



# O Manual HTML

## Índice

- [Índice](#)
- [1. Prefácio](#)
- [2.HTML Noções básicas](#)
- [2.1.HTML estrutura da página](#)
- [2.2. Tags vs elementos](#)
- [2.3. Atributos](#)
- [2.4. Insensível a maiúsculas e minúsculas](#)
- [2.5. Espaço em branco](#)
- [3. O título do documento](#)
- [3.1. A tag de título](#)
- [3.2. A tag de script](#)
- [3.3. A tag noscript](#)
- [3.4. A tag de link](#)
- [3.5. A tag de estilo](#)
- [3.6. A etiqueta base](#)
- [3.7. A meta tag](#)
- [4. O corpo do documento](#)
- [4.1. Elementos de bloco vs elementos embutidos](#)
- [5. Tags que interagem com o texto](#)
- [5.1. A tag\\_p](#)
- [5.2. A tag\\_span](#)
- [5.3. A tag\\_br](#)
- [5.4. As etiquetas de cabeçalho](#)
- [5.5. A etiqueta forte](#)

St Adobe Stock

Obtenha 10 fotos gratuitas do Adobe Stock. Comece a baixar incríveis fotos royalty-free hoje mesmo.

ANÚNCIOS VIA CARBON

- [5.6. A etiqueta em](#)
- [5.6.1. Cotações](#)
- [5.7. Linha horizontal](#)
- [5.8. Blocos de código](#)
- [5.9. Listas](#)
- [5.10. Outras etiquetas de texto](#)
- [6. Ligações](#)
- [7. Tags de contêiner e HTML de estrutura de página](#)
- [7.1. Etiquetas de contentor](#)
- [7.1.1. Artigo](#)
- [7.1.2. secção](#)
- [7.1.3. div](#)
- [7.2. Tags relacionadas à página](#)
- [7.2.1. navegação](#)
- [7.2.2. Anulação](#)
- [7.2.3. cabeçalho](#)
- [7.2.4. Principal](#)
- [7.2.5. rodapé](#)
- [8. Formulários](#)
- [8.1. A etiqueta de entrada](#)
- [8.1.1. E-mail](#)
- [8.1.2. Palavra-passe](#)
- [8.1.3. Números](#)
- [8.1.4. Campo oculto](#)
- [8.1.5. Definindo um valor padrão](#)
- [8.2. Envio do formulário](#)
- [8.3. Validação do formulário](#)
- [8.3.1. Definir campos conforme necessário](#)
- [8.3.2. Impor um formato específico](#)
- [8.4. Outros campos](#)
- [8.4.1. Uploads de arquivos](#)
- [8.4.2. Botões](#)
- [8.4.3. Botões de opção](#)
- [8.4.4. Caixas de verificação](#)
- [8.4.5. Data e hora](#)
- [8.4.6. Seletor de cores](#)
- [8.4.7. Gama](#)

- [8.4.8. Telefone](#)
- [8.4.9. URL](#)
- [8.5. A tag `textarea`](#)
- [8.6. A tag de seleção](#)
- [9. Tabelas](#)
- [9.0.1. A etiqueta da tabela](#)
- [9.0.2. Linhas](#)
- [9.0.3. Cabeçalhos de coluna](#)
- [9.0.4. Conteúdo do quadro](#)
- [9.0.5. Estender colunas e linhas](#)
- [9.0.6. Cabeçalhos de linha](#)
- [9.0.7. Mais tags para organizar a tabela](#)
- [9.1. Legenda da tabela](#)
- [10. Tags multimídia: áudio e vídeo](#)
- [10.1. A etiqueta de áudio](#)
- [10.2. A etiqueta de vídeo](#)
- [11. iframes](#)
- [11.1. Srcdoc](#)
- [11.2. Caixa de areia](#)
- [11.3. Permitir](#)
- [11.4. Referenciador](#)
- [12. Imagens](#)
- [12.1. A etiqueta da figura](#)
- [12.2. Imagens responsivas usando srcset](#)
- [12.3. A etiqueta de imagem](#)
- [13. Acessibilidade](#)
- [13.1. Usar HTML semântico](#)
- [13.2. Usar atributos alt para imagens](#)
- [13.3. Usar o atributo role](#)
- [13.4. Use o atributo tabindex](#)
- [13.5. Usar os atributos aria](#)
- [13.5.1. Rótulo de ária](#)
- [13.5.2. Rotulado com ária](#)
- [13.5.3. Ária descrita](#)
- [13.5.4. Usar ária oculta para ocultar conteúdo](#)

# 1. Prefácio

Este livro tem como objetivo ajudá-lo a aprender HTML rapidamente e se familiarizar com os tópicos avançados de HTML.

HTML, uma abreviação de Hyper Text Markup Language, é um dos blocos de construção mais fundamentais da Web.

O HTML nasceu oficialmente em 1993 e, desde então, evoluiu para o seu estado atual, passando de simples documentos de texto para alimentar aplicativos Web ricos.

Este manual destina-se a um vasto público.

Primeiro, o iniciante. Eu explico HTML do zero de uma forma sucinta, mas abrangente, para que você possa usar este livro para aprender HTML a partir do básico.

Depois, o profissional. HTML é muitas vezes considerado como uma coisa secundária para aprender. Pode ser dado como certo.

No entanto, muitas coisas são obscuras para muitas pessoas. Eu incluído. Eu escrevi este manual para ajudar a minha compreensão do tópico, porque quando eu preciso explicar algo, é melhor eu me certificar de que eu primeiro sei a coisa de dentro para fora.

Mesmo que você não escreva HTML no seu trabalho do dia a dia, saber como o HTML funciona pode ajudar a poupar algumas dores de cabeça quando você precisa entendê-lo de tempos em tempos, por exemplo, ao ajustar uma página da web.

HTML é a base da maravilha chamada Web.

Há um poder incrível por trás desse conjunto bastante simples e limitado de regras, que nos permite – desenvolvedores, criadores, designers, escritores e funileiros – criar documentos, aplicativos e experiências para pessoas de todo o mundo.

Meu primeiro livro em HTML saiu em 1997 e foi chamado de "HTML Unleashed". Um grande, muitas páginas, longo tomo.

Mais de 20 anos se passaram, e o HTML ainda é a base da Web, com mudanças mínimas a partir de então.

Claro, temos mais tags semânticas, HTML de apresentação não é mais uma coisa, e CSS cuidou do design das coisas.

O sucesso do HTML é baseado em uma coisa: **simplicidade**.

Ele resistiu a ser sequestrado em um dialeto XML via XHTML, quando eventualmente as pessoas perceberam que a coisa era muito complexa.

Fê-lo por causa de outra característica que nos proporciona: **o perdão**. Existem *algumas* regras, certo, mas depois de aprendê-las, você tem muita liberdade.

Os navegadores aprenderam a ser resilientes e a sempre tentar fazer o seu melhor ao analisar e apresentar HTML aos usuários.

E toda a plataforma Web fez uma coisa certa: nunca quebrou a compatibilidade com versões anteriores. Incrivelmente, podemos voltar aos documentos HTML escritos em 1991, e eles parecem praticamente como pareciam naquela época.

Nós até sabemos qual foi a primeira página da web. É o seguinte:  
<http://info.cern.ch/hypertext/WWW/TheProject.html>

E você pode ver a fonte da página, graças a outra grande característica da Web e HTML: **podemos inspecionar o HTML de qualquer página da web**.

Não tome isso como garantido. Não conheço nenhuma outra plataforma que nos dê essa capacidade.

As excepcionais Ferramentas de Desenvolvedor embutidas em qualquer navegador nos permitem inspecionar e nos inspirar em HTML escrito por qualquer pessoa no mundo.

Se você é novo em HTML, este livro tem como objetivo ajudá-lo a começar. Se você é um desenvolvedor Web experiente, este livro irá melhorar o seu conhecimento.

Eu aprendi muito enquanto o escrevia, mesmo que eu tenha trabalhado com a Web por mais de 20 anos, e tenho certeza de que você encontrará algo novo também.

Ou você vai repreender algo antigo que você esqueceu.

De qualquer forma, o objetivo do livro é ser útil para você, e espero que tenha sucesso.

## 2.HTML Noções básicas

HTML é um padrão definido pelo **WHATWG**, um acrônimo para Web Hypertext Application Technology Working Group, uma organização formada por pessoas que trabalham no navegador da Web mais popular. Isso significa que é basicamente controlado pelo Google, Mozilla, Apple e Microsoft.

No passado, o **W3C** (World Wide Web Consortium) era a organização encarregada de criar o padrão HTML.

O controle mudou informalmente do W3C para o WHATWG quando ficou claro que o impulso do W3C para o XHTML não era uma boa ideia.

Se você nunca ouviu falar de XHTML, aqui está um conto. No início dos anos 2000, todos nós acreditávamos que o futuro da Web era XML (sério). Assim, o HTML passou de uma linguagem de criação baseada em SGML para uma linguagem de marcação XML.

Foi uma grande mudança. Tínhamos que conhecer, e respeitar, mais regras. Regras mais rígidas.

Eventualmente, os fornecedores de navegadores perceberam que esse não era o caminho certo para a Web e recuaram, criando o que agora é conhecido como HTML5.

O W3C realmente não concordou em desistir do controle do HTML, e por anos tivemos 2 padrões concorrentes, cada um com o objetivo de ser o oficial. Eventualmente, em 28 de maio de 2019, foi oficializado pelo W3C que a versão HTML "verdadeira" era a publicada pela WHATWG.

Eu mencionei HTML5. Deixe-me explicar esta pequena história. Eu sei, é meio confuso até agora, como acontece com muitas coisas na vida quando muitos atores estão envolvidos, mas também é fascinante.

Tínhamos **HTML versão 1** em 1993. Aqui está a RFC original.

**HTML 2** seguido em 1995.

Obtivemos o **HTML 3** em janeiro de 1997 e o **HTML 4** em dezembro de 1997.

Horários de pico!

Mais de 20 anos se passaram, tivemos toda essa coisa de XHTML e, eventualmente, chegamos a essa "coisa" HTML5, que não é mais *apenas* HTML.

HTML5 é um termo que agora define todo um conjunto de tecnologias, que inclui HTML, mas adiciona muitas APIs e padrões como WebGL, SVG e muito mais.

