representado por uma sequência de bits (zeros e uns), inclusive nossas páginas HTML. Um encoding é um mapeamento de um conjunto de caracteres para o conjunto de sequências de bits que representam cada um deles. Um dos mais comuns e utilizados hoje é o UTF-8. Quando um arquivo é salvo (qualquer arquivo, HTML, txt ou qualquer outro), o editor de texto utilizado faz uso de um encoding para armazenar o conteúdo digitado. Assim, o conteúdo armazenado será a sequência de bits que representa o conteúdo digitado, de acordo com o encoding. Para entender melhor, considere os mapeamentos a seguir. São mapeamentos fictícios, apenas para exemplificar. Ali, mostramos o código binário para três possíveis caracteres.

Encoding 1

Caractere	Codificação
a	001
b	010

c	011
---	-----

Encoding 2

Caractere	Codificação
a	010
b	001
c	011

Agora, considere um arquivo textual cujo conteúdo é o seguinte:

abc

Suponha que ele seja armazenado utilizando-se o encoding 1:

Armazenamento com Encoding 1

001010011

Agora, suponha que o arquivo seja aberto por outro aplicativo, configurado para utilizar o encoding 2. O que ele exibirá?

Conteúdo exibido por aplicativo usando o encoding 2

bac

Isso explica o seguinte fenômeno, veja:

Este é um documento armazenado com encoding utf-8. Escreva seu conteúdo num arquivo novo, chamado **encoding.html** para alguns testes.

```
<html>
<head>
  <title>Encoding</title>
</head>
<body>
  <p>Ação</p>
</body>
</html>
```

Esta é a página sendo exibida por um navegador configurado para utilizar o encoding correto, ou seja, o utf-8:

Ação

Esta é a página sendo exibida por um navegador configurado para utilizar um encoding diferente daquele que foi utilizado para salvar o arquivo. Neste exemplo, usamos o encoding Arabic (ISO-8859-6):

أفءفءو

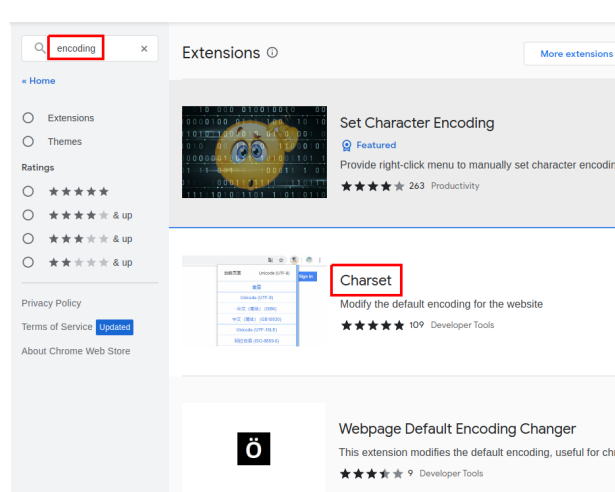
Para saber mais sobre encodings e, em particular, sobre o UTF-8, visite:

<https://en.wikipedia.org/wiki/UTF-8>

(Usando o Google Chrome para abrir um arquivo utf-8 com encoding diferente) Façamos o seguinte teste. Vamos abrir o nosso arquivo armazenado como utf-8 com o Google Chrome, porém, utilizando um encoding diferente. O Google Chrome, desde sua versão 55, removeu a opção de escolha manual do encoding a ser utilizado, segundo a documentação oficial, por conta da opção ser utilizada por pouquíssimos usuários. Segundo sua documentação, ele é capaz de detectar com sucesso o encoding correto a ser usado para tratar o conteúdo das páginas. Assim, para testar o conteúdo deste passo, precisamos instalar uma extensão. Comece visitando este link:

<https://chrome.google.com/webstore/category/extensions>

A seguir, faça a busca por “encoding” e instale o plugin chamado “Charset”. Veja:





Charset

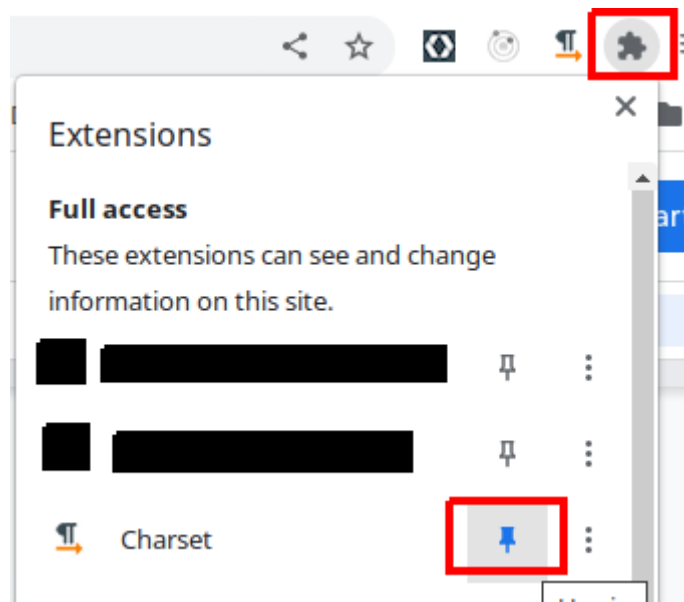
★★★★★ 109 ⓘ

Ferramentas para desenvolvedores

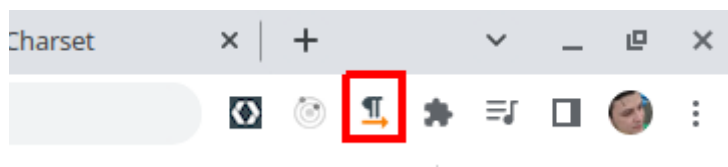
100.000+ usuários

Usar no Chrome

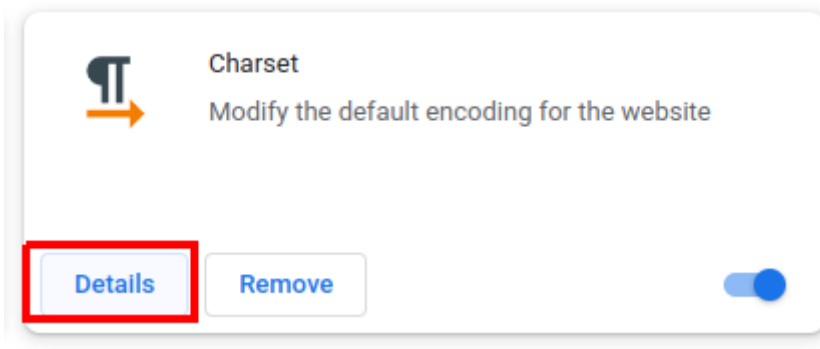
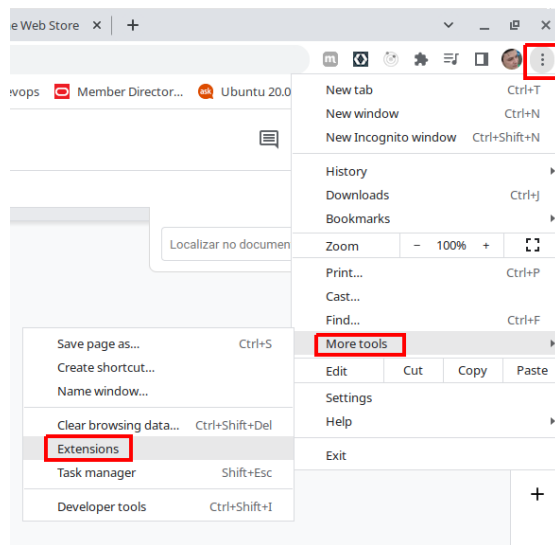
Feita a instalação, podemos usar o plugin para alterar o encoding a ser utilizado pelo navegador para interpretar nossa página. Se necessário, clique no botão de extensões do Chrome (canto superior direito) e escolha a extensão Charset, pois talvez ela não fique visível para você:

