Aula 01 - HTML

**Introdução**

Vamos mergulhar no HTML (Linguagem de Marcação de Hipertexto). O HTML é uma linguagem de marcação que fornece um modo estruturado de criar páginas web. Ao usarmos as tags que o HTML nos disponibiliza, podemos controlar como quase tudo é apresentado no navegador do usuário. É o navegador quem interpreta as tags para apresentar o que o usuário final vê.

* [Referência de Elementos/Tags HTML](https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element)
* [Referência do HTML 5](https://developer.mozilla.org/es/docs/Web/HTML/Element)

**O HTML e suas Tags**

O HTML é uma linguagem de marcação interpretada pelo navegador. Ela usa tags ou elementos específicos para dizer ao navegador como as coisas devem ser apresentadas visualmente ao usuário. Vamos dar uma olhada em alguns conceitos por trás de como essas tags funcionam antes de vermos as tags em si.

**Nem todas as tags desses exemplos são tags HTML reais. Elas estão sendo usadas para explicar os conceitos.**

* As tags são colocadas dentro de sinais de maior que (<) e menor que (>). Perceba que a tag de fechamento leva o caractere /. Isto diz ao navegador que esta é a tag de fechamento correspondente. As tags são quase sempre seguidas de tags de fechamento correspondentes.
* <tag>conteúdo</tag>
* As tags também podem ser aninhadas umas dentro das outras.

<div>

<p>abacaxi</p>

</div>

<div>

<p>queijo</p>

<p>pizza</p>

<p>presunto</p>

</div>

- As tags usam o que chamamos de atributos para adicionar funcionalidades ou descrever comportamentos.

- Algumas tags têm fechamento próprio. Essas tags normalmente não envolvem algum conteúdo. A tag `<img>` é uma tag de   
 fechamento próprio bem comum. Como você pode ver, ainda há uma / no final da tag. Observe também o uso dos atributos.

1. <img src="caminho/para/arquivo/de/imagem.jpg" alt="descrição da imagem" />

**Estrutura básica de páginas HTML**

Nesta seção vamos dar uma olhada na estrutura de um arquivo html. Vamos discutir as tags básicas necessárias para ter uma página bem formada para exibição na internet. A partir de agora, vamos falar no contexto do HTML5.

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title></title>
5. </head>
6. <body>
7. </body>
8. </html>

**Vamos dar uma olhada em algumas dessas tags/elementos**

Doctype

* <!DOCTYPE html> informa ao navegador qual versão do HTML será usada neste documento. Neste caso, estamos declarando o HTML5. Doctype é especial no sentido de que ele não é considerado uma tag de verdade, mas sim uma declaração. O que você realmente precisa entender sobre esta tag/declaração é que ela deve estar presente, e deve estar especificamente na primeira linha do seu arquivo HTML.

html

* <html> é a tag que representa o elemento raiz (elemento de nível superior) de um documento HTML, por isso ela também é chamada de elemento raiz. Todos os outros elementos devem ser descendentes deste elemento. Dito isto, observe a tag de fechamento </html> na última linha.

head

* <head> é a tag que contém informações descritivas sobre a sua página, como o título <title> das páginas. Também é aqui onde você insere coisas como <meta> e planilhas de estilo. Falaremos mais sobre essas coisas mais para frente nesse curso. Observe que a tag <title> possui uma tag de fechamento, </title>, assim como a tag <html>. Acredito que você esteja percebendo o padrão.
* [Leitura Recomendada](https://developer.mozilla.org/pt-BR/docs/Learn/HTML/Introduction_to_HTML/The_head_metadata_in_HTML)

title

* <title> é a tag que define o título do documento, ele é exibido na barra de título do navegador ou na aba da página. Ela pode conter apenas texto, e todas as tags colocadas aqui são ignoradas. Ela é seguida da tag de fechamento </title>.

body

* <body> representa o conteúdo de um documento HTML. Pode haver apenas um elemento <body> em um documento HTML. É aqui onde ficam as coisas que queremos que as pessoas vejam. Vou parar de apontar as tags de fechamento. Acho que você já pegou a ideia.

