

# HTML e CSS

As linguagens da web

*Um guia de desenvolvimento web*

*Por Jefferson Chaves*

# **HTML e CSS**

As linguagens da web

*Um guia de desenvolvimento web*

*Por Jefferson Chaves*

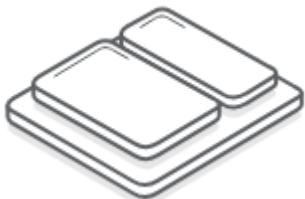
# HTML e CSS

## Introdução



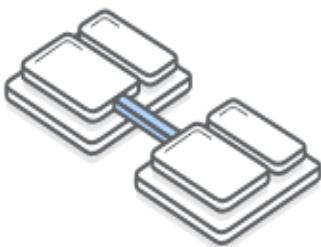
Saiba qual o propósito de se usar HTML, CSS e JavaScript, Conheça a diferença entre frameworks e linguagens e como se orientar em um projeto básico de site com o [VSCode](#).

## Páginas da web básicas



Conheça a estrutura básica de cada página web na Internet, bem como elementos HTML fundamentais como títulos, parágrafos, listas, etc.

## Links e imagens

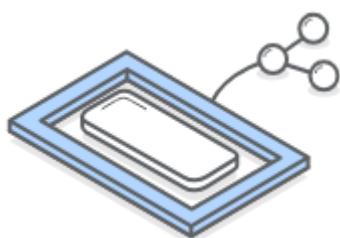


Descubra como criar links para outras páginas, referindo-se a elas com caminhos absolutos, relativos e relativos à raiz, incorporando imagens e retoques finais em nosso esqueleto básico de página da web.



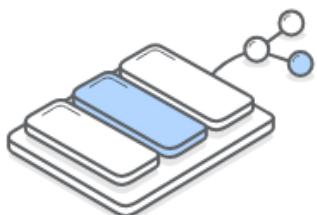
## Olá, CSS

Adicione folhas de estilo, estilizando elementos HTML com todos os tipos de propriedades CSS, selecionando diferentes elementos e utilizando estilos em várias páginas da web.



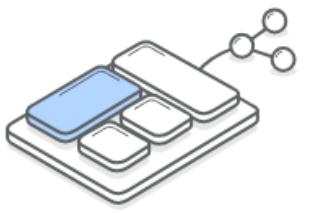
## O Modelo de Caixa

Caixas de bloco, caixas embutidas, preenchimento, bordas, margens, dimensões e uma introdução sobre como belos sites são construídos com caixas.



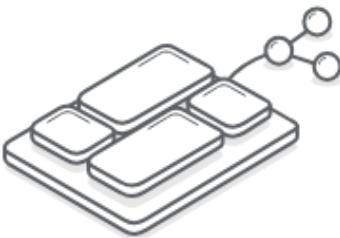
## Seletores CSS

Seletores de classe, seletores descendentes, estilos de links com pseudoclasses e por que seletores de ID são uma má ideia.



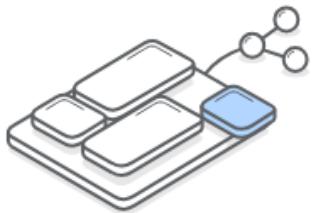
## Flutuadores

Introdução aos layouts CSS baseados em float, controlando o fluxo horizontal da página.



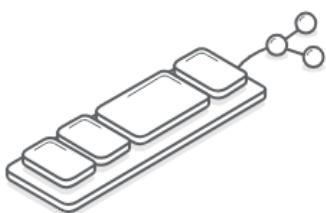
## Flexbox

Saiba como usar **contêineres flexíveis** e **itens flexíveis**, alinhando-os vertical e horizontalmente, distribuindo, agrupando, reordenando e criando layouts CSS modernos.



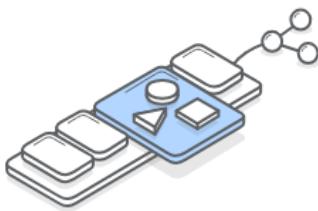
## Posicionamento avançado

Posicionamento estático, relativo, absoluto e fixo, menus suspensos, **índice z** e outros códigos CSS sofisticados que farão você se sentir realmente um ninja.



## Design responsivo

Saiba como realizar consultas de mídia para controlar quando as regras CSS são aplicadas, desenvolvimento mobile-first e desativação do zoom da janela de visualização.



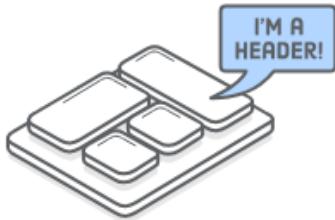
## Imagens responsivas

A parte difícil do design responsivo: telas de alta resolução, imagens fluidas, otimização de dispositivos com a propriedade `srcset` e direção de arte com o elemento `<picture>`.



## Formulários

A interação com o usuário acontece por meio de formulários! Saiba como criar campos de texto, botões de opção, menus suspensos, caixas de seleção e botões, bem como o CSS para estilizar todos eles.



## HTML semântico

O esboço do documento, artigos, seções, navegação, cabeçalhos, rodapés, apartes, figuras e outras maneiras de tornar sua marcação HTML mais informativa.

## JavaScript

O esboço do documento, artigos, seções, navegação, cabeçalhos, rodapés, apartes, figuras e outras maneiras de tornar sua marcação HTML mais informativa.



## Tipografia da web

Onde encontrar fontes da web, hospedagem local ou externa, lidar com vários tipos de fontes e princípios tipográficos básicos, como alinhamento de texto, entrelinhamento e medida.

# Introdução

## Capítulo 1

*O encontro mais amigável que você terá com desenvolvimento web*

Aprender HTML e CSS pode parecer complicado à primeira vista, mas não é. Neste guia de amigável para aprendizado de HTML e CSS, você será orientado em tudo, desde a seleção de um bom editor de texto (o que é surpreendentemente importante) até a criação de páginas da Web completas e de qualidade profissional a partir do zero.

Esse guia foi idealizado para ser a única introdução ao HTML e CSS que você precisará. Lendo cada seção e escrevendo cada trecho de código, este tutorial tem o potencial de substituir centenas de páginas ou dezenas de horas em cursos online. O objetivo é deixar o primeiro contato com o desenvolvimento web o mais fácil possível para desenvolvedores web iniciantes.

## HTML, CSS e JavaScript

HyperText Markup Language (HTML), Cascading Style Sheets (CSS) e JavaScript são as linguagens que formam a **base de uma página web**. Eles estão intimamente relacionados, mas cada um projetado para tarefas muito específicas. Compreender

como eles interagem será um grande passo para se tornar um desenvolvedor web. Estaremos expandindo isso ao longo do guia, mas a essência é:

- HTML serve para adicionar significado ao conteúdo bruto, marcando-o;
- CSS serve para formatar os elementos HTML e seu conteúdo;
- JavaScript serve para tornar esse conteúdo e formatação interativos.

Pense no HTML como a estrutura, o texto e as imagens por trás de uma página da web, no CSS como o responsável por definir como a página será realmente exibida e no JavaScript como os comportamentos que podem manipular o HTML e o CSS.



Por exemplo, você pode marcar uma sequência específica de texto como um parágrafo com este HTML:

```
<p id='some-paragraph'>Isso é um parágrafo.</p>
```

Então, você pode definir o tamanho e a cor desse parágrafo com algum CSS:

```
p {  
    font-size: 20px;  
    color: blue;  
}
```

E, se quiser ser sofisticado, você pode reescrever esse parágrafo quando o usuário clicar nele com algum JavaScript (algo mais sofisticado será apresentado no futuro):

```
var p = document.getElementById('some-paragraph');

p.addEventListener('click', function(event) {
    p.innerHTML = 'Você clicou aqui!';
});
```

Como você pode ver, HTML, CSS e JavaScript são linguagens totalmente diferentes, mas todas se referem umas às outras de alguma forma. A maioria dos sites depende dos três, mas a aparência de cada site é determinada por HTML e CSS. Isso torna este tutorial um excelente ponto de partida para sua jornada de desenvolvimento web.

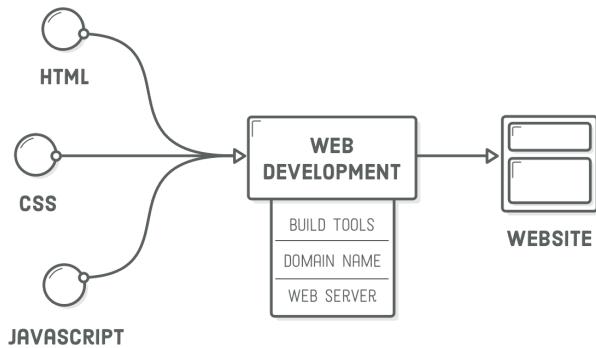
## Linguagens de Programação vs Desenvolvimento web

Dominar HTML, CSS e JavaScript é apenas um pré-requisito para se tornar um desenvolvedor web profissional. Existem várias outras habilidades práticas que você precisa para administrar um site:

- Organizar seu código HTML em modelos reutilizáveis (os templates);
- Saber usar um servidor web;
- Mover arquivos do seu computador local para o seu servidor web;
- Reverter para uma versão anterior quando você estraga alguma coisa;
- Configurar um nome de domínio para o seu site;
- etc;

Lidar com essas complexidades envolve a criação de vários “ambientes” para organizar seus arquivos e lidar com a implementação e implantação de sua aplicação web ou seu site. Tudo isso é paralelo ao código HTML, CSS e JavaScript. Deixe isso

para quando você for um Jedi em desenvolvimento Web: por enquanto concentre-se no HTML, CSS E JavaScript

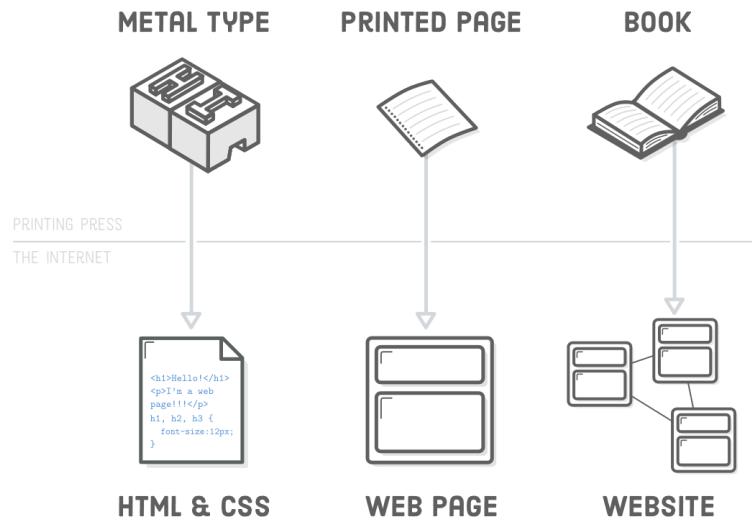


Ganhar fluência em HTML e CSS é um primeiro passo significativo para se tornar um verdadeiro desenvolvedor web. Embora desafios e outras adversidades possam aparecer, após a leitura desse guia, você terá as habilidades necessárias para recriar a grande maioria das páginas da Web na Internet. **Confia!**

## Publicação na web

Então, o que é “aprender” HTML e CSS? Gostamos de olhar para isso através das lentes históricas da indústria gráfica. Na época das primeiras impressões, os impressores criavam documentos organizando caracteres de metal, mergulhando-os em tinta e pressionando-os sobre um pedaço de papel.

De muitas maneiras, é exatamente isso que os desenvolvedores web fazem, exceto que em vez de organizar tipos móveis, eles escrevem HTML e CSS. Estamos preocupados com a mesma tarefa que eles: **transmitir conteúdo de maneira significativa**. Lidamos até com as mesmas questões de apresentação que eles, como selecionar a fonte a ser usada, definir o tamanho dos títulos e determinar o espaço entre as linhas do texto.



As impressoras imprimiam um monte de páginas para serem encadernadas como um livro. Hoje em dia, criamos vários arquivos HTML e os vinculamos em um site. Aprender HTML e CSS é uma questão de compreender a marcação HTML disponível e as regras CSS para fazer um navegador renderizar esses arquivos exatamente como deveriam.

## Foque nos fundamentos, não em Frameworks

Existem todos os tipos de estruturas de desenvolvimento web front-end por aí (Bootstrap , Pure CSS, Tailwind CSS , só para citar alguns). O objetivo de cada um deles é abstrair alguns dos aspectos redundantes da criação de páginas web do zero. Esse tipo de estrutura são uma parte importante do desenvolvimento web no mundo real e definitivamente vale a pena explorá-las, mas somente depois de dominar o básico do HTML e CSS (veremos frameworks em nossa disciplina, fiquem tranquilos!)

Este guia é sobre fundamentos de HTML e CSS. Você terá a capacidade de criar praticamente tudo o que precisar como desenvolvedor web com HTML e CSS brutos. Isso é algo que permanece com você para sempre , apesar das novas adições aos

padrões HTML e CSS ou das novas estruturas da moda que ajudam você a fazer as coisas com mais rapidez.

## Aprendizagem prática

Esse guia tem tudo a ver com **aprendizado na prática** (me contradizendo com os parágrafos anteriores, que foram só teoria). Iremos explorar exemplos concretos, explicando os aspectos conceituais de HTML e CSS ao longo do caminho.

Para aproveitar ao máximo este guia, você deve criar páginas da webativamente e acompanhar cada etapa de cada capítulo. Se você realmente quer se tornar um desenvolvedor web, digite cada trecho de código caractere por caractere, em vez de copiá-los e colá-los em seu editor de texto.

Por que? Porque é isso que você realmente fará como um verdadeiro desenvolvedor web. Digitar exemplos de código exercita a memória muscular que será útil quando você estiver codificando códigos para sites reais.

## Caixa de Ferramentas

Você precisará, basicamente, de duas ferramentas: um editor de texto e um navegador da web **decente**. Seu trabalho básico é escrever o código em seu editor de texto e, em seguida, abri-lo em um navegador da web para ver sua aparência. Ao começar a criar seus próprios sites, você eventualmente adicionará mais ferramentas à sua caixa de ferramentas, mas é importante começar de forma mínima e aprender completamente os fundamentos de HTML e CSS.



Editores decentes vêm com recursos que permitem escrever código mais rápido do que normalmente, como tags de preenchimento automático, pular texto e navegar em seu sistema de arquivos. Aproveitar totalmente o seu editor de texto é a parte artesanal do aprendizado de HTML e CSS. Dito isso, reserve um tempo para conhecer, razoavelmente, seu editor de texto. Pesquise, veja vídeos sobre o seu editor!

O único pré-requisito para um bom navegador da web é que ele esteja atualizado e em uso comum. **Chrome** e **Firefox** são os favoritos entre os desenvolvedores web. A maioria dos tutoriais pela internet usará um destes navegadores. O Safari está bem se você também estiver executando o OS X. Sugiro fortemente não criar sites com o Internet Explorer ou Edge. O desenvolvimento web profissional geralmente requer uma maneira eficiente de testar código em todos esses navegadores, mas isso é um pouco mais complicado do que precisamos agora.

## Editor de texto VSCode

Recomendo o editor de texto VSCode . É fácil de usar, mesmo para iniciantes, oferece todos os recursos úteis mencionados acima e está disponível para todos os principais sistemas operacionais. Também é infinitamente configurável, o que se tornará importante à medida que você identificar tarefas repetitivas que podem ser automatizadas.

Se você ainda não tem o VSCode, faça o [download agora](#) , pois você precisará dele no próximo capítulo. Depois de baixá-lo, abra-o para que possamos fazer um breve tour

por seus principais recursos. Ao iniciá-lo pela primeira vez, você deverá ver uma tela de boas-vindas. Feche essas telas clicando no ícone X nas guias correspondentes.

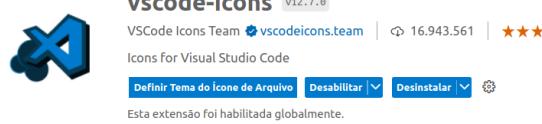
Configurar seu ambiente de desenvolvimento faz parte da rotina dos desenvolvedores. Seu VScode pode ser customizado por meio de temas, cores, fontes e etc. Além disso, os recursos do VScode podem ser potencializados por meio de extensões! Aqui segue uma lista de extensões úteis:

Fecha automaticamente uma tag html aberta. Muda o fechamento da tag automaticamente ao mudar a tag de abertura	 <p><b>Auto Close Tag</b> v0.5.15 Jun Han Automatically add HTML/XML close tag, same as Visual Studio IDE or ... <a href="#">Desabilitar</a> <a href="#">Desinstalar</a>  Esta extensão foi habilitada globalmente.</p>
Importação inteligente de arquivos.	 <p><b>Auto Import</b> v1.5.4 steoates Automatically finds, parses and provides code actions and code comp... <a href="#">Desabilitar</a> <a href="#">Desinstalar</a>  Esta extensão foi habilitada globalmente.</p>
Aponta a maioria dos erros de sintaxe e descreve o erro cometido.	 <p><b>Error Lens</b> v3.16.0 Alexander   ⚡ 3.453.008   ★★★★★ (146) Improve highlighting of errors, warnings and other language diagnostics. <a href="#">Habilitar</a> <a href="#">Desinstalar</a>  Esta extensão foi desabilitada globalmente pelo usuário.</p>
Permite executar a aplicação e ver as mudanças do código em tempo real.	 <p><b>Live Server</b> v5.7.9 Ritwick Dey   ⚡ 44.692.144   ★★★★★ (473) Launch a development local Server with live reload feature for static &amp; dynamic pages <a href="#">Desabilitar</a> <a href="#">Desinstalar</a>  Esta extensão foi habilitada globalmente.</p>
Permite criar sessão compartilhada de codificação, tipo Google Docs	 <p><b>Live Share</b> v1.0.598 Microsoft   <a href="#">microsoft.com</a>   ⚡ 15.680.725   ★★★★★ (150) Real-time collaborative development from the comfort of your favorite tools. <a href="#">Desabilitar</a> <a href="#">Desinstalar</a>  Esta extensão foi habilitada globalmente.</p>
Assistente para sugerir caminhos para arquivos, tais como imagens ou arquivos CSS.	 <p><b>Path Intellisense</b> v2.8.5 Christian Kohler   <a href="#">christiankohler.net</a>   ⚡ 12.072.047   ★★★★★ (119) Visual Studio Code plugin that autocompletes filenames <a href="#">Desabilitar</a> <a href="#">Desinstalar</a>  Esta extensão foi habilitada globalmente.</p>

Assistente para sugerir nomes de classes ou formatações a partir de um arquivo CSS linkado.



Deixa os ícones mais bonitinhos.



## Criando um Projeto

Cada site em que você trabalha no VSCode é um “projeto”, que é essencialmente apenas uma pasta em seu sistema de arquivos que contém vários arquivos HTML e CSS. Vamos explorar o VSCode por meio da criação de um projeto exemplo. Por enquanto, clique em Arquivo > Abrir Pasta na barra de menu para abrir uma janela de arquivo e selecione Nova pasta para criar uma nova pasta. Chame-a de `hello-vscode` e clique em Abrir. É só isso, um projeto em VSCode nada mais é do que uma pasta aberta com alguns arquivos dentro.

## Resumo

Parabéns! Você concluiu o primeiro passo para se tornar um desenvolvedor web Jedi! Espero que este capítulo tenha lhe dado uma compreensão básica de como HTML e CSS funcionam em sintonia para se criar uma aplicação web.

O editor VSCode fará parte de nossa rotina quase que diária da disciplina, então certifique-se de estar confortável com ele. Deixo aqui meu incentivo para que você exercente um pouco com o projeto de exemplo, adicionando arquivos e praticando a navegação de um para outro. Dominar a arte de escrever HTML e CSS é o que distingue os desenvolvedores incrivelmente produtivos dos programadores girinos.

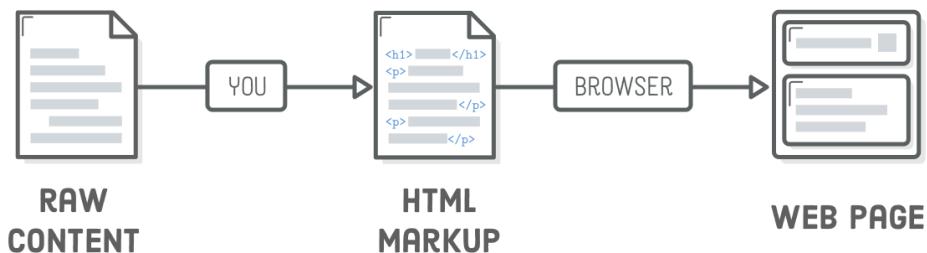
Agora que temos um editor de texto adequado, estamos prontos para começar a codificar algumas páginas da web reais. Começaremos explorando os elementos HTML mais comuns nos próximos dois capítulos e depois adicionaremos um pouco de CSS às nossas páginas.

# Páginas Web Básicas

## Capítulo 2

*Um guia para criar páginas da web (realmente) simples*

**HTML define o conteúdo de cada página da web na Internet.** Ao “marcar” seu conteúdo bruto com **tags HTML**, você pode informar aos navegadores como deseja que diferentes partes de seu conteúdo sejam exibidas. Criar um documento HTML com conteúdo devidamente marcado é o **primeiro passo no desenvolvimento de uma página web**.



Neste capítulo, construiremos nossa primeira página da web. Não se preocupe se ela não parecer **com a página daqueles sites famosos**. Nosso objetivo aqui é fazer uma introdução completa aos elementos HTML com os quais os desenvolvedores web trabalham diariamente. À medida que você avança nos exemplos, digite-os em seu VSCode.

## Configurações

Vamos começar criando um novo projeto com no VScode chamado “`ola-html`”. Em seguida, crie um novo arquivo chamado `index.html`. Este arquivo HTML representa uma única página web e é onde colocaremos todo o código deste capítulo. Se você ainda não configurou o VSCode, pare o que está fazendo e o configure com as extensões que discutimos em aula.



Lembre-se de que o fluxo de trabalho básico para desenvolvedores web é editar HTML em seu editor de texto e visualizar essas alterações em um navegador web, então isso é exatamente o que você deve fazer em cada seção deste capítulo.