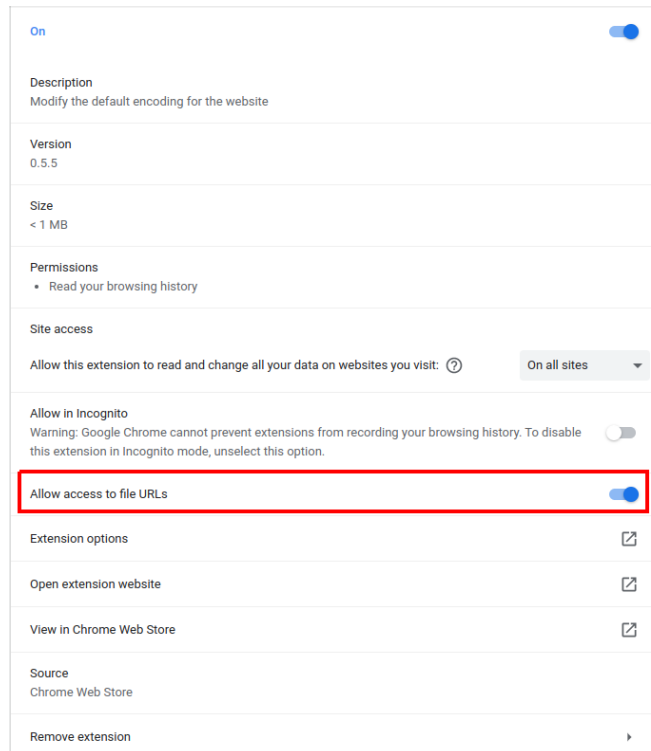


Assim ela deverá ficar fixa no menu de exibição do Google Chrome:



Também pode ser necessário permitir que o plugin acesse arquivos em nosso sistema de arquivos. Para isso, clique no **símbolo de três bolinhas no canto superior direito do Chrome** e então **More Tools >> Extensions**. Na extensão Charset, clique em **Details** e então escolha a opção **Allow access to File URLs**:





Antes de clicar na extensão, vá até a aba em que o seu arquivo .HTML está aberto. Clique no ícone da extensão e escolha um encoding qualquer, como Arabic (ISO-8859-6). O resultado deve ser mais ou menos esse aqui:

Alí?í?o

Repare que os caracteres com acento ficam com exibição incorreta. Para voltar ao normal, escolha UTF-8. Por que UTF-8? Porque, por padrão, o VS Code salvou nosso arquivo com essa codificação. Dá pra ver isso na parte inferior direita dele:



Clique sobre UTF-8 e veja que dá para reabrir o arquivo com outros tipos de encodings. Faça alguns testes para ver os resultados.

A fim de auxiliar o navegador a identificar o encoding utilizado para salvar seu arquivo, podemos especificar um metadado:

```
<html>
  <head>
    <title>Documentos HTML</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>Estrutura de um documento HTML</h1>
    <p>Estamos usando os elementos html, head e body.</p>
  </body>
</html>
```

Essa é apenas uma dica que estamos dando ao navegador do usuário sobre qual encoding foi utilizado para salvar este documento. É claro que essa informação deve ser condizente com aquela que foi de fato utilizada pelo seu editor de texto.

É interessante saber que os navegadores mais modernos são capazes de inferir essa informação a partir do conteúdo inicial dos documentos.

Nota. A notação “charset=utf-8” indica que charset é um **atributo** da tag meta cujo valor associado é utf-8. Utilizaremos a noção de atributos de tags muitas vezes ao longo do curso.

Nota. A notação a seguir era comumente utilizada em versões anteriores do HTML:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

Entretanto, o HTML5 admite também o uso da versão mais simples que especificamos previamente.

(Explicando para o navegador que o documento foi escrito com a versão 5 do HTML) Como vimos, a linguagem HTML possui diversas versões. É comum utilizarmos uma notação específica para explicar para o navegador qual a versão do HTML que está sendo utilizada no documento atual. Como veremos, inclusive, isso será fundamental ao utilizarmos arcabouços CSS como o Bootstrap. Veja como podemos contar para o navegador que estamos utilizando a versão 5 do HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Documentos HTML</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>Estrutura de um documento HTML</h1>
    <p>Estamos usando os elementos html, head e body.</p>
  </body>
</html>
```

Nota. O símbolo **DOCTYPE** não é **case sensitive**. Também funciona se escrevermos algo como **<!DoCtYpe html>**. Mas não faça isso!

Nota. Veja como documentos escritos com outras versões do HTML tinham a sua versão especificada. As URLs especificadas fazem parte do padrão. Não quer dizer que estão sendo acessadas pelo navegador.

HTML 1.1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

HTML 4.01

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

(Preocupação com acessibilidade: especificando o idioma da página) Obviamente, é fundamental que nos preocupemos com a acessibilidade. Pessoas com deficiência visual podem acessar as nossas páginas utilizando dispositivos capazes de ler (e reproduzir sonoramente) seu conteúdo, por exemplo. Há um atributo chamado **lang** que nos permite especificar o idioma utilizado para criar a página. Veja:


```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>Documentos HTML</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>Estrutura de um documento HTML</h1>
    <p>Estamos usando os elementos html, head e body.</p>
  </body>
</html>
```

O que acontece **se escrevermos a página em português e especificarmos que ela está em inglês?** O navegador exibirá a página normalmente. Ou seja, o atributo lang não tem nada a ver com instruir o navegador a traduzir o conteúdo da página, por exemplo. Fazer isso seria uma péssima ideia do ponto de vista da acessibilidade. Dispositivos de acessibilidade tentariam ler o conteúdo da página em inglês, o que traria um péssimo resultado.

Os códigos que representam idiomas/regiões seguem uma especificação ISO. Veja um bom resumo:

<https://gist.github.com/JamieMason/3748498>

(Links entre páginas: elemento a) O funcionamento da web tem como ideia central a ligação (ou links) entre os documentos. O elemento HTML utilizado para representar um link é o elemento **a** (de anchor). Para este exemplo, crie um novo arquivo chamado **homepage.html** com o conteúdo a seguir.

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Página inicial</title>
  </head>
  <body>
    <h1>Página inicial</h1>
    <p>Você está na página inicial. Veja uma página com nomes de
    pessoas.</p>
  </body>
</html>
```

Desejamos que o texto “**página com nomes de pessoas**” seja renderizado como um link. Quando clicado, ele deverá fazer com que uma nova página seja exibida. Para tal, usamos o elemento **a**, da seguinte forma:

- O texto a ser exibido como link para o usuário fica entre as tags de abertura e fechamento
- Especificamos o atributo **href** (de **hypertext reference**) associado ao endereço desejado.