A principal coisa a entender aqui é o seguinte: não existe tal coisa (mais) como uma versão HTML agora. É um padrão de vida. Como o CSS, que é chamado de "3", mas na realidade é um monte de módulos independentes desenvolvidos separadamente. Como o JavaScript, onde temos uma nova edição a cada ano, mas hoje em dia, a única coisa que importa é quais recursos individuais são implementados pelo mecanismo.

Yes we call it HTML5, but HTML4 is from 1997. That's a long time for anything, let alone for the web.

This is where the standard now "lives":

<https://html.spec.whatwg.org/multipage>.

HTML is the markup language we use to structure content that we consume on the Web.

HTML is served to the browser in different ways.

- It can be generated by a server-side application that builds it depending on the request or the session data, for example a Rails or Laravel or Django application.
- It can be generated by a JavaScript client-side application that generates HTML on the fly.
- In the simplest case, it can be stored in a file and served to the browser by a Web server.

Let's dive into this last case. Although in practice it's probably the least popular way to generate HTML, it's still essential to know the basic building blocks.

By convention, an HTML file is saved with a .html extension..html.htm

Inside this file, we organize the content using **tags**.

Tags wrap the content, and each tag gives a special meaning to the text it wraps.

Let's make a few examples.

This HTML snippet creates a paragraph using the tag:p

```
<p>A paragraph of text</p>
```

This HTML snippet creates a list of items using the tag, which means *unordered list*, and the tags, which mean *list item*:ul li

```
<ul>
  <li>First item</li>
  <li>Second item</li>
```

```
<li>Third item</li>
</ul>
```

Quando uma página HTML é servida pelo navegador, as tags são interpretadas e o navegador renderiza os elementos de acordo com as regras que definem sua aparência visual.

Algumas dessas regras são internas, como a forma como uma lista é renderizada ou como um link é sublinhado em azul.

Algumas outras regras são definidas por você com CSS.

HTML não é de apresentação. Não está preocupado com a *aparência das coisas*. Em vez disso, está preocupado com o que as coisas *significam*.

Cabe ao navegador determinar a aparência das coisas, com as diretivas definidas por quem constrói a página, com a linguagem CSS.

Agora, esses dois exemplos que fiz são trechos HTML tirados fora do contexto de uma página.

## 2.1.HTML estrutura da página

Vamos fazer um exemplo de uma página HTML adequada.

As coisas começam com a Declaração de Tipo de Documento (também conhecida como *doctype*), uma maneira de dizer ao navegador que esta é uma página HTML e qual versão do HTML estamos usando.

O HTML moderno usa este tipo de documento:

```
<!DOCTYPE html>
```

Então temos o elemento, que tem uma tag de abertura e fechamento:`html`

```
<!DOCTYPE html>
<html>
```

```
...  
</html>
```

A maioria das tags vem em pares com uma tag de abertura e uma tag de fechamento. A tag de fechamento é escrita da mesma forma que a tag de abertura, mas com um :/

```
<sometag>some content</sometag>
```

Existem algumas tags de fechamento automático, o que significa que elas não precisam de uma tag de fechamento separada, pois não contêm nada *nelas*.

A marca inicial é usada no início do documento, logo após a declaração de tipo de documento.html

A marca final é a última coisa presente em um documento HTML.html

Dentro do elemento temos 2 elementos: e :htmlheadbody

```
<!DOCTYPE html>  
<html>  
  <head>  
    ...  
  </head>  
  <body>  
    ...  
  </body>  
</html>
```

Dentro teremos tags que são essenciais para a criação de uma página web, como o título, os metadados e CSS e JavaScript internos ou externos.

Principalmente coisas que não aparecem diretamente na página, mas apenas ajudam o navegador (ou bots como o bot de pesquisa do Google) a exibi-lo corretamente.head

Dentro teremos o conteúdo da página. As **coisas visíveis**.body

## 2.2. Tags vs elementos

Eu mencionei tags e elementos. Qual a diferença?

Os elementos têm uma tag inicial e uma tag de fechamento. Neste exemplo, usamos as marcas inicial e de fechamento para criar um elemento:`pp`

```
<p>A paragraph of text</p>
```

Assim, um elemento constitui todo o *pacote*:

- tag inicial
- conteúdo do texto (e possivelmente outros elementos)
- tag de fechamento

Se um elemento não tiver uma tag de fechamento, ele só será escrito com a tag inicial e não poderá conter nenhum conteúdo de texto.

Dito isto, eu poderia usar o termo tag ou elemento no livro significando a mesma coisa, exceto se eu mencionar explicitamente a tag inicial ou a tag final.

## 2.3. Atributos

A tag inicial de um elemento pode ter trechos especiais de informações que podemos anexar, chamados **atributos**.

Os atributos têm a sintaxe:`key="value"`

```
<p class="a-class">A paragraph of text</p>
```

Você também pode usar aspas simples, mas usar aspas duplas em HTML é uma boa convenção.

Podemos ter muitos deles:

```
<p class="a-class" id="an-id">A paragraph of text</p>
```

e alguns atributos são booleanos, o que significa que você só precisa da chave:

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

Os atributos e são dois dos mais comuns que você encontrará usados.`class``id`

Eles têm um significado especial e são úteis tanto em CSS quanto em JavaScript.

A diferença entre os dois é que um é único no contexto de uma página da web; ele não pode ser duplicado.`id`

As classes, por outro lado, podem aparecer várias vezes em vários elementos.

Além disso, um é apenas um valor. pode conter vários valores, separados por um espaço:`id``class`

```
<p class="a-class another-class">A paragraph of text</p>
```

É comum usar o traço para separar palavras em um valor de classe, mas é apenas uma convenção.-

Esses são apenas dois dos atributos possíveis que você pode ter. Alguns atributos são usados apenas para uma tag. Eles são altamente especializados.

Outros atributos podem ser usados de uma maneira mais geral. Você acabou de ver `e`, mas temos outros também, como `o` que pode ser usado para inserir regras CSS embutidas em um elemento.`id``class``style`

## 2.4. Insensível a maiúsculas e minúsculas

HTML não diferencia maiúsculas de minúsculas. As tags podem ser escritas em maiúsculas ou minúsculas. Nos primeiros dias, os bonés eram a norma. Hoje minúsculas são a norma. É uma convenção.

Você geralmente escreve assim:

```
<p>A paragraph of text</p>
```

não é assim:

```
<P>A paragraph of text</P>
```

## 2.5. Espaço em branco

Muito importante. Em HTML, mesmo que você adicione vários espaços em branco a uma linha, ela será recolhida pelo mecanismo CSS do navegador.

Por exemplo, a renderização deste parágrafo

```
<p>A paragraph of text</p>
```

é o mesmo que isto:

```
<p>A paragraph of text      </p>
```

e o mesmo que isto:

```
<p>
```

A paragraph of text

```
</p>
```

Using the white-space CSS property you can change how things behave. You can find more information on how CSS processes white space in the CSS Spec

I'd say use the syntax that makes things visually more organized and easier to read, but you can use any syntax you like.

I typically favor

```
<p>A paragraph of text</p>
```

or

```
<p>
  A paragraph of text
</p>
```

Nested tags should be indented with 2 or 4 characters, depending on your preference:

```
<body>
  <p>A paragraph of text</p>
  <ul>
    <li>A list item</li>
  </ul>
</body>
```

Note: this “white space is not relevant” feature means that if you want to add additional space, it can make you pretty mad. I suggest you use CSS to make more space when needed.

Note: in special cases, you can use the HTML entity (an acronym that means *non-breaking space*) - more on HTML entities later on. I think this should not be abused. CSS is always preferred to alter the visual presentation.&nbsp;

## 3. The document heading

The tag contains special tags that define the document properties.

It's always written before the tag, right after the opening tag:

```
<!DOCTYPE html>
<html>
  <head>
    ...
  </head>
  ...
</html>
```

We never use attributes on this tag. And we don't write content in it.

It's just a container for other tags. Inside it we can have a wide variety of tags, depending on what you need to do:

- title
- script
- noscript
- link
- style
- base
- meta

### 3.1. The tag title

The tag determines the page title. The title is displayed in the browser, and it's especially important as it's one of the key factors for Search Engine Optimization (SEO).

## 3.2. The `tagscript`

This tag is used to add JavaScript into the page.

You can include it inline, using an opening tag, the JavaScript code and then the closing tag:

```
<script>  
  ..some JS  
</script>
```

Or you can load an external JavaScript file by using the attribute:src

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

The attribute by default is set to , so it's completely optional.type=text/javascript

There is something pretty important to know about this tag.

Sometimes this tag is used at the bottom of the page, just before the closing tag. Why? For performance reasons.</body>

Loading scripts by default blocks the rendering of the page until the script is parsed and loaded.

By putting it at the bottom of the page, the script is loaded and executed after the whole page is already parsed and loaded, giving a better experience to the user over keeping it in the tag.head

My opinion is that this is now bad practice. Let live in the tag.scripthead

In modern JavaScript we have an alternative this is more performant than keeping the script at the bottom of the page — the attribute. This is an example that loads a file, relative to the current URL:deferfile.js

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

This is the scenario that triggers the faster path to a fast-loading page, and fast-loading JavaScript.

Note: the attribute is similar, but in my opinion a worse option than `.defer`. I describe why, in more detail, on page <https://flaviocopes.com/javascript-async-defer/asyncdefer>

### 3.3. The `tag noscript`

This tag is used to detect when scripts are disabled in the browser.

Note: users can choose to disable JavaScript scripts in the browser settings. Or the browser might not support them by default.

It is used differently depending on whether it's put in the document head or in the document body.

We're talking about the document head now, so let's first introduce this usage.

In this case, the tag can only contain other tags:`:noscript`

- `link` tags
- `style` tags
- `meta` tags

to alter the resources served by the page, or the information, if scripts are disabled.`:meta`

In this example I set an element with the class to display if scripts are disabled, as it was by default:`:no-script`  
`display: none`

```
<!DOCTYPE html>
<html>
```

```
<head>
  ...
  <noscript>
    <style>
      .no-script-alert {
        display: block;
      }
    </style>
  </noscript>

  ...
</head>
...
</html>
```

Let's solve the other case: if put in the body, it can contain content, like paragraphs and other tags, which are rendered in the UI.

## 3.4. The taglink

A tag é usada para definir relações entre um documento e outros recursos.[link](#)

É usado principalmente para vincular um arquivo CSS externo a ser carregado.