Leitura adicional

* [Introdução ao HTML no MDN](https://developer.mozilla.org/pt-BR/docs/Learn/HTML/Introduction_to_HTML/Getting_started)
* [O que há no head? no MDN](https://developer.mozilla.org/pt-BR/docs/Learn/HTML/Introduction_to_HTML/The_head_metadata_in_HTML)
* [O elemento meta no MDN](https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/meta)
* [Lista de códigos ISO 639-1 no Wikipedia](https://pt.wikipedia.org/wiki/ISO_639)

Aula 02 - HTML Elementos de Bloco e Em Linha

Leitura adicional

* [Elementos inline no MDN](https://curso.kenzie.com.br/modulos)
* [Elementos em nível de bloco no MDN](https://curso.kenzie.com.br/modulos)

**Transcrição do vídeo**

Olá, a todos. Sejam bem-vindos à Kenzie Academy. Meu nome é Davey Strus.

Aninhamos tranquilamente elementos uns dentro dos outros, e até agora tudo certo.

Desde que você feche suas tags adequadamente, você pode colocar os elementos uns dentro dos outros.

<p>

Estou <em>muito</em> entediado!

</p>

Ao fazer isso, o elemento externo é chamado de pai do elemento interno, e o elemento interno é o filho do elemento externo.

Ah, mas e se o filho tiver seus próprios filhos?

<body>

<p>

Vamos ler <cite>Carlos Drummond de Andrade!</cite>

</p>

</body>

O parágrafo é filho do body, e a citação é filha do parágrafo. Portanto qual a relação entre o body e cite? Ué, a cite é o neto do body!

Vamos mais fundo.

<html>

<body>

<p>

Vamos ler <cite>Carlos Drummond de Andrade!</cite>

</p>

</body>

</html>

Termos como bisavô não são muito úteis em termos de marcação. Neste ponto, os chamamos apenas de ancestrais e descendentes. O elemento html é um ancestral de todos os elementos da página, e todos os outros elementos são descendentes do elemento html Falaremos mais sobre isso e sobre a estrutura de árvore que isso cria mais para frente. Isso é tudo que você precisa saber por enquanto.

Então agora sabemos tudo sobre aninhar elementos uns dentro dos outros, e você provavelmente pode aninhar qualquer elemento dentro de outro, certo? Não! Há regras.

Para determinar quais elementos podem ser aninhados dentro de outro, primeiro temos que entender as diferenças entre elementos de nível de bloco e elementos em linha. O HTML 5 reclassificou os elementos HTML usando um sistema muito mais complicado que, sinceramente, não é muito útil no começo, e entender as antigas classificações de nível de bloco e em linha são úteis, então vamos por este caminho para ajudar no aprendizado.

A versão super curta é que os elementos de nível de bloco começam uma nova linha e preenchem toda a largura de seu container. Elementos em linha podem iniciar e terminar no meio de uma linha.

Exemplos de elementos de nível de bloco que você já conhece incluem h1 até h6 e os parágrafos. Os elementos em linha que você já está familiarizado incluem em, strong, cite, i, b, e u.

Mas quais são as restrições de quais elementos podem ser aninhados dentro de outros elementos? Bom, como você deve ter imaginado, você não pode colocar um elemento de nível de bloco dentro de um elemento em linha. Você pode colocar um elemento em linha dentro de outro elemento em linha. E você pode colocar um elemento de nível de bloco dentro de outro elemento de nível de bloco… às vezes. Há algumas exceções bem grandes. Na verdade, você não pode colocar elementos de nível de bloco dentro de um parágrafo ou dentro de títulos, h1 até h6. Isso significa que ainda não falamos sobre nenhum elemento que possa conter outros elementos de nível de bloco.