Veja:

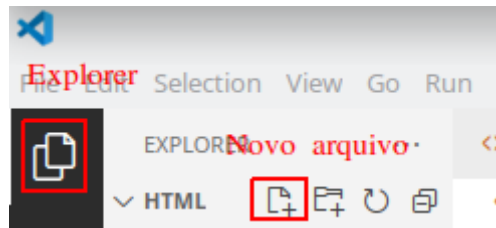
```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Página inicial</title>
  </head>
  <body>
    <h1>Página inicial</h1>
    <p>Você está na página inicial. Veja uma
      <a href="nomes.html">página com nomes de pessoas</a>.
    </p>
  </body>
</html>
```

Observe o resultado esperado:

Página inicial

Você está na página inicial. Veja uma [página com nomes de pessoas](#).

Como ainda não existe um arquivo chamado **nomes.html**, caso clique no link, o navegador deverá exibir uma mensagem parecida com “Cannot GET /nomes.html”. Assim, crie um novo arquivo chamado **nomes.html** na mesma pasta em que você está trabalhando no momento. No Explorer do VS Code, você pode clicar da seguinte forma:



Veja o conteúdo inicial do arquivo:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Nomes</title>
  </head>
  <body>

  </body>
</html>
```

A seguir, vamos especificar uma lista de nomes. Já aproveitamos para exemplificar o uso do elemento **ol** (de **ordered list**) do HTML. Veja:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Nomes</title>
  </head>
  <body>
    <h1>Lista de nomes</h1>
    <ol>
      <li>Antônio</li>
      <li>Daniela</li>
      <li>Jaqueline</li>
      <li>João</li>
    </ol>
  </body>
</html>
```

Volte à página inicial e clique no link. Você deverá ser levado à página recém criada:

Lista de nomes

1. Antônio
2. Daniela
3. Jaqueline
4. João

A página que mostra a lista de nomes pode ter um link que permite ao usuário voltar à página inicial. Façamos a sua definição:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Nomes</title>
  </head>
  <body>
    <h1>Lista de nomes</h1>
    <ol>
      <li>Antônio</li>
      <li>Daniela</li>
      <li>Jaqueline</li>
      <li>João</li>
    </ol>
    <a href="homepage.html">Página inicial</a>
  </body>
</html>
```

Observe o resultado:

Lista de nomes

1. Antônio
2. Daniela
3. Jaqueline
4. João

[Página inicial](#)

Faça alguns testes clicando nos dois links. Observe que temos dois documentos “ligados entre si”, formando uma teia (web).

(Imagens: elemento img) Evidentemente, páginas HTML podem exibir imagens. Para tal, usamos o elemento **img** (de image, como você deve ter desconfiado) da seguinte forma:

- Associamos o endereço da imagem (que pode ser local ou remoto) ao atributo **src** (de source) do elemento img.
- Associamos um texto alternativo que descreve a imagem ao atributo **alt** do elemento img. Este texto será exibido caso o navegador falhe em obter a imagem e também é fundamental do ponto de vista da acessibilidade (um dispositivo que lê o conteúdo da página para um usuário com deficiência visual poderá “falar” para ele do que se trata a imagem).

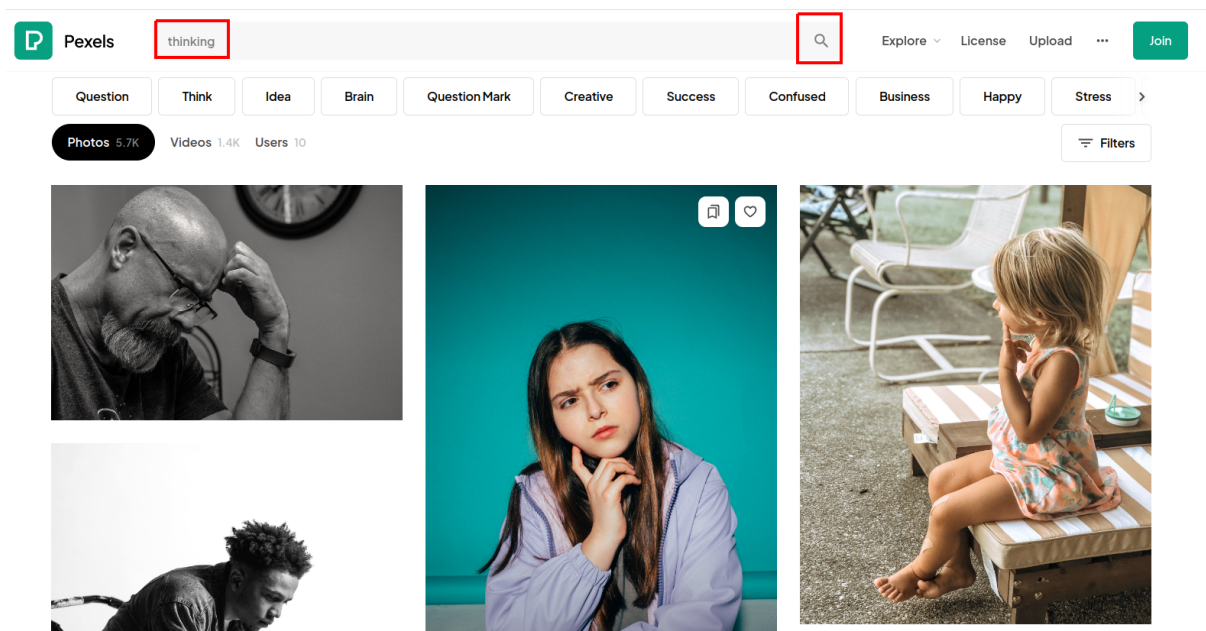
Para fazer os próximos testes, obtenha imagens quaisquer na Internet. Algumas excelentes fontes são o Unsplash, o Pexels e o Pixabay. Veja:

<https://unsplash.com/>

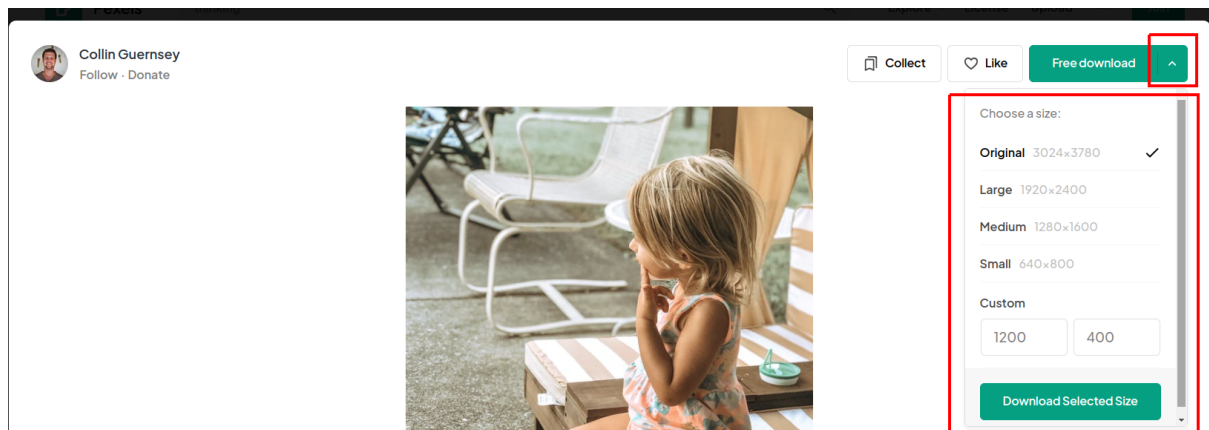
<https://www.pexels.com/>

<https://pixabay.com/>

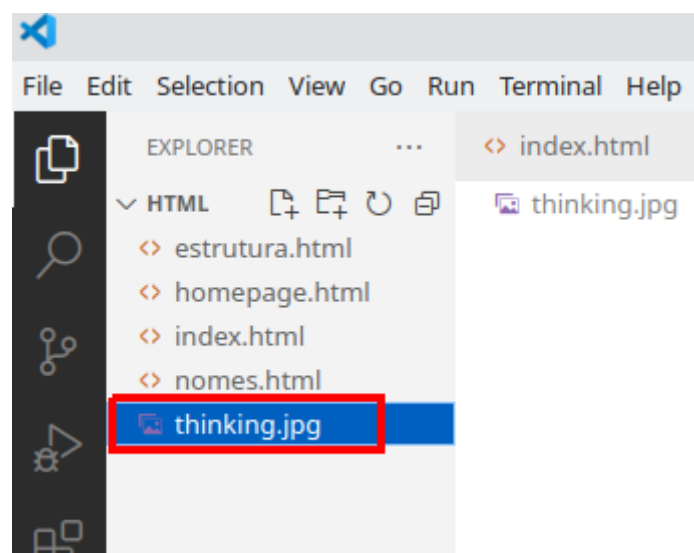
Se você utilizar o Pexels, poderá escolher as dimensões da figura. Primeiro uma busca:



Depois de clicar sobre uma foto que você goste, você pode escolher uma das dimensões pré-definidas ou escolher uma personalizada:



Para este exemplo, escolha a versão “large”. Depois de fazer o download, armazene a figura na mesma pasta em que se encontram os seus arquivos .HTML. Você pode, inclusive, arrastar e soltar a figura do navegador de arquivos para o Explorer do VS Code. O resultado esperado é o seguinte. Observe que, para facilitar, renomeamos o arquivo:



O código fica assim. Ainda estamos no arquivo **homepage.html**:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Página inicial</title>
  </head>
  <body>
    <h1>Página inicial</h1>
    <p>Você está na página inicial. Veja uma
      <a href="nomes.html">página com nomes de pessoas</a>.
    </p>
    
  </body>
</html>
```