Esse elemento não tem nenhuma tag de fechamento.

Uso:

```
<!DOCTYPE html>
<html>
  <head>
    ...
    <link href="file.css" rel="stylesheet" />
    ...
  </head>
```

```
 . . .
```

```
</html>
```

O atributo permite o carregamento de diferentes folhas de estilo, dependendo dos recursos do dispositivo:media

```
<link href="file.css" media="screen" rel="stylesheet" />
<link href="print.css" media="print" rel="stylesheet" />
```

Também podemos vincular a recursos que não sejam folhas de estilo.

Por exemplo, podemos associar um feed RSS usando

```
<link rel="alternate" type="application/rss+xml" href="/index.xml" />
```

Ou podemos associar um favicon usando:

```
<link
  rel="apple-touch-icon"
  sizes="180x180"
  href="/assets/apple-touch-icon.png"
/>
```

```
<link
  rel="icon"
  type="image/png"
  sizes="32x32"
  href="/assets/favicon-32x32.png"
/>
```

```
<link
  rel="icon"
  type="image/png"
  sizes="16x16"
```

```
    href="/assets/favicon-16x16.png"  
  />
```

Essa tag também *foi* usada para conteúdo de várias páginas, para indicar a página anterior e a próxima usando e . Principalmente para o Google. A partir de 2019, o Google anunciou que não usa mais essa tag porque pode encontrar a estrutura de página correta sem ela.`rel="prev" rel="next"`

## 3.5. A etiqueta `style`

Essa marca pode ser usada para adicionar estilos ao documento, em vez de carregar uma folha de estilos externa.

Uso:

```
<style>  
  .some-css {  
  }  
</style>
```

Assim como na tag, você pode usar o atributo para usar esse CSS somente na mídia especificada:`media`

```
<style media="print">  
  .some-css {  
  }  
</style>
```

## 3.6. The tag `base`

This tag is used to set a base URL for all relative URLs contained in the page.

```
<!DOCTYPE html>  
<html>  
  <head>
```

```
...
<base href="https://flaviocopes.com/" />
...
</head>
...
</html>
```

## 3.7. The tagmeta

As meta tags executam uma variedade de tarefas e são muito, muito importantes.

Especialmente para SEO.

meta os elementos só têm a tag inicial.

A mais básica é a meta tag:description

```
<meta name="description" content="A nice page" />
```

Isso pode ser usado pelo Google para gerar a descrição da página em suas páginas de resultados, se achar que descreve melhor a página do que o conteúdo na página (não me pergunte como).

A meta tag é usada para definir a codificação de caracteres da página. na maioria dos casos:charsetutf-8

```
<meta charset="utf-8" />
```

A meta tag instrui os bots do mecanismo de pesquisa a indexar uma página ou não:robots

```
<meta name="robots" content="noindex" />
```

Ou se devem seguir links ou não:

```
<meta name="robots" content="nofollow" />
```

Você também pode definir nofollow em links individuais. É assim que você pode definir globalmente nofollow

Você pode combiná-los:

```
<meta name="robots" content="noindex, nofollow" />
```

O comportamento padrão é .index, follow

Você pode usar outras propriedades, incluindo , e muito mais.nosnippetnoarchiveimageindex

Você também pode simplesmente dizer ao Google em vez de segmentar todos os mecanismos de pesquisa:

```
<meta name="googlebot" content="noindex, nofollow" />
```

E outros mecanismos de pesquisa também podem ter sua própria meta tag.

Falando nisso, podemos dizer ao Google para desativar alguns recursos. Isso impede a funcionalidade de tradução nos resultados do mecanismo de pesquisa:

```
<meta name="google" content="notranslate" />
```

A meta tag é usada para dizer ao navegador para definir a largura da página com base na largura do dispositivo.viewport

```
<meta name="viewport" content="width=device-width, initial-scale=1" />
```

[Veja mais sobre esta tag.](#)

Outra meta tag bastante popular é a única. Esta linha diz ao navegador para aguardar 3 segundos e, em seguida, redirecionar para essa outra página:`http-equiv="refresh"`

```
<meta  
    http-equiv="refresh"  
    content="3;url=http://flaviocopes.com/another-page"  
/>
```

Usar 0 em vez de 3 redirecionará o mais rápido possível.

Esta não é uma referência completa; Existem outras metatags menos usadas.

Após a introdução deste título do documento, podemos começar a mergulhar no corpo do documento.

## 4. O corpo do documento

Após a tag head de fechamento, só podemos ter uma coisa em um documento HTML: o elemento body

```
<!DOCTYPE html>  
<html>  
    <head>  
        ...  
    </head>  
    <body>  
        ...  
    </body>  
</html>
```

Assim como as tags e, só podemos ter uma tag em uma página.`head``html``body`

Dentro da tag temos todas as tags que definem o conteúdo da página.`body`

Tecnicamente, as tags de início e término são opcionais. Mas considero uma boa prática adicioná-los. Só para maior clareza.

Nos próximos capítulos, definiremos a variedade de tags que você pode usar dentro do corpo da página.

Mas antes, devemos introduzir uma diferença entre elementos de bloco e elementos embutidos.

## 4.1. Elementos de bloco vs elementos embutidos

Os elementos visuais, aqueles definidos no corpo da página, podem ser geralmente classificados em 2 categorias:

- elementos de bloco (, , elementos de título, listas e itens de lista, ...)pdiv
- elementos embutidos (, , ...)aspanimg

Qual a diferença?

Bloquear elementos, quando posicionados na página, não permitem outros elementos ao lado deles. À esquerda ou à direita.

Em vez disso, os elementos embutidos podem ficar ao lado de outros elementos embutidos.

A diferença também está nas propriedades visuais que podemos editar usando CSS. Podemos alterar a largura/altura, margem, preenchimento e borda dos elementos do bloco. Não podemos fazer isso para elementos embutidos.

Observe que usando CSS podemos alterar o padrão para cada elemento, definindo uma tag para ser embutida, por exemplo, ou a para ser um elemento de bloco.pspan

Outra diferença é que os elementos embutidos podem estar contidos em elementos de bloco. O inverso não é verdadeiro.

Alguns elementos de bloco podem conter outros elementos de bloco, mas depende. A tag, por exemplo, não permite essa opção.p

## 5. Tags que interagem com o texto

### 5.1. A etiqueta p

Essa tag define um parágrafo de texto.

```
<p>Some text</p>
```

É um elemento de bloco.

Dentro dele, podemos adicionar qualquer elemento embutido que gostamos, gostamos ou .spana

Não podemos adicionar elementos de bloco.

Não podemos aninhar um elemento em outro.p

Por padrão, os navegadores estilizam um parágrafo com uma margem na parte superior e inferior. no Chrome, mas o valor exato pode variar entre os navegadores.16px

Isso faz com que dois parágrafos consecutivos sejam espaçados, replicando o que pensamos de um "parágrafo" no texto impresso.

### 5.2. A etiquetaspan

Esta é uma marca embutida que pode ser usada para criar uma seção em um parágrafo que pode ser direcionada usando CSS:

```
<p>A part of the text <span>and here another part</span></p>
```

### 5.3. A etiquetabr

Essa marca representa uma quebra de linha. É um elemento embutido e não precisa de uma tag de fechamento.

Nós o usamos para criar uma nova linha dentro de uma tag, sem criar um novo parágrafo.  
`p`

E em comparação com a criação de um novo parágrafo, ele não adiciona espaçoamento adicional.

```
<p>Some text<br />A new line</p>
```

## 5.4. As etiquetas de cabeçalho

HTML nos fornece 6 tags de título. Do mais importante ao menos importante, temos , , , , .  
`h1h2h3h4h5h6`

Normalmente, uma página terá um elemento, que é o título da página. Então você pode ter um ou mais elementos, dependendo do conteúdo da página.  
`h1h2`

Os títulos, especialmente a organização do título, também são essenciais para o SEO, e os mecanismos de pesquisa os usam de várias maneiras.

O navegador, por padrão, tornará a tag maior e tornará o tamanho dos elementos menor à medida que o número próximo aumentar:  
`h1h`

```
<h1>h1</h1>
<h2>h1</h2>
<h3>h1</h3>
<h4>h1</h4>
<h5>h1</h5>
<h6>h1</h6>
<p>p</p>
```

Todos os títulos são elementos de bloco. Eles não podem conter outros elementos, apenas texto.

## 5.5. A etiqueta **strong**

Essa tag é usada para marcar o texto dentro dela como *forte*. Isso é muito importante, não é uma dica visual, mas uma dica semântica. Dependendo do meio utilizado, sua interpretação variará.

Por padrão, os navegadores colocam o texto nessa tag em **negrito**.

## 5.6. A etiqueta **em**

Essa tag é usada para marcar o texto dentro dela como *enfatizado*. Como com **strong**, não é uma dica visual, mas uma dica semântica.

Os navegadores, por padrão, tornam o texto em **itálico**.

## 5.6.1. Cotações

A tag HTML é útil para inserir citações no texto.

Por padrão, os navegadores aplicam uma margem ao elemento. O Chrome aplica uma margem esquerda e direita de 40px e uma margem superior e inferior de 10px.

A tag HTML é usada para aspas embutidas.

## 5.7. Linha horizontal

Não é realmente baseado em texto, mas a tag é frequentemente usada dentro de uma página. Significa , e adiciona uma linha horizontal na página.

Útil para separar seções na página.

## 5.8. Blocos de código

A tag é especialmente útil para mostrar o código, porque os navegadores lhe dão uma fonte monoespaçada.

Essa é tipicamente a única coisa que os navegadores fazem. Este é o CSS aplicado pelo Chrome:

```
code {  
    font-family: monospace;  
}
```

Essa marca geralmente é encapsulada em uma marca, porque o elemento ignora espaços em branco e quebras de linha. Como a tag.

O Chrome oferece este estilo padrão:

```
pre {  
    display: block;  
    font-family: monospace;  
    white-space: pre;  
    margin: 1em 0px;  
}
```

o que impede o colapso do espaço em branco e o torna um elemento de bloco.