Conheça o <div>. É uma abreviação de divisão de conteúdo, e é um elemento sem significado semântico. Ele não tem função a não ser servir de container para outros elementos. É um elemento de nível de bloco e pode conter qualquer outro elemento que pertence ao corpo. Não é muito útil até que você comece a mexer com CSS, mas é bom saber que ele existe para aquelas ocasiões onde você precisa de um tipo de container e não há um elemento mais apropriado. Ele é essencialmente o elemento de nível de bloco genérico.

E como você deve estar imaginando, há um equivalente em linha: . Ele não significa nada. É apenas um container em linha genérico. Mas às vezes é disso que você precisa, porque é necessário aplicar algum CSS em linha, e não há um elemento semântico melhor que este.

E é isso. Até a próxima, meu nome é Davey Strus. Boooom desenvolvimento!

Aula 03 - HTML Listas

Leitura adicional

* [Listas em HTML no w3schools](https://www.w3schools.com/html/html_lists.asp)
* [Listas não ordenadas no MDN](https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/ul)
* [Listas ordenadas no MDN](https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/ol)
* [Lista de itens no MDN](https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/li)
* [Lista de itens no MDN](https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/dl)

**Transcrição do vídeo**

Olá, pessoal. Sejam bem-vindos à Kenzie Academy. Meu nome é Davey Strus.

Estamos progredindo bem com o HTML, mas nossas páginas ainda estão um pouco tediosas, apenas com títulos e parágrafos, e com algumas frases em negrito ou itálico. Vamos agitar um pouco as coisas com listas. Um pequeno agito, mas não deixa de ser um agito.

Estamos interessados em dois tipos de listas: Listas não ordenadas e listas ordenadas. Ambas são constituídas de itens de lista. Há um terceiro tipo de lista em HTML chamado de lista de definições, mas ela não é constituída de itens de lista, e geralmente não é tão parecida (nem tão comum) com as outras duas, então iremos focar nas listas não ordenadas e ordenadas por enquanto. Sinta-se à vontade para seguir o link e ler mais sobre listas de definição quando quiser.

Tanto as listas não ordenadas quanto as listas ordenadas são constituídas de itens de lista, e a marcação é virtualmente idêntica. Então qual a diferença? Bem, está no nome. Em uma lista não ordenada, a ordem dos itens não importa. Você pode reorganizar a lista e ela ainda fará sentido. O navegador não irá reorganizá-la para você de maneira inesperada, mas ao decidir qual tipo de lista usar, a questão é: A ordem dos itens é importante? Se sim, então você precisa de uma lista ordenada. Vamos dar uma olhada na marcação.

<h1>Filmes de 1978</h1>

<h2>Filmes de maior bilheteria (EUA)</h2>

<ol>

<li>Grease — Nos Tempos da Brilhantina</li>

<li>Superman</li>

<li>Clube dos Cafajestes</li>

<li>Doido para Brigar... Louco para Amar</li>

<li>O Diabo Disse Não</li>

</ol>

<h2>Indicados para melhor filme</h2>

<ul>

<li>Expresso da Meia-noite</li>

<li>O paraíso Pode Esperar</li>

<li>O Franco Atirador</li>

<li>Amargo Regresso</li>

<li>Uma Mulher Descasada</li>

</ul>

À esquerda, há uma lista ordenada, marcada com o elemento <ol> À direita, há uma lista não ordenada, <ul>. Ambas são constituídas de itens de lista, marcadas com <li>. a lista da esquerda, a ordem é importante. São os dez filmes de maior bilheteria do ano, em ordem. Na lista da direita, a ordem não interessa. São as indicações de melhor filme sem ordem específica.

Então como elas serão apresentadas no navegador?

Bem, devo admitir que eu trapaceei um pouquinho aqui. Eu usei alguns divs e um pouco de CSS para fazer as listas aparecerem lado a lado. As suas aparecerão uma embaixo da outra se você copiar o HTML que eu acabei de mostrar. Isso facilita na hora de encontrar as diferenças. Em suma, a lista ordenada está numerada, e a lista não ordenada está em tópicos. Faz sentido, certo? Já que a ordem faz diferença em uma lista ordenada, a numeração faz sentido. E na lista não ordenada, os pontos de tópico deixam claro que isso é, de fato, uma lista, mesmo que não tenha uma ordem específica. E esta é a diferença básica.