Nota. Observe que o elemento `img` não tem tag de fechamento. Ele é um elemento de auto-fechamento. Isso acontece pois ele não pode ter conteúdo ou elementos aninhados. Ele cumpre a sua missão utilizando apenas seus atributos, dispensando um corpo e, por consequência, uma tag de fechamento. Ele também pode ser escrito assim:

``

mas essa barra é dispensável.

A foto que baixamos é muito grande e deve ter tomado a tela quase inteira. Podemos especificar largura e altura da figura com os atributos **width** e **height** do elemento **img**, respectivamente. Veja como reduzir a largura da imagem. Observe que a altura é reduzida de modo proporcional:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Página inicial</title>
  </head>
  <body>
    <h1>Página inicial</h1>
    <p>Você está na página inicial. Veja uma
      <a href="nomes.html">página com nomes de pessoas</a>.
    </p>
    
  </body>
</html>
```

E a altura pode ser alterada assim. Da mesma forma, a largura será alterada automaticamente proporcionalmente:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Página inicial</title>
  </head>
  <body>
    <h1>Página inicial</h1>
    <p>Você está na página inicial. Veja uma
      <a href="nomes.html">página com nomes de pessoas</a>.
    </p>
    
  </body>
</html>
```


Alterar largura e altura simultaneamente pode não ser uma boa ideia. A imagem resultante pode ficar distorcida. Faça esse teste:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Página inicial</title>
  </head>
  <body>
    <h1>Página inicial</h1>
    <p>Você está na página inicial. Veja uma
      <a href="nomes.html">página com nomes de pessoas</a>.
    </p>
    
    </body>
  </html>
```

Veja como a figura ficou distorcida:



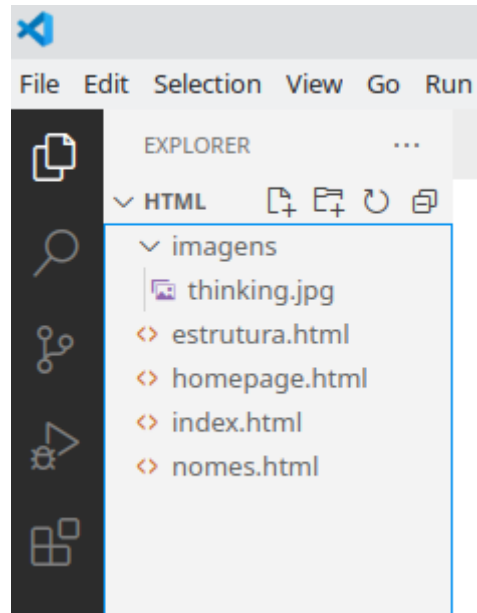
Assim, remova um dos atributos (width ou height). **Neste material, removeremos o height.**

(Fotos remotas) As fotos que nossas páginas utilizam podem ser armazenadas remotamente também. Veja:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Página inicial</title>
  </head>
  <body>
    <h1>Página inicial</h1>
    <p>Você está na página inicial. Veja uma
      <a href="nomes.html">página com nomes de pessoas</a>.
    </p>
    

    
    </body>
</html>
```


(Pasta exclusiva para imagens) É uma excelente prática organizar os arquivos que fazem parte de seu projeto. É bastante comum, por exemplo, ter uma pasta exclusiva para as imagens. Passemos a adotar esta prática. No VS Code, crie uma pasta chamada **imagens** e arraste a figura para dentro dela. Assim:



Observe que agora a imagem não aparece mais no navegador. Como ele não a encontrou, o texto é exibido:

Página inicial

Você está na página inicial. Veja uma [página com nomes de pessoas](#).

 Uma meninininha pensando

Precisamos ajustar a referência à imagem no arquivo **homepage.html**. Veja:

```

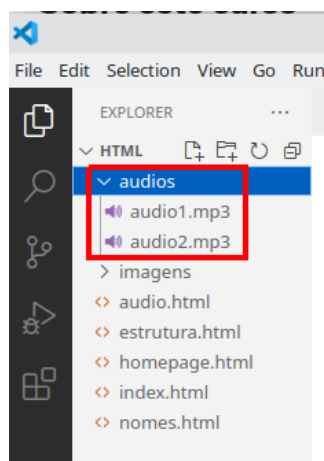
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Página inicial</title>
  </head>
  <body>
    <h1>Página inicial</h1>
    <p>Você está na página inicial. Veja uma
      <a href="nomes.html">página com nomes de pessoas</a>.
    </p>
    
    </body>
  </html>

```

(Arquivos de áudio: elemento audio) Uma página HTML pode dar acesso a arquivos de áudio também. Utilizando o Pixabay, você pode fazer o download de diversos arquivos de áudio. Visite, por exemplo, este link:

[https://pixabay.com/music/search/theme/music%20for%20videos/?genre=electronic&mood=suspense](https://pixabay.com/music/search/theme/music%20for%20videos/?genre=electronic& mood=suspense)

Faça o download de dois arquivos quaisquer ou vasculhe o Pixabay em busca de outros. A seguir, crie uma pasta chamada **audios**, renomeie os arquivos e arraste-os para dentro dessa nova pasta, da seguinte forma:



Podemos incluir um áudio na página da seguinte forma:

- Utilizamos um elemento HTML do tipo **audio** e especificamos o atributo **controls**. Esse atributo indica que os controles (play, volume etc) devem ser exibidos.
- Aninhado dentro do elemento audio, especificamos um elemento do tipo **source**. Seu atributo **src** fica associado ao nome do arquivo de áudio e seu atributo type é utilizado para instruir o navegador sobre qual é o tipo de arquivo sendo utilizado.