## 5.9. Listas

Temos 3 tipos de listas:

- listas não ordenadas
- listas ordenadas
- listas de definições

As listas não ordenadas são criadas usando a tag. Cada item na lista é criado com a marca:`ul li`

```
<ul>  
    <li>First</li>  
    <li>Second</li>  
</ul>
```

As listas ordenadas são semelhantes, apenas feitas com a tag:`ol`

```
<ol>  
    <li>First</li>  
    <li>Second</li>  
</ol>
```

A diferença entre os dois é que as listas ordenadas têm um número antes de cada item:

```
<ul>
  <li>First</li>
  <li>Second</li>
</ul>

<ol>
  <li>First</li>
  <li>Second</li>
</ol>
```

- First
  - Second
1. First
  2. Second

As listas de definições são um pouco diferentes. Você tem um termo e sua definição:

```
<dl>
  <dt>Flavio</dt>
  <dd>The name</dd>
  <dt>Copes</dt>
  <dd>The surname</dd>
</dl>
```

É assim que os navegadores normalmente os renderizam:

```
<dl>
  <dt>Flavio</dt>
  <dd>The name</dd>
  <dt>Copes</dt>
  <dd>The surname</dd>
</dl>
```

Flavio  
The name  
Copes  
The surname

Devo dizer que você raramente os vê na natureza, com certeza não muito como e , mas às vezes eles podem ser úteis.ulol

## 5.10. Outras etiquetas de texto

Há uma série de tags com propósitos de apresentação:

- a tagmark
- a tagins
- a tagdel
- a tagsup
- a tagsub
- a tagsmall
- a tagi
- a tagb

Este é um exemplo da renderização visual deles que é aplicada por padrão pelos navegadores:

The screenshot shows a code editor with two columns. The left column contains the following HTML code:

```
<mark>mark</mark>
<ins>ins</ins>
<del>del</del>
<sup>sup</sup>
<sub>sub</sub>
<small>small</small>
<i>i</i>
<b>b</b>
```

The right column shows the visual representation of these tags as rendered by a browser. The words are highlighted with different styles:

mark ins del sup sub small i b

Você pode se perguntar, como é diferente de ? E como é diferente de ?  
bstrongiem

A diferença está no significado semântico. Enquanto e são uma dica direta no navegador para fazer um pedaço de texto em negrito ou itálico, e dar ao texto um significado especial, e cabe ao navegador dar o estilo. Que passa a

ser exatamente o mesmo que é, por padrão. Embora você possa alterar isso usando CSS.

Há uma série de outras tags menos usadas relacionadas ao texto. Acabei de mencionar os que eu vejo mais usados.

## 6. Ligações

Os links são definidos usando a tag. O destino do link é definido por meio de seu atributo `ahref`

Exemplo:

```
<a href="https://flaviocopes.com">click here</a>
```

Entre a tag inicial e de fechamento, temos o texto do link.

O exemplo acima é uma URL absoluta. Os links também funcionam com URLs relativas:

```
<a href="/test">click here</a>
```

Nesse caso, ao clicar no link, o usuário é movido para a URL na origem atual `./test`

Tenha cuidado com o personagem. Se omitido, em vez de começar a partir da origem, o navegador apenas adicionará a cadeia de caracteres à URL atual `./test`

Exemplo, estou na página e tenho estes links: `https://flaviocopes.com/axios/`

- `/test` uma vez clicado me leva a `https://flaviocopes.com/test`
- `test` uma vez clicado me leva a  
`https://flaviocopes.com/axios/test`

As tags de link podem incluir outras coisas dentro delas, não apenas texto.  
Por exemplo, imagens:

```
<a href="https://flaviocopes.com">  
    
</a>
```

ou quaisquer outros elementos, exceto outras tags.<a>

Se você quiser abrir o link em uma nova guia, você pode usar o atributo:`target`

```
<a href="https://flaviocopes.com" target="_blank">open in new tab</a>
```

## 7. Tags de contêiner e HTML de estrutura de página

### 7.1. Etiquetas de contendor

HTML fornece um conjunto de tags de contêiner. Essas tags podem conter um conjunto não especificado de outras tags.

Temos:

- `article`
- `section`
- `div`

e pode ser confuso entender a diferença entre eles.

Vamos ver quando usar cada um deles.

#### 7.1.1. `article`

A tag de artigo identifica uma *coisa* que pode ser independente de outras *coisas* em uma página.

Por exemplo, uma lista de postagens de blog na página inicial.

Ou uma lista de links.

```
<div>
  <article>
    <h2>A blog post</h2>
    <a ...>Read more</a>
  </article>
  <article>
    <h2>Another blog post</h2>
    <a ...>Read more</a>
  </article>
</div>
```

Não estamos limitados a listas: um artigo pode ser o elemento principal de uma página.

```
<article>
  <h2>A blog post</h2>
  <p>Here is the content...</p>
</article>
```

Dentro de uma tag devemos ter um título (-) e articleh1h6

## 7.1.2. section

Representa uma seção de um documento. Cada seção tem uma tag de título (-) e, em seguida, o *corpo da seção*.h1h6

Exemplo:

```
<section>
  <h2>A section of the page</h2>
  <p>...</p>
  <img ... />
</section>
```

É útil dividir um artigo longo em **seções** diferentes.

Não deve ser usado como um elemento de contêiner genérico. é feito para isso.`div`

### 7.1.3. `div`

`div` é o elemento genérico do contêiner:

```
<div>...</div>
```

Muitas vezes, você adiciona um atributo ou a esse elemento, para permitir que ele seja estilizado usando CSS.`classid`

Usamos em qualquer lugar onde precisamos de um contêiner, mas as tags existentes não são adequadas.`div`

## 7.2. Tags relacionadas à página

### 7.2.1. `nav`

Essa tag é usada para criar a marcação que define a navegação da página. Nisso, normalmente adicionamos uma ou lista:`ul``ol`

```
<nav>
  <ol>
    <li><a href="/">Home</a></li>
    <li><a href="/blog">Blog</a></li>
```

```
</ol>  
</nav>
```

## 7.2.2. aside

A tag é usada para adicionar uma parte do conteúdo que está relacionada ao conteúdo principal.`aside`

Uma caixa onde adicionar uma cotação, por exemplo. Ou uma barra lateral.

Exemplo:

```
<div>  
  <p>some text...</p>  
  <aside>  
    <p>A quote...</p>  
  </aside>  
  <p>other text...</p>  
</div>
```

O uso é um sinal de que as coisas que ele contém não fazem parte do fluxo regular da seção em que vive.`aside`

## 7.2.3. header

A tag representa uma parte da página que é a introdução. Ele pode, por exemplo, conter uma ou mais tags de título (-), o slogan do artigo, uma imagem.`header`

```
<article>  
  <header>  
    <h1>Article title</h1>  
  </header>  
  ...  
</div>
```

## 7.2.4. main

A tag representa a parte principal de uma página:`main`

```
<body>
  ....
  <main>
    <p>....</p>
  </main>
</body>
```

## 7.2.5. footer

A tag é usada para determinar o rodapé de um artigo ou o rodapé da página:`footer`

```
<article>
  ....
  <footer>
    <p>Footer notes..</p>
  </footer>
</div>
```

# 8. Formulários

Os formulários são a maneira como você pode interagir com uma página ou um aplicativo criado com tecnologias da Web.

Você tem um conjunto de controles e, quando enviar o formulário, com um clique em um botão "enviar" ou programaticamente, o navegador enviará os dados para o servidor.

Por padrão, esse envio de dados faz com que a página seja recarregada depois que os dados são enviados, mas usando JavaScript você pode alterar esse comportamento (não vou explicar como neste livro).

Um formulário é criado usando a marca:`form`

```
<form>...</form>
```

Por padrão, os formulários são enviados usando o método HTTP GET. Que tem suas desvantagens, e geralmente você quer usar o POST.

Você pode definir o formulário para usar o POST quando enviado usando o atributo:`method`

```
<form method="POST">...</form>
```

O formulário é enviado, usando GET ou POST, para a mesma URL onde reside.

Portanto, se o formulário estiver na página, pressionar o botão "enviar" fará uma solicitação para esse mesmo URL.<https://flaviocopes.com/contacts>

O que pode resultar em nada acontecendo.

Você precisa de algo do lado do servidor para lidar com a solicitação e, normalmente, você "ouve" esses eventos de envio de formulário em uma URL dedicada.

Você pode especificar a URL através do parâmetro:`action`

```
<form action="/new-contact" method="POST">...</form>
```

Isso fará com que o navegador envie os dados do formulário usando o POST para a URL na mesma origem./new-contact

Se a origem (protocolo + domínio + porta) for (a porta 80 é o padrão), isso significa que os dados do formulário serão enviados para  
<https://flaviocopes.com/new-contact>

Falei sobre dados. Quais dados?

Os dados são fornecidos pelos usuários por meio do conjunto de controles disponíveis na plataforma Web:

- caixas de entrada (texto de linha única)
- áreas de texto (texto de várias linhas)
- caixas de seleção (escolha uma opção em um menu suspenso)
- botões de opção (escolha uma opção em uma lista sempre visível)
- caixas de seleção (escolha zero, uma ou mais opções)
- uploads de arquivos
- e muito mais!

Vamos apresentar cada um deles na visão geral dos campos de formulário a seguir.

## 8.1. A etiqueta `input`

O campo é um dos elementos de forma mais utilizados. Também é um elemento muito versátil e pode mudar completamente o comportamento com base no atributo `input type`

O comportamento padrão é ser um controle de entrada de texto de linha única:

```
<input />
```

Equivalente a usar:

```
<input type="text" />
```

Tal como acontece com todos os outros campos que se seguem, você precisa dar ao campo um nome para que seu conteúdo seja enviado para o servidor quando o formulário for enviado:

```
<input type="text" name="username" />
```

O atributo é usado para que algum texto apareça, em cinza claro, quando o campo está vazio. Útil para adicionar uma dica ao usuário sobre o que digitar:`placeholder`

```
<input type="text" name="username" placeholder="Your username" />
```

## 8.1.1. E-mail

O uso validará o lado do cliente (no navegador) de um e-mail para correção (correção semântica, não garantindo que o endereço de e-mail exista) antes de enviar.`type="email"`

```
<input type="email" name="email" placeholder="Your email" />
```

## 8.1.2. Palavra-passe

O uso fará com que cada chave inserida apareça como um asterisco (\*) ou ponto, útil para campos que hospedam uma senha.`type="password"`

```
<input type="password" name="password" placeholder="Your password" />
```

## 8.1.3. Números

Você pode fazer com que um elemento de entrada aceite apenas números:

```
<input type="number" name="age" placeholder="Your age" />
```

Você pode especificar um valor mínimo e máximo aceito:

```
<input type="number" name="age" placeholder="Your age" min="18" max="110" />
```

O atributo ajuda a identificar as etapas entre diferentes valores. Por exemplo, isso aceita um valor entre 10 e 50, nas etapas de 5:`step`

```
<input type="number" name="a-number" min="10" max="50" step="5" />
```

## 8.1.4. Campo oculto

Os campos podem ser ocultos do usuário. Eles ainda serão enviados para o servidor após o envio do formulário:

```
<input type="hidden" name="some-hidden-field" value="some-value" />
```

Isso é comumente usado para armazenar valores como um token CSRF, usado para segurança e identificação de usuários, ou mesmo para detectar robôs enviando spam, usando técnicas especiais.