## Estrutura de uma página da web

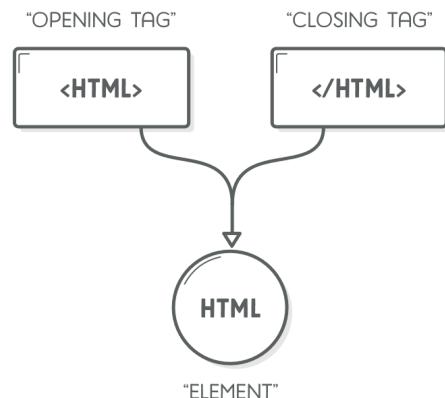
Adicione a seguinte marcação HTML ao nosso arquivo `index.html`. É com isso que você começará para cada página da web que irá produzir. Normalmente, você usaria algum tipo de mecanismo de modelagem para evitar digitar as partes redundantes, mas para este guia, nos concentraremos no HTML bruto.

```
<!DOCTYPE html>
<html>
<head>
    <!-- Metadados vão aqui -->
</head>
<body>
    <!-- Conteúdos vão aqui -->
</body>
</html>
```

Primeiro, precisamos informar aos navegadores que esta é uma página HTML5 com a linha `<!DOCTYPE html>`. Esta é apenas uma string especial que os navegadores procuram quando tentam exibir nossa página web. Sem essa linha a página web até

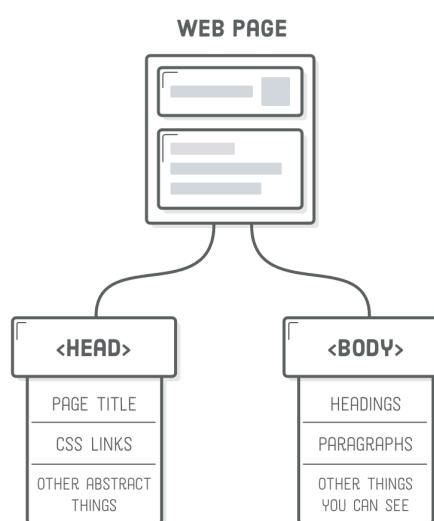
funciona, mas o navegador precisa deduzir qual versão do HTML estamos usando e nem sempre o navegador deduzirá da forma correta.

Nossa página da web precisa ser envolvida por tags `<html>`. O elemento `<html>` é chamado de “tag de abertura”, enquanto `</html>` é chamado de “tag de fechamento”. Tudo dentro dessas tags é considerado parte do elemento `<html>`, que é aquela coisa etérea que é criada quando um navegador analisa suas tags HTML.



Dentro do elemento `<html>`, temos mais dois elementos chamados `<head>` e `<body>`. O cabeçalho de uma página web contém todos os seus metadados, como o título da página, quaisquer folhas de estilo CSS e outras coisas que são necessárias para renderizar a página, mas que você não quer necessariamente que o usuário veja. A maior parte da nossa marcação HTML ficará no `<body>`, que representa o conteúdo visível da página. Observe que abrir nossa página em um navegador da web não exibirá nada.

O propósito desta divisão entre `<head>` e `<body>` ficará mais claro quando começarmos a trabalhar com CSS.



Observe também a sintaxe do comentário HTML no trecho acima. Qualquer coisa que comece `<!--` e termine com `-->` é conhecida como comentário e será completamente ignorada pelo navegador. Isso é útil para documentar seu código e fazer anotações para você mesmo.

## Títulos de Páginas

Uma das peças mais importantes dos metadados é o título da sua página da web, definido pelo elemento `<title>`, apropriadamente nomeado. Os navegadores exibem isso na guia da página. Já buscadores como o Google, exibem esse valor nos resultados do mecanismo de pesquisa, por exemplo. Atualize o arquivo `index.html` para corresponder ao seguinte:

```
<!DOCTYPE html>
<html>
<head>
    <title>Desenvolvimento Web é Moleza!</title>
</head>
<body>
    <!-- Conteúdos vão aqui -->
</body>
</html>
```

Ao recarregar a página em seu navegador, você ainda deverá ver uma página vazia, mas também verá **Desenvolvimento Web é Moleza!** na aba do navegador:



Observe como todas as tags HTML em nossa página da web estão perfeitamente **aninhadas**<sup>1</sup>. É muito importante garantir que não haja elementos sobrepostos. Por exemplo, o elemento `<title>` deveria estar dentro de `<head>`, então você nunca iria querer adicionar a tag `</head>` de fechamento antes da tag `</title>` que realiza o fechamento:

```
←— (Cuidado, não faça isso!) →  
<head>  
  <title>  
    Desenvolvimento Web é Moleza!  
  </head>  
</title>
```

---

<sup>1</sup> Não confunda **aninhado** com **alinhado**. Um código aninhado segue ordem de recuos que organiza o código e facilita a leitura. Já fazíamos isso na disciplina de programação!

# Parágrafos

Os títulos são muito bons, mas vamos fazer algo que possamos realmente ver. O elemento `<p>` marca todo o texto dentro dele como um parágrafo. Tente adicionar o seguinte elemento `<p>` ao corpo da nossa página web:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Desenvolvimento Web é Moleza!</title>
  </head>
  <body>
    <p>Inicialmente, precisamos aprender o HTML básico.</p>
  </body>
</html>
```

Agora você deve conseguir ver algum conteúdo da página. Novamente, como este é o conteúdo que queremos exibir, ele precisa estar no elemento `<body>` (não no `<head>`).



Observe também como os elementos `<p>` e `<title>` são recuados duas vezes, enquanto `<body>` e `<head>` são recuados uma vez. Recuar elementos aninhados como este é uma prática recomendada importante que torna seu HTML mais fácil de ler

para outros desenvolvedores (ou para você mesmo, se voltar daqui a 5 meses e quiser alterar algumas coisas).

Cabe a você decidir se deseja usar espaços ou caracteres de tabulação para recuos. Você pode definir essa preferência em seu editor de texto.

## Títulos

Já vimos o elemento `<title>`, mas os títulos que vamos tratar aqui são diferentes. Eles são usados para definir títulos que serão exibidos na página. HTML fornece seis níveis de títulos e os elementos correspondentes são:

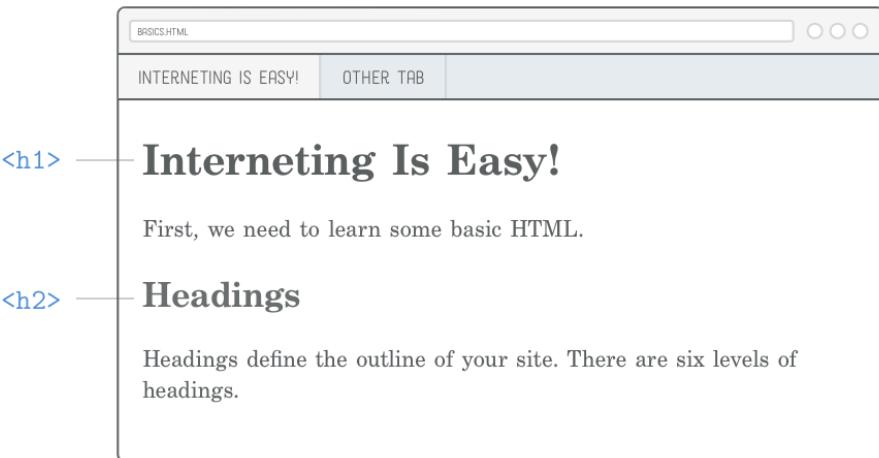
O primeiro título de uma página normalmente deve ser um `<h1>`, então vamos inserir um acima do nosso elemento `<p>` existente. É muito comum (mas não uma exigência ou regra) que o primeiro elemento `<h1>` corresponda ao `<title>` do documento, como acontece aqui:

```
<body>
  <h1>Desenvolvimento Web é Moleza!</h1>
  <p>Inicialmente, precisamos aprender o HTML básico.</p>
</body>
```

Por padrão, os navegadores renderizam títulos menos importantes em fontes menores. Por exemplo, vamos incluir um título de segundo nível e ver o que acontece:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Desenvolvimento Web é Moleza!</title>
  </head>
  <body>
    <h1>Desenvolvimento Web é Moleza!</h1>
    <p>Inicialmente, precisamos aprender o HTML básico.</p>
  </body>
</html>
```

Isso deve resultar em uma página da web parecida com esta:



Os títulos são a principal forma de marcar diferentes seções do seu conteúdo. Eles definem o contorno da sua página da web como os **humanos** e os **mecanismos de pesquisa** o veem, o que torna a seleção de títulos relevantes essencial para uma página da web de alta qualidade.

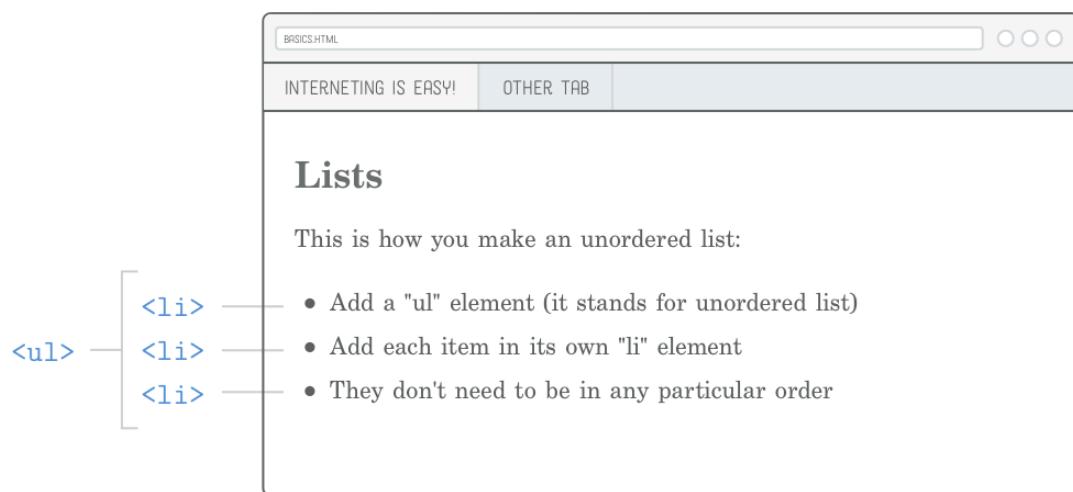
# Listas não ordenadas

Sempre que você envolver um trecho de texto com tags HTML, você adiciona um novo significado a esse texto. Agrupar o conteúdo em tags `<ul>` informa ao navegador que tudo o que está dentro deve ser renderizado como uma “lista não ordenada”. Para denotar itens individuais nesta lista, você os agrupa em tags `<li>`, da seguinte forma assim:

```
<h2>Listas!</h2>
<p>É assim que se faz uma lista não ordenada:</p>
```

```
<ul>
  <li>Adicione um "ul" (isso significa Unordered List)</li>
  <li>Adicione cada item em seu próprio "li" (List Item)</li>
  <li>Eles não precisam estar em ordem específica</li>
</ul>
```

Depois de adicionar esta marcação ao elemento `<body>` (abaixo do conteúdo existente), você deverá ver uma lista com marcadores com um marcador dedicado para cada elemento `<li>`:



A **especificação HTML define regras** estritas sobre quais elementos podem estar dentro de outros elementos. Neste caso, os elementos `<ul>` devem conter apenas elementos `<li>`, o que significa que você nunca deve escrever algo assim:

```
←— (Isso é ruim!) →  
<ul>  
  <p>Adicione um "ul" (isso significa Unordered List)</p>  
</ul>
```

Em vez disso, você deve envolver esse parágrafo com tags `<li>`:

```
←— (Isso é melhor) →  
<ul>  
  <li><p>Adicione um "ul" (isso significa Unordered List)</p></li>  
</ul>
```

Como sabemos que elementos `<ul>` só aceitam elementos `<li>` e que elementos `<li>` podem aceitar parágrafos? Porque a [Mozilla Developer Network \(MDN\)](#) diz isso. MDN é uma excelente referência de elementos HTML. Tentaremos cobrir o máximo que pudermos sobre como usar elementos HTML básicos, mas sempre que você não tiver certeza sobre um elemento específico, faça uma pesquisa rápida no Google por “MDN `<nome-elemento>`”.

# Listas ordenadas

Em listas não ordenadas, a reorganização dos elementos `<li>` não deve alterar o significado da lista. Se a sequência dos itens da lista for importante, você deverá usar uma “lista ordenada”. Para criar uma lista ordenada, basta alterar o elemento `<ul>` para `<ol>`. Adicione o seguinte conteúdo à seção Listas de `index.html`:

```
<p>Esta é a aparência de uma lista ordenada:</p>
```

```
<ol>
  <li>Observe o novo elemento "ol" envolvendo tudo</li>
  <li>Mas os elementos dos itens da lista são os mesmos</li>
  <li>Observe como os números aumentam por conta própria</li>
  <li>As coisas estão em ordem! </li>
</ol>
```

Ao recarregar a página em seu navegador, você notará que o navegador incrementou automaticamente a contagem de cada elemento `<li>`. No capítulo Olá, CSS (ainda em desenvolvimento), aprenderemos como alterar o tipo de números exibidos.

This is how you make an unordered list:

- Add a "ul" element (it stands for unordered list)
- Add each item in its own "li" element
- They don't need to be in any particular order

This is what an ordered list looks like:

1. Notice the new "ol" element wrapping everything
2. But, the list item elements are the same
3. Also note how the numbers increment on their own
4. You should be noticing things in this precise order, because this is an ordered list

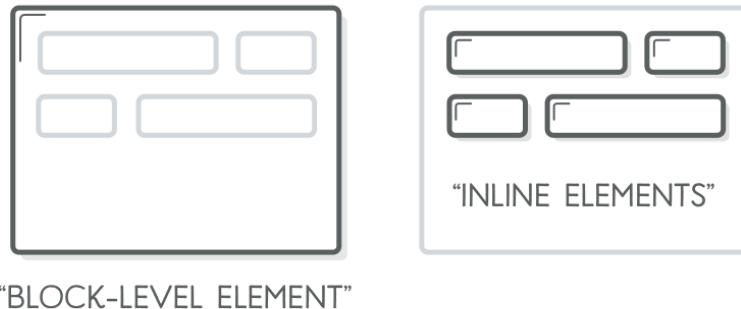
A diferença entre uma lista não ordenada e uma lista ordenada pode parecer boba, mas realmente tem significado para navegadores da web, mecanismos de pesquisa e, claro, para leitores humanos. Também é mais fácil do que enumerar manualmente cada item da lista.

Procedimentos passo a passo, como receitas, instruções e até índices, são bons candidatos para listas ordenadas, enquanto listas `<ul>` são melhores para representar inventários de itens, características de produtos, comparações de prós e contras e menus de navegação.

## Elementos de ênfase (itálico)

Até agora, trabalhamos apenas com “elementos de nível de bloco” (também chamados de “conteúdo de fluxo”). O outro tipo principal de conteúdo são “elementos inline”, “elementos em linha”, ou “conteúdo de frase”, que são tratados de maneira um pouco diferente. Os elementos em nível de bloco são sempre renderizados em uma

nova linha, enquanto os elementos embutidos podem afetar seções de texto em qualquer lugar dentro de uma linha.

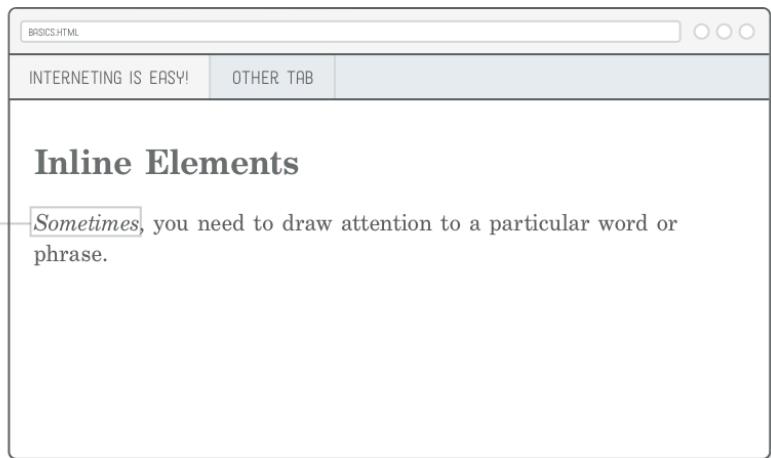


Por exemplo, `<p>` é um elemento de nível de bloco, enquanto `<em>` é um elemento embutido que afeta uma extensão de texto dentro de um parágrafo. Significa “ênfase” ou “emphasis” em inglês, e normalmente é exibido como texto em itálico. Tente adicionar uma nova seção demonstrando o texto enfatizado à nossa página da web de exemplo:

```
<h2>Elementos inline</h2>
```

```
<p><em>Às vezes</em>, você precisa chamar a atenção para uma determinada palavra ou frase.</p>
```

A parte envolvida pelas tags `<em>` deve ser renderizada em itálico, conforme mostrado abaixo. Observe como apenas parte de uma linha foi afetada, o que é característico de elementos inline. No capítulo CSS Box Model, descobriremos como os elementos inline e de bloco se comportam no layout de uma página.



Caso ainda não tenha entendido, é muito importante que você aninhe corretamente seus elementos HTML. É mais fácil bagunçar a ordem das tags quando você usa vários elementos embutidos, então certifique-se de verificar se sua marcação nunca fica assim:

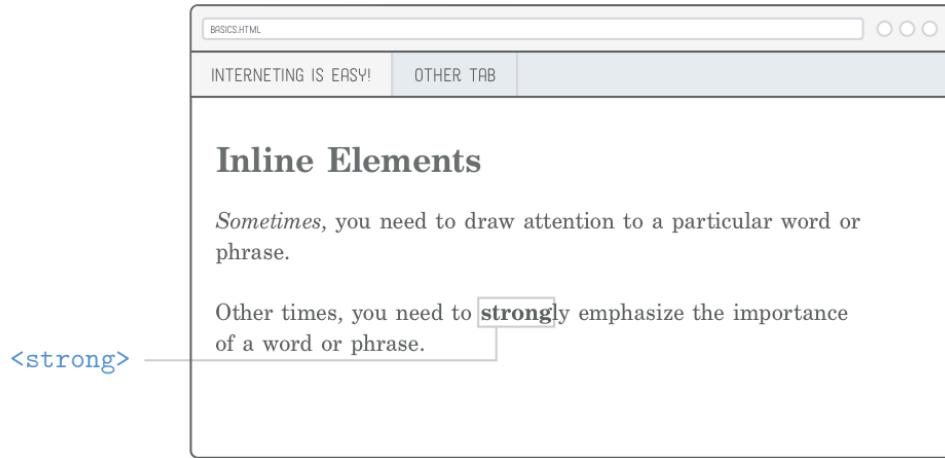
```
←— (Novamente, não faça isso!) —→  
<p>Este é um texto <em>enfatizado</p></em>
```

## Elementos Fortes (negrito)

Se quiser ser mais enfático do que uma tag `<em>`, você pode usar tag `<strong>`. É um elemento inline como `<em>` e se parece com isto:

```
<h2>Elementos inline</h2>  
  
<p><em>Às vezes</em>, você precisa chamar a atenção para uma determinada palavra ou frase.</p>  
  
<p>Outras vezes você precisa enfatizar <strong>fortemente</strong> a importância de uma palavra ou frase.</p>
```

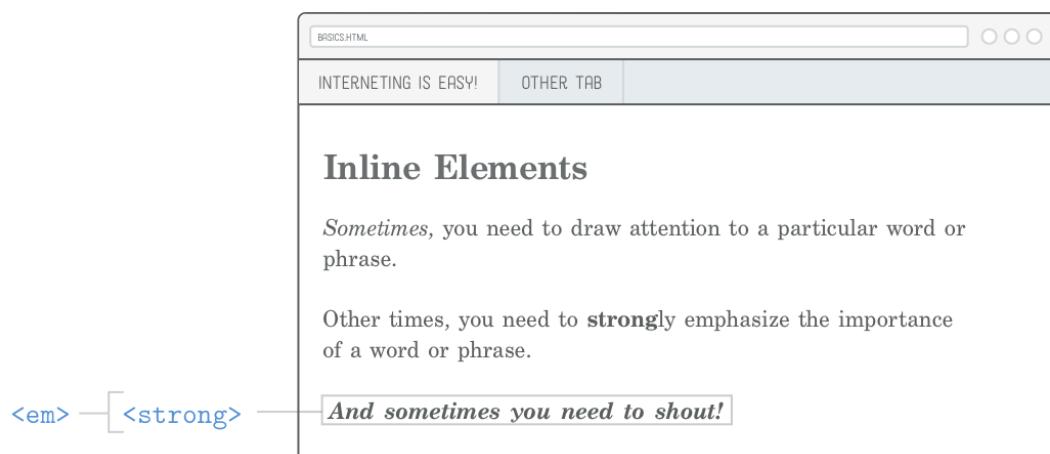
Deve ser renderizado em negrito, assim:



Para chamar ainda mais atenção para um trecho de texto, você pode aninhar um elemento `<strong>` em um elemento `<em>` (ou vice-versa). Isso lhe dará um texto forte e enfatizado:

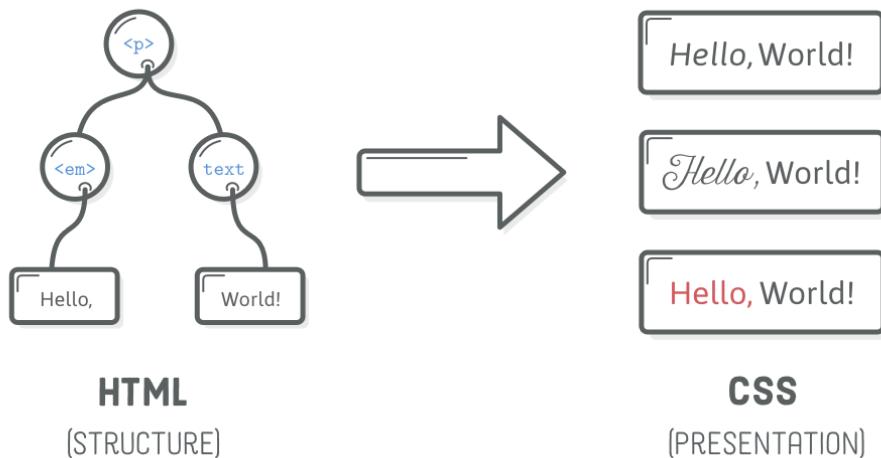
```
<p><em><strong>E às vezes você precisa gritar!</strong></em></p>
```

Como sugere o texto de exemplo, este é efetivamente o equivalente tipográfico de gritar. Talvez exista um capítulo sobre Tipografia da Web. Leia-o antes de enlouquecer com as fontes em negrito e itálico.



# Estrutura versus Apresentação

Você deve estar se perguntando por que usamos os termos “ênfase” e “forte” em vez de “itálico” e “negrito”. Isso nos leva a uma distinção importante entre HTML e CSS. A marcação HTML deve fornecer informações semânticas sobre o seu conteúdo – não informações de apresentação . Em outras palavras, o HTML deve definir a estrutura do seu documento, deixando sua aparência para o CSS.



Os elementos “pseudo-obsoletos” `<b>` e `<i>` são exemplos clássicos disso. Eles costumavam significar “negrito” e “itálico”, respectivamente, mas o HTML5 tentou criar uma separação clara entre a estrutura de um documento e sua apresentação. Assim, o elemento `<i>` foi substituído por `<em>`, uma vez que o texto enfatizado pode ser exibido de várias maneiras, exceto em itálico (por exemplo, em uma fonte diferente, em uma cor diferente ou em um tamanho maior). O mesmo vale para `<b>` e `<strong>`.