Nota. audio/mpeg é um MIME type. Esse assunto será estudado mais adiante. Se estiver muito curioso sobre isso, visite esta página:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types

Para testar os arquivos de áudio, crie um novo arquivo chamado **audio.html**. Veja seu conteúdo:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Músicas</title>
  </head>
  <body>
    <h1>Minhas músicas</h1>

    <p>Ouça algumas músicas que eu gosto.</p>

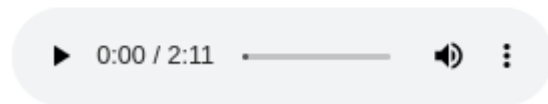
    <audio controls>
      <source src="audios/audio1.mp3" type="audio/mpeg">
    </audio>

  </body>
</html>
```

Veja o resultado esperado:

Minhas músicas

Ouçã algumas músicas que eu gosto.



Nota. A forma como o elemento é apresentado é específica do navegador. Isso pode ser personalizado com Javascript, assunto que está fora de escopo no momento.

Um elemento audio pode ter diversos elementos do tipo source aninhados a ele. Isso pode ser útil caso tenhamos dois arquivos de áudio iguais porém de formato diferente (mp3 e ogg, por exemplo). Especificando os dois arquivos nesta ordem, estamos instruindo o navegador a tocar o arquivo em mp3 e, caso ele não seja capaz de fazê-lo, o que pode ser o caso de navegadores antigos, ele deverá tocar o arquivo em formato ogg. Também podemos especificar um link simples (elemento a) para navegadores mais antigos que não tenham suporte ao elemento audio. Veja:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Músicas</title>
  </head>
  <body>
    <h1>Minhas músicas</h1>

    <p>Ouça algumas músicas que eu gosto.</p>

    <audio controls>
      <source src="audios/audio1.mp3" type="audio/mpeg">
      <a href="audios/audio1.mp3">Áudio 1</a>
    </audio>

    <audio
      controls
      autoplay
      loop
      muted
      src="audios/audio2.mp3"
    >
    </audio>

  </body>
</html>
```

Nota. Se especificarmos diversos arquivos de áudio em diferentes formatos, o navegador não fará o download de todos eles. Ele fará o download um a um, na ordem especificada, até que um deles seja compatível e possa ser reproduzido.

Como estamos usando o Google Chrome e ele tem suporte ao elemento audio, a página não deve ter sido alterada. Abrir esse arquivo com um navegador mais antigo pode resultar em algo parecido com o seguinte. Lembre-se de acessar o endereço

<http://127.0.0.1:5500/audio.html>

no navegador, claro, usando a porta que o Live Server estiver utilizando nos outros exemplos.

Minhas músicas

Ouçã algumas músicas que eu gosto.

[Áudio 1](#)

Observe como podemos simplificar, utilizando o atributo **src** do próprio elemento audio:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Músicas</title>
  </head>
  <body>
    <h1>Minhas músicas</h1>

    <p>Ouça algumas músicas que eu gosto.</p>

    <audio controls>
      <source src="audios/audio1.mp3" type="audio/mpeg">
    </audio>

    <audio controls src="audios/audio2.mp3"></audio>

  </body>
</html>
```


Veja alguns outros exemplos de atributos que um elemento audio pode ter:

- **autoplay**: toca automaticamente assim que a página carrega
- **loop**: fica em modo de repetição
- **muted**: começa "mutado"

Fica assim:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Músicas</title>
  </head>
  <body>
    <h1>Minhas músicas</h1>

    <p>Ouça algumas músicas que eu gosto.</p>

    <audio controls>
      <source src="audios/audio1.mp3" type="audio/mpeg">
    </audio>

    <audio
      controls
      autoplay
      loop
      muted
      src="audios/audio2.mp3"
    >
    </audio>

  </body>
</html>
```

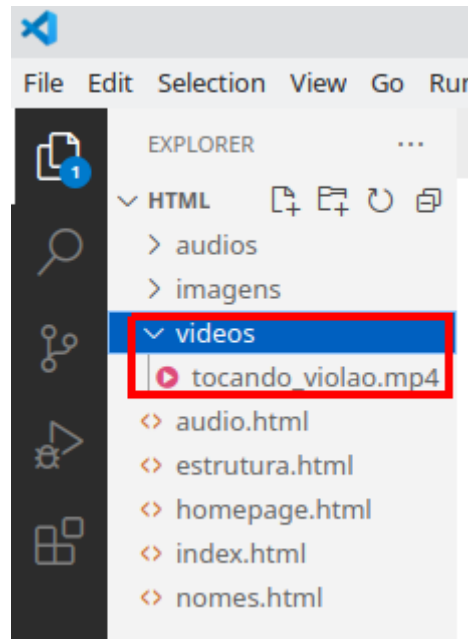
Se quiser saber mais sobre o elemento audio, visite:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio>

(Arquivos de vídeo: elemento video) Para este exemplo, visite o Pixabay e faça o download de um arquivo de vídeo em formato mp4. O que se encontra disponível neste link pode ser uma boa opção:

<https://pixabay.com/videos/alone-person-man-people-guitar-46637/>

Usando o Explorer do VS Code, crie uma pasta chamada **videos**, renomeie o arquivo e arraste-o para lá, da seguinte forma:



A seguir, crie um novo arquivo chamado **video.html**. Veja seu conteúdo inicial:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Meus vídeos</title>
  </head>
  <body>
    <h1>Meus vídeos</h1>
    <p>Veja alguns vídeos que gravei.</p>
  </body>
</html>
```

Para ver os resultados, visite

<http://127.0.0.1:5500/video.html>

no seu navegador.

Observe como a adição de um vídeo à página é semelhante àquela que fizemos com os áudios. Apenas trocamos o elemento do tipo audio por outro de tipo video:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Meus vídeos</title>
  </head>
  <body>
    <h1>Meus vídeos</h1>
    <p>Veja alguns vídeos que gravei.</p>

    <video controls>
      <source src="videos/tocando_violao.mp4" type="video/mp4">
    </video>
  </body>
</html>
```

Também podemos aplicar os atributos **width** e **height** a um elemento de tipo video. Veja:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Meus vídeos</title>
  </head>
  <body>
    <h1>Meus vídeos</h1>
    <p>Veja alguns vídeos que gravei.</p>

    <video width="400" controls>
      <source src="videos/tocando_violao.mp4" type="video/mp4">
    </video>
  </body>
</html>
```

Nota. Esse assunto pode te fazer pensar sobre o conhecido **Flash**. É importante saber que ele foi descontinuado pela Adobe desde 2020 e que ela recomenda que nós, desenvolvedores, utilizemos recursos próprios do HTML5 para esse tipo de manipulação. Se desejar, visite este link:

<https://www.adobe.com/products/flashplayer/end-of-life-alternative.html>

Também podemos adicionar um texto abaixo do elemento source para explicar para o usuário que o seu navegador não tem suporte à manipulação de vídeos. Observe:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Meus vídeos</title>
  </head>
  <body>
    <h1>Meus vídeos</h1>
    <p>Veja alguns vídeos que gravei.</p>

    <video controls>
      <source src="videos/tocando_violao.mp4" type="video/mp4">
      Seu navegador não é capaz de lidar com vídeos.
    </video>
  </body>
</html>
```

O texto somente será exibido por navegadores que não saibam lidar com o elemento video.