As listas ordenadas também possuem alguns atributos opcionais que as listas não ordenadas não têm. Se você quiser começar uma lista ordenada, digamos, no 6º item, simplesmente adicione start="6".

<h2>Filmes de maior bilheteria (EUA)</h2>

<ol start="6">

<li>Hooper, o Homem das Mil Façanhas</li>

<li>Tubarão 2</li>

<li>A Vingança da Pantera Cor-de-Rosa</li>

<li>O Franco Atirador</li>

<li>Halloween - O Início</li>

</ol>

Quer listar as coisas em ordem invertida? Use o atributo booleano reversed.

<h2>Top 5</h2>

<ol reversed>

<li>O Paraíso Pode Esperar</li>

<li>Doido para Brigar... Louco para Amar</li>

<li>Clube dos Cafajestes</li>

<li>Superman</li>

<li>Grease — Nos Tempos da Brilhantina</li>

</ol>

Lembra dos atributos booleanos? Os valores deles são opcionais. Se você incluir o valor, ele deve ser o mesmo do nome do atributo. Neste caso, reversed="reversed".

<h2>Top 5</h2>

<ol reversed="reversed">

<li>O Paraíso Pode Esperar</li>

<li>Doido para Brigar... Louco para Amar</li>

<li>Clube dos Cafajestes</li>

<li>Superman</li>

<li>Grease — Nos Tempos da Brilhantina</li>

</ol>

Você também pode mudar a numeração para usar letras maiúsculas ou minúsculas, ou algarismos romanos, usando o atributo type.

<h1>Review de Roger Ebert's sobre Grease — Nos Tempos da Brilhantina</h1>

<ol type="I">

<li>Introdução</li>

<li>Elenco</li>

<li>Resumo do enredo</li>

<li>Pontos Fortes</li>

<li>Pontos Fracos</li>

<li>Conclusão</li>

</ol>

Coloque type="I" para algarismos romanos maiúsculos, "i" para algarismos romanos minúsculos, "A" para letras maiúsculas, "a" para letras minúsculas, ou "1" para números, que é o padrão.

Portanto as listas ordenadas possuem alguns atributos que podemos usar: start, reversed, e type. As listas não ordenadas não possuem atributos que são parte da especificação atual. Se você quiser exibi-las de maneira diferente, você precisará usar CSS.

E quanto aos itens de lista? Eles possuem um atributo muito útil: value, que funciona apenas em listas ordenadas. Ele especifica o valor de um item de lista específico, e depois dele a lista continuará sendo numerada automaticamente. Isso é útil se houver um empate, por exemplo.

<h1>Filmes com a maioria dos prêmios da Academia</h1>

<ol>

<li>Ben-Hur (empate)</li>

<li value="1">Titanic (empate)</li>

<li value="1">O Retorno do Rei (empate)</li>

<li value="4">Amor, Sublime Amor</li>

</ol>

E adivinha só. Você pode aninhar listas dentro de listas! Eita! Simplesmente inclua outra lista dentro de um item de lista antes de fechar a tag.

<h1>Review de Roger Ebert's sobre Grease — Nos Tempos da Brilhantina</h1>

<ol type="I">

<li>Introdução</li>

<li>Elenco

<ul>

<li>Olivia Newton-John</li>

<li>John Travolta</li>

</ul>

</li>

<li>Resumo do enredo</li>

<li>Pontos Fortes</li>

<li>Pontos Fracos</li>

<li>Conclusão</li>

</ol>

Aqui eu aninhei uma lista não ordenada dentro de uma lista ordenada. É possível fazer isso e vice-versa. Naturalmente, você também pode aninhar uma lista dentro de outra lista do mesmo tipo.