Como descobriremos em Olá, CSS, podemos alterar a renderização padrão dos elementos `<strong>` e do navegador `<em>`. Isso reforça o ponto de que não devemos chamá-lo como texto em itálico ou negrito no HTML - isso é algo que cabe ao CSS decidir.

# Elementos HTML vazios

As tags HTML que encontramos até agora envolvem conteúdo de texto (por exemplo, `<p>`) ou outros elementos HTML (por exemplo, `<ol>`). Esse não é o caso de todos os elementos HTML. Alguns deles podem estar “vazios” ou “fechados automaticamente”. Quebras de linha e linhas horizontais são os principais exemplos de elementos vazios.

## Quebras de linha

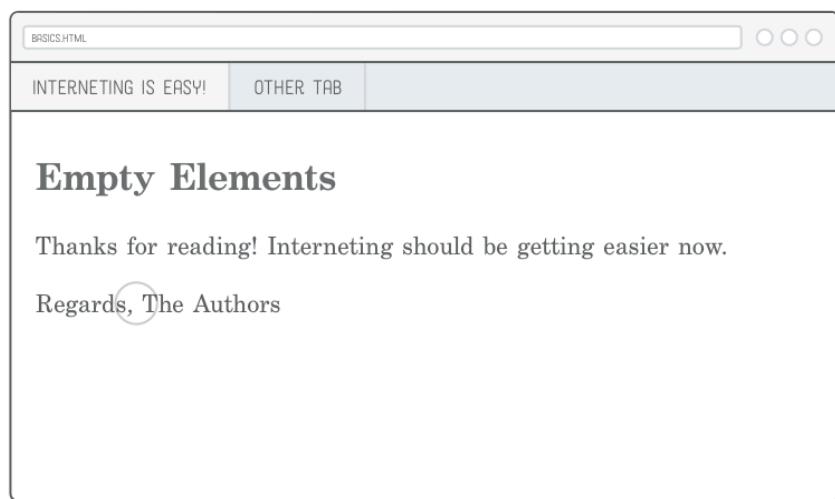
HTML condensa espaços consecutivos, tabulações ou novas linhas (conjuntamente conhecidos como “espaços em branco”) em um único espaço. Para ver do que estamos falando, adicione a seguinte seção ao nosso arquivo `index.html`:

```
<h2>Elementos vazios</h2>
```

```
<p>Obrigado por ler! A Internet deve estar ficando mais fácil  
agora.</p>
```

```
<p>Atenciosamente,  
Os Autores</p>
```

A nova linha após o trecho Atenciosamente acima, será transformada em um espaço em vez de ser exibida como uma quebra de linha:

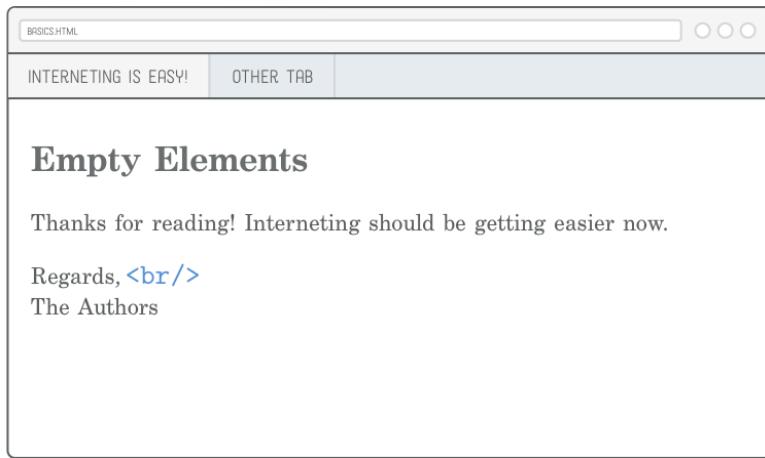


Esse comportamento pode parecer contra-intuitivo, mas os desenvolvedores web geralmente configuram seu editor de texto para limitar o comprimento da linha a cerca de 80 caracteres. Como programador, é mais fácil gerenciar o código dessa maneira, mas fazer com que cada uma das novas linhas aparecesse na página renderizada atrapalharia gravemente o layout de página pretendido.

Para informar ao navegador que queremos uma quebra de linha, precisamos usar um elemento `<br/>` que explica uma quebra de linha, como este:

```
<p>Atenciosamente, <br/> Os Autores</p>
```

O elemento `<br/>` é útil em qualquer lugar que a formatação de texto seja importante. Assinaturas e letras de músicas são apenas alguns exemplos em que podem ser úteis.



Porém, tome muito cuidado para não abusar da tag `<br/>`. Lembre-se que cada tag que você usar deve transmitir significado - você não deve usá-lo para, digamos, adicionar muito espaço entre os parágrafos:

```
←— (Você pode ser expulso de Valinor por isso) →  
<p>Este parágrafo precisa de algum espaço abaixo ...</p>  
<br/><br/><br/><br/><br/><br/><br/><br/>  
<p>Então, adicionei algumas quebras de linha.</p>
```

Conforme discutido na seção anterior, esse tipo de informação de apresentação deve ser definido em seu CSS em vez de em seu HTML.

## Linhas Horizontais

O elemento `<hr />` é uma “linha horizontal”, que representa uma **ruptura temática**. A transição de uma cena de uma história para a seguinte ou entre o final de uma carta e um pós-escrito (P.S) são bons exemplos de quando uma regra horizontal pode ser apropriada. Por exemplo:

```
<h2>Elementos vazios</h2>
<p>Obrigado por ler! A Internet deve estar ficando mais fácil
agora.</p>

<p>Atenciosamente,<br />
Os Autores</p>

<hr />

<p>P.S. Esta página pode parecer sem graça, meio pra baixo, mas
vamos consertar isso com algum CSS em breve.</p>
```

Um dos temas deste capítulo foi a separação entre conteúdo (HTML) e apresentação (CSS), e com `<hr />` não é diferente. Assim como `<em>` e `<strong>`, ele tem uma aparência padrão (uma linha horizontal), mas assim que começarmos a trabalhar com CSS, seremos capazes de personalizar esse elemento, adicionando, por exemplo, mais espaço entre seções, um caractere de destaque decorativo ou praticamente qualquer outra coisa que desejarmos.



Assim como o elemento `<br />`, o elemento `<hr />` deve ter significado – não o use quando quiser apenas exibir uma linha por uma questão de estética. Para isso, você vai querer usar a propriedade border CSS, que discutiremos em breve.

Outra maneira de pensar sobre o elemento `<hr />` é que ele carrega menos significado do que a separação criada por um novo elemento de título, mas mais significado do que um novo parágrafo.

## Barra Final Opcional

A barra final (/) em todos os elementos HTML vazios é totalmente opcional. O trecho de código acima também pode ser marcado assim (observe a falta da barra nas tags `<br>` e `<hr>`):

```
<p>Atenciosamente,<br>
Os Autores</p>
```

```
<hr>
```

Realmente não faz diferença qual convenção você escolhe, mas escolha uma e siga-a por uma questão de consistência. Neste tutorial, incluiremos a barra final porque mostra claramente que é um elemento de fechamento automático. Isso ajudará a evitar que seus olhos procurem a tag de fechamento em outras partes do documento.

# Resumo

Este capítulo pode ter parecido uma lista interminável de elementos HTML e, bem, basicamente era. HTML é bastante simples quando se trata disso. As páginas da Web são compostas de elementos HTML, cada elemento adiciona um significado diferente ao texto que contém e os elementos podem ser aninhados uns dentro dos outros.

O que fizemos neste capítulo é sempre o primeiro passo no processo de desenvolvimento web – você precisa definir o que quer dizer (HTML) antes de definir como isso será apresentado esteticamente (CSS). Esperamos que o arquivo `index.html` que criamos neste capítulo sirva como uma referência rápida e útil dos principais elementos HTML. Se acontecer de você perdê-lo, veja como deve ser:

The screenshot shows a browser window with the title bar 'INTERNETING IS EASY!' and a tab labeled 'OTHER TAB'. The main content area displays an HTML document with the following structure:

```

<title> INTERNETING IS EASY!
<h1> Interneting Is Easy!
<p> First, we need to learn some basic HTML.
<h2> Headings
      Headings define the outline of your site. There are six levels of headings.
<h3> Lists
      This is how you make an unordered list:
      <ul>
          <li> Add a "ul" element (it stands for unordered list)
          <li> Add each item in its own "li" element
          <li> They don't need to be in any particular order
      This is what an ordered list looks like:
      <ol>
          <li> Notice the new "ol" element wrapping everything
          <li> But, the list item elements are the same
          <li> Also note how the numbers increment on their own
          <li> You should be noticing things in this precise order, because this is an ordered list
<h3> Inline Elements
      <em> Sometimes, you need to draw attention to a particular word or phrase.
      <strong> Other times, you need to strongly emphasize the importance of a word or phrase.
      <em><strong> And sometimes you need to shout!
<h3> Empty Elements
      Thanks for reading! Interneting should be getting easier now.
      Regards, <br/>
      The Authors
      P.S. This page might look like crap, but we'll learn how to fix that with some CSS soon.
  
```

Conversamos sobre como escrever HTML e como manipular conteúdo em um editor de código. Desenvolver HTML é obviamente um processo muito artesanal e criativo. Você irá precisar conhecer as tags disponíveis para saber explorá-las quando precisar.

Embora o número grande de tags disponíveis possa confundir no início, isso traz uma incrível flexibilidade ao HTML: você pode usar HTML para exibi-lo em uma página web, em um dispositivo móvel, em um tablet ou mesmo em um pedaço de papel impresso, cada um com layouts diferentes. Você pode até remodelar vários documentos apenas alterando uma única linha de CSS. O Microsoft Word, por exemplo, não chega nem perto do potencial do HTML e CSS como meio de conteúdo.

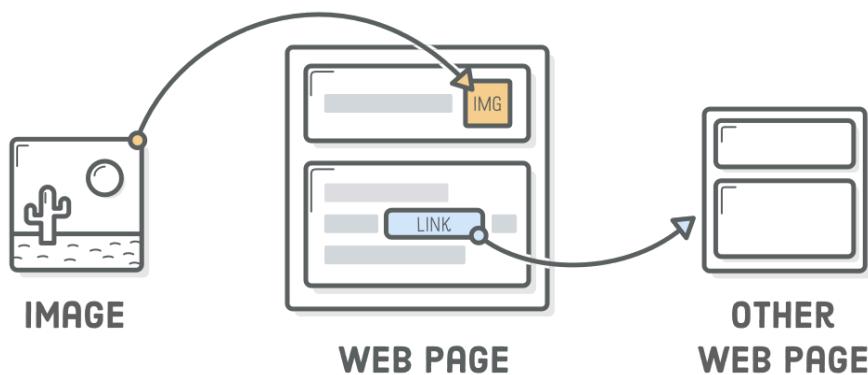
No próximo capítulo, aprofundaremos nossos conhecimentos sobre HTML com os demais elementos que você encontrará diariamente: links e imagens. Para os elementos mais obscuros, deixaremos que você explore [a referência de elemento HTML do MDN](#) por conta própria.

# Links e Imagens

## Capítulo 3

*Um tutorial HTML amigável sobre como conectar páginas da web entre si*

O [capítulo anterior](#) abordou alguns elementos HTML muito importantes, mas estávamos lidando apenas com uma única página da web. Links e imagens são fundamentalmente diferentes desses elementos porque lidam com recursos externos. Os links direcionam o usuário para um documento HTML diferente e as imagens puxam outro recurso para a página.



Para usar links e imagens, também precisaremos aprender sobre outro componente da sintaxe HTML: [atributos](#). Os atributos abrirão um mundo totalmente novo de possibilidades para nossas páginas da web.

Neste capítulo, criaremos um site simples composto por vários documentos HTML e arquivos de imagem. Este capítulo também tratará questões em torno da organização de arquivos e pastas. À medida que começarmos a trabalhar com vários arquivos, descobriremos a importância de ser um desenvolvedor web organizado.

# Configurações

Este capítulo é sobre como vincular páginas da web, então precisaremos criar alguns novos arquivos HTML antes de codificarmos qualquer coisa. Trabalharemos com três páginas da web separadas neste capítulo, junto com alguns arquivos de imagem de vários formatos:



LINKS.HTML



IMAGES.HTML



MISC/EXTRAS.HTML



MOCHI.JPG



MOCHI.GIF



MOCHI.PNG



MOCHI.SVG

Para começar, crie uma nova pasta para nosso projeto, chamada `links-e-imagens`. Essa pasta irá armazenar todos os nossos arquivos.

## Página de Links

Após ter criado a pasta, conforme orientado, adicione um novo arquivo a essa pasta chamada `links.html` e insira o seguinte modelo HTML.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Links</title>
  </head>
  <body>
    <h1>Links</h1>
  </body>
</html>
```

## Página de Imagens

Na mesma pasta, crie outro arquivo chamado `images.html`:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Images</title>
  </head>
  <body>
    <h1>Images</h1>
  </body>
</html>
```

## Páginas Extras

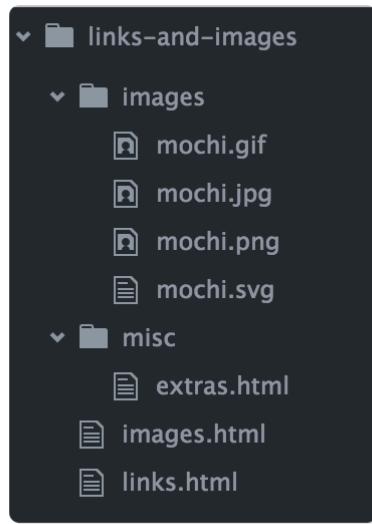
Nossa última página nos ajudará a demonstrar links relativos. Crie uma nova pasta dentro de `links-e-imagens` chamada de `misc` e adicione um novo arquivo chamado `extras.html`:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Extras</title>
  </head>
  <body>
    <h1>Extras</h1>
  </body>
</html>
```

Observe que você pode criar uma nova pasta no VSCode clicando com o botão direito no painel do navegador de arquivos (explorador) e selecionando Nova pasta... no menu contextual. A vida é melhor quando se usa um bom editor de código!

## Download de Imagens

Incluiremos imagens em nosso arquivo `imagens.html`, portanto, baixe também esses [exemplos de imagens mochi](#). Descompacte-os em sua pasta `links-e-imagens`, mantendo a pasta `imagens` do arquivo ZIP. Seu projeto agora deve ficar assim:



## Âncoras

Os links são criados com o elemento `<a>`, que significa “âncora”. Funciona exatamente como todos os elementos do capítulo anterior: quando você envolve algum texto entre tags `<a>`, o significado desse conteúdo é alterado. Vamos dar uma olhada adicionando o seguinte parágrafo ao elemento `<body>` do arquivo `links.html`:

```
<p>Este exemplo é sobre links e <a>imagens</a>.</p>
```

Se você carregar a página em um navegador da web, notará que o elemento `<a>` não se parece em nada com um link. Sim, infelizmente, o elemento `<a>` por si só não faz muita coisa.

# Ligações

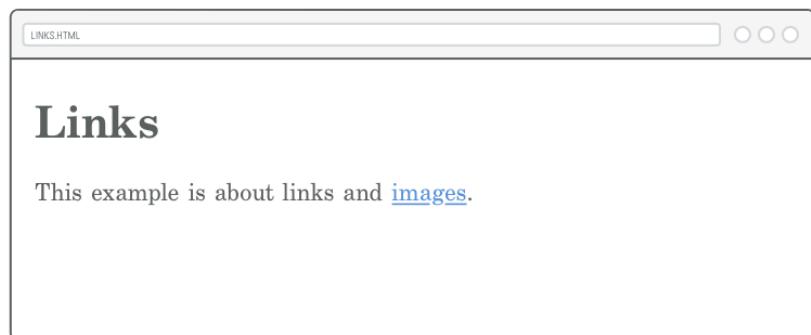
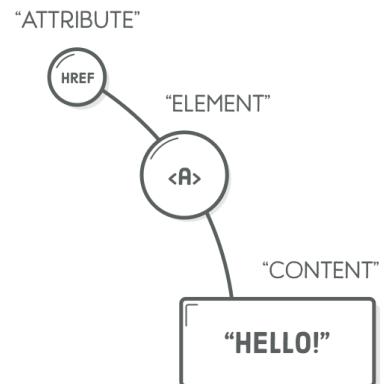
Da mesma forma que um elemento acrescenta significado ao conteúdo que contém, um “**atributo**” HTML **adiciona significado ao elemento** ao qual está atribuído.

Elementos diferentes permitem atributos diferentes e você pode consultar o [MDN](#) para obter detalhes sobre quais elementos aceitam quais atributos. No momento, estamos preocupados com o atributo href (http reference) porque ele determina para onde o usuário será redirecionado ao clicar em um elemento <a>. Atualize seu link para corresponder ao seguinte:

```
<p>Este exemplo é sobre links e <a href='images.html'>imagens</a>.</p>
```

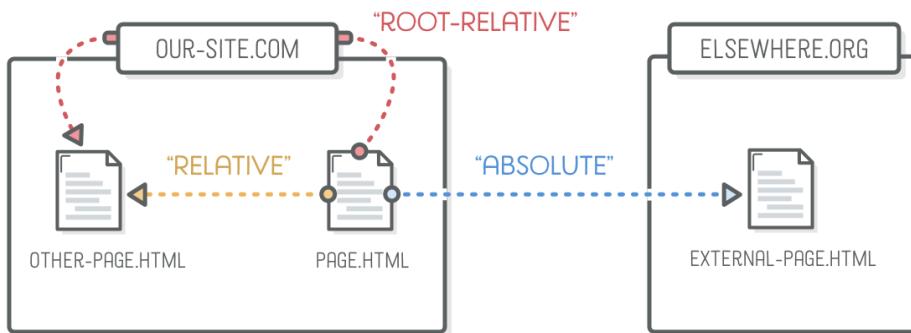
Observe como os atributos ficam dentro da tag de abertura . O nome do atributo vem primeiro, depois um sinal de igual e, em seguida, o “valor” do atributo entre aspas simples ou duplas. Esta sintaxe distingue atributos de conteúdo (que fica entre as tags).

A informação extra fornecida pelo atributo href informa ao navegador que este elemento <a> é na verdade um link e deve renderizar o conteúdo em seu texto azul padrão:



# Links absolutos, relativos e relativos à raiz

Agora que estamos trabalhando com links, precisamos entender como um site é estruturado. Por enquanto, podemos compreender um site como sendo apenas uma coleção de arquivos HTML organizados em pastas. Para se referir a esses arquivos dentro de outro arquivo, a Internet usa “localizadores uniformes de recursos” (URLs). Dependendo do que você está se referindo, os URLs podem assumir diferentes formatos. Os três tipos de URLs mais utilizados são i) links absolutos, ii) links relativos; e iii) relativos à raiz:



Links absolutos, relativos e relativos à raiz referem-se ao valor do atributo `href`. As próximas seções explicam como e quando usar cada um deles. Mas primeiro, vamos adicionar o seguinte conteúdo ao nosso arquivo `links.html`:

```
<p>Esta página é sobre links! Existem três tipos de links:</p>
```

```
<ul>
  <!— Adicione elementos <li> aqui →
</ul>
```

## Links Absolutos

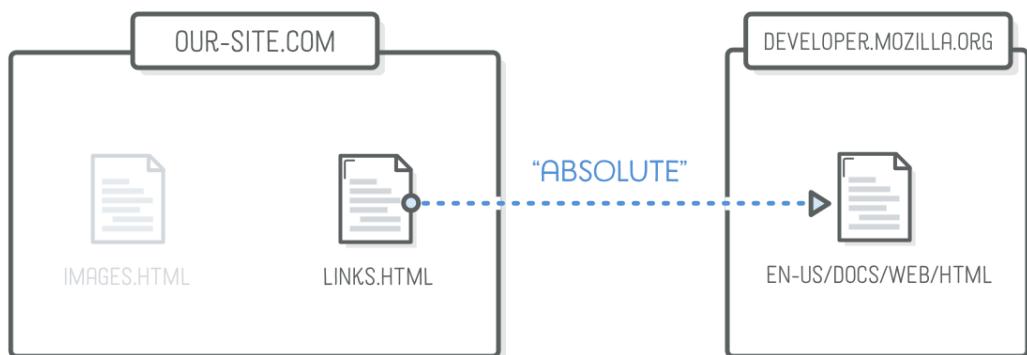
Links “absolutos” são a forma mais detalhada de se referir a um recurso da web. Eles começam com o “esquema” (normalmente http://ou https://), seguido pelo nome de domínio do site e, em seguida, pelo caminho da página da web de destino.



Por exemplo, tente criar um link para a referência do elemento HTML da Mozilla Developer Network:

```
<li>Links absolutos: 
```

É possível usar links absolutos para se referir a páginas do seu próprio site, mas codificar seu nome de domínio em qualquer lugar pode criar algumas situações complicadas. Geralmente é melhor reservar links absolutos apenas para direcionar os usuários a um site diferente.



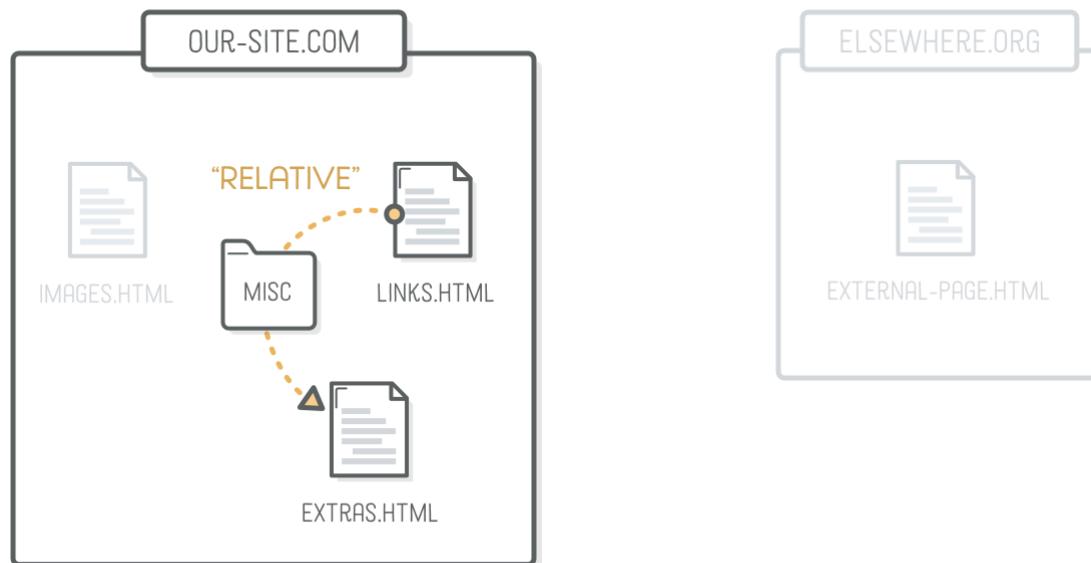
## Links Relativos

Links “relativos” apontam para outro arquivo em seu site do ponto de vista do arquivo que você está editando. Em links relativos não informamos o esquema nem tampouco

o nome do domínio: está implícito que o esquema e o nome de domínio são iguais aos da página atual, portanto, a única coisa que você precisa fornecer é o caminho. Veja como podemos vincular ao nosso arquivo `extras.html` de dentro de `links.html`:

```
<li>Links relativos, como nossos <a href='misc/extras.html'>extras  
página</a>.</li>
```

Nesse caso, o atributo `href` representa o caminho do arquivo para `extras.html` desde o arquivo `links.html`. Como `extras.html` não está na mesma pasta que `links.html`, precisamos incluir a pasta `misc` no URL.



Cada pasta e arquivo em um caminho é separado por uma barra (/). Portanto, se estivéssemos tentando acessar um arquivo com duas pastas, precisaríamos de uma URL como esta:

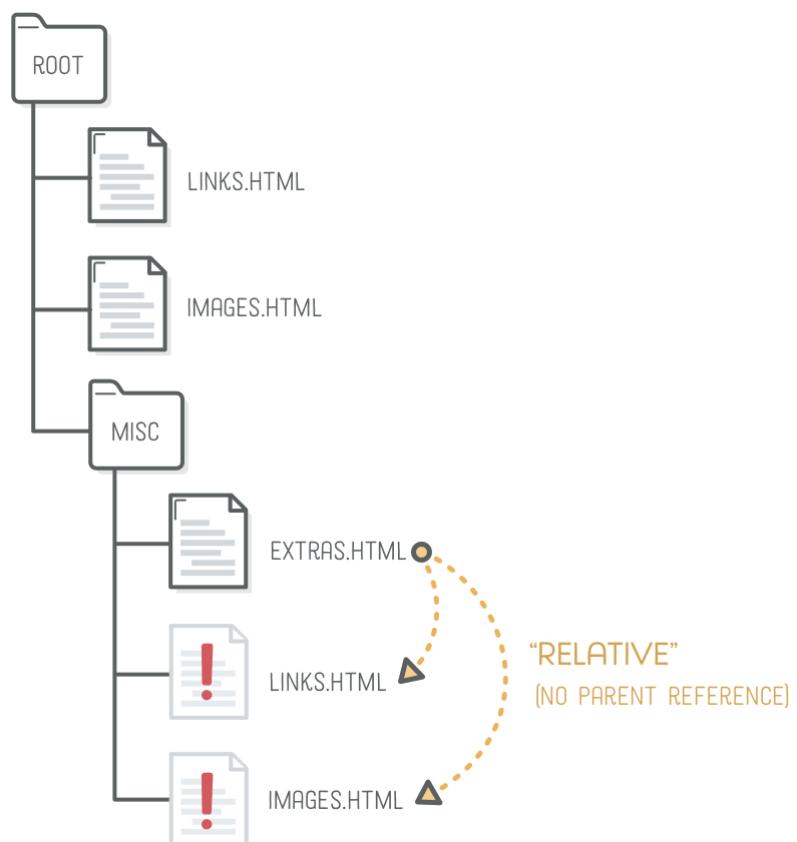
```
misc/outra-pasta/extras.html
```

## Pastas Pai

Isso funciona para se referir a arquivos que estão na mesma pasta ou em uma pasta mais profunda. Que tal vincular páginas que estão em um diretório acima do arquivo atual? Vamos tentar criar links relativos de `links.html` e `imagens.html` para nosso arquivo `extras.html`. Adicione isto a `extras.html`:

```
<p>Aqui estão descritos diversos assuntos HTML. Talvez você esteja interessado em <a href='links.html'>links</a> ou <a href='images.html'>imagens</a>.</p>
```

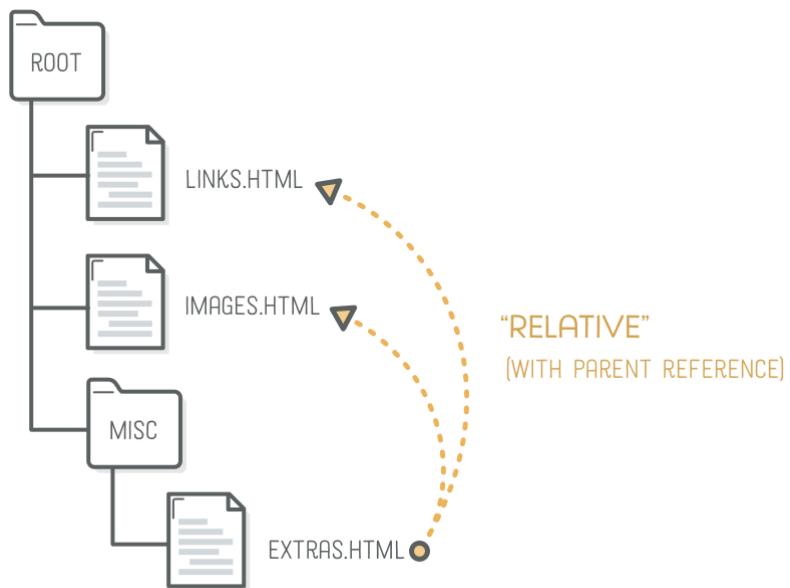
Quando você clica em qualquer um desses links em um navegador da web, ele reclama que a página não existe. Examinando a barra de endereço (**faça esse exercício**), você descobrirá que o navegador está tentando carregar `misc/links.html` e `misc/images.html` está procurando na pasta errada! Isso porque nossos links são relativos à localização de `extras.html`, que fica na pasta `misc`.



Para corrigir isso, precisamos da sintaxe de dois pontos (..). Dois pontos consecutivos em um caminho de arquivo representam um ponteiro para o diretório pai:

```
<p>Aqui estão descritos diversos assuntos HTML. Talvez você esteja interessado em <a href='..;/links.html'>links</a> ou <a href='..;/images.html'>imagens</a></p>
```

É como dizer: “Eu sei que extras.html está na pasta misc. Vá até uma pasta e procure links.html ou imagens.html por lá.”



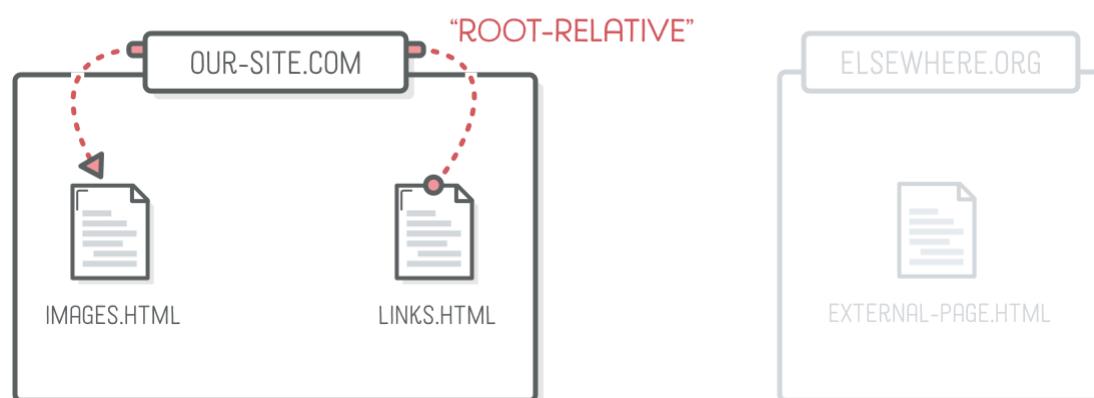
Para navegar por vários diretórios, use várias vezes o elemento .., como por exemplo:

```
..;/..;/elsewhere.html
```

Links relativos são bons porque permitem que você move pastas inteiras sem ter que atualizar todas as propriedades `<href>` dos elementos `<a>`. Entretanto, tome cuidado, pode ficar um pouco confuso quando todos os seus links começam com um monte de pontos. Eles são melhores para se referir a recursos na mesma pasta ou em uma seção independente do seu site.

## Links Relativos à Raiz

Os links “relativos à raiz” são semelhantes à seção anterior, mas em vez de serem relativos à página atual, são relativos à “raiz” de todo o site. Por exemplo, se o seu site estiver hospedado em `nosso-site.com`, todos os URLs relativos à raiz serão relativos a `nosso-site.com`.



Infelizmente, há uma ressalva em nossa discussão sobre links relativos à raiz: todo este guia usa arquivos HTML hospedados em nossa própria máquina em vez de um site hospedado em um servidor web. Isso significa que não poderemos experimentar links relativos à raiz. Mas, se tivéssemos um servidor real, o link para nossa página inicial ficaria assim:

```
←— Isso não funcionará para nossos arquivos HTML locais —→  
<li>Links relativos à raiz, como a <a href='/'>página inicial</a> do  
nossa site, mas eles não são úteis para nós no momento.</li>
```

A única diferença entre um link relativo à raiz e um link relativo é que o primeiro começa com uma barra. Essa barra inicial representa a raiz do seu site. Você pode adicionar mais pastas e arquivos ao caminho após a barra inicial, assim como links relativos. O caminho a seguir funcionará corretamente, não importa onde a página atual esteja localizada (mesmo em `misc/extras.html`):

`/images.html`

Links relativos à raiz são alguns dos tipos de links mais úteis. Eles são explícitos o suficiente para evitar a confusão potencial de links relativos, mas não são excessivamente explícitos como links absolutos. Você verá muitos deles ao longo de sua carreira de desenvolvimento web, especialmente em sites maiores, onde é difícil acompanhar as referências relativas.

## Alvos de links

Os atributos alteram o significado dos elementos HTML e, às vezes, é necessário modificar mais de um aspecto de um elemento. Por exemplo, os elementos `<a>` também aceitam um atributo `target` que define onde exibir a página quando o usuário clica no link. Por padrão, a maioria dos navegadores substitui a página atual pela nova. Podemos usar o atributo `target` para pedir ao navegador para abrir um link em uma nova janela/aba.

Tente alterar nosso link absoluto `links.html` para corresponder ao código a seguir: observe como o segundo atributo se parece com o primeiro, mas estão separados um do outro por um espaço (ou uma nova linha):

```
<li>Links absolutos, gostaria de acessar <a  
href='https://developer.mozilla.org/en-US/docs/Web/HTML'  
target='_blank'>Mozilla Developer Network</a>, que é uma rede muito  
boa recurso para desenvolvedores web.</li>
```

O atributo `target` possui alguns valores predefinidos que têm um significado especial para navegadores da web, mas o mais comum é `_blank`, que especifica uma nova guia ou janela. Você pode ler sobre o resto no [MDN](#).

# Convenções de nomenclatura

Você notará que nenhum de nossos arquivos ou pastas possui espaços em seus nomes. Isso é de propósito. Espaços em URLs requerem tratamento especial e devem ser evitados a todo custo. Para ver do que estamos falando, tente criar um novo arquivo em nosso projeto `links-e-imagens` chamado `espaços são ruins.html`. Adicione um pouco de texto e abra-o no Google Chrome ou Safari (o Firefox trapaceia e preserva os espaços). Observe a URL gerada:

```
links-e-imagens/espaços%20são%20ruins.html
```

Na barra de endereço, você verá que todos os nossos espaços foram substituídos por `%20`, conforme mostrado acima. **Espaços não são permitidos em URLs e essa é a codificação especial usada para representá-los. Em vez de espaço, você deve sempre usar um hífen**, como vimos fazendo ao longo deste tutorial. Também é uma boa ideia usar todos os caracteres minúsculos para manter a consistência e o padrão.

Observe como há uma conexão direta entre nossos nomes de arquivos e pastas e o URL da página da web que eles representam. Os nomes de nossas pastas e arquivos determinam as rotas (às vezes chamados de slugs) de nossas páginas da web. Eles, muitas vezes, serão visíveis para o usuário, o que significa que você deve se esforçar tanto para nomear seus arquivos quanto para criar o conteúdo que eles contêm.

Essas convenções de nomenclatura se aplicam a todos os arquivos do seu site, não apenas aos arquivos HTML. Arquivos CSS, arquivos JavaScript e imagens devem evitar espaços e também ter letras maiúsculas consistentes.

## Imagens

Ao contrário de todos os elementos HTML que encontramos até agora, o conteúdo da imagem é definido fora da página web que a renderiza. Felizmente para nós, já temos

uma maneira de nos referirmos a recursos externos de dentro de um documento HTML: URLs absolutos, relativos e relativos à raiz.

Imagens são incluídas em páginas da web com a tag `<img />` adicionando o atributo `src`, que aponta para o arquivo de imagem que você deseja exibir.

```
<img src='some-photo.jpg' />
```

Telas de alta resolução e dispositivos móveis tornam o manuseio de imagens um pouco mais complicado do que simplesmente usar a tag `<img />`. Deixaremos essas complexidades para o futuro, quando tratarmos de Imagens Responsivas. Certifique-se também de verificar o elemento `<figure>` e `<figcaption>` no capítulo HTML semântico. Por enquanto, vamos nos concentrar nos vários formatos de imagem que, normalmente, circulam pela Internet.

## Formatos de imagem

Existem quatro ou cinco formatos principais de imagem em uso na web e todos foram projetados para fazer coisas diferentes. Compreender a finalidade pretendida ajuda muito a melhorar a qualidade de suas páginas da web.



JPG



GIF



PNG



SVG

As próximas seções apresentarão casos de uso para cada um desses formatos de imagem. Certifique-se de descompactar essas imagens mochi em seu projeto links-e-imagens antes de prosseguir.

## Imagens JPG

As imagens JPG são projetadas para lidar com grandes paletas de cores sem aumentar exorbitantemente o tamanho do arquivo. Isso os torna ótimos para fotos e imagens da web. Por outro lado, JPGs não permitem pixels transparentes, que você pode ver nas bordas brancas da imagem abaixo se olhar bem de perto:



Incorpore a imagem `mochi.jpg` em nossa página `imagens.html` com o seguinte trecho de código (isso também inclui um pouco de navegação para nossas outras páginas):

```
<p>Esta página abrange formatos de imagem comuns, mas você também pode estar procurando por <a href='links.html'>links</a> e <a href='misc/extras.html'>úteis extras</a>.</p>
```

## <h2>JPG</h2>

```
<p>Imagens JPG são boas para fotos e imagens na web.</p>
<img src='images/mochi.jpg' />
```

## Imagens GIF

GIFs são a opção ideal para animações simples, mas a desvantagem é que eles são um tanto limitados em termos de paleta de cores – nunca os use para fotos.

Pixels transparentes são uma opção binária para GIFs, o que significa que você não pode ter pixels semi-opacos. Isso pode dificultar a obtenção de altos níveis de detalhes em um fundo transparente.

Por esse motivo, geralmente é melhor usar imagens PNG se você não precisar de animação.



Você pode adicionar esse carinha ao arquivo `imagens.html` com o seguinte trecho de código:

```
<h2>GIFs</h2>

<p>GIFs são bons para animações.</p>
<img src='images/mochi.gif' />
```

## Imagens PNG

PNGs são ótimos para qualquer coisa que não seja uma foto ou animação. Para fotos, um arquivo PNG da mesma qualidade (conforme percebido pelo olho humano) geralmente seria maior que um arquivo JPG equivalente. No entanto, eles lidam perfeitamente com transparências e não têm limitações de paleta de cores. Isso os torna ideais para ícones, diagramas técnicos, logotipos, etc.



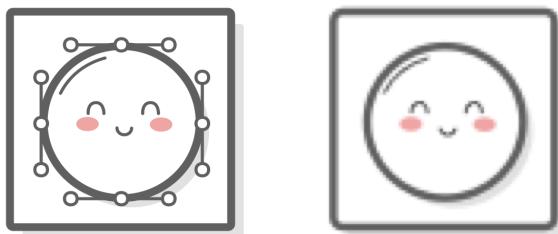
Vamos adicionar esta imagem PNG ao nosso projeto de exemplo também:

```
<h2>PNGs</h2>

<p>PNGs são bons para transparências e ícones.</p>
<img src='images/mochi.png' />
```

## Imagens SVG

Ao contrário dos formatos de imagem baseados em pixels acima, SVG é um formato gráfico baseado em vetor, o que significa que pode ser ampliado ou reduzido para qualquer dimensão sem perda de qualidade. Essa propriedade torna as imagens SVG uma ferramenta maravilhosa para design responsivo. Eles são bons para praticamente todos os mesmos casos de uso dos PNGs, e você deve usá-los sempre que puder.



Essas imagens de 300x300 pixels tinham originalmente 100x100 pixels, mas aumentá-las mostra claramente a diferença entre elas. Observe as linhas nítidas e limpas na imagem SVG esquerda, enquanto a imagem PNG à direita agora está muito pixelizada.

Apesar de ser um formato vetorial, os SVGs podem ser usados exatamente como seus equivalentes não vetoriais. Vá em frente e adicione `mochi.svg` à nossa página `images.html`:

## <h2>SVGs</h2>

```
<p>SVGs são <em>da hora</em>! Use onde quiser.</p>
<img src='images/mochi.svg' />
```

Há um problema potencial com SVGs: para que eles sejam exibidos de forma consistente em todos os navegadores, você precisa converter quaisquer campos de texto em contornos usando seu editor de imagens (por exemplo, o Adobe Illustrator). Se suas imagens contiverem muito texto, isso poderá ter um grande impacto no tamanho do arquivo. Nesses casos, é melhor usar os formatos PNGs ou o formato Webp em vez de SVGs, embora os SVGs sejam incríveis.

## Imagens Webp

O tipo de imagens Webp desenvolvido pelo Google, tem como objetivo de diminuir o tamanho dos arquivos, garantindo uma transferência mais rápida e com qualidade. Une o que há de melhor em outros formatos de imagem, como a possibilidade de compressão do arquivo (semelhante ao JPEG), a capacidade de usar transparência (semelhante ao PNG), além do suporte a animações (do mesmo modo que o GIF). De acordo com o Google, imagens e gráficos em formato Webp são aproximadamente 30% menores do que arquivos PNG ou JPEG com a mesma qualidade de imagem.



A criação do formato Webp não é uma coisa nova, foi em 2010. Contudo, a adoção pela maioria dos navegadores demorou um pouco mais. Ainda hoje, alguns navegadores suportam o formato Webp apenas parcialmente, como é o caso do navegador Safari. Vamos adicionar uma imagem Webp a nossa página `images.html`:

```
<h2>Webp</h2>
```

```
<p>Webp são o futuro! E o futuro é logo ali!</p>
<img src='images/mochi.webp' />
```

# Dimensões da imagem

Por padrão, o elemento `<img/>` usa as dimensões herdadas de seu arquivo de imagem. Nossas imagens JPG, GIF, PNG e WEBP têm, na verdade, 150×150 pixels, enquanto nosso mochi SVG tem apenas 75×75 pixels.

This page covers common image formats, but you may also be looking for [links](#) and [useful extras](#).

## JPGs

JPG images are good for photos.



## GIFs

GIFs are good for animations.



## PNGs

PNGs are good for diagrams and icons.



## SVGs

SVGs are *amazing*. Use them wherever you can.



Como discutiremos mais adiante em Imagens responsivas, os formatos de imagem baseados em pixels precisam ser pelo menos duas vezes maiores do que você deseja que apareçam em telas de alta resolução. Para reduzir nossas imagens baseadas em pixels ao tamanho pretendido (75x75), podemos usar o atributo `width` do elemento `<img/>`. Em `imagens.html`, atualize todas as nossas imagens baseadas em pixels para corresponder ao seguinte:

```
← In JPGs section →  
<img src='images/mochi.jpg' width='75' />  
  
← In GIFs section →  
<img src='images/mochi.gif' width='75' />  
  
← In PNGs section →  
<img src='images/mochi.png' width='75' />
```

O atributo `width` define uma dimensão (largura) explícita para a imagem. Também existe um atributo correspondente para a altura, o atributo `height`. Definir apenas um deles fará com que a imagem seja dimensionada proporcionalmente, enquanto definir ambos irá esticar a imagem (às vezes até distorcer). Os valores das dimensões são especificados em pixels e você não precisa incluir uma unidade (por exemplo, `width='75px'` seria incorreto).

Os atributos `width` e `height` podem ser úteis, mas geralmente é melhor definir as dimensões da imagem com CSS para que você possa alterá-las com consultas de mídia. Discutiremos isso com mais detalhes quando chegarmos ao design responsivo .

## Alternativas de texto

Adicionar atributos `alt` aos seus elementos `<img/>` é uma prática recomendada. Definir um “texto alternativo” à imagem exibida tem impacto tanto nos motores de busca como nos utilizadores com navegadores apenas de texto (por exemplo, pessoas que utilizam software de conversão de texto em voz devido a uma deficiência visual).

Atualize todas as nossas imagens para incluir atributos alt descritivos:

```
←— In JPGs section →  
<img src='images/mochi.jpg' width='75' alt='Um mochi em uma bolha'↗  
  
←— In GIFs section →  
<img src='images/mochi.gif' width='75' alt='Um mochi dançando'↗  
  
←— In PNGs section →  
<img src='images/mochi.png' width='75' alt='Um mochi'↗  
  
←— In SVGs section →  
<img src='images/mochi.svg' alt='Um mochi com curvas de Bézier'↗  
  
←— In Webp section →  
<img src='images/mochi.webp alt='Mais um mochi'↗
```

Para obter mais exemplos de como usar o atributo alt, consulte a [documentação oficial](#) ridiculamente detalhada.

## Mais atributos HTML

Agora que estamos (espero) mais do que confortáveis com a sintaxe do atributo HTML, podemos adicionar alguns retoques finais ao nosso código padrão HTML. Cada página da web que você cria deve definir o idioma em que está escrita e seu conjunto de caracteres.

### Idioma do Documento

O idioma padrão de uma página web é definido pelo atributo lang no elemento `<html>`. Nosso documento está em português, então usaremos `pt-br`, que é o código do país, como valor do atributo (faça isso para todas as páginas que criamos):

```
<html lang='pt-br'>
```

Fazer esse ajuste, permite ao navegador melhor servir o conteúdo, sem sugestões de tradução entre outras coisas. Se não tiver certeza de qual é o código do país do seu idioma, você pode procurá-lo [aqui](#) no campo Subtag .

## Conjuntos de Caracteres

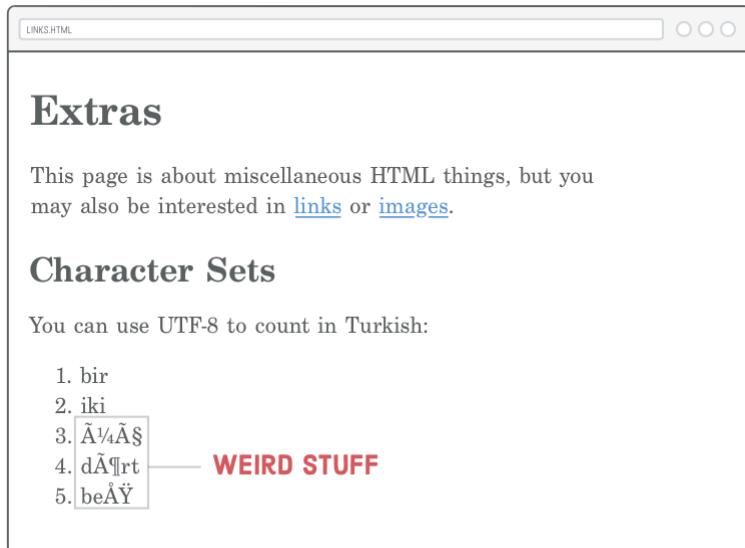
Um “conjunto de caracteres” é como um alfabeto digital para o seu navegador. É diferente do idioma do seu documento porque afeta apenas a forma como as letras são renderizadas, e não o idioma do conteúdo. Vamos copiar e colar alguns caracteres internacionais em nossa página `misc/extras.html` e ver o que acontece.