Ele também pode ser usado apenas para identificar uma forma e sua ação.

## 8.1.5. Definindo um valor padrão

Todos esses campos aceitam um valor predefinido. Se o usuário não alterá-lo, este será o valor enviado para o servidor:

```
<input type="number" name="age" value="18" />
```

Se você definir um espaço reservado, esse valor aparecerá se o usuário limpar o valor do campo de entrada:

```
<input type="number" name="age" placeholder="Your age" value="18" />
```

## 8.2. Envio do formulário

O campo é um botão que, uma vez pressionado pelo usuário, envia o formulário:`type="submit"`

```
<input type="submit" />
```

O atributo define o texto no botão, que, se estiver faltando, mostra o texto "Enviar":value

```
<input type="submit" value="Click me" />
```

## 8.3. Validação do formulário

Os navegadores fornecem funcionalidade de validação do lado do cliente para formulários.

Você pode definir campos conforme necessário, garantindo que eles sejam preenchidos e impor um formato específico para a entrada de cada campo.

Vamos ver as duas opções.

### 8.3.1. Definir campos conforme necessário

O atributo ajuda você com a validação. Se o campo não estiver definido, a validação do lado do cliente falhará e o navegador não enviará o formulário:required

```
<input type="text" name="username" required />
```

### 8.3.2. Impor um formato específico

Descrevi o campo acima. Ele valida automaticamente o endereço de e-mail de acordo com um formato definido na especificação.type="email"

No campo, mencionei o atributo e aos valores-limite inseridos em um intervalo.type="number"minmax

Você pode impor um formato específico em qualquer campo por meio do atributo, o que lhe dá a capacidade de definir uma expressão regular para validar o valor.pattern

Recomendo a leitura do meu Guia de Expressões Regulares em [flaviocopes.com/javascript-regular-expressions/](https://flaviocopes.com/javascript-regular-expressions/).

Exemplo:

```
<input type="text" name="username" pattern="[a-zA-Z]{8}" />
```

## 8.4. Outros campos

### 8.4.1. Uploads de arquivos

Você pode carregar arquivos do computador local e enviá-los para o servidor usando um elemento de entrada:`type="file"`

```
<input type="file" name="secret-documents" />
```

Você pode anexar vários arquivos:

```
<input type="file" name="secret-documents" multiple />
```

Você pode especificar um ou mais tipos de arquivo permitidos usando o atributo. Isso aceita imagens:`accept`

```
<input type="file" name="secret-documents" accept="image/*" />
```

Você pode usar um tipo MIME específico, como ou definir uma extensão de arquivo como o `.jpg`. Ou defina várias extensões de arquivo, como esta:`application/json.pdf`

```
<input type="file" name="secret-documents" accept=".jpg, .jpeg, .png" />
```

### 8.4.2. Botões

Os campos de entrada podem ser usados para adicionar botões adicionais ao formulário, que não são botões de envio:type="button"

```
<input type="button" value="Click me" />
```

Eles são usados para fazer algo programaticamente, usando JavaScript.

Há um campo especial renderizado como um botão, cuja ação especial é limpar todo o formulário e trazer de volta o estado dos campos para o inicial:

```
<input type="reset" />
```

### 8.4.3. Botões de opção

Os botões de opção são usados para criar um conjunto de opções, das quais uma é pressionada e todas as outras são desabilitadas.

O nome vem de antigos rádios de carro que tinham esse tipo de interface.

Você define um conjunto de entradas, todas com o mesmo atributo e um atributo diferente:type="radio"namevalue

```
<input type="radio" name="color" value="yellow" />
<input type="radio" name="color" value="red" />
<input type="radio" name="color" value="blue" />
```

Depois que o formulário for enviado, a propriedade data terá um único valor.color

Há sempre um elemento verificado. O primeiro item é aquele verificado por padrão.

Você pode definir o valor pré-selecionado usando o atributo. Você pode usá-lo apenas uma vez por grupo de entradas de rádio.checked

## 8.4.4. Caixas de verificação

Semelhante às caixas de rádio, mas elas permitem que vários valores sejam escolhidos, ou nenhum.

Você define um conjunto de entradas, todas com o mesmo atributo e um atributo diferente:`type="checkbox"``name``value`

```
<input type="checkbox" name="color" value="yellow" />
<input type="checkbox" name="color" value="red" />
<input type="checkbox" name="color" value="blue" />
```

Todas essas caixas de seleção serão desmarcadas por padrão. Use o atributo para habilitá-los no carregamento da página.`checked`

Como esse campo de entrada permite vários valores, no envio do formulário, o(s) valor(es) será(ão) enviado(s) para o servidor como uma matriz.

## 8.4.5. Data e hora

Temos alguns tipos de entrada para aceitar valores de data.

O campo de entrada permite que o usuário insira uma data e mostra um seletor de data, se necessário:`type="date"`

```
<input type="date" name="birthday" />
```

O campo de entrada permite que o usuário insira uma hora e mostra um seletor de tempo, se necessário:`type="time"`

```
<input type="time" name="time-to-pickup" />
```

O campo de entrada permite que o usuário insira um mês e um ano:`type="month"`

```
<input type="month" name="choose-release-month" />
```

O campo de entrada permite que o usuário insira uma semana e um ano:  
type="week"

```
<input type="week" name="choose-week" />
```

Todos esses campos permitem limitar o intervalo e a etapa entre cada valor. Eu recomendo verificar MDN para os pequenos detalhes sobre o seu uso.

O campo permite que você escolha uma data e uma hora.  
type="datetime-local"

```
<input type="datetime-local" name="date-and-time" />
```

Aqui está uma página para testar todos eles:

<https://codepen.io/flaviocopes/pen/ZdWQPm>

## 8.4.6. Seletor de cores

Você pode permitir que os usuários escolham uma cor usando o elemento:  
type="color"

```
<input type="color" name="car-color" />
```

Você define um valor padrão usando o atributo:  
value

```
<input type="color" name="car-color" value="#000000" />
```

O navegador se encarregará de mostrar um seletor de cores para o usuário.

## 8.4.7. Gama

Esse elemento de entrada mostra um elemento deslizante. As pessoas podem usá-lo para passar de um valor inicial para um valor final:

```
<input type="range" name="age" min="0" max="100" value="30" />
```

Você pode fornecer uma etapa opcional:

```
<input type="range" name="age" min="0" max="100" value="30" step="10" />
```

## 8.4.8. Telefone

O campo de entrada é usado para inserir um número de telefone:`type="tel"`

```
<input type="tel" name="telephone-number" />
```

O principal ponto de venda para usar over é no celular, onde o dispositivo pode optar por mostrar um teclado numérico.`teltext`

Especifique um atributo para validação adicional:`pattern`

```
<input type="tel" pattern="[0-9]{3}-[0-9]{8}" name="telephone-number" />
```

## 8.4.9. URL

O campo é usado para inserir uma URL.`type="url"`

```
<input type="url" name="website" />
```

Você pode validá-lo usando o atributo:`pattern`

```
<input type="url" name="website" pattern="https://.*" />
```

## 8.5. A etiqueta textarea

O elemento permite que os usuários insiram texto de várias linhas. Em comparação com , requer uma tag de terminação:`textareainput`

```
<textarea></textarea>
```

Você pode definir as dimensões usando CSS, mas também usando os atributos `e:rowscols`

```
<textarea rows="20" cols="10"></textarea>
```

Assim como acontece com as outras marcas de formulário, o atributo `determina o nome nos dados enviados ao servidor: name`

```
<textarea name="article"></textarea>
```

## 8.6. A etiqueta select

Essa tag é usada para criar um menu suspenso.

O usuário pode escolher uma das opções disponíveis.

Cada opção é criada usando a tag. Você adiciona um nome à seleção e um valor a cada opção:`option`

```
<select name="color">
  <option value="red">Red</option>
  <option value="yellow">Yellow</option>
</select>
```

Você pode definir uma opção desativada:

```
<select name="color">
  <option value="red" disabled>Red</option>
  <option value="yellow">Yellow</option>
</select>
```

Você pode ter uma opção vazia:

```
<select name="color">
  <option value="">None</option>
  <option value="red">Red</option>
  <option value="yellow">Yellow</option>
</select>
```

As opções podem ser agrupadas usando a tag. Cada grupo de opções tem um atributo: optgroup label

```
<select name="color">
  <optgroup label="Primary">
    <option value="red">Red</option>
    <option value="yellow">Yellow</option>
    <option value="blue">Blue</option>
  </optgroup>
  <optgroup label="Others">
    <option value="green">Green</option>
    <option value="pink">Pink</option>
  </optgroup>
</select>
```

## 9. Tabelas

Nos primórdios da web, as tabelas eram uma parte muito importante da construção de layouts.

Mais tarde, eles foram substituídos pelo CSS e seus recursos de layout, e hoje temos ferramentas poderosas como CSS Flexbox e CSS Grid para criar

layouts. As mesas agora são usadas apenas para, adivinhe, construir tabelas!

## 9.0.1. A etiqueta `table`

Você define uma tabela usando a tag:`table`

```
<table></table>
```

Dentro da tabela, definiremos os dados. Raciocinamos em termos de linhas, o que significa que adicionamos linhas a uma tabela (não colunas). Definiremos colunas dentro de uma linha.

## 9.0.2. Linhas

Uma linha é adicionada usando a tag, e essa é a única coisa que podemos adicionar a um elemento:`tr`

```
<table>
  <tr></tr>
  <tr></tr>
  <tr></tr>
</table>
```

Esta é uma tabela com 3 linhas.

A primeira linha *pode* assumir o papel do cabeçalho.

## 9.0.3. Cabeçalhos de coluna

O cabeçalho da tabela contém o nome de uma coluna, normalmente em negrito.