Na verdade, você pode colocar quase qualquer outro tipo de elemento dentro de um item de lista: parágrafos, títulos, divs… como quiser! E claro, você pode colocar elementos em linha também.

<h1>Review de Roger Ebert's sobre Grease — Nos Tempos da Brilhantina</h1>

<ol type="I">

<li>Introdução

<p><cite>Grease — Nos Tempos da Brilhantina</cite> vale a pena ver pela nostalfia, mas é apenas um musical mediano.</p>

</li>

<li>Elenco</li>

<li>Resumo do enredo</li>

<li>Pontos Fortes</li>

<li>Pontos Fracos</li>

<li>Conclusão</li>

</ol>

E aí, você está pronto para temperar um pouco suas páginas com listas? Sim, é claro que sim, seu fazedor de listas! Até a próxima, meu nome é Davey Strus. Booooas codificações!

Aula 04 - HTML Hyperlinks

**Leitura adicional**

* [HTML tag <a>](https://www.w3schools.com/tags/tag_a.asp)on w3schools
* [O elemento âncora](https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/a) on MDN
* [sobre rel=noopener](https://mathiasbynens.github.io/rel-noopener/) by Mathias Bynens

**Transcrição do vídeo**

Olá, pessoal. Sejam bem-vindos à Kenzie Academy. Meu nome é Davey Strus.

Aprendemos como fazer todo tipo de coisa chique com textos, mas ainda é só texto. Queremos *hipertexto*! Para isso, precisamos de **hiperlinks**.

Um hiperlink, ou apenas *link*, se você preferir assim, nos envia para outro lugar na internet. Pode ser outra página do mesmo site, outro site, ou para uma parte diferente da página onde estamos. Pode até mesmo ser um link para baixar um arquivo. Se o objeto tiver uma URL, você pode criar um link até ele.

Você faz isso com o elemento <a>.

<a>

É um a de *âncora*, apesar de seu uso como âncora agora ser obsoleto. Hoje em dia, ele é usado apenas para links. O atributo crucial aqui é o href.

<a href="about.html">Sobre nós</a>

Quando um href estiver presente, o elemento se torna um hiperlink, e o conteúdo entre as tags de abertura e de fechamento podem ser ativados para seguir o link. O href aponta para uma URL, ou para um fragmento de URL.

Um fragmento de URL começa com uma cerquilha (#) e aponta para um elemento específico da página atual, usando o id do elemento. Você pode dar um id a qualquer elemento usando, com antecedência suficiente, o atributo id.

<a href="#products">Produtos</a>

<h2 id="products">Nossos produtos</h2>

Este link aponta para aquele h2, pois o nome que segue a cerquilha do fragmento de URL bate com o id do h2. Ao ativar aquele link, o navegador irá rolar a página até onde o h2 está. Você também pode usar um fragmento no final de uma URL para linkar a um id específico de outra página.

<a href="about.html#founders">

Fundadores

</a>

Este link nos enviaria para o about.html e em seguida iria rolar a nova página até o elemento com o id "founders".

Você também pode linkar ao topo da página atual usando #top ou apenas #. Você não precisar ter um elemento com o id "top" para que isso funcione.

O conteúdo de um link, ou seja, as coisas que aparecem entre as tags de abertura e fechamento, e as coisas que se tornam clicáveis, podem ser praticamente qualquer coisa, exceto outro link. Nada de links dentro de links. Você pode pensar em links como elementos em linha. Como mencionei antes, o HTML redefiniu os tipos de conteúdo, pois precisávamos de mais confusão em nossas vidas, mas os links não inserem novas linhas automaticamente nem preenchem a largura de seus containers por padrão. Portanto, eles agem como elementos em linha. Apesar disso, você pode colocar um elemento de nível de bloco dentro de um link, por mais esquisito que isso se pareça para caretões como eu, que se lembram de quando não se podia fazer isso.