```
<h2>Conjuntos de caracteres</h2>
```

```
<p>Você pode usar UTF-8 para contar em turco:</p>
```

```
<ol>
  <li>bir</li>
  <li>iki</li>
  <li>üç</li>
  <li>dormir</li>
  <li>beş</li>
</ol>
```

Ao visualizar isso em um navegador, você verá algumas coisas estranhas onde os caracteres ü, , e deveriam estar:çöş



Isso ocorre porque o conjunto de caracteres padrão da maioria dos navegadores não acomoda esses caracteres “especiais”. Para corrigir isso, especifique uma codificação de caracteres UTF-8 adicionando um elemento `<meta>` com um atributo `charset` ao `<head>` do nosso arquivo `misc/extras.html`:

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

Os caracteres especiais agora devem ser renderizados corretamente. Hoje em dia, o UTF-8 é como um alfabeto universal para a Internet. Cada página da web que você criar deve ter esta linha no `<head>` de seu arquivo.

## Entidades HTML

Esta última seção na verdade não tem nada a ver com links ou imagens, mas precisamos discutir mais uma coisa antes de mudar para CSS: “entidades HTML”. Entidades HTML são caracteres especiais que não podem ser representados como texto simples em um documento HTML. Isso normalmente significa que é um caractere reservado em HTML ou que você não possui uma tecla para isso no teclado.

## Caracteres reservados

Os caracteres <, > e & são chamados de “caracteres reservados” porque não podem ser inseridos em um documento HTML sem serem interpretados como códigos. Isso ocorre porque eles significam algo na sintaxe HTML: < inicia uma nova tag, > termina uma tag e, como iremos aprender, & desencadeia uma entidade HTML. Em misc/extras.html, adicione o seguinte:

<h2>Entidades HTML</h2>

<p>Existem três caracteres reservados em HTML: &lt;; &gt;; e &amp;. Você deve sempre usar entidades HTML para esses três caracteres.</p>

As entidades sempre começam com um E comercial (&) e terminam com um ponto e vírgula (;). No meio, você coloca um código especial que seu navegador interpreta como um símbolo. Nesse caso, ele interpreta lt, gt e amp como símbolos de “menor que”, “maior que” e “e comercial”, respectivamente. Há uma tonelada de entidades HTML. Deixaremos que você [explore a maioria deles](#) por conta própria.

## Citações

Aspas curvas serão algumas das entidades HTML mais comuns que você usará. Existem quatro tipos diferentes de aspas curvas (abertura e fechamento de aspas simples e duplas):

- &ldquo;;
- &rdquo;;
- &lsquo;;
- &rsquo;;

Você pode usá-los no lugar de 'aspas "retas, assim:

<p>Se você&ecirc; gosta de &ldquo;tipografia da web&rdquo;; você também encontrará você mesmo usa aspas curvas bastante.</p>

Ao contrário das aspas retas, essas entidades de aspas curvas devem abraçar o texto.

## Entidades UTF-8 e HTML

Nos velhos tempos da web, os arquivos HTML não podiam conter caracteres especiais, tornando as entidades muito mais úteis. Mas, como agora estamos usando um conjunto de caracteres UTF-8, podemos inserir qualquer caractere diretamente no documento HTML. Isso torna as entidades úteis principalmente como caracteres reservados ou por conveniência ao criar HTML direto.

# Resumo

Um site é basicamente um monte de arquivos HTML, imagens e (como aprenderemos em breve) arquivos CSS vinculados entre si. Você deve começar a pensar em um site como uma bela maneira para os usuários navegarem nas pastas e arquivos que criamos como parte do processo de desenvolvimento web. Com essa perspectiva, deve ficar claro que manter um sistema de arquivos bem organizado é um aspecto crítico da criação de um site.

Também aprendemos sobre alguns atributos importantes (`lang` e `charset`) que nos fornecem o modelo básico que você deve usar como início de cada página da web que você criar:

```
<!DOCTYPE html>
<html lang='pt-br'>
<head>
  <meta charset='UTF-8' />
  <title>Alguma página da web</title>
</head>
```

```
<body>
  <h1>Alguma página da web</h1>
  <!-- Resto do conteúdo da página -->
</body>
</html>
```

Para nossa página ficar completa e alinhada ao que se usa no mercado atualmente, ainda falta uma componente muito importante: CSS. No próximo capítulo, descobriremos mais elementos e atributos HTML que nos permitirão anexar estilos CSS a todo o nosso site. A capacidade de trabalhar com vários arquivos e vinculá-los entre si de maneira inteligente se tornará ainda mais importante do que foi neste capítulo.

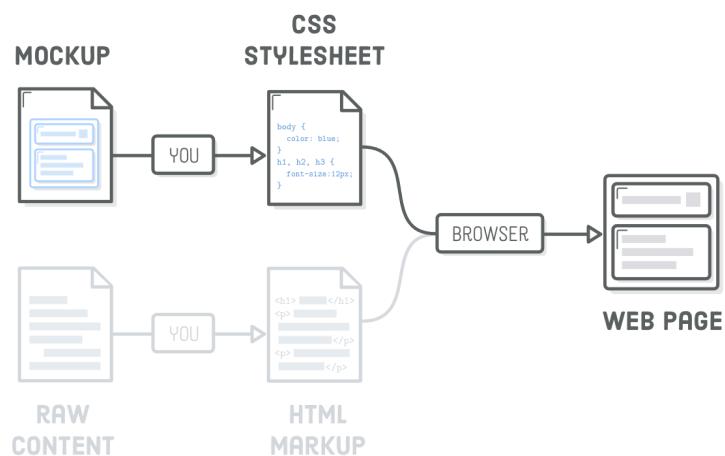
# Olá, CSS

## Capítulo 4

*Um tutorial amigável para criar sites um pouco mais bonitos*

Os primeiros capítulos deste guia focaram exclusivamente em HTML. Agora é hora de deixar as coisas bonitas (mais ou menos) com **Cascading Style Sheets (CSS)**. Você pode pensar em CSS como uma definição do “design” de uma página da web. Ele determina coisas como tamanho da fonte, margens e cores usando uma linguagem totalmente separada do HTML.

Por que é uma linguagem separada? Bem, isso serve a um propósito completamente diferente. HTML representa o conteúdo da sua página web, enquanto CSS define como esse conteúdo é apresentado ao usuário. Esta é uma distinção fundamental central para o desenvolvimento web moderno.



CSS fornece o vocabulário para dizer a um navegador da web coisas como: “Quero que meus títulos sejam bem grandes e que minha barra lateral apareça à esquerda do artigo principal”. O HTML não tem a terminologia necessária para tomar esse tipo de decisão de layout – tudo o que ele pode dizer é: “isso é um título e isso é uma barra lateral”.

Neste capítulo, exploraremos a sintaxe básica do CSS e também como conectá-lo aos nossos documentos HTML. O objetivo não é tanto se tornar um especialista em CSS ou memorizar todos os estilos disponíveis, mas sim entender como CSS e HTML interagem. CSS normalmente reside em seu próprio arquivo, portanto, como no capítulo anterior, uma boa organização de arquivos será fundamental.

## Configurações

Para simplificar, armazenaremos o exemplo de cada capítulo deste tutorial em uma pasta separada. Crie um novo projeto chamado `hello-css`. Iremos estilizar uma página existente chamada `hello-css.html`, então vá em frente e crie-a e adicione a seguinte marcação:

```
<!DOCTYPE html>
<html lang='pt'>
<head>
  <meta charset='UTF-8' />
  <title>Olá, CSS</title>
</head>
<body>
  <h1>Olá, CSS</h1>
  <p>CSS nos permite estilizar elementos HTML.</p>
  <p>Há também <a href='ficticio.html'>outra página</a> associada a
este exemplo.</p>

  <h2>Estilos de lista</h2>
  <p>Você pode estilizar listas não ordenadas:</p>
  <ul>
    <li>disco</li>
    <li>círculo</li>
    <li>quadrado</li>
  </ul>
```

```

<p>E você pode numerar listas ordenadas com o seguinte:</p>

<ol>
    <li>decimais</li>
    <li>romano maiúsculos</li>
    <li>romano minúsculos</li>
    <li>alfa maiúsculos</li>
    <li>alfa minúsculos</li>
    <li>(e muito mais!)</li>
</ol>
</body>
</html>

```

Além disso, precisaremos de uma pequena página fictícia para aprender como os estilos CSS podem ser aplicados a várias páginas da web. Crie um arquivo chamado fictício.html e adicione o seguinte trecho de código:

```

<!DOCTYPE html>
<html lang='pt-br'>
<head>
    <meta charset='UTF-8' />
    <title>Página Fictícia</title>
</head>
<body>
    <h1>Página Fictícia</h1>

    <p>Esta é uma página fictícia que nos ajuda a demonstrar folhas de estilo CSS reutilizáveis.<a href='hello-css.html'>Voltar</a>.</p>

    <p>Quer tentar arriscar um <a href=lugar-nenhum.html'>link duvidoso</a>? Essa é sua chance!</p>

</body>
</html>

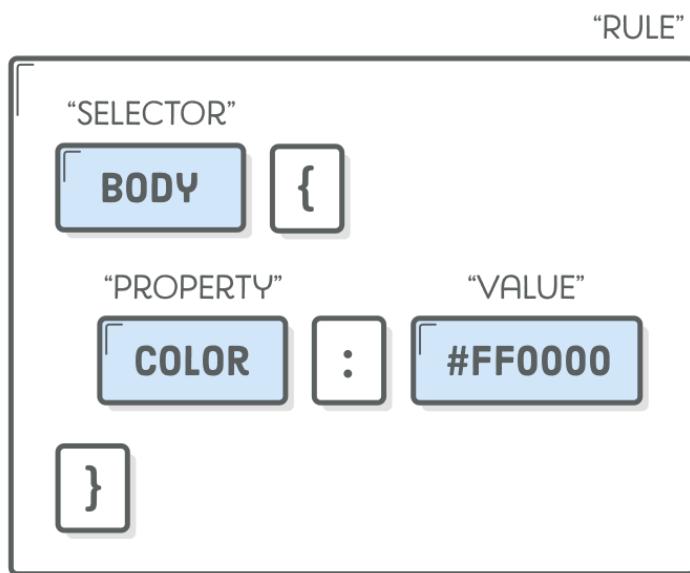
```

# Folhas de estilo CSS

As folhas de estilo CSS residem em arquivos de texto simples com a extensão `.css`. Crie um novo arquivo chamado `styles.css` dentro de nossa pasta `hello-css`. Vamos adicionar uma regra CSS para que possamos saber se nossa folha de estilo está conectada corretamente às nossas páginas HTML.

```
body {  
    color: #FF0000;  
}
```

Uma “regra” CSS sempre começa com um “seletor” que define a quais elementos HTML ela se aplica. Neste caso, estamos tentando estilizar o elemento `<body>`. Após o seletor, temos o “bloco de declarações” dentro de algumas chaves. Quaisquer “propriedades” que definirmos aqui afetarão o elemento `<body>`.



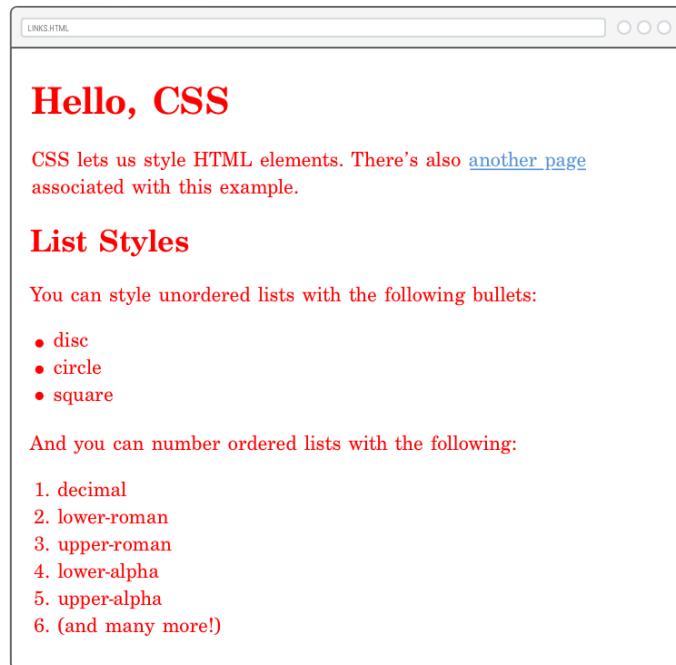
A propriedade `color`, definida pela especificação CSS, determina que será modificada a cor do texto de quaisquer elementos HTML selecionados. Essa propriedade aceita um valor hexadecimal representando uma cor. O valor `#FF0000`, por exemplo, representa o código hexadecimal para vermelho brilhante.

# Vinculando folhas de estilo CSS

Se você tentar carregar qualquer uma das páginas HTML em um navegador, perceberá que nossa folha de estilo não foi carregada. Isso porque ainda não a vinculamos. Para informar ao navegador, no momento em que ele renderizar uma página que temos uma folha de estilos, devemos utilizar o elemento `<link/>`. No arquivo `hello-css.html`, mude o trecho de código dentro do elemento `<head>` para o seguinte:

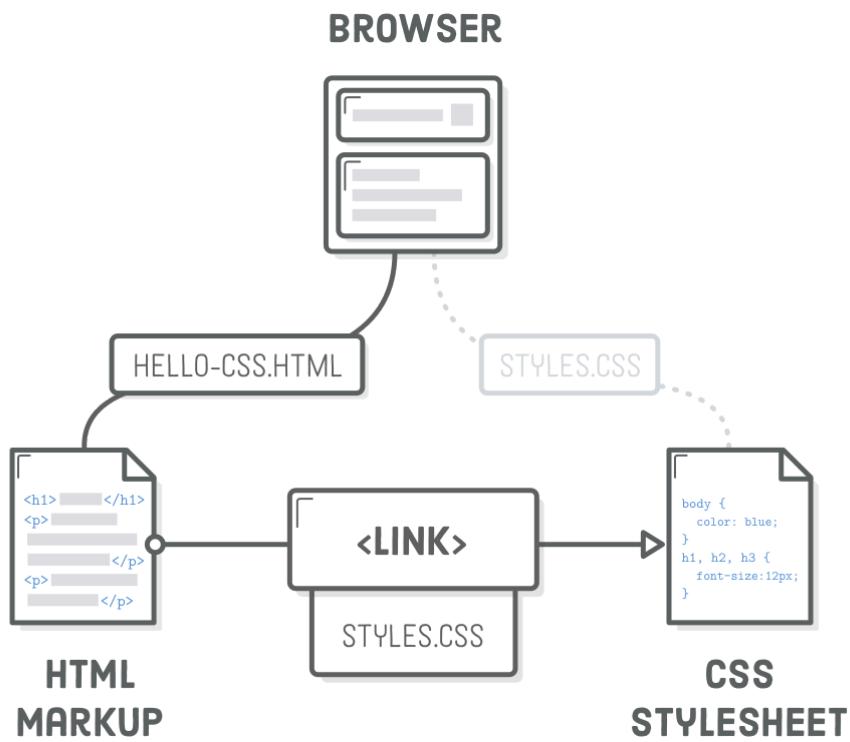
```
<head>
  <meta charset='UTF-8' />
  <title>Hello, CSS</title>
  <link rel='stylesheet' href='styles.css' />
</head>
```

O elemento `<link/>` é a forma como os navegadores sabem que precisam carregar `styles.css` quando tentam renderizar nossa página `hello-css.html`. Devemos agora ver um texto em vermelho, de gosto questionável, em todos os lugares:



O elemento `<link/>` é igual ao elemento `<a>`, mas só deve ser usado dentro de tag `<head>`. Como está no cabeçalho do documento, `<link/>` pode conectar-se aos metadados definidos fora do documento atual. Observe também que `<link/>` é um elemento vazio , portanto não precisa de uma tag de fechamento.

O atributo `rel` define o relacionamento entre o recurso e o documento HTML. De longe, o valor mais comum é `stylesheet`, mas existem algumas outras opções . O atributo `href` funciona da mesma forma que no capítulo anterior, só que deve apontar para um arquivo `.css` em vez de outra página da web. O valor de `href` pode ser um link absoluto, relativo ou relativo à raiz.



Observe que não há conexão direta entre o navegador e nossa folha de estilo. É somente através da marcação HTML que o navegador pode encontrá-lo. CSS, imagens e até mesmo JavaScript dependem de uma página HTML para unir tudo, tornando o HTML o coração da maioria dos sites.

# Comentários CSS

Agora que nossa folha de estilo está configurada, vamos trabalhar um pouco com ela. Esse vermelho é horrível. Vamos diminuir o tom para um belo cinza:

```
body {  
    color: #414141;      /* essa cor é o Dark gray */  
}
```

Observe que os comentários em CSS são um pouco diferentes dos comentários em HTML. Em vez da sintaxe `<!-- -->`, o CSS interpreta como comentários tudo entre os caracteres `/* e */`.

## Estilizando com Múltiplas Propriedades

Você pode inserir quantas propriedades quiser no bloco de declarações de uma regra CSS. Tente definir a cor de fundo de toda a página da web alterando nossa regra para o seguinte:

```
body {  
    color: #414141;          /* Dark gray */  
    background-color: #EEEEEE; /* Light gray */  
}
```

A propriedade `background-color` é muito semelhante à propriedade `color`, mas define a cor de fundo de qualquer elemento selecionado. Reserve um segundo para admirar os pontos e vírgulas no final de cada declaração. Removê-los violará a regra CSS, portanto, lembre-se sempre dos pontos e vírgulas!

Por que escolhemos tons de cinza em vez de preto e branco? Usar um background #000000 com cores de texto #FFFFFF, por exemplo, apresenta um contraste muito alto. Faz parecer que a página está vibrando, o que pode distrair muito os leitores.

## Selecionando Elementos Diferentes

Você desejará aplicar estilos a outros elementos além de <body>. Para isso, basta adicionar mais regras CSS com seletores diferentes. Podemos alterar o tamanho da fonte de nossos títulos <h1> da seguinte forma:

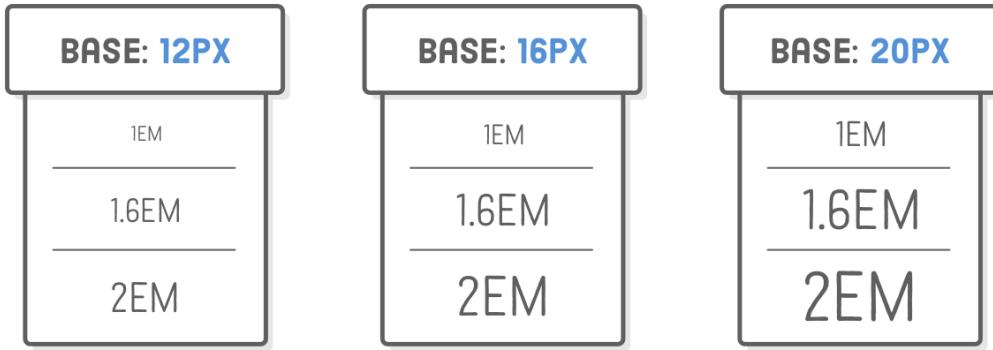
```
body {  
    color: #414141;          /* Dark gray */  
    background-color: #EEEEEE; /* Light gray */  
}  
  
h1 {  
    font-size: 36px;  
}
```

E, se quiser alterar os títulos <h2>, adicione outra regra:

```
h2 {  
    font-size: 28px;  
}
```

## Unidades de medida

Muitas propriedades CSS requerem uma unidade de medida. Há muitas unidades disponíveis, mas as mais comuns que você encontrará são px(pixel) e em (pronunciado como a letra m). O primeiro é o que você chamaria intuitivamente de pixel, independentemente de o usuário ter uma tela retina ou não, e o último é o tamanho da fonte atual do elemento em questão.



A unidade `em` é muito útil para definir tamanhos proporcionais à fonte definida pelo seletor `body`. Na imagem acima, você pode ver unidades `em` sendo dimensionadas para corresponder a um tamanho de fonte base de 12px, 16px e 20px. Para um exemplo concreto, considere a seguinte alternativa ao trecho de código anterior:

```
body {
  color: #414141;           /* Dark gray */
  background-color: #EEEEEE;  /* Light gray */
  font-size: 18px;
}

h1 {
  font-size: 2em;
}

h2 {
  font-size: 1.6em;
}
```

O código acima define o tamanho base da fonte do documento como 18px. Isso significa que elementos `<h1>` terão 36px, pois `2em` representa o dobro do tamanho da fonte de base. Já o tamanho dos elementos `<h2>` será de 28,8px, pois devem ser 1,6 vezes o tamanho da fonte base. Se nós (ou o usuário) quisessemos aumentar ou diminuir a fonte base, as unidades `em`'s permitiriam que toda a nossa página fosse dimensionada de acordo.

# Selecionando Vários Elementos

E se quisermos adicionar alguns estilos a todos os nossos títulos? Não queremos ter regras redundantes, pois isso acabaria se tornando um pesadelo para manter:

```
/* (Você se arrependerá de criar estilos redundantes como este) */
h1 {
    font-family: "Helvetica", "Arial", sans-serif;
}

h2 {
    font-family: "Helvetica", "Arial", sans-serif;
}

h3 {
    font-family: "Helvetica", "Arial", sans-serif;
}
/* (etc) */
```

Em vez disso, podemos selecionar vários elementos HTML na mesma regra CSS, separando-os com vírgulas. Adicione isto ao nosso arquivo `styles.css`:

```
h1, h2, h3, h4, h5, h6 {
    font-family: "Helvetica", "Arial", sans-serif;
}
```

Isso define a fonte a ser usada para todos os nossos títulos com uma única regra. Isso é ótimo, porque se quisermos mudar isso, só teremos que fazê-lo em um só lugar. Copiar e colar código geralmente é uma má ideia para programadores, e vários seletores podem ajudar a reduzir um pouco esse tipo de comportamento.

The screenshot shows a web browser window with a title bar labeled "LINKS.HTML". The main content area displays the text "Hello, CSS" in a large, bold, dark font. Below it, a smaller text says "CSS lets us style HTML elements. There's also [another page](#) associated with this example." A section titled "List Styles" follows, with the subtext "You can style unordered lists with the following bullets:" and a list of bullet types: "• disc", "• circle", and "• square". Further down, it says "And you can number ordered lists with the following:" followed by a numbered list from 1 to 6: "1. decimal", "2. lower-roman", "3. upper-roman", "4. lower-alpha", "5. upper-alpha", and "6. (and many more!)".

## Definição de Fontes

A propriedade `font-family` é a propriedade CSS que permite definir o tipo de fonte de qualquer elemento selecionado. Aceita vários valores porque nem todos os usuários terão as mesmas fontes instaladas em seus computadores. O trecho de código acima, que definiu o tipo de fonte para os títulos `h1` a `h6` funciona de seguinte forma: o navegador tenta carregar primeiro o mais à esquerda (`Helvetica`), se não houver essa fonte, tenta carregar a fonte `Arial` e, finalmente, escolhe a fonte sem serifas, padrão do sistema.



Depender das fontes do sistema operacional do usuário pode ser extremamente limitante para desenvolvedores web. Hoje em dia, as fontes do sistema foram amplamente substituídas pelas fontes da web. Você pode baixá-las e colocá-las em

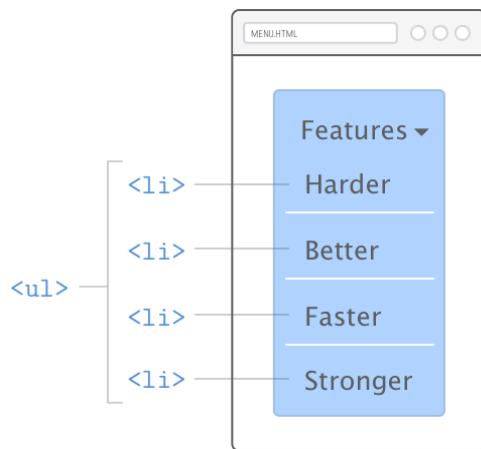
seu projeto ou mesmo usar fontes disponíveis na web, como é o caso do [Google Fonts](#). No futuro falaremos mais sobre fontes.

## Estilos de Lista

A propriedade `list-style-type` permite alterar o ícone do marcador usado para os elementos `<li>`. Normalmente você desejará defini-lo no elemento pai, `<ul>` ou `<ol>`:

```
ul {  
    list-style-type: circle;  
}  
  
ol {  
    list-style-type: lower-roman;  
}
```

Um valor interessante (para essa e outras propriedades) é o valor `none`, que é comumente usado ao marcar a navegação no menu com uma lista `<ul>`. O valor `none` permite que os itens da lista do menu tenham um estilo mais parecido com botões. Em discussões adiante, usaremos essa técnica para criar o menu de navegação mostrado abaixo.



Este é um bom exemplo da separação entre conteúdo e apresentação. Um menu de navegação é uma lista não ordenada, mas também faz sentido exibi-los como botões em vez de uma lista típica com marcadores. O HTML foi projetado de forma inteligente permitindo que os mecanismos de pesquisa possam inferir a estrutura do nosso conteúdo, enquanto o CSS nos permite exibi-lo aos humanos de maneiras bonitas.

Você pode até criar marcadores personalizados para elementos `<li>` com a propriedade `list-style-image` ([consulte MDN para obter detalhes](#)).

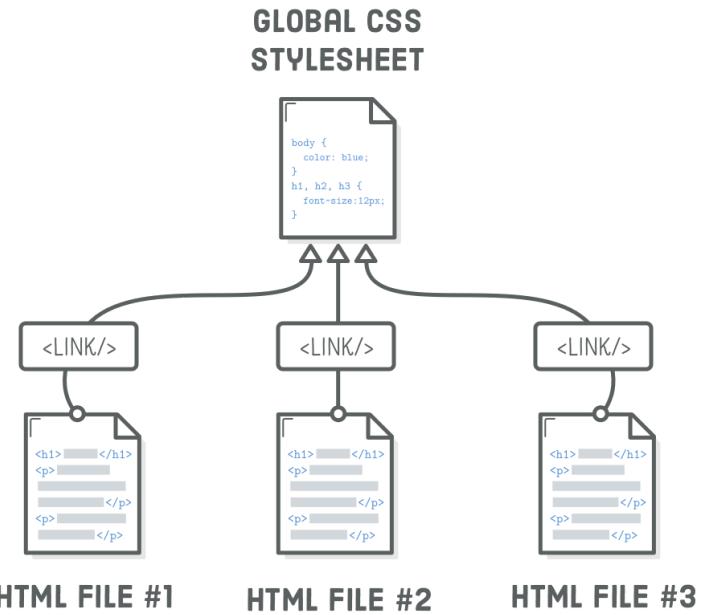
Definir a cor do seu texto e a aparência dos marcadores pode parecer trivial, e de certa forma é. Mas veja o panorama geral: trata-se de obter controle total sobre a aparência de um documento HTML. Sozinha, uma única propriedade CSS é bobagem. Junte todos eles e você poderá criar uma página da web totalmente personalizada.

## Folhas de Estilo Reutilizáveis

Então, acabamos de definir alguns estilos básicos para uma de nossas páginas web. Seria muito conveniente se pudéssemos reutilizá-los também em outras páginas. Para isso, basta adicionar o elemento `<link>` a quaisquer outras páginas que queiramos estilizar. Tente adicionar a seguinte linha ao `<head>` do arquivo `ficticio.html`:

```
<link rel='stylesheet' href='styles.css' />
```

Agora, nossa página `ficticio.html` deve corresponder aos nossos estilos definidos no arquivo `hello-css`. Sempre que alterarmos um estilo no arquivo `styles.css`, essas alterações serão refletidas automaticamente em ambas as nossas páginas da web. É assim que você obtém uma aparência consistente em todo o site.



Quase sempre você terá pelo menos uma folha de estilo aplicada a todo o site. Geralmente é uma boa ideia usar caminhos relativos à raiz ao vincular folhas de estilo globais para evitar problemas em páginas aninhadas. Por exemplo, para a página localizada em `alguma-pasta/pagina.html` seria necessário usar o caminho `../styles.css` para referenciar nosso arquivo `styles.css`. Redobre a atenção nesses caminhos, pois eles podem ficar confusos rapidamente.

## Mais estilos de Texto

Há um monte de propriedades CSS diferentes que apresentaremos ao longo deste guia, mas por enquanto, vamos terminar com algumas das formas mais comuns de formatar texto.

### Sublinhadas

A propriedade `text-decoration` determina se o texto está sublinhado ou não. Ao defini-lo como `none`, podemos remover o sublinhado padrão de todos os nossos links. Discutiremos detalhadamente os estilos de link adiante.

```
a {
  text-decoration: none;
}
```

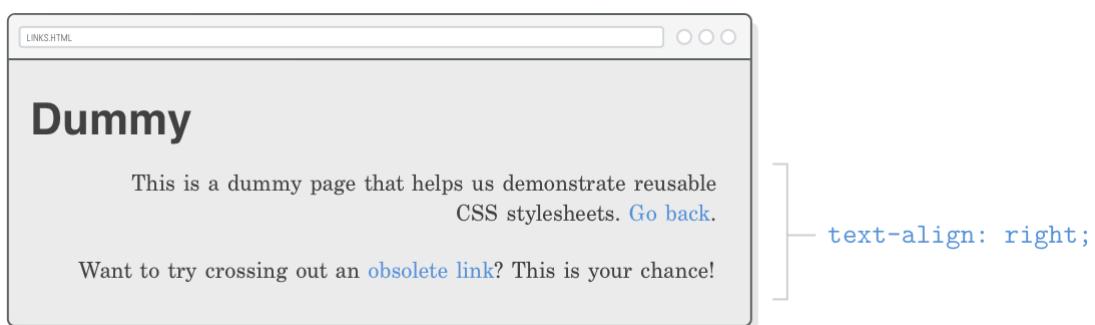
O outro valor comum para a propriedade `text-decoration` é `line-through`, que traz um efeito de riscar ou tachar o texto. Mas lembre-se de que o significado deve sempre ser transmitido por meio de HTML – não de CSS. Por exemplo, quando quisermos indicar que um texto foi excluído, como ocorre em [leis brasileiras](#), é melhor usar os elementos `<ins>` e `<del>` em vez de adicionar o estilo `line-through` a um elemento `<p>`, por exemplo. Esses elementos fazem com que o estilo tachado não seja apenas estético, mas faz com que o navegador, que leitores de tela e o Google, por exemplo, saibam que esse trecho do texto não é mais tão relevante assim.

## Alinhamento de Texto

A propriedade apropriadamente chamada de `text-align` define o alinhamento do texto em um elemento HTML

```
p {  
  text-align: left;  
}
```

Além do valor `left`, os outros valores possíveis são `right`, `center`, ou `justify`:



## Peso e Estilos da Fonte

A propriedade `font-weight` define o “negriticidade” do texto em um elemento. Quanto maior o valor da propriedade `font-weight` mais escuro fica o texto. Já a propriedade `font-style` indica se está em itálico ou não.

Digamos que não queremos que nossos títulos sejam tão ousados assim. Atualize nossa regra de fonte de título no arquivo `styles.css`:

```
h1, h2, h3, h4, h5, h6 {  
    font-family: "Helvetica", "Arial", sans-serif;  
    font-weight: normal; /* Adicione essa linha */  
}
```

Essas propriedades demonstram claramente a separação entre conteúdo (HTML) e apresentação (CSS). As regras a seguir trocam a aparência dos elementos `<em>` e `<strong>`:

```
/* Você provavelmente não irá querer fazer isso */  
em {  
    font-weight: bold;  
    font-style: normal;  
}  
  
strong {  
    font-weight: normal;  
    font-style: italic;  
}
```

Porém, não sugerimos fazer isso para sites reais. Os pesos e estilos das fontes, no entanto, se tornarão muito mais importantes quando começarmos a brincar com fontes personalizadas.

## A Cascata do CSS

A parte “em cascata” do CSS se deve ao fato de que as regras descem em cascata de múltiplas fontes. Até agora, vimos apenas um lugar onde o CSS pode ser definido: arquivos com a extensão `.css` externos. No entanto, folhas de estilo externas são apenas um dos muitos lugares onde você pode colocar seu código CSS.

A hierarquia CSS para cada página da web é semelhante a esta:

- A folha de estilo padrão do navegador
- Folhas de estilo definidas pelo usuário
- Folhas de estilo externas (usadas até agora)
- Estilos específicos da página
- Estilos `inline` ou embutidos

Isso é ordenado da menor para a maior precedência, o que significa que os estilos definidos em cada etapa subsequente substituem os anteriores. Por exemplo, os estilos embutidos sempre farão com que o navegador ignore seus estilos padrão. As próximas seções se concentram nas duas últimas opções porque é sobre elas que temos controle como desenvolvedores web (além dos estilos externos com os quais já trabalhamos).

Neste guia houve um esforço para que você começasse no caminho certo, que é o uso de folhas de estilo externas. É importante entender os estilos `inline` ou embutidos e os estilos específicos da página, porque você certamente os encontrará por aí, em projetos e exemplos da internet. Contudo é importante deixar claro que as folhas de estilo externas são de longe o melhor lugar para definir a aparência do seu site.

## Estilos Específicos de Página

O elemento `<style>` é usado para adicionar regras CSS específicas da página a documentos HTML individuais. O elemento `<style>` sempre está dentro do elemento `<head>` de uma página da web, o que faz sentido porque são metadados, não conteúdo visível ao usuário.

Como exemplo, vamos aplicar alguns estilos à nossa página fictício.html atualizando o conteúdo do elemento <head>:

```
<head>
  <meta charset='UTF-8' />
  <title>Dummy</title>
  <link rel='stylesheet' href='styles.css' />
  <style>
    body {
      color: #0000FF; /* Azul */
    }
  </style>
</head>
```

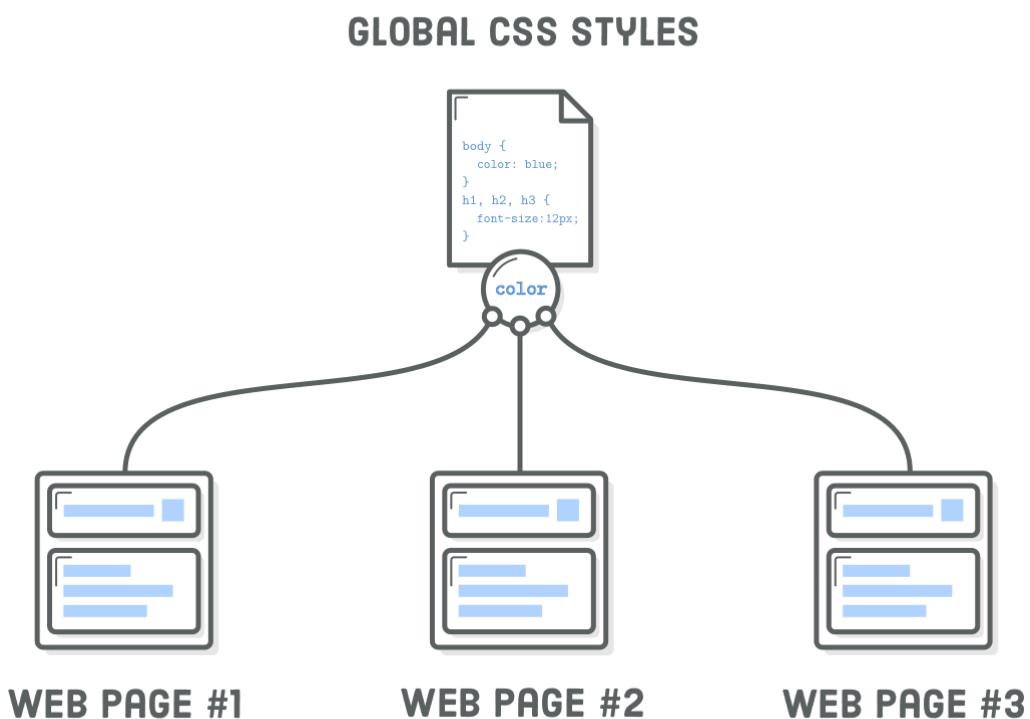
Esses estilos se aplicam apenas a página fictício.html. Nossa página hello-css.html não será afetada. Se tudo deu certo, deverá ver um texto em azul brilhante ao carregar a página fictício.html em um navegador.



Qualquer coisa que você colocar em nosso arquivo styles.css pode também ser colocada no elemento <style>: ele usa exatamente a mesma sintaxe CSS de uma folha de estilo externa. Como vimos anteriormente, todas as regras informadas aqui substituirão as regras em nosso arquivo styles.css. Neste caso, estamos dizendo ao navegador para ignorar a propriedade color que definimos para o elemento <body> em nossa folha de estilo externa e passar a usar esse azul maravilhoso, definido pelo hexadecimal #0000FF.



O problema com estilos específicos de página é que eles são difíceis de manter. Conforme mostrado na figura acima, quando você deseja aplicar esses estilos a outra página, você deve copiá-los e colá-los no arquivo `<head>`. Tentar rastrear regras CSS redundantes em vários arquivos `.html` é muito mais difícil do que editar um único arquivo `.css`.



Estilos específicos de página ocasionalmente são úteis quando você está com pressa, mas quase sempre é melhor armazenar todo o seu CSS em folhas de estilo externas em vez de elementos `<style>`.

## Estilos Inline

Você também pode inserir regras CSS no atributo `style` de um elemento HTML. Em nossa página `fictício.html`, temos um link que na verdade não leva a lugar nenhum. Vamos torná-lo vermelho através de um estilo `inline` para lembrarmos que é um link morto:

```
<p>Quer tentar arriscar um <a href="lugar-nenhum.html' style='color: #990000; text-decoration: line-through;'>link duvidoso</a>? Essa é sua chance!</p>
```

Assim como os estilos específicos de página, esta é a mesma sintaxe CSS com a qual estamos trabalhando. Porém, como está em um atributo, precisa ser condensado em uma única linha. Os estilos embutidos são a forma mais específica de definir CSS. As propriedades `color` e `text-decoration` que definimos aqui tem prioridade sobre as regras definidas anteriormente. Mesmo se voltássemos e adicionássemos a `text-decoration: none` ao nosso elemento `<style>`, isso não teria nenhum efeito.



Os estilos embutidos devem ser evitados a todo custo. Se você quiser alterar os estilos do seu site no futuro, não poderá simplesmente alterar algumas regras em seu

arquivo `styles.css` – você teria que passar por cada página e atualizar cada elemento HTML que possui um atributo `style`. Além disso, você precisaria copiar e colar os estilos em cada elemento. É horrível.

Dito isto, muitas vezes você precisará aplicar estilos apenas a um elemento HTML específico. Para isso, você deve sempre usar classes CSS em vez de estilos inline. Exploraremos as classes adiante .

## Várias Folhas de Estilo

As regras CSS podem ser espalhadas por várias folhas de estilo externas adicionando vários elementos `<link/>` à mesma página. Um caso de uso comum é separar estilos para diferentes seções do seu site. Isso permite aplicar seletivamente estilos consistentes a categorias distintas de páginas da web.

Por exemplo, se tivéssemos um monte de páginas de produtos que pareciam totalmente diferentes do nosso blog, poderíamos usar o seguinte (Não temos essas folhas de estilo definidas, então não se preocupe em adicioná-las ao nosso projeto, é só um exemplo):

←— Todas as páginas de produtos têm isto →

```
<head>
  <link rel='stylesheet' href='styles.css' />
  <link rel='stylesheet' href='produtos.css' />
</head>
```

←— Todas as postagens do blog têm isso →

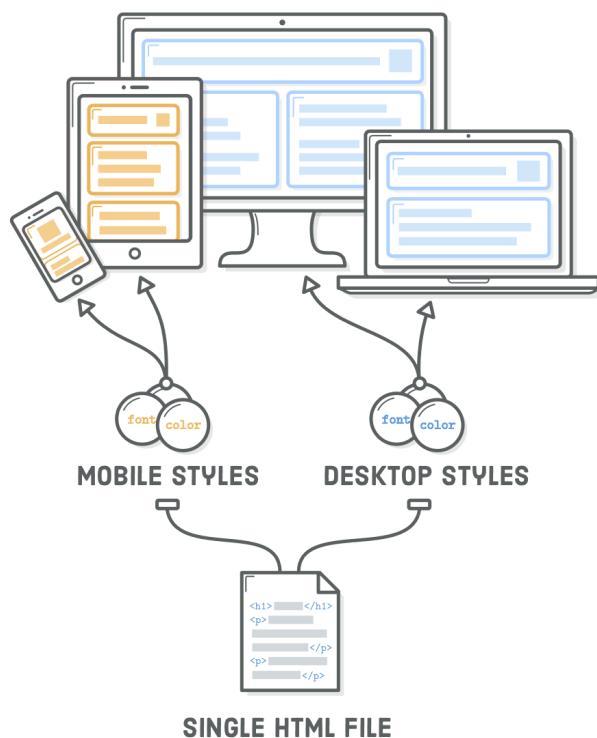
```
<head>
  <link rel='stylesheet' href='styles.css' />
  <link rel='stylesheet' href='blog.css' />
</head>
```

A ordem dos elementos `<link>` é importante. Caso um mesmo elemento esteja sendo utilizado em duas folhas separadas, a estilização feita em folhas de estilo posteriores substituirão os estilos anteriores (existem outras maneiras de se alterar a precedência, veremos isso adiante).

Normalmente, você colocará seus estilos “base” ou “padrão” em uma folha de estilo global (`styles.css`) e os complementará com folhas de estilo específicas da seção (`produto.css` e `blog.css`). Isso permite que você organize regras CSS em arquivos gerenciáveis, evitando os perigos de estilos embutidos e específicos de página.

## Resumo

Falamos muito sobre separar o conteúdo da apresentação neste capítulo. Isso não apenas nos permite reutilizar a mesma folha de estilo CSS em vários documentos HTML, mas também nos permite aplicar condicionalmente regras CSS diferentes ao mesmo conteúdo HTML, dependendo se o usuário está em um telefone celular, tablet ou computador desktop. Esta última parte é chamada de Design Responsivo .



Como desenvolvedor web, você (esperançosamente) receberá um design sofisticado para trabalhar. Seu trabalho é transformar esse modelo em uma página da web real, aproveitando seu conhecimento de CSS. Como mencionamos anteriormente, definir propriedades CSS individuais é bastante simples. A parte difícil é combinar o grande número de propriedades integradas para criar exatamente o que seu web designer pediu – e fazer isso rapidamente.

Este capítulo se concentrou principalmente na formatação de texto, mas a linguagem Cascading Style Sheet pode fazer muito mais. No próximo capítulo, começaremos a explorar como o CSS define o layout de nossas páginas web. Por fim, lembre-se de que você sempre pode consultar a Referência CSS do [MDN](#) quando não tiver certeza de como uma propriedade específica funciona.

# Modelos de Caixa

## Capítulo 5

*Uma introdução amigável ao preenchimento, bordas e margens*

O capítulo anterior introduziu as propriedades básicas de formatação de texto do CSS, mas esse foi apenas um aspecto do estilo de páginas. **Definir o layout** de uma página da web é algo totalmente diferente. É disso que trata este capítulo.

O “modelo de caixa” do CSS nada mais é que um conjunto de regras que definem como cada página da web na Internet é renderizada. CSS trata cada elemento do seu documento HTML como uma “caixa” com várias propriedades diferentes que determinam onde ele aparece na página. Até agora, todas as nossas páginas da web eram apenas um monte de elementos renderizados um após o outro. O modelo de caixa é nosso kit de ferramentas para personalizar esse esquema de layout padrão.

Uma grande parte do seu trabalho como desenvolvedor web será aplicar regras do modelo de caixa CSS para transformar um modelo de design em uma página web. À medida que você trabalha neste capítulo, você pode se perguntar por que precisamos aprender todas essas regras em vez de apenas enviar uma imagem estática gigante de uma página da web para um servidor da Web e encerrar o dia.

Na verdade, isto tornaria a vida muito mais fácil. Entretanto, se não separássemos nosso conteúdo em HTML, os mecanismos de pesquisa não teriam como inferir a estrutura de nossas páginas da web, não poderíamos tornar nosso site responsivo e não haveria como adicionar animações sofisticadas ou interatividade com JavaScript. Essa é uma compensação grande o suficiente para tornar o CSS uma causa válida.

Este capítulo aborda os principais componentes do modelo de caixa CSS: preenchimento, bordas, margens, caixas de bloco e caixas embutidas. Você pode pensar nisso como a visão “micro” dos layouts CSS, pois define o comportamento individual das caixas. Nos próximos capítulos, aprenderemos mais sobre como a

estrutura HTML e o modelo de caixa CSS se combinam para formar todos os tipos de layouts de páginas complexas.

## Configurações

Para começar, vamos criar uma nova pasta chamada `css-modelos-de-caixa` e inserir nela uma nova página da web chamada `caixas.html`. Adicione o seguinte código:

```
<!DOCTYPE html>
<html lang='en'>
  <head>
    <meta charset='UTF-8' />
    <title>Caixas são moleza!</title>
    <link rel='stylesheet' href=modelelos-de-caixa.css' />
  </head>
  <body>
    <h1>Cabeçalhos são elementos de bloco</h1>

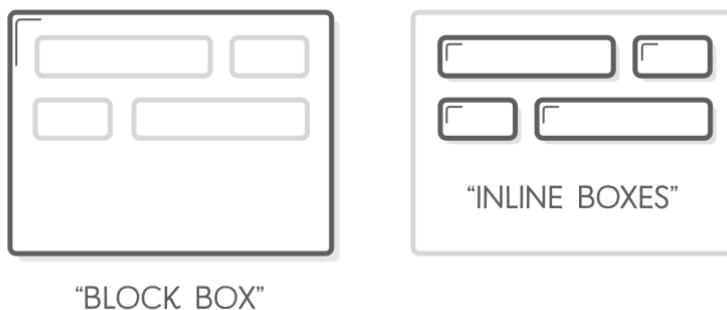
    <p>Parágrafos são elementos de bloco, também. <em>Já<br/>
       elementos</em>, &lt;em&gt; e &lt;strong&gt; não são: eles são<br/>
       elementos <strong>inline</strong>.</p>

    <p>Elementos de bloco definem o fluxo do documento HTML,<br/>
       enquanto elementos inline não.</p>
  </body>
</html>
```

Assim como foi feito projeto Hello CSS, esse arquivo HTML está vinculado a uma folha de estilo CSS armazenada em um arquivo chamado `estilos-de-caixa.css`. Vá em frente e crie este arquivo também (deixe-o vazio por enquanto).

# Elementos de bloco e elementos de linha

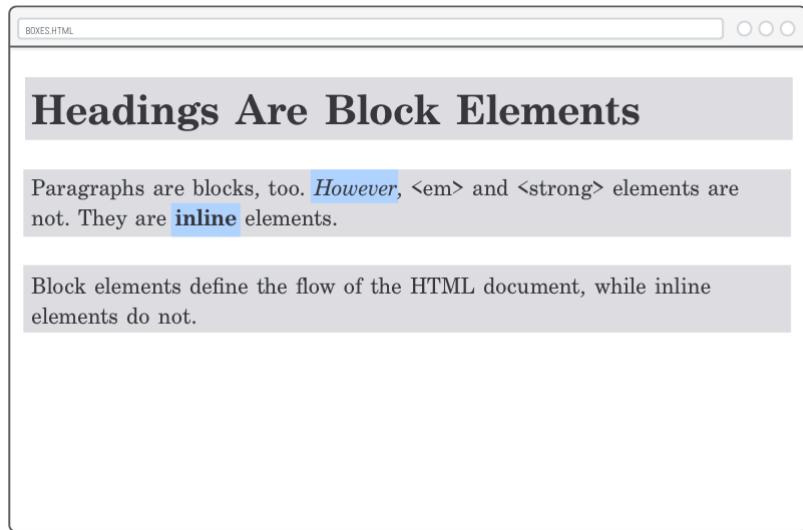
Anteriormente, abordamos brevemente como o CSS usa “caixas” para definir o layout de uma página da web. Cada elemento HTML renderizado na tela é uma caixa, e eles vêm em dois tipos: caixas “bloco” e caixas “inline”.



Todos os elementos HTML com os quais estamos trabalhando possuem um tipo de caixa padrão. Por exemplo, `<h1>` e `<p>` são elementos de nível de bloco, enquanto `<em>` e `<strong>` são elementos de linha. Vamos dar uma olhada melhor em nossas caixas adicionando o seguinte a `modelos-de-caixa.css`:

```
h1, p {  
    background-color: #DDE0E3;      /* Light gray */  
}  
  
em, strong {  
    background-color: #B2D6FF;      /* Light blue */  
}
```

A propriedade `background-color` preenche apenas o plano de fundo da caixa selecionada, portanto, isso nos dará uma visão clara da estrutura da página de amostra atual. Nossos títulos e parágrafos devem ter fundo cinza, enquanto nossa ênfase e elementos fortes devem ser azuis claros.



Isso nos mostra alguns comportamentos muito importantes associados a caixas de bloco e inline:

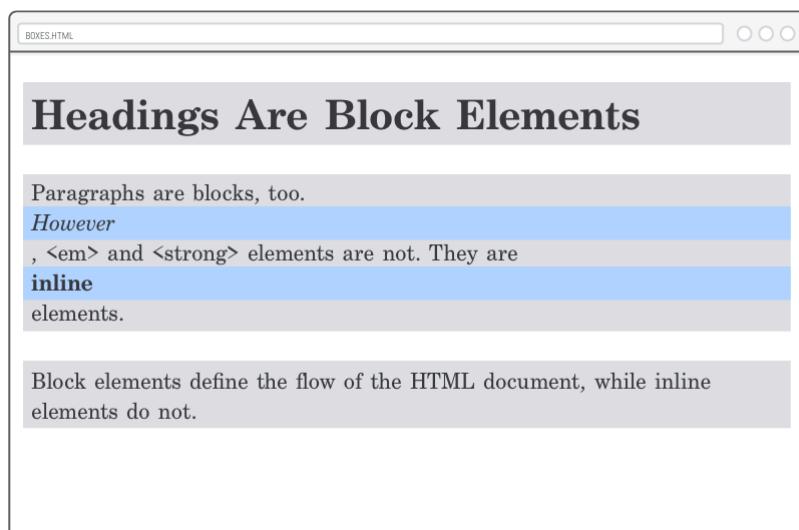
- **As caixas de bloco** sempre aparecem **abaixo** do elemento de bloco anterior. Este é o fluxo “natural” ou “estático” de um documento HTML quando ele é renderizado por um navegador da web.
- A **largura das caixas de bloco** é definida automaticamente com base na largura do contêiner pai. Até agora, nossos blocos tiveram sempre a largura da janela do navegador.
- A **altura padrão das caixas de bloco** é baseada no conteúdo que elas contêm. Quando você estreita a janela do navegador, ela `<h1>` é dividida em duas linhas e sua altura é ajustada de acordo.
- **As caixas inline** não afetam o **espaçamento vertical**. Eles não servem para determinar o layout - servem para estilizar coisas *dentro* de um bloco.
- A **largura das caixas em linha** é baseada no conteúdo que elas contêm, não na largura do elemento pai.

# Alterando o comportamento da Caixa

Podemos substituir o tipo de caixa padrão dos elementos HTML pela propriedade `display` do CSS. Por exemplo, se quiséssemos fazer nossos blocos de elementos `<em>` e `<strong>` em vez de elementos inline, poderíamos atualizar nossa regra da `estilos-de-caixa.css` seguinte forma:

```
em, strong {  
    background-color: #B2D6FF;  
    display: block;  
}
```

Agora, esses elementos funcionam como nossos títulos e parágrafos: eles começam em sua própria linha e preenchem toda a largura do navegador. Isso é útil quando você está tentando transformar elementos `<a>` em botões ou formatar elementos `<img />` (ambos são caixas embutidas por padrão).



No entanto, quase nunca é uma boa ideia transformar `<em>` e `<strong>` em elementos de bloco, então vamos transformá-los novamente em caixas inline alterando sua propriedade `display` para `inline`, assim:

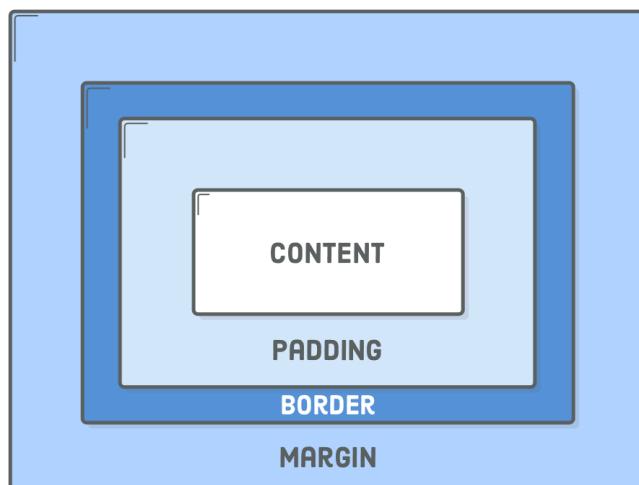
```
em, strong {  
background-color: #B2D6FF;  
display: inline; /* Esse é o padrão para em e strong */  
}
```

## Conteúdo, Preenchimento, borda e margem

O “modelo de caixa CSS” é um conjunto de regras que determinam as dimensões de cada elemento em uma página web. Ele fornece a cada caixa (em linha e em bloco) quatro propriedades:

- **Conteúdo** – O texto, imagem ou outro conteúdo de mídia no elemento.
- **Padding (preenchimento)** – O espaço entre o conteúdo da caixa e sua borda.
- **Border (borda)** – A linha entre o preenchimento e a margem da caixa.
- **Margin (margem)** – O espaço entre a caixa e as caixas circundantes.

Juntos, isso é tudo que um navegador precisa para renderizar a caixa de um elemento. O conteúdo é o que você cria em um documento HTML e é o único que possui algum valor semântico ([é por isso que está no HTML](#)). O restante deles é puramente de apresentação, portanto são definidos por regras CSS.

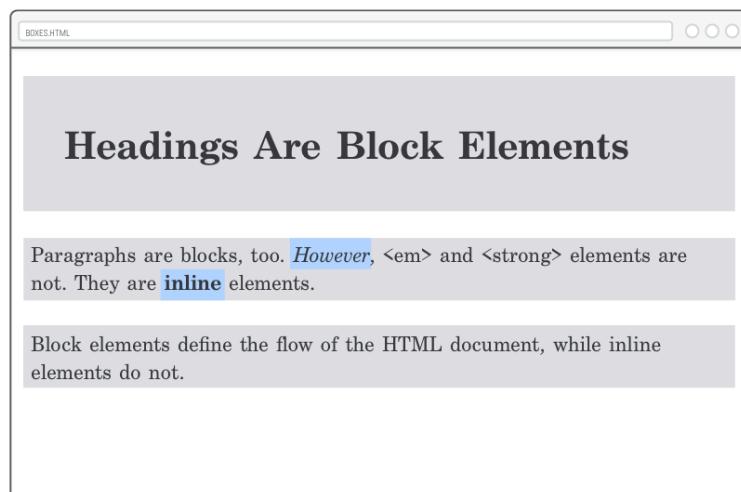


# Preenchimento

Vamos começar de dentro para fora. A propriedade padding define o preenchimento do elemento selecionado:

```
h1 {  
  padding: 50px;  
}
```

Isso adiciona 50 pixels a cada lado do título `<h1>`. Observe como a cor de fundo se expande para preencher esse espaço. Isso ocorre porque está dentro da borda e tudo dentro da borda ganha cor de fundo.



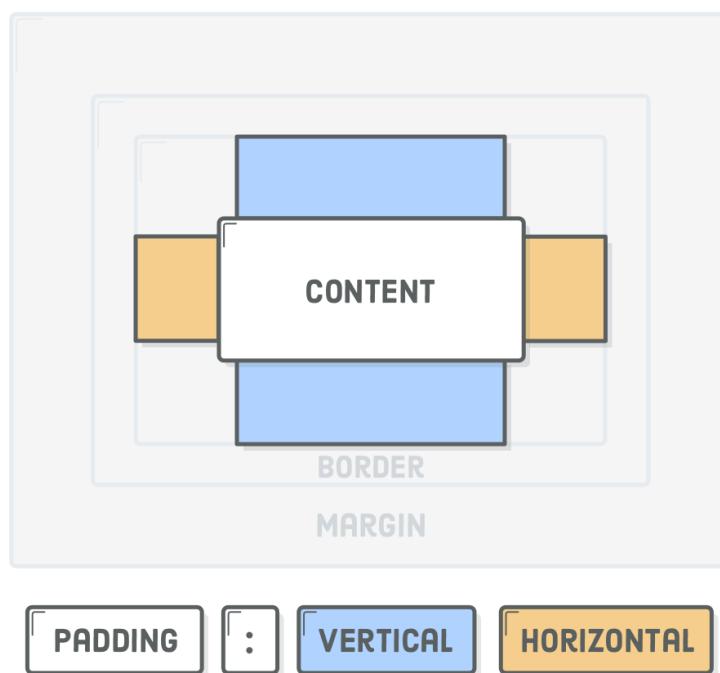
Às vezes, você desejará estilizar apenas um lado de um elemento. Para isso, o CSS fornece as seguintes propriedades:

```
p {  
  padding-top: 20px;  
  padding-bottom: 20px;  
  padding-left: 10px;  
  padding-right: 10px;  
}
```

Você pode usar qualquer unidade para o preenchimento de um elemento, não apenas pixels. Novamente, a [unidade de medida em](#) é particularmente útil para dimensionar suas margens com o tamanho base da fonte.

## Formatos Abreviados

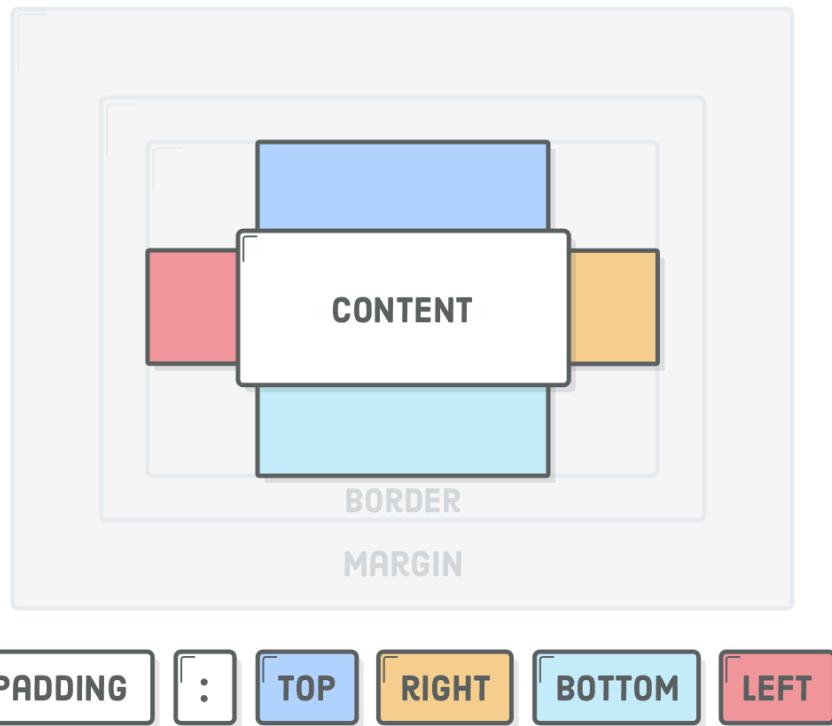
Digitar todas essas propriedades pode ser cansativo. Como um atalho, o CSS fornece uma forma alternativa de “abreviação” da propriedade padding que permite definir o preenchimento superior/inferior e esquerdo/direito com apenas uma linha de CSS. Quando você fornece dois valores à propriedade padding, eles são interpretados como valores de preenchimento vertical e horizontal, respectivamente.



Isso significa que nossa regra anterior pode ser reescrita como:

```
p {  
padding: 20px 10px; /* Vertical Horizontal */  
}
```

Alternativamente, se você fornecer quatro valores, poderá definir o preenchimento para cada lado de um elemento individualmente. Os valores são interpretados no sentido horário, começando no topo:



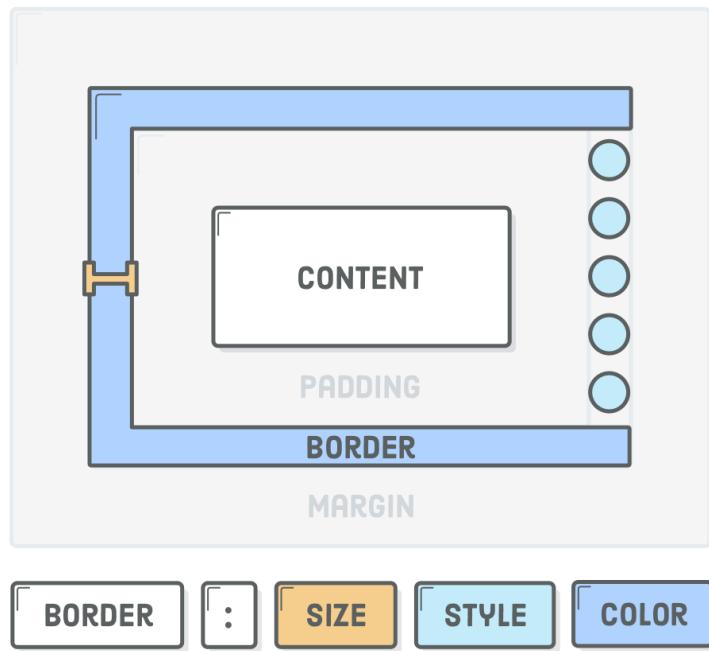
Vamos tentar remover o preenchimento de 10px correto da regra anterior. Isso deve nos dar 20 pixels na parte superior e inferior de cada parágrafo, 10 pixels à esquerda, mas nenhum à direita:

```
p {  
padding: 20px 0 20px 10px; /* Topo Direita Baixo Esquerda */  
}
```

Se você deseja ou não usar propriedades CSS de forma abreviada, é em grande parte uma questão de preferência pessoal e convenções da equipe de trabalho. Alguns desenvolvedores gostam do fato de a forma abreviada ser mais condensada, enquanto outros acham que a forma longa é mais fácil de entender à primeira vista (e, portanto, mais fácil de manter). Independentemente disso, você provavelmente encontrará todos eles em algum momento de sua carreira de desenvolvimento web.

# Bordas

Continuando nossa jornada para fora do centro do modelo de caixa CSS, temos a borda: uma linha desenhada ao redor do conteúdo e do preenchimento de um elemento. A propriedade `border` requer uma nova sintaxe que nunca vimos antes. Primeiro definimos a largura do traço da borda, depois o seu estilo, seguido da sua cor.



Tente adicionar uma borda ao redor do nosso título `<h1>` atualizando a regra em `estilos-caixa.css`:

```
h1 {  
  padding: 50px;  
  border: 1px solid #5D6063;  
}
```

Isso diz ao navegador para desenhar uma linha cinza fina ao redor do nosso título. Observe como a borda se eleva ao lado do preenchimento, sem espaço entre elas. E, se você reduzir o navegador o suficiente para que o título seja dividido em duas linhas, tanto o preenchimento quanto a borda ainda estarão lá.

Desenhar uma borda ao redor do nosso título faz com que pareça um pouco dos anos 1990, então que tal limitá-lo ao final do título? Assim como na propriedade padding, existem variantes `-top`, `-bottom`, `-left` e `-right` para a propriedade border:

```
border-bottom: 1px solid #5D6063;
```

As bordas são elementos de design comuns, mas também são inestimáveis para depuração e testes. Quando você não tiver certeza de como uma caixa está sendo renderizada, adicione a ela uma regra, como por exemplo: `border: 1px solid red`. Isso mostrará claramente o preenchimento, a margem e as dimensões gerais da caixa com apenas uma única linha de CSS. Depois de descobrir por que suas coisas estão quebradas, simplesmente exclua a regra.

Consulte a [Mozilla Developer Network](#) para obter mais informações sobre estilos de borda.

## Margens

As margens definem o espaço fora da borda de um elemento. Ou melhor, o espaço entre uma caixa e as caixas que a rodeiam. Vamos adicionar margens na parte inferior de cada elemento `<p>`:

```
p {  
  padding: 20px 0 20px 10px;  
  margin-bottom: 50px;          /* Adicione esta linha*/  
}
```

Isso demonstra uma variante específica da propriedade margin, mas também aceita os mesmos [formatos abreviados](#) que padding.

Margens e preenchimento podem realizar a mesma coisa em muitas situações, tornando difícil determinar qual é a escolha “certa”. Os motivos mais comuns pelos quais você escolheria um em vez de outro são:

- O preenchimento de uma caixa tem fundo, enquanto as margens são sempre transparentes.
- O preenchimento está incluído na área de clique de um elemento, enquanto as margens não.
- As margens colapsam verticalmente, enquanto o preenchimento não (discutiremos isso mais na próxima seção).

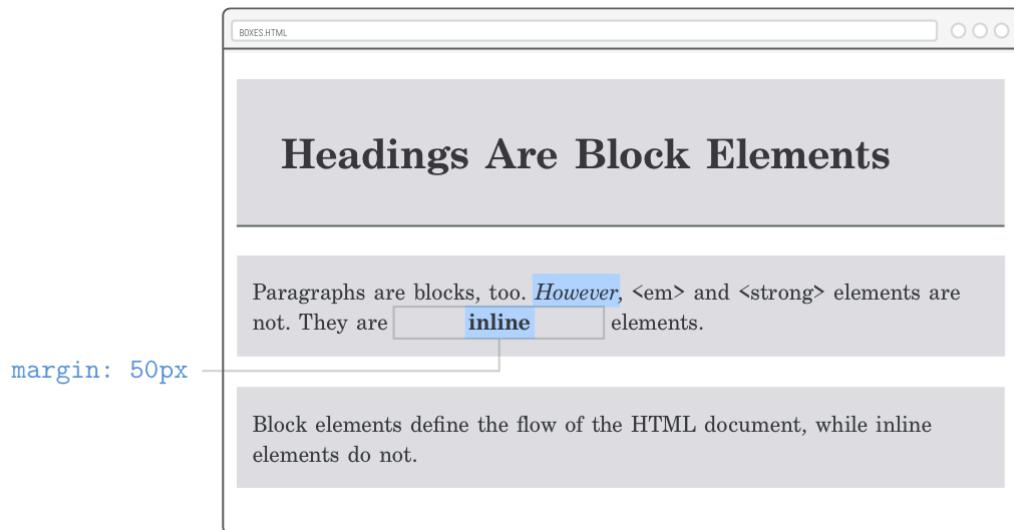
Se nada disso ajudar você a decidir se deve usar a propriedade `padding` ou a propriedade `margin`, não se preocupe - basta escolher um. Em CSS, geralmente há mais de uma maneira de resolver seu problema.

## Margens em elementos embutidos

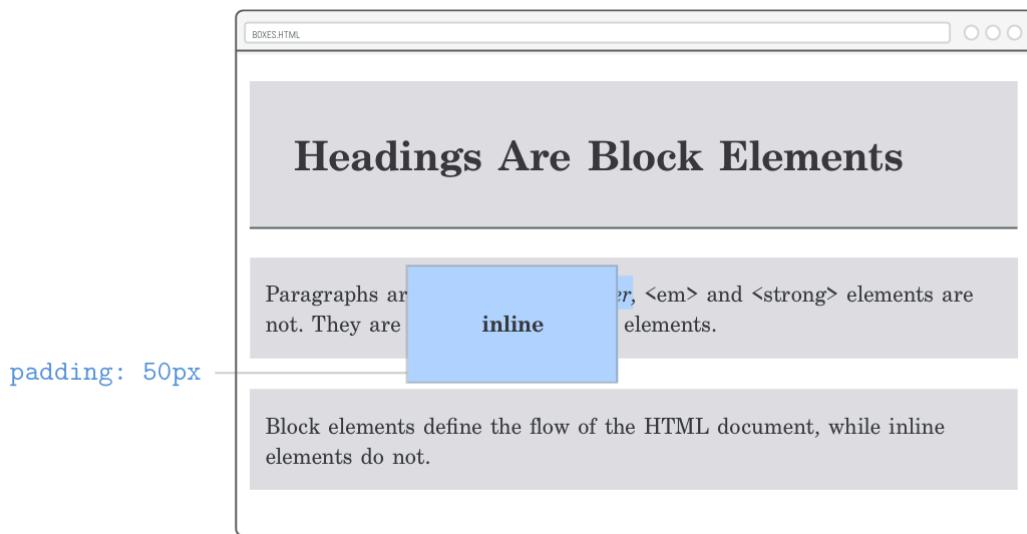
Um dos contrastes mais marcantes entre os elementos de nível de bloco e os inline é o tratamento das margens. As caixas embutidas ignoram completamente as margens superior e inferior de um elemento. Por exemplo, observe o que acontece quando adicionamos uma grande margem ao nosso elemento `<strong>`:

```
strong {  
  margin: 50px;  
}
```

As margens horizontais são exibidas exatamente como esperávamos, mas isso não altera nem um pouco o espaço vertical ao redor do nosso elemento `<strong>`.



Se mudarmos `margin` para `padding`, descobriremos que este não é exatamente o caso do preenchimento de uma caixa. Irá exibir o fundo azul; entretanto, isso não afetará o layout vertical das caixas circundantes.



A lógica por trás disso remonta ao fato de que as caixas em linha formatam trechos de texto dentro de um bloco e, portanto, têm impacto limitado no layout geral de uma página. Se você quiser brincar com o espaço vertical de uma página, você deve trabalhar com elementos em nível de bloco (felizmente, já sabemos [como alterar o tipo de caixa](#) de um elemento).

Portanto, antes de começar a bater a cabeça na parede tentando descobrir por que sua margem superior ou inferior não está funcionando, lembre-se de verificar sua propriedade display. Pode acreditar, isso acontecerá com você eventualmente.

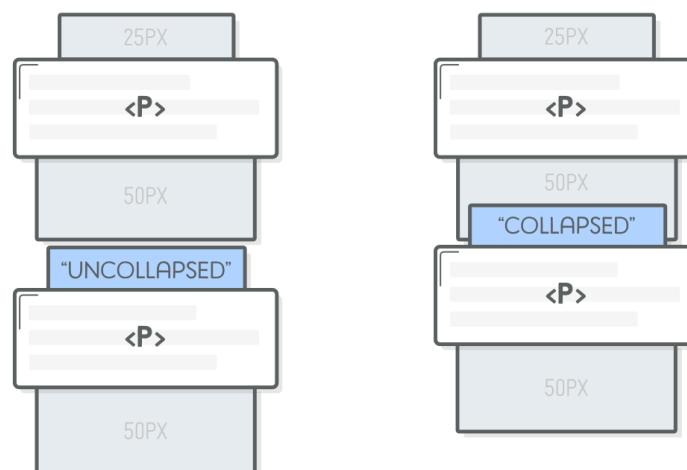
## Colapso da margem vertical

Outra peculiaridade do modelo de caixa CSS é o “colapso da margem vertical”. Quando você tiver duas caixas com margens verticais próximas uma da outra, elas entrarão em colapso. Em vez de somar as margens como seria de esperar, apenas a maior delas é exibida.

Por exemplo, vamos adicionar uma margem superior de 25 pixels ao nosso elemento <p>:

```
p {  
    padding: 20px 0 20px 10px;  
    margin-top: 25px;  
    margin-bottom: 50px;  
}
```

Cada parágrafo deve ter 50 pixels na parte inferior e 25 pixels na parte superior. São 75 pixels entre nossos elementos <p>, certo? Errado! Ainda só haverá 50px entre eles porque a margem superior menor colapsa na margem inferior maior.



Este comportamento pode ser muito útil quando você está trabalhando com vários tipos diferentes de elementos e deseja definir seu layout como o espaço mínimo entre outros elementos.

## Caixas genéricas

Até agora, cada elemento HTML que vimos confere um significado adicional ao conteúdo que contém. Na verdade, esse é o objetivo do HTML, mas muitas vezes precisamos de uma caixa genérica apenas para estilizar uma página da web. Para isso o HTML disponibiliza os elementos `<div>` e `<span>`.

Ambos, `<div>` e `<span>`, são elementos “contêiner” que não afetam a estrutura semântica de um documento HTML. No entanto, eles fornecem um gancho para adicionar estilos CSS a seções de uma página da web. Por exemplo, às vezes você precisa adicionar uma caixa invisível para evitar o colapso da margem, ou talvez queira agrupar os primeiros parágrafos de um artigo em uma sinopse com formatação de texto ligeiramente diferente.

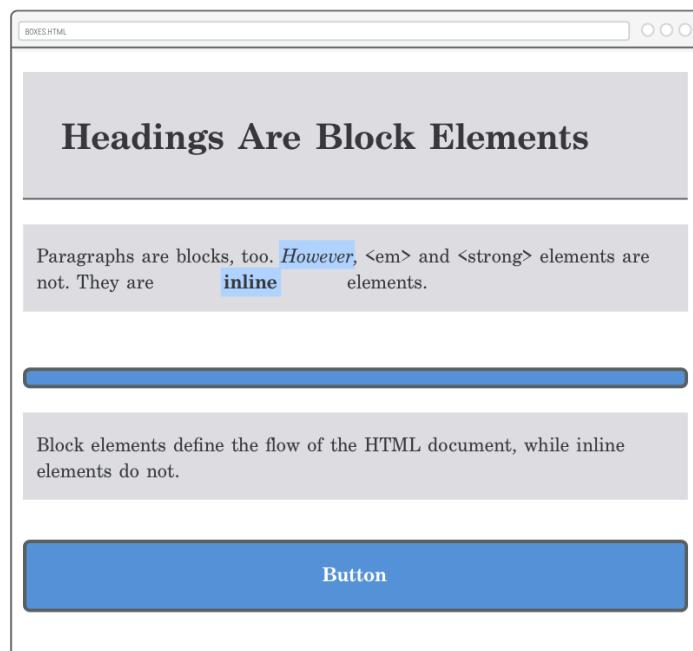
Usaremos muitos `<div>`'s ao longo do restante deste guia. Por enquanto, vamos criar um botão simples adicionando o seguinte ao final do nosso arquivo `caixas.html`:

```
<div>Button</div>
```

E aqui estão os estilos associados que precisam ser incluídos `modelos-de-caixa.css`. A essa altura, já estamos familiarizados com as propriedades CSS. Aqui apenas estamos introduzindo uma nova propriedade chamada de `border-radius`:

```
div {  
  color: #FFF;  
  background-color: #5995DA;  
  font-weight: bold;  
  padding: 20px;  
  text-align: center;  
  border: 2px solid #5D6063;  
  border-radius: 5px;  
}
```

Isso nos dará um grande botão azul que ocupa toda a largura do navegador:

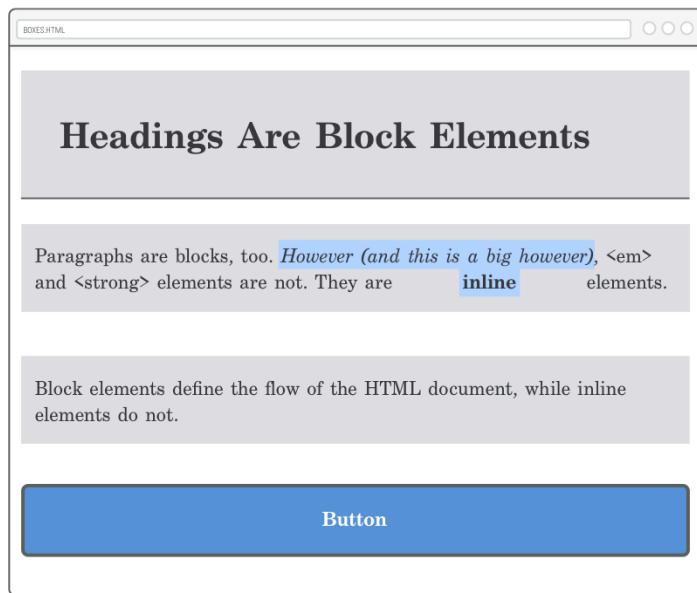


Obviamente, precisamos de uma maneira de selecionar os elementos `<div>` se quisermos que eles tenham alguma utilidade prática para nós. É para isso que servem os seletores de classe, que veremos adiante.

Aqui cabe destacar que a única diferença real entre `<div>` e `<span>` é que o primeiro se destina a conteúdo em nível de bloco, enquanto o segundo se destina a conteúdo inline.

# Dimensões Explícitas

Até agora, deixamos nossos elementos HTML definirem suas dimensões automaticamente. Os preenchimentos, bordas e margens com os quais estivemos brincando envolvem qualquer conteúdo que esteja dentro da caixa do elemento. Se você adicionar mais texto ao nosso elemento `<em>`, tudo será expandido para acomodá-lo:

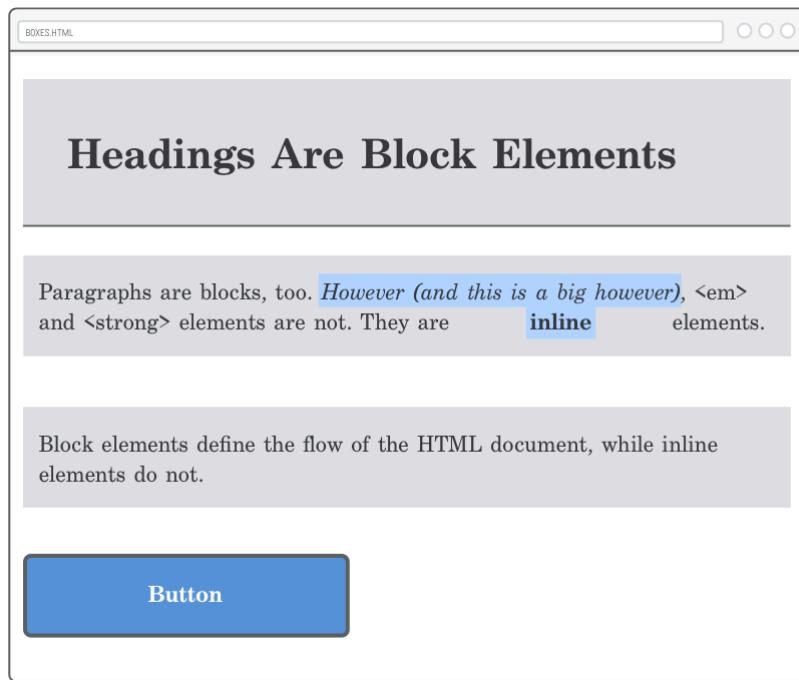


Mas, às vezes, o layout desejado exige uma dimensão explícita, como uma barra lateral com exatamente 250 pixels de largura. Para isso, CSS fornece as propriedades `width` e `height`. Eles têm precedência sobre o tamanho padrão do conteúdo de uma caixa.

Vamos dar ao nosso botão uma largura explícita adicionando a seguinte propriedade a `estilos-de-caixa.css`:

```
div {  
  /* [demais declarações] */  
  width: 200px;  
}
```

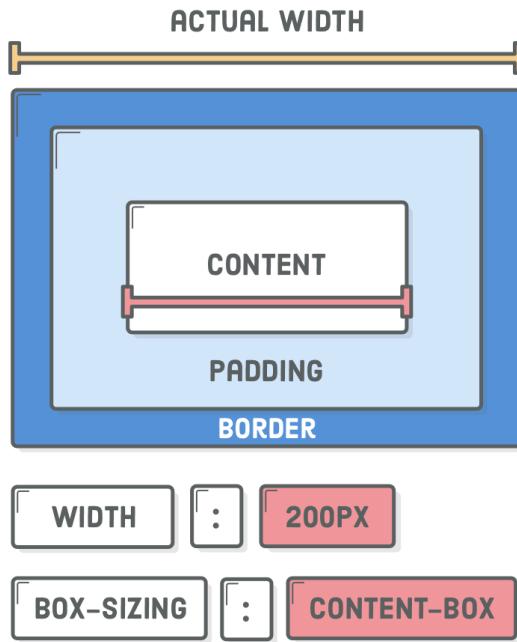
Em vez de ser tão largo quanto a janela do navegador, nosso botão agora tem 200 pixels e abrange o lado esquerdo da página:



Observe também que se você aumentar o título do botão, ele passará automaticamente para a próxima linha e o elemento se expandirá verticalmente para acomodar o novo conteúdo. Você pode alterar esse comportamento padrão com as propriedades [white-space](#) e [overflow](#).

## Caixas de conteúdo e caixas de borda

As propriedades `width` e `height` definem apenas o tamanho do conteúdo de uma caixa. Seu preenchimento e borda são adicionados sobre quaisquer dimensões explícitas que você definir. Isso explica por que você obterá uma imagem com 244 pixels de largura ao tirar uma captura de tela do nosso botão, apesar de ele ter uma propriedade que explicita uma largura de 200 pixels com `width: 200px`.

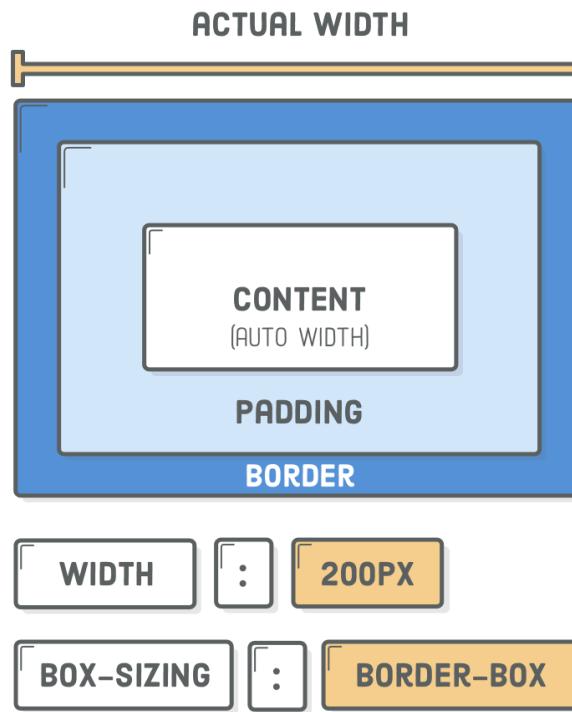


Isso pode ser um pouco contra-intuitivo quando você está tentando criar o layout de uma página. Imagine tentar encher um contêiner de 600px com três caixas que possuem `width: 200px` cada, mas elas não cabem porque todas têm uma 1px borda (fazendo sua largura real 202px).

Felizmente, o CSS permite alterar a forma como a largura de uma caixa é calculada por meio da propriedade `box-sizing`. Por padrão, ele possui um valor `content-box`, o que leva ao comportamento descrito acima. Vamos ver o que acontece quando mudamos para `border-box`:

```
div {
  color: #FFF;
  background-color: #5995DA;
  font-weight: bold;
  padding: 20px;
  text-align: center;
  border: 2px solid #5D6063;
  border-radius: 5px;
  width: 200px;
  box-sizing: border-box; /* Adicione esta linha */
}
```

Isso força a largura real da caixa a ser 200px—incluindo preenchimento e bordas. Claro, isso significa que a largura do conteúdo agora é determinada automaticamente:



Isso é muito mais intuitivo e, como resultado, usar border-box para todas as suas caixas é considerado uma prática recomendada entre os desenvolvedores web modernos.

## Alinhando caixas

Alinhar caixas horizontalmente é uma tarefa comum para desenvolvedores web, e o modelo de caixa oferece várias maneiras de fazer isso. Já vimos a propriedade `text-align`, que alinha o conteúdo e as caixas embutidas dentro de um elemento em nível de bloco. Alinhar caixas de blocos é outra história. Tente adicionar a seguinte regra à nossa folha de estilo:

```
body {  
  text-align: center;  
}
```

Isso apenas alinhará o conteúdo dentro de nossas caixas de blocos – não os próprios blocos. Nosso “botão <div>” ainda está alinhado à esquerda, independentemente do alinhamento do texto.

Existem três métodos para alinhar horizontalmente elementos em nível de bloco:

1. “Margens automáticas” para alinhamento central;
2. “Flutuadores” para alinhamento esquerdo/direito e;
3. “Flexbox” para controle completo sobre o alinhamento.

Sim, infelizmente o alinhamento em nível de bloco não tem nenhuma relação com a propriedade `text-align`.

## Centralizando Caixas com margens automáticas

FLOATS e Flexbox são tópicos modernos e amplamente utilizados para trabalhar com caixas. Contudo, são tópicos complicados aos quais serão dedicados capítulos inteiros. Contudo, já temos o conhecimento necessário para lidar com margens automáticas agora. Quando você define as margens esquerda e direita de um elemento no nível do bloco como `auto`, isso centralizará o bloco em seu elemento pai.

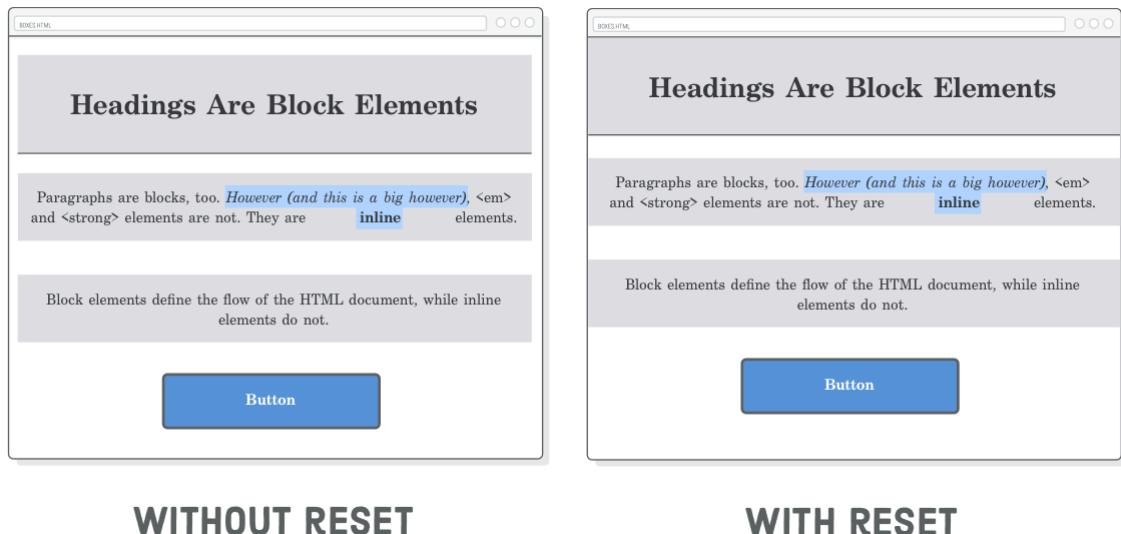
Por exemplo, podemos centralizar nosso botão com o seguinte:

```
div {  
  color: #FFF;  
  background-color: #4A90E2;  
  font-weight: bold;  
  padding: 20px;  
  text-align: center;  
  width: 200px;  
  box-sizing: border-box;  
  margin: 20px auto; /* Vertical Horizontal */  
}
```

Observe que isso funciona apenas em blocos que possuem uma largura explícita definida neles. Remova a linha `width: 200px` e nosso botão terá toda a largura do navegador, tornando o “alinhamento central” sem sentido.

## “Resetando” Estilos

Observe aquela faixa branca ao redor de nossa página? Essa é uma margem/preenchimento padrão adicionado pelo seu navegador. Navegadores diferentes têm estilos padrão diferentes para todos os seus elementos HTML, dificultando a criação de folhas de estilo consistentes.



Geralmente é uma boa ideia substituir os estilos padrão por um valor previsível usando o seletor CSS “universal” (\*). Tente adicionar isto ao topo do nosso arquivo `modelos-de-caixa.css`:

```
* {  
  margin: 0;  
  padding: 0;  
  box-sizing: border-box;  
}
```

Este seletor corresponde a cada elemento HTML, redefinindo efetivamente as propriedades `margin` e `padding` de nossa página da web. Também convertemos todas as nossas caixas para `border-box`, o que, novamente, é uma prática recomendada.

Você encontrará uma redefinição semelhante no topo de quase todas as folhas de estilo CSS globais na web. Eles podem ficar muito mais complicados, mas as três declarações simples mostradas acima nos permitem ajustar com segurança o modelo de caixa CSS para nossos próprios propósitos, sem nos preocupar com interações imprevistas com estilos de navegador padrão.

## Resumo

Aprenderemos mais sobre os usos práticos do modelo de caixa CSS à medida que nos aprofundamos na construção de páginas da web complexas. Por enquanto, pense nisso como uma nova ferramenta em sua caixa de ferramentas CSS. Com alguns conceitos-chave deste capítulo, você se sentirá muito mais preparado para converter um modelo de design em uma página da web real:

- Tudo é uma caixa.
- As caixas podem ser embutidas ou em nível de bloco.
- As caixas possuem conteúdo, preenchimento, bordas e margens.
- Eles também têm regras aparentemente arbitrárias sobre como interagem.
- Dominar o modelo de caixa CSS significa que você pode criar o layout da maioria das páginas da web.

Assim como no último capítulo, as propriedades CSS que acabamos de abordar podem parecer simples — e de certa forma são. Mas, comece a olhar os sites que você visita através das lentes do modelo de caixa CSS e você verá essas coisas literalmente em todos os lugares.

Nossa exploração de caixas genéricas (`<div>` e `<span>`) foi um pouco limitada porque não tínhamos como extrair um elemento HTML individual de nossa página

web. Corrigiremos isso no próximo capítulo com uma discussão mais aprofundada sobre seletores CSS.