Se desejar saber mais sobre o elemento video, visite

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/video>

Observe que no final desta página há uma tabela que mostra quais versões de quais navegadores são compatíveis com esse elemento. Aliás, estas páginas do MDN são ótimas para isso. Fique atento.

(Vídeos do Youtube: elemento iframe) Para inserir um vídeo do Youtube na sua página, use um elemento **iframe** do HTML5. Veja a sua descrição:

The <iframe> HTML element represents a nested browsing context, embedding another HTML page into the current one.

Veja um exemplo de uso.

Nota. O atributo **frameborder** permite especificar se desejamos uma borda no vídeo ou não. O valor 0 indica que não. O valor 1 indica que sim.

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Meus vídeos</title>
  </head>
  <body>
    <h1>Meus vídeos</h1>
    <p>Veja alguns vídeos que gravei.</p>

    <video controls>
      <source src="videos/tocando_violao.mp4" type="video/mp4">
      Seu navegador não é capaz de lidar com vídeos.
    </video>

    <iframe src="https://www.youtube.com/embed/5mF4pKS2SU"
frameborder="0"></iframe>
  </body>
</html>
```

Nota. Quando assistimos um vídeo no Youtube, geralmente o link inclui a palavra **“watch”** e o identificador do vídeo. Para adicionar esse vídeo a uma página usando um iframe, precisamos fazer o seguinte ajuste. Se o link usado para assistir o vídeo é <https://www.youtube.com/watch?v=5mF4pKS2SU>, então o link a ser usado no iframe é <https://www.youtube.com/embed/5mF4pKS2SU>.

Nota. Há discussões sobre o elemento iframe ser obsoleto e seu uso não ser recomendado. Na verdade, seu uso é muito comum e é a principal forma de incorporação de conteúdo de páginas. Um de seus “problemas” é tornar mais difícil que buscadores como o Google “entendam” o conteúdo da página que foi incorporada.

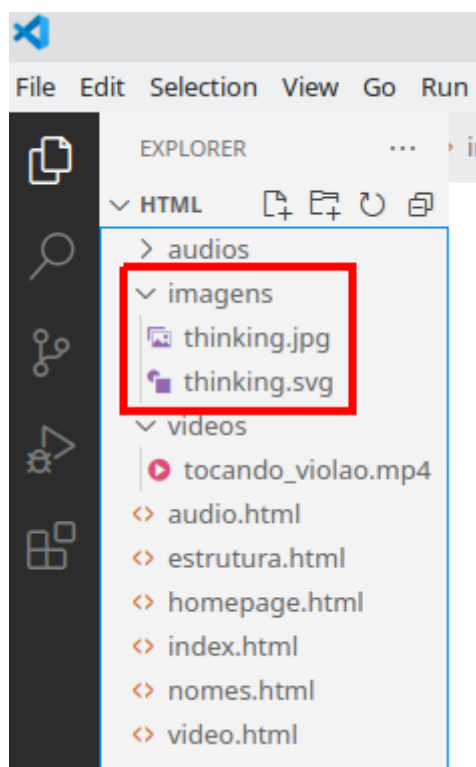
(Figuras SVG) SVG significa **Scalable Vector Graphics**. Uma figura em formato SVG é descrita por meio de “fórmulas matemáticas” utilizando a linguagem XML. Veja uma comparação entre SVG e JPEG.

SVG	JPEG
Descritas textualmente utilizando a linguagem XML	Formato digital composto por uma sequência de bytes que representam os pixels que caracterizam a imagem
Um zoom não faz com que a qualidade seja perdida	Quanto maior for o nível de zoom, mais evidente fica a existência dos pixels e dilatada a imagem
Recomendado sempre que possível: Quando você for criar uma imagem opte por este formato. Exemplos: logo da empresa, personagens de jogo etc.	Utilizado por câmeras digitais para representar fotografias reais, não criadas em computador.

Visite o link a seguir e faça o download do SVG exibido:

<https://freesvg.org/thinking-about-writing>

Arraste-o para a pasta de imagens do seu projeto, além de renomeá-lo:



No VS Code, clique duas vezes sobre o arquivo .SVG. Ele deverá ser aberto textualmente. Observe como a figura é totalmente descrita textualmente utilizando XML. Um **path**, por exemplo, é uma curva livre de um ponto a outro.

Crie um arquivo chamado **teste_svg.html**. Veja seu conteúdo inicial:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>SVG</title>
  </head>
  <body>
    <h1>Exemplo SVG</h1>
    <p>
      Veja uma figura SVG.
    </p>
  </body>
</html>
```

Lembre-se de acessar

http://127.0.0.1:5500/teste_svg.html

para ver o resultado. Como uma figura SVG é apenas texto, podemos simplesmente copiar e colar todo o conteúdo do arquivo thinking.svg para o arquivo teste_svg.html. Observe:

Nota. Não copie a primeira linha. Comece a partir da tag **<svg**. Copie todo o conteúdo a partir daí.


```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>SVG</title>
  </head>
  <body>
    <h1>Exemplo SVG</h1>
    <p>
      Veja uma figura SVG.
    </p>
    <svg
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:cc="http://creativecommons.org/ns#"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    ...
```

Ao invés de copiar e colar o conteúdo inteiro, também podemos usar um simples elemento `img`. Veja:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>SVG</title>
  </head>
  <body>
    <h1>Exemplo SVG</h1>
    <p>
      Veja uma figura SVG.
    </p>
    
  </body>
</html>
```

(Caracteres especiais) Há muitos caracteres que não podemos digitar. Para exibi-los, podemos fazer uso de sequências especiais, previstas na especificação do HTML. Crie um arquivo chamado **caracteres_especiais.html**. Seu conteúdo é o seguinte:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>Caracteres especiais</title>
    <meta charset="utf-8">
  </head>
  <body>
    <p>
      Um "e comercial": &amp;
    </p>
    <p>
      Um símbolo de menor: &lt;
    </p>
    <p>
      Um símbolo de maior: &gt;
    </p>
    <p>
      Raiz quadrada: &#8730;
    </p>
    <p>
      Portanto: &there4;
    </p>
    <p>
      Copyright: &copy;
    </p>
    <p>
      Marca registrada comercial: &reg;
    </p>
  </body>
</html>
```

Visite

http://127.0.0.1:5500/caracteres_especiais.html

para ver o resultado esperado:

Um "e comercial": &

Um símbolo de menor: <

Um símbolo de maior: >

Raiz quadrada: √

Portanto: ∴

Copyright: ©

Marca registrada comercial: ®

Não deixe de visitar o link a seguir para conhecer a lista completa:

<https://html.spec.whatwg.org/multipage/named-characters.html#named-character-references>

(Elementos estruturais) A linguagem HTML possui muitos **elementos estruturais**. Um elemento estrutural, entre outras coisas, serve para:

- organizar outros elementos na página
- associar semântica (mais sobre isso em breve) a regiões da página
- agrupar elementos a fim de aplicar a eles um conjunto de regras de estilos com CSS