Pense em um documento do Excel / Planilhas Google. O cabeçalho superior.A-B-C-D...

	A	B	C	D	E
1					
2					
3					
4					
5					
6					

Definimos o cabeçalho usando a tag:th

```
<table>
  <tr>
    <th>Column 1</th>
    <th>Column 2</th>
    <th>Column 3</th>
  </tr>
  <tr></tr>
  <tr></tr>
</table>
```

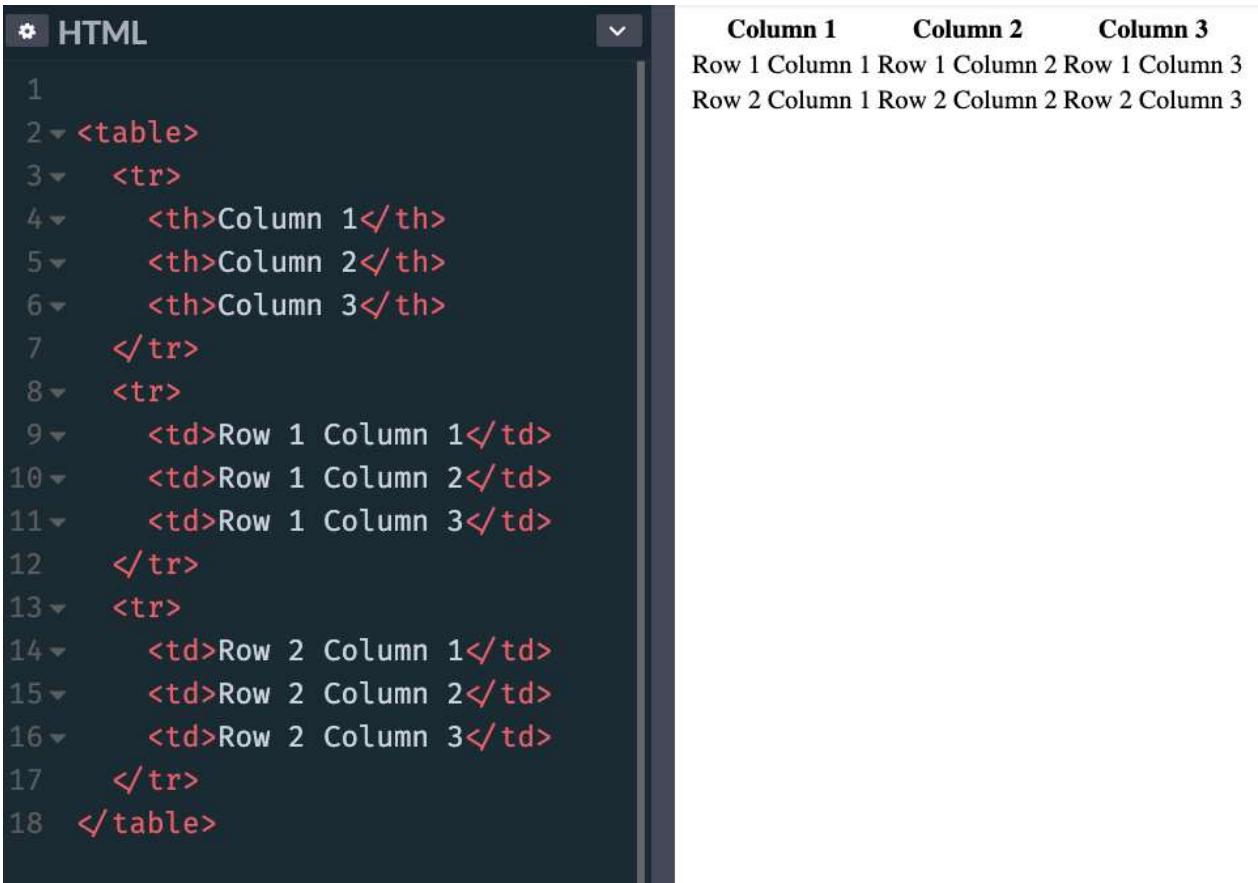
## 9.0.4. Conteúdo do quadro

O conteúdo da tabela é definido usando tags, dentro dos outros elementos:tdtr

```
<table>
  <tr>
    <th>Column 1</th>
    <th>Column 2</th>
    <th>Column 3</th>
  </tr>
  <tr>
    <td>Row 1 Column 1</td>
    <td>Row 1 Column 2</td>
    <td>Row 1 Column 3</td>
  </tr>
```

```
</tr>  
  
<tr>  
  <td>Row 2 Column 1</td>  
  <td>Row 2 Column 2</td>  
  <td>Row 2 Column 3</td>  
</tr>  
</table>
```

É assim que os navegadores o renderizam, se você não adicionar nenhum estilo CSS:



The screenshot shows a code editor with two panes. The left pane is titled 'HTML' and contains the following code:

```
1  
2 <table>  
3   <tr>  
4     <th>Column 1</th>  
5     <th>Column 2</th>  
6     <th>Column 3</th>  
7   </tr>  
8   <tr>  
9     <td>Row 1 Column 1</td>  
10    <td>Row 1 Column 2</td>  
11    <td>Row 1 Column 3</td>  
12  </tr>  
13  <tr>  
14    <td>Row 2 Column 1</td>  
15    <td>Row 2 Column 2</td>  
16    <td>Row 2 Column 3</td>  
17  </tr>  
18 </table>
```

The right pane shows the rendered HTML table with three columns labeled 'Column 1', 'Column 2', and 'Column 3'. The rows are labeled 'Row 1' and 'Row 2'.

Adicionando este CSS:

```
th,  
td {  
  padding: 10px;  
  border: 1px solid #333;  
}
```

faz com que a tabela se pareça mais com uma tabela adequada:

Column 1	Column 2	Column 3
Row 1 Column 1	Row 1 Column 2	Row 1 Column 3
Row 2 Column 1	Row 2 Column 2	Row 2 Column 3

## 9.0.5. Estender colunas e linhas

Uma linha pode decidir abranger mais de 2 ou mais colunas, usando o atributo:colspan

```
<table>
  <tr>
    <th>Column 1</th>
    <th>Column 2</th>
    <th>Column 3</th>
  </tr>
  <tr>
    <td colspan="2">Row 1 Columns 1-2</td>
    <td>Row 1 Column 3</td>
  </tr>
  <tr>
    <td colspan="3">Row 2 Columns 1-3</td>
  </tr>
</table>
```

Column 1	Column 2	Column 3
Row 1 Columns 1-2		Row 1 Column 3
Row 2 Columns 1-3		

Ou pode abranger mais de 2 ou mais linhas, usando o atributo:rowspan

```

<table>
  <tr>
    <th>Column 1</th>
    <th>Column 2</th>
    <th>Column 3</th>
  </tr>
  <tr>
    <td colspan="2" rowspan="2">Rows 1-2 Columns 1-2</td>
    <td>Row 1 Column 3</td>
  </tr>
  <tr>
    <td>Row 2 Column 3</td>
  </tr>
</table>

```

Column 1	Column 2	Column 3
Rows 1-2 Columns 1-2		Row 1 Column 3
		Row 2 Column 3

## 9.0.6. Cabeçalhos de linha

Antes eu expliquei como você pode ter cabeçalhos de coluna, usando a tag dentro da primeira tag da tabela.`thtr`

Você pode adicionar uma tag como o primeiro elemento dentro de um que não é o primeiro da tabela, para ter cabeçalhos de linha:`thtrtr`

```

<table>
  <tr>
    <th></th>
    <th>Column 2</th>
    <th>Column 3</th>
  </tr>

```

```

<tr>
  <th>Row 1</th>
  <td>Col 2</td>
  <td>Col 3</td>
</tr>
<tr>
  <th>Row 2</th>
  <td>Col 2</td>
  <td>Col 3</td>
</tr>
</table>

```

	<b>Column 2</b>	<b>Column 3</b>
<b>Row 1</b>	Col 2	Col 3
<b>Row 2</b>	Col 2	Col 3

### 9.0.7. Mais tags para organizar a tabela

Você pode adicionar mais 3 tags em uma tabela, para tê-la mais organizada.

Isso é melhor ao usar tabelas grandes. E para definir corretamente um cabeçalho e um rodapé também.

Essas tags são

- `thead`
- `tbody`
- `tfoot`

Eles envolvem as tags para definir claramente as diferentes seções da tabela.  
Veja um exemplo:

```
<table>
  <thead>
    <tr>
      <th></th>
      <th>Column 2</th>
      <th>Column 3</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>Row 1</th>
      <td>Col 2</td>
      <td>Col 3</td>
    </tr>
    <tr>
      <th>Row 2</th>
      <td>Col 2</td>
      <td>Col 3</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td></td>
      <td>Footer of Col 1</td>
      <td>Footer of Col 2</td>
    </tr>
  </tfoot>
</table>
```

	<b>Column 2</b>	<b>Column 3</b>
<b>Row 1</b>	Col 2	Col 3
<b>Row 2</b>	Col 2	Col 3
	Footer of Col 1	Footer of Col 2

## 9.1. Legenda da tabela

Uma tabela deve ter uma marca que descreva seu conteúdo. Essa tag deve ser colocada imediatamente após a tag de abertura: `<caption>`

```
<table>
  <caption>
    Dogs age
  </caption>
  <tr>
    <th>Dog</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Roger</td>
    <td>7</td>
  </tr>
</table>
```

## 10. Tags multimídia: e audiovideo

Nesta seção, quero mostrar-lhe as tags e.audiovideo

### 10.1. A etiqueta `audio`

Essa tag permite que você incorpore conteúdo de áudio em suas páginas HTML.

Este elemento pode transmitir áudio, talvez usando um microfone via , ou pode reproduzir uma fonte de áudio que você referencia usando o atributo: `getUserMedia()` `src`

```
<audio src="file.mp3"></audio>
```

Por padrão, o navegador não mostra nenhum controle para esse elemento. O que significa que o áudio será reproduzido somente se definido para reprodução automática (mais sobre isso mais tarde) e o usuário não pode ver como pará-lo ou controlar o volume ou mover-se pela faixa.

Para mostrar os controles internos, você pode adicionar o atributo:`controls`

```
<audio src="file.mp3" controls></audio>
```

Os controles podem ter uma capa personalizada.