<h1>

<a href="https://kenzie.com.br/">Kenzie Academy Brasil</a>

</h1>

<a href="https://kenzie.com.br/">

<h1>Kenzie Academy Brasil</h1>

</a>

Aqui, fiz das duas formas. Aninhei um link dentro de um h1 e um h1 dentro de um link. Qual a diferença prática?

No primeiro link, apenas o texto é clicável. Observe o ponteiro do mouse. Eu preciso colocá-lo diretamente sobre a palavra. No segundo link, todo o h1, e não apenas o texto dentro do h1, é clicável. Já que o h1 é um elemento de nível de bloco, ele ocupa toda a largura. Então todo o espaço à direita da palavra ainda faz parte do h1 e é clicável.

Os links têm vários atributos além do href, mas o que você provavelmente irá usar mais é o target. A maioria dos valores possíveis para o target tem a ver com frames, os quais não mexeremos agora. Mas o uso mais comum do target é o de dar o valor \_blank para abrir um link em uma nova aba ou janela.

<a href="https://kenzie.com.br/" target="\_blank">

Kenzie Academy Brasil

</a>

Isso é comum ao linkar a recursos externos, tipo ao linkar a outro site, sob certas circunstâncias em que você acredita que o usuário provavelmente irá desejar manter seu site aberto. Por exemplo, usamos isso nas páginas deste curso ao linkar a um recurso externo, pois imaginamos que você provavelmente quer manter a aula aberta enquanto lê outra coisa. Mas cuidado para não exagerar com o target="\_blank". Ele está fazendo uma suposição sobre a intenção do usuário, e ao passo que ele sempre pode clicar com o botão direito ou com a tecla CTRL pressionada para abrir qualquer link em uma nova aba, se você marcá-lo com target="\_blank", o usuário não terá escolha a não ser abri-lo em uma nova aba. Eles não têm a opção de ignorar este comportamento.

Ao usar target="\_blank", é uma boa ideia adicionar rel="noopener" ao link.

<a

href="https://kenzie.com.br/"

target="\_blank"

rel="noopener"

>

Kenzie Academy Brasil

</a>

O atributo rel especifica a relação entre a página atual e a página para qual o link está apontando. Você pode não usá-lo muito, e deve sentir-se à vontade para ler mais sobre isso por conta própria; mas neste caso, você pode usar isso para lidar com um possível problema de segurança ao usar target"\_blank". Você pode ler mais sobre a natureza exata deste problema de segurança seguindo um link nesta aula. Não vou falar sobre isso neste vídeo.

Você deve ter notado que eu coloquei algumas linhas em branco no meio da tag de abertura aqui, pois o número de atributos estava tornando a tag muito grande. Isso é totalmente válido. Se você fizer isso, é uma boa ideia indentar seus atributos e alinhá-los verticalmente, como eu fiz, para mantê-los agradáveis de ler. Em seguida, eu alinhei o símbolo de fechamento com o símbolo de abertura para ficar fácil identificar onde a tag de abertura começa e termina.

Agora você tem o poder de tornar seu texto bom e hiper. Use este novo poder com responsabilidade. Até a próxima, meu nome é Davey Strus. Cá entre nós, eu provavelmente continuarei sendo o Davey Strus até mesmo depois da próxima. Mas até lá, booooas codificações!

Aula 05 - HTML Incorporando imagens

**Leitura adicional**

* [Imagens no HTML](https://developer.mozilla.org/pt-BR/docs/Aprender/HTML/Multimedia_and_embedding/Images_in_HTML) no MDN
* [O elemento de incorporação de imagem](https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/img) no MDN
* [Guia de tipo e formato de arquivo de imagem](https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Image_types) no MDN (em inglês)
* [HTML images](https://www.w3schools.com/html/html_images.asp) no w3schools (em inglês)
* [HTML <img> tag](https://www.w3schools.com/tags/tag_img.asp) no w3schools (em inglês)

**Transcrição do vídeo**

Olá, pessoal. Sejam bem-vindos à Kenzie Academy. Meu nome é Davey Strus.