A fim de ilustrar seu uso, começamos com uma página fictícia em que algumas vagas de empregos são publicadas. Crie um novo arquivo chamado **elementos_estruturais.html**. Veja seu conteúdo inicial. Não copie e cole. Escreva item a item.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Elementos estruturais</title>
  </head>
  <body>
    <!-- esse é um menu de navegação superior com algumas opções -->
    <ul>
      <li>Home</li>
      <li>Vagas</li>
      <li>Meu cadastro</li>
      <li>Sair</li>
    </ul>

    <!-- esse é o cabeçalho principal da página -->
    <h1>Portal de oportunidades</h1>
    <p>Encontre novas vagas. Atualizado diariamente.</p>

    <!-- essa é uma vaga -->
    <h2>Desenvolvedor Javascript</h2>
    <p>Salário: R$4000</p>
    <p>
      Experiência com desenvolvimento Front End.
    </p>
    <p>
      Data de publicação: 19/04/2022
    </p>

    <!-- mais uma vaga -->
    <h2>Desenvolvedor Java</h2>
    <p>Salário: R$4200</p>
    <p>
      Experiência com desenvolvimento Back End.
    </p>
    <p>
      Data de publicação: 22/04/2022
    </p>
```

```
<!-- barra lateral com anúncios -->

<!-- primeiro anúncio -->
Lorem ipsum dolor sit amet consectetur adipisicing elit.
Quibusdam, est.

<!-- segundo anúncio -->
Lorem ipsum dolor sit amet consectetur adipisicing elit.
Voluptatibus cumque odio hic, nesciunt ipsa sit.

<!-- rodapé da página -->
<p>&copy; 2022 - Todos os direitos reservados</p>

</body>
</html>
```

Lembre-se de visitar

http://127.0.0.1:5500/elementos_estruturais.html

para ver o resultado:

- Home
- Vagas
- Meu cadastro
- Sair

Portal de oportunidades

Encontre novas vagas. Atualizado diariamente.

Desenvolvedor Javascript

Salário: R\$4000

Experiência com desenvolvimento Front End.

Data de publicação: 19/04/2022

Desenvolvedor Java

Salário: R\$4200

Experiência com desenvolvimento Back End.

Data de publicação: 22/04/2022

Lorem ipsum dolor sit amet consectetur adipisicing elit. Quibusdam, est.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptatibus cumque odio hic, nesciunt ipsa sit.

© 2022 - Todos os direitos reservados

(Elementos estruturais sem semântica: div e span) É muito comum que desenvolvedores utilizem elementos estruturais sem semântica a fim de organizar as suas páginas e de agrupar elementos para aplicar-lhes estilos com CSS. Os mais comuns são `div` (de **division**) e `span` (vem do verbo **span** mesmo e passa a ideia de “um depois do outro”). Veja a sua definição no MDN:

div: The `<div>` HTML element is the generic container for flow content. It has no effect on the content or layout until styled in some way using CSS. As a "pure" container, the `<div>` element does not inherently represent anything.

span: The `` HTML element is a generic inline container for phrasing content, which does not inherently represent anything.

Observe como eles não têm um significado, uma semântica. Isso é ruim para desenvolvedores que lêem o código e especialmente ruim para leitores de tela. Como os elementos não têm significado, um leitor de tela não consegue identificar com clareza a sua razão de ser. Por exemplo: um `div` é uma seção lateral ou o conteúdo principal da página?

Nota. Essencialmente, `div` e `span` têm o mesmo propósito. A principal diferença está na sua forma de exibição. Enquanto um `div` toma a tela inteira horizontalmente, um `span` ocupa somente o espaço necessário para o seu conteúdo, admitindo que mais conteúdo seja exibido do seu lado. Isso ficará mais claro quando estudarmos a propriedade `display` do CSS.

Digamos que desejamos aplicar regras de estilos às principais regiões da página. Para tal, aplicaremos elementos estruturais. Neste primeiro exemplo, utilizaremos apenas elementos `div`. Veja como pode ficar:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Elementos estruturais</title>
  </head>
  <body>
    <!-- esse é um menu de navegação superior com algumas opções -->
    <div>
      <ul>
        <li>Home</li>
        <li>Vagas</li>
        <li>Meu cadastro</li>
        <li>Sair</li>
      </ul>
    </div>
    <!-- conteúdo principal da página -->
    <div>
      <!-- esse é o cabeçalho principal da página -->
      <h1>Portal de oportunidades</h1>
      <p>Encontre novas vagas. Atualizado diariamente.</p>

      <!-- agrupamento de vagas -->
      <div>
        <!-- essa é uma vaga -->
        <h2>Desenvolvedor Javascript</h2>
        <p>Salário: R$4000</p>
        <p>
          Experiência com desenvolvimento Front End.
        </p>
        <p>
          Data de publicação: 19/04/2022
        </p>

        <!-- mais uma vaga -->
        <h2>Desenvolvedor Java</h2>
        <p>Salário: R$4200</p>
        <p>
          Experiência com desenvolvimento Back End.
        </p>
        <p>
```

```
        Data de publicação: 22/04/2022
    </p>
</div>
</div>

<!-- barra lateral com anúncios -->
<div>
    <!-- primeiro anúncio -->
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
Quibusdam, est.</p>
    <!-- segundo anúncio -->
    <p> Lorem ipsum dolor sit amet consectetur adipisicing elit.
Voluptatibus cumque odio hic, nesciunt ipsa sit.</p>
</div>

<!-- rodapé da página -->
<div>
    <p>&copy; 2022 - Todos os direitos reservados</p>
</div>

</body>
</html>
```

Repare mais uma vez como o uso dos elementos div não é muito bom do ponto de vista da semântica da página. Como saber que o último div, por exemplo, é o seu rodapé?

(Usando classes ou ids) Há a possibilidade de atribuir classes ou um id a elementos div tornando seu significado mais claro. Veja um exemplo com ids.


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Elementos estruturais</title>
  </head>
  <body>
    <!-- esse é um menu de navegação superior com algumas opções -->
    <div id="menu-navegacao">
      <ul>
        <li>Home</li>
        <li>Vagas</li>
        <li>Meu cadastro</li>
        <li>Sair</li>
      </ul>
    </div>
    <!-- conteúdo principal da página -->
    <div id="principal">
      <!-- esse é o cabeçalho principal da página -->
      <h1>Portal de oportunidades</h1>
      <p>Encontre novas vagas. Atualizado diariamente.</p>

      <!-- agrupamento de vagas -->
      <div id="vagas">
        <!-- essa é uma vaga -->
        <h2>Desenvolvedor Javascript</h2>
        <p>Salário: R$4000</p>
        <p>
          Experiência com desenvolvimento Front End.
        </p>
        <p>
          Data de publicação: 19/04/2022
        </p>

        <!-- mais uma vaga -->
        <h2>Desenvolvedor Java</h2>
        <p>Salário: R$4200</p>
        <p>
          Experiência com desenvolvimento Back End.
        </p>
        <p>
```

```
        Data de publicação: 22/04/2022
    </p>
</div>
</div>

<!-- barra lateral com anúncios -->
<div id="anuncios">
    <!-- primeiro anúncio -->
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
Quibusdam, est.</p>
    <!-- segundo anúncio -->
    <p> Lorem ipsum dolor sit amet consectetur adipisicing elit.
Voluptatibus cumque odio hic, nesciunt ipsa sit.</p>
</div>

<!-- rodapé da página -->
<div id="rodape">
    <p>&copy; 2022 - Todos os direitos reservados</p>
</div>