Você pode especificar o tipo MIME do arquivo de áudio usando o atributo. Se não estiver definido, o navegador tentará determiná-lo automaticamente:`type`

```
<audio src="file.mp3" controls type="audio/mpeg"></audio>
```

Um arquivo de áudio por padrão não é reproduzido automaticamente. Adicione o atributo para reproduzir o áudio automaticamente:`autoplay`

```
<audio src="file.mp3" controls autoplay></audio>
```

Observação: os navegadores móveis não permitem a reprodução automática

O atributo reinicia a reprodução de áudio às 0:00 se definido; caso contrário, se não estiver presente, o áudio pára no final do arquivo:`loop`

```
<audio src="file.mp3" controls autoplay loop></audio>
```

Você também pode reproduzir um arquivo de áudio silenciado usando o atributo (não tenho certeza de qual é a utilidade disso):`muted`

```
<audio src="file.mp3" controls autoplay loop muted></audio>
```

Usando JavaScript, você pode ouvir vários eventos que acontecem em um elemento, os mais básicos dos quais são:`audio`

- `play` quando o arquivo começa a ser reproduzido
- `pause` quando a reprodução de áudio foi pausada
- `playing` quando o áudio é retomado de uma pausa
- `ended` quando o fim do arquivo de áudio foi atingido

## 10.2. A etiqueta `video`

Essa tag permite que você incorpore conteúdo de vídeo em suas páginas HTML.

Este elemento pode transmitir vídeo, usando uma webcam via ou **WebRTC**, ou pode reproduzir uma fonte de vídeo que você referencia usando o atributo:`getUserMedia()``src`

```
<video src="file.mp4"></video>
```

Por padrão, o navegador não mostra nenhum controle para esse elemento, apenas o vídeo.

O que significa que o vídeo será reproduzido somente se estiver configurado para reprodução automática (mais sobre isso mais tarde) e o usuário não poderá ver como pará-lo, pausá-lo, controlar o volume ou pular para uma posição específica no vídeo.

Para mostrar os controles internos, você pode adicionar o atributo:`controls`

```
<video src="file.mp4" controls></video>
```

Os controles podem ter uma capa personalizada.

Você pode especificar o tipo MIME do arquivo de vídeo usando o atributo. Se não estiver definido, o navegador tentará determiná-lo automaticamente:`type`

```
<video src="file.mp4" controls type="video/mp4"></video>
```

Um arquivo de vídeo por padrão não é reproduzido automaticamente. Adicione o atributo para reproduzir o vídeo automaticamente:autoplay

```
<video src="file.mp4" controls autoplay></video>
```

Alguns navegadores também exigem o atributo para reprodução automática. O vídeo é reproduzido automaticamente somente se estiver sem som:muted

```
<audio src="file.mp3" controls autoplay muted></audio>
```

O atributo reinicia a reprodução de vídeo às 0:00 se definido; caso contrário, se não estiver presente, o vídeo pára no final do arquivo:loop

```
<video src="file.mp4" controls autoplay loop></video>
```

Você pode definir uma imagem para ser a imagem do pôster:

```
<video src="file.mp4" poster="picture.png"></video>
```

Se não estiver presente, o navegador exibirá o primeiro quadro do vídeo assim que estiver disponível.

Você pode definir os atributos e para definir o espaço que o elemento ocupará para que o navegador possa contabilizá-lo e não altere o layout quando for finalmente carregado. É preciso um valor numérico, expresso em pixels.widthheight

Usando JavaScript, você pode ouvir vários eventos que acontecem em um elemento, os mais básicos dos quais são:video

- play quando o arquivo começa a ser reproduzido

- pause quando o vídeo foi pausado
- playing quando o vídeo é retomado de uma pausa
- ended quando o fim do arquivo de vídeo foi atingido

## 11. iframes

A tag nos permite incorporar conteúdo proveniente de outras origens (outros sites) em nossa página da web.`iframe`

Tecnicamente, um iframe cria um novo contexto de navegação aninhado. Isso significa que qualquer coisa no iframe não interfere na página pai e vice-versa. JavaScript e CSS não "vazam" de/para iframes.

Muitos sites usam iframes para executar várias coisas. Você pode estar familiarizado com Codepen, Glitch ou outros sites que permitem codificar em uma parte da página e você vê o resultado em uma caixa. Isso é um iframe.

Você cria um desta maneira:

```
<iframe src="page.html"></iframe>
```

Você também pode carregar um URL absoluto:

```
<iframe src="https://site.com/page.html"></iframe>
```

Você pode definir um conjunto de parâmetros de largura e altura (ou defini-los usando CSS), caso contrário, o iframe usará os padrões, uma caixa de 300x150 pixels:

```
<iframe src="page.html" width="800" height="400"></iframe>
```

### 11.1. Srcdoc

O atributo permite que você especifique algum HTML embutido para mostrar. É uma alternativa ao , mas recente e não suportada no Edge 18 e inferior, e no IE:srcdocsrc

```
<iframe srcdoc=<p>My dog is a good dog</p>></iframe>
```

## 11.2. Caixa de areia

O atributo nos permite limitar as operações permitidas nos iframes.sandbox

Se o omitirmos, tudo é permitido:

```
<iframe src="page.html"></iframe>
```

Se o definirmos como "", nada é permitido:

```
<iframe src="page.html" sandbox=""></iframe>
```

Podemos selecionar o que permitir adicionando opções no atributo. Você pode permitir vários adicionando um espaço no meio. Aqui está uma lista incompleta das opções que você pode usar:sandbox

- allow-forms: permitir o envio de formulários
- allow-modals permitir abrir janelas modais, incluindo chamadas em JavaScript alert()
- allow-orientation-lock permitir bloquear a orientação da tela
- allow-popups permitir pop-ups, uso e linkswindow.open()target=\_blank"
- allow-same-origin tratar o recurso que está sendo carregado como a mesma origem
- allow-scripts permite que o iframe carregado execute scripts (mas não crie pop-ups).
- allow-top-navigation dá acesso ao iframe para o contexto de navegação de nível superior

## 11.3. Permitir

Atualmente experimental e suportado apenas por navegadores baseados no Chromium, este é o futuro do compartilhamento de recursos entre a janela pai e o iframe.

É semelhante ao atributo, mas nos permite permitir recursos específicos, incluindo: sandbox

- accelerometer dá acesso à interface do Acelerômetro API de Sensores
- ambient-light-sensor dá acesso à interface Sensorial AmbientLightSensor API
- autoplay permite reproduzir automaticamente arquivos de vídeo e áudio
- camera permite acessar a câmera a partir da API getUserMedia
- display-capture permite acessar o conteúdo da tela usando a API getDisplayMedia
- fullscreen permite acessar o modo de tela cheia
- geolocation permite acessar a API de Localização Geográfica
- gyroscope dá acesso à interface do Giroscópio API de Sensores
- magnetometer dá acesso à interface do magnetômetro API de sensores
- microphone dá acesso ao microfone do dispositivo usando a API getUserMedia
- midi permite o acesso à API MIDI da Web
- payment dá acesso à API de solicitação de pagamento
- speaker permite o acesso à reprodução de áudio através dos alto-falantes do dispositivo
- usb dá acesso à API WebUSB.
- vibrate dá acesso à API de vibração
- vr dá acesso à API WebVR

## 11.4. Referenciador

Ao carregar um iframe, o navegador envia informações importantes sobre quem o está carregando no cabeçalho (observe o único, um erro de digitação com o qual devemos conviver). Refererr

O erro ortográfico do referenciador originou-se na proposta original do cientista da computação Phillip Hallam-Baker de incorporar o campo na especificação HTTP. O erro ortográfico foi definido em pedra no momento de sua incorporação no documento de padrões de Solicitação de Comentários RFC 1945

O atributo nos permite definir o referenciador para enviar ao iframe ao carregá-lo. O referenciador é um cabeçalho HTTP que permite que a página saiba quem está carregando-a. Estes são os valores permitidos:`referrerpolicy`

- `no-referrer-when-downgrade` é o padrão e não envia o referenciador quando a página atual é carregada por HTTPS e o iframe é carregado no protocolo HTTP
- `no-referrer` não envia o cabeçalho do referenciador
- `origin` o referenciador é enviado e contém apenas a origem (porta, protocolo, domínio), não a origem + caminho que é o padrão
- `origin-when-cross-origin` ao carregar a partir da mesma origem (porta, protocolo, domínio) no iframe, o referenciador é enviado em sua forma completa (origem + caminho). Caso contrário, apenas a origem é enviada
- `same-origin` o referenciador é enviado somente ao carregar da mesma origem (porta, protocolo, domínio) no iframe
- `strict-origin` envia a origem como o referenciador se a página atual for carregada por HTTPS e o iframe também for carregado no protocolo HTTPS. Não envia nada se o iframe for carregado por HTTP
- `strict-origin-when-cross-origin` envia a origem + caminho como o referenciador ao trabalhar na mesma origem. Envia a origem como o referenciador se a página atual for carregada por HTTPS e o iframe também for carregado no protocolo HTTPS. Não envia nada se o iframe for carregado por HTTP
- `unsafe-url`: envia a origem + o caminho como o referenciador mesmo ao carregar recursos de HTTP e a página atual é carregada por HTTPS

## 12. Imagens

As imagens podem ser exibidas usando a tag `img`

Essa tag aceita um atributo, que usamos para definir a origem da imagem:`src`

```

```

Podemos usar um amplo conjunto de imagens. Os mais comuns são PNG, JPEG, GIF, SVG e mais recentemente WebP.

O padrão HTML requer que um atributo esteja presente para descrever a imagem. Isso é usado por leitores de tela e também por bots de mecanismos de pesquisa:`alt`

```

```

Você pode definir os atributos `width` e `height` para definir o espaço que o elemento ocupará, para que o navegador possa contabilizá-lo e não altere o layout quando estiver totalmente carregado. É preciso um valor numérico, expresso em pixels.

```

```

## 12.1. A etiqueta `figure`

A tag é frequentemente usada junto com a tag `img`.

`figure` é uma tag semântica frequentemente usada quando você deseja exibir uma imagem com uma legenda. Você o usa assim:

```
<figure>
  
  <figcaption>A nice dog</figcaption>
</figure>
```

A tag quebra o texto da legenda `figcaption`

## 12.2. Imagens responsivas usando srcset

O atributo permite que você defina imagens responsivas que o navegador pode usar dependendo da densidade de pixels ou da largura da janela, de acordo com suas preferências. Dessa forma, ele só pode baixar os recursos necessários para renderizar a página, sem baixar uma imagem maior se estiver em um dispositivo móvel, por exemplo.

Aqui está um exemplo, onde damos 4 imagens adicionais para 4 tamanhos de tela diferentes:

```

```

No usamos a medida para indicar a largura da janela.

Como fazemos isso, também precisamos usar o atributo sizes

```


Neste exemplo, a cadeia de caracteres no atributo descreve o tamanho da imagem em relação ao visor, com várias condições separadas por uma vírgula. (max-width: 500px) 100vw, (max-width: 900px) 50vw, 800px sizes

A condição de mídia define o tamanho da imagem em correlação com a largura do visor. Em suma, se o tamanho da janela for < 500px, ele renderiza a imagem em 100% do tamanho da janela. max-width: 500px

Se o tamanho da janela for maior, mas <, ele renderizará a imagem em 50% do tamanho da janela. 900px

E se ainda maior, ele renderiza a imagem em 800px.

A unidade de medida pode ser nova para você e, em suma, podemos dizer que 1 é 1% da largura da janela, assim como 100% da largura da janela. vw vw 100vw

Um site útil para gerar as imagens e progressivamente menores é <https://responsivebreakpoints.com/.srcset>

## 12.3. A etiqueta picture

HTML também nos dá a tag, que faz um trabalho muito semelhante ao , e as diferenças são muito sutis. picture srcset

Você usa quando, em vez de apenas servir uma versão menor de um arquivo, você deseja alterá-lo completamente. Ou servir um formato de imagem diferente. picture

O melhor caso de uso que encontrei é ao servir uma imagem WebP, que é um formato ainda não amplamente suportado. Na tag, você especifica uma lista de imagens e elas serão usadas em ordem, portanto, no próximo

exemplo, os navegadores que oferecem suporte a WebP usarão a primeira imagem e retornarão ao JPG se não:picture

```
<picture>
  <source type="image/webp" srcset="image.webp" />
  
</picture>
```

A tag define um (ou mais) formatos para as imagens. A tag é o fallback caso o navegador seja muito antigo e não suporte a tag source img picture

Na tag dentro, você pode adicionar um atributo para definir consultas de mídia source picture media

O exemplo que se segue funciona como o exemplo acima com:srcset

```
<picture>
  <source media="(min-width: 500w)" srcset="dog-500.png" sizes="100vw"
/>
  <source media="(min-width: 800w)" srcset="dog-800.png" sizes="100vw"
/>
  <source media="(min-width: 1000w)" srcset="dog-1000.png"
sizes="800px" />
  <source media="(min-width: 1400w)" srcset="dog-1400.png"
sizes="800px" />
  
</picture>
```

Mas esse não é o seu caso de uso, porque, como você pode ver, é muito mais detalhado.

A tag é recente, mas agora é suportada por todos os principais navegadores, exceto Opera Mini e IE (todas as versões).picture

## 13. Acessibilidade

É importante projetarmos nosso HTML com a acessibilidade em mente.

Ter HTML acessível significa que as pessoas com deficiência podem usar a Web. Existem usuários totalmente cegos ou com deficiência visual, pessoas com problemas de perda auditiva e uma infinidade de outras deficiências diferentes.

Infelizmente este tópico não tem a importância que precisa, e não parece tão legal quanto outros.

E se uma pessoa não puder *ver* sua página, mas ainda quiser consumir seu conteúdo? Primeiro, como eles fazem isso? Eles não podem usar o mouse, eles usam algo chamado leitor de **tela**. Você não precisa imaginar isso. Você pode experimentar um agora: o Google fornece a extensão gratuita do ChromeVox Chrome. A acessibilidade também deve cuidar de permitir que as ferramentas selecionem facilmente elementos ou naveguem pelas páginas.

Páginas da Web e aplicativos Web nem sempre são criados com acessibilidade como um de seus primeiros objetivos, e talvez a versão 1 seja lançada não acessível, mas é possível tornar uma página da Web acessível após o fato. Mais cedo é melhor, mas nunca é tarde demais.

É importante e, no meu país, os sites construídos pelo governo ou por outras organizações públicas devem ser acessíveis.

O que isso significa para tornar um HTML acessível? Deixe-me ilustrar as principais coisas que você precisa pensar.

Nota: existem várias outras coisas para cuidar, que podem ir no tópico CSS, como cores, contraste e fontes. Ou como tornar as imagens SVG acessíveis. Eu não falo sobre eles aqui.

## 13.1. Usar HTML semântico

HTML semântico é muito importante e é uma das principais coisas que você precisa cuidar. Deixe-me ilustrar alguns cenários comuns.

É importante usar a estrutura correta para as tags de título. O mais importante é , e você usa números mais altos para os menos importantes, mas todos os títulos de mesmo nível devem ter o mesmo significado (pense nisso como uma estrutura de árvore)h1

h1  
h2  
h3  
h2  
h2  
h3  
h4

Use e em vez de e . Visualmente eles parecem os mesmos, mas os 2 primeiros têm mais significado associado a eles. e são elementos mais visuais.strongembibi

As listas são importantes. Um leitor de tela pode detectar uma lista e fornecer uma visão geral e, em seguida, permitir que o usuário escolha entrar na lista ou não.

Uma tabela deve ter uma marca que descreva seu conteúdo:caption

```
<table>
  <caption>
    Dogs age
  </caption>
  <tr>
    <th>Dog</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Roger</td>
    <td>7</td>
  </tr>
</table>
```

## 13.2. Usar atributos para imagens alt

Todas as imagens devem ter uma tag descrevendo o conteúdo da imagem. Não é apenas uma boa prática, é exigido pelo padrão HTML e seu HTML sem ele não é validado.

```

```

Também é bom para os mecanismos de pesquisa, se isso for um incentivo para você adicioná-lo.

## 13.3. Usar o atributo role

O atributo permite que você atribua funções específicas aos vários elementos em sua página.

Você pode atribuir muitas funções diferentes: complementar, lista, item, principal, navegação, região, guia, alerta, aplicativo, artigo, banner, botão, célula, caixa de seleção, contentinfo, diálogo, documento, feed, figura, formulário, grade, gridcell, cabeçalho, img, caixa de listagem, linha, grupo de linhas, pesquisa, alternância, tabela, painel de guias, caixa de texto, temporizador.

É muito e para a referência completa de cada um deles eu dou-lhe [este link MDN](#). Mas você não precisa atribuir uma função a cada elemento da página. Os leitores de tela podem inferir a partir da tag HTML na maioria dos casos. Por exemplo, você não precisa adicionar uma tag de função a tags semânticas como , , .navbuttonform

Vamos pegar o exemplo da tag. Você pode usá-lo para definir a navegação da página da seguinte forma:

```
<nav>
  <ul>
    <li><a href="/">Home</a></li>
    <li><a href="/blog">Blog</a></li>
```

```
</ul>  
</nav>
```

Se você fosse *forçado* a usar uma tag em vez de , você usaria a função:`divnavnavigation`

```
<div role="navigation">  
  <ul>  
    <li><a href="/">Home</a></li>  
    <li><a href="/blog">Blog</a></li>  
  </ul>  
</div>
```

Então, aqui você tem um exemplo prático: é usado para atribuir um valor significativo quando a tag ainda não transmite o significado.`role`

## 13.4. Usar o atributo `tabindex`

O atributo permite que você altere a ordem de como pressionar a tecla Tab seleciona elementos "selecionáveis". Por padrão, apenas links e elementos de formulário são "selecionáveis" pela navegação usando a tecla Tab (e você não precisa defini-los).

A adição torna um elemento selecionável:`tabindex="0"`

```
<div tabindex="0">...</div>
```

Em vez disso, o uso remove um elemento dessa navegação baseada em guias e pode ser bastante útil.`tabindex="-1"`

## 13.5. Usar os atributos `saria`

ARIA é um acrônimo que significa Accessible Rich Internet Applications e define semântica que pode ser aplicada a elementos.

## 13.5.1. aria-label

Esse atributo é usado para adicionar uma cadeia de caracteres para descrever um elemento.

Exemplo:

```
<p aria-label="The description of the product">...</p>
```

Eu uso esse atributo na barra lateral do meu blog, onde tenho uma caixa de entrada para pesquisa sem um rótulo explícito, pois tem um atributo de espaço reservado.

## 13.5.2. aria-labelledby

Esse atributo define uma correlação entre o elemento atual e aquele que o rotula.

Se você sabe como um elemento pode ser associado a um elemento, isso é semelhante `inputlabel`

Passamos o id do item que descreve o elemento atual.

Exemplo:

```
<h3 id="description">The description of the product</h3>
```

```
<p aria-labelledby="description">...</p>
```

## 13.5.3. aria-describedby

Esse atributo nos permite associar um elemento a outro elemento que serve como descrição.

Exemplo:

```
<button aria-describedby="payNowDescription">Pay now</button>

<div id="payNowDescription">
  Clicking the button will send you to our Stripe form!
</div>
```

### 13.5.4. Usar ária oculta para ocultar conteúdo

Eu gosto de um design minimalista em meus sites. Meu blog, por exemplo, é principalmente apenas conteúdo, com alguns links na barra lateral. Mas algumas coisas na barra lateral são apenas elementos visuais que não se somam à experiência de uma pessoa que não pode ver a página. Como a imagem do meu logotipo ou o seletor de temas escuros / brilhantes.

Adicionar o atributo dirá aos leitores de tela para ignorar esse elemento.`aria-hidden="true"`

 → Você pode me seguir no [Twitter](#)

 → Todos os anos organizo um [BOOTCAMP](#) de codificação para lhe ensinar JS, Tailwind, React e Next.js (próxima edição Q1 2024)

 → Quer aprender JavaScript AGORA MESMO com um currículo bem organizado? [O CURSO JS](#) é o lugar

 → Vá para [o SOLO LAB](#) se você estiver em iniciar e administrar um negócio na Internet como uma pessoa solo (NOVO CURSO EM BREVE, junte-se à lista de espera para ficar por dentro)

 → Leia meus outros ebooks O Manual C O Manual da Linha de Comando O Manual CSS O Manual do Expresso O Manual do Go O Manual HTML O Manual do JS O Próximo.js Manual O Nó.js Manual O Manual do PHP O Manual do Python O Manual do [React](#)