</body>
</html>
```

Repare que já melhorou mas talvez não seja o suficiente. Os ids usados seguem uma convenção própria da equipe responsável por essa página. Outros desenvolvedores podem adotar convenções diferentes. Também, os ids estão em um idioma específico. Talvez leitores de tela não sejam capazes de lidar com esse idioma.

(Elementos estruturais com semântica) O uso de divs foi o cenário mais comum até o momento em que surgiram os elementos estruturais com semântica. O seu uso não causa efeito visual nenhum. Entretanto, cada um possui uma semântica ou significado previamente estabelecido pela especificação. Assim, desenvolvedores e leitores de tela sempre saberão o que cada um representa. A seguir, ajustemos a nossa página para que ela faça uso deles.

(Elemento nav) O primeiro elemento estrutural que conheceremos é o elemento nav. Veja a sua descrição no MDN:

The <nav> HTML element represents a section of a page whose purpose is to provide navigation links, either within the current document or to other documents. Common examples of navigation sections are menus, tables of contents, and indexes.

Ele parece uma boa opção para a barra de navegação, não? Pode ficar assim:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Elementos estruturais</title>
  </head>
  <body>
    <!-- esse é um menu de navegação superior com algumas opções -->
    <nav>
      <ul>
        <li>Home</li>
        <li>Vagas</li>
        <li>Meu cadastro</li>
        <li>Sair</li>
      </ul>
    </nav>
    <!-- conteúdo principal da página -->
    <div id="principal">
      ...
```

(Elemento main) A seguir, temos a parte principal da página. Veja o significado do elemento main:

The <main> HTML element represents the dominant content of the <body> of a document. The main content area consists of content that is directly related to or expands upon the central topic of a document, or the central functionality of an application.

Apliquemo-lo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Elementos estruturais</title>
  </head>
  <body>
    <!-- esse é um menu de navegação superior com algumas opções -->
    <nav>
      <ul>
        <li>Home</li>
        <li>Vagas</li>
        <li>Meu cadastro</li>
        <li>Sair</li>
      </ul>
    </nav>
    <!-- conteúdo principal da página -->
    <main>
      <!-- esse é o cabeçalho principal da página -->
      <h1>Portal de oportunidades</h1>
      ...
    </main>

    <!-- barra lateral com anúncios -->
    <div id="anuncios">
      ...
    </div>
  </body>
</html>
```

(Elemento header) A página possui um cabeçalho principal composto por um elemento h1 e outro p. O elemento header pode abrigá-los muito bem. Veja a sua definição:

The header element represents introductory content for its nearest ancestor sectioning content or sectioning root element. A header typically contains a group of introductory or navigational aids. When the nearest ancestor sectioning content or sectioning root element is the body element, then it applies to the whole page.

Olha como fica:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Elementos estruturais</title>
  </head>
  <body>
    <!-- esse é um menu de navegação superior com algumas opções -->
    <nav>
      <ul>
        <li>Home</li>
        <li>Vagas</li>
        <li>Meu cadastro</li>
        <li>Sair</li>
      </ul>
    </nav>
    <!-- conteúdo principal da página -->
    <main>
      <header>
        <!-- esse é o cabeçalho principal da página -->
        <h1>Portal de oportunidades</h1>
        <p>Encontre novas vagas. Atualizado diariamente.</p>
      </header>

      <!-- agrupamento de vagas -->
      <div id="vagas">
        ...

```

(Elemento section) Talvez utilizemos o elemento **section** para abrigar a coleção de vagas. Olha a sua definição:

The section element represents a generic section of a document or application. A section, in this context, is a thematic grouping of content. Each section should be identified, typically by including a heading (h1- h6 element) as a child of the section element.

Observe que um section é um “grupo temático de conteúdo” e que, tipicamente, possui um cabeçalho. Além de trocar a div por um section, vamos também adicionar o cabeçalho. Veja:

```

...
<main>
  <header>
    <!-- esse é o cabeçalho principal da página -->
    <h1>Portal de oportunidades</h1>
    <p>Encontre novas vagas. Atualizado diariamente.</p>
  </header>

  <!-- agrupamento de vagas -->
  <section>
    <h1>Vagas de hoje</h1>
    <!-- essa é uma vaga -->
    <h2>Desenvolvedor Javascript</h2>
    <p>Salário: R$4000</p>
    <p>
      Experiência com desenvolvimento Front End.
    </p>
    <p>
      Data de publicação: 19/04/2022
    </p>

    <!-- mais uma vaga -->
    <h2>Desenvolvedor Java</h2>
    <p>Salário: R$4200</p>
    <p>
      Experiência com desenvolvimento Back End.
    </p>
    <p>
      Data de publicação: 22/04/2022
    </p>
  </section>
</main>

```

(Cada vaga tem cabeçalho, conteúdo e rodapé: article, header, p e footer). Observe como a especificação de uma vaga será um tanto sofisticada. Os elementos que utilizaremos agora e que não utilizamos ainda junto com as suas especificações são os seguintes.

article: The <article> HTML element represents a self-contained composition in a document, page, application, or site, which is intended to be independently distributable or reusable (e.g., in syndication). Examples include: a forum post, a magazine or newspaper article, or a blog entry, a product card, a user-submitted comment, an interactive widget or gadget, or any other independent item of content.

footer: The <footer> HTML element represents a footer for its nearest ancestor sectioning content or sectioning root element. A <footer> typically contains information about the author of the section, copyright data or links to related documents.

Observe que cada vaga pode ter o seu próprio rodapé. Não é algo exclusivo da página inteira. Veja como podemos fazer:

```
...
<!-- agrupamento de vagas -->
  <section>
    <h1>Vagas de hoje</h1>
    <!-- essa é uma vaga -->
    <article>
      <header>
        <h2>Desenvolvedor Javascript</h2>
      </header>
      <p>Salário: R$4000</p>
      <p>
        Experiência com desenvolvimento Front End.
      </p>
      <footer>
        <p>
          Data de publicação: 19/04/2022
        </p>
      </footer>
    </article>

    <!-- mais uma vaga -->
    <article>
      <header>
        <h2>Desenvolvedor Java</h2>
      </header>
      <p>Salário: R$4200</p>
      <p>
        Experiência com desenvolvimento Back End.
      </p>
```

```

        <footer>
            <p>
                Data de publicação: 22/04/2022
            </p>
        </footer>
    </article>
</section>
...

```

(Elemento aside) Veja a definição do elemento aside.

The <aside> HTML element represents a portion of a document whose content is only indirectly related to the document's main content. Asides are frequently presented as sidebars or call-out boxes.

Que tal utilizarmos ele para a barra de anúncios? Observe:

```

...
</section>
</main>
<!-- barra lateral com anúncios -->
<aside>
    <!-- primeiro anúncio -->
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
        Quibusdam, est.</p>
    <!-- segundo anúncio -->
    <p> Lorem ipsum dolor sit amet consectetur adipisicing elit.
        Voluptatibus cumque odio hic, nesciunt ipsa sit.</p>
</aside>

<!-- rodapé da página -->
<div id="rodape">
    <p>&copy; 2022 - Todos os direitos reservados</p>
</div>

</body>
</html>

```


(Rodapé da página) Finalmente, especificamos o rodapé da página, usando um footer mais uma vez:

```
...
</aside>

    <!-- rodapé da página -->
    <footer>
        <p>&copy; 2022 - Todos os direitos reservados</p>
    </footer>

</body>
</html>
```