Vamos falar de imagens. É claro que HTML significa Linguagem de Marcação de Hipertextos, mas nossas páginas não estão limitadas a apenas textos. Nós podemos inserir outras mídias em nossas páginas. E qual o exemplo mais simples? Uma simples imagem. Para inserir uma imagem, use o elemento de inserção de imagem, <img>.

<img>

O <img> é diferente de todos os outros elementos que usamos até agora no corpo, no sentido de que é um **elemento vazio**. Quer dizer, ele não tem tag de fechamento. Usamos outro elemento vazio no cabeçalho: <meta>. Mas esta é a primeira vez que usamos um no corpo. E a tag de fechamento não é meramente *opcional*, ela é *proibida*. Ele simplesmente não possui tag de fechamento.

Contudo, ele possui atributos. Primeiro, o atributo source, src.

<img src="davey.png">

É aqui que você coloca a URL da imagem. Facinho. Falaremos mais sobre formatos de imagem depois, mas de longe os formatos de imagem mais comuns que você encontrará na web são JPGs, GIFs e PNGs.

Em seguida, temos o atributo alt.

<img src="davey.png" alt="Davey Strus">

O atributo alt fornece uma alternativa textual, caso o usuário esteja impossibilitado de ver a imagem por algum motivo. Pode ser porque está usando um leitor de tela, ou porque a URL está errada, ou porque a conexão está muito lenta. Seja como for, o valor aqui deve ser o texto que você gostaria que aparecesse caso a imagem não possa ser vista. Não seja preguiçoso de colocar o nome do arquivo de imagem ou algo assim. Isso não ajuda muito. Pense sobre o que a imagem está tentando comunicar e converta isso em palavras.

<img src="decoration.gif" alt="">

Se a imagem for puramente decorativa, você pode dar um valor vazio ao atributo alt. Os leitores de tela irão ignorá-lo e as pessoas com conexões lentas simplesmente não verão nada. Isso é provavelmente mais útil do que ver ou ouvir as palavras "fronteira florida" ou algo do tipo.

Entretanto, você não deveria deixar o atributo alt totalmente desligado. Coloque algo útil nele ou aponte-o para uma string vazia se ele for puramente decorativo. Mais especificamente, a especificação do HTML 5 permite que o alt seja omitido sob certas circunstâncias, mas isso tem a ver com ferramentas que geram HTML automaticamente e coisas do tipo, e não com cenários que você mesmo encontrará ao escrever HTML. Você deveria considerar o alt como algo obrigatório.

Você também pode especificar a largura e a altura da imagem usando esses atributos.

<img

src="davey.png"

alt="Davey Strus"

width="600"

height="600"

>

Espera-se que os valores numéricos que você usar aqui sejam em pixels. Por padrão, a imagem aparecerá no tamanho que ela de fato é. Você não precisa dizer isso ao navegador. Ele irá entender. A vantagem de especificar a largura e altura é que um placeholder do tamanho correto irá aparecer enquanto a imagem está carregando. Sem isso, você verá o conteúdo mudando de lugar na página enquanto a imagem carrega. Não é o fim do mundo, mas não é o adequado.

Isso é o básico. E como sempre, sinta-se à vontade para seguir os links nesta aula para aprender mais. Até a próxima, meu nome é Davey Strus. Booooas codificações!

Aula 06 - HTML Títulos

Os elementos h1 - h6 são títulos.

Esses elementos têm uma classificação dada pelo número em seu nome. O <h1> elemento é dito ter o posto mais alto, o <h6> elemento tem o menor grau, e dois elementos com o mesmo nome têm o mesmo valor.

<h1>This is heading 1</h1>

<h2>Esse é um cabeçalho 2</h2>

<h3>Esse é um cabeçalho 3</h3>

<h4>Esse é um cabeçalho 4</h4>

<h5>Esse é um cabeçalho 5</h5>

<h6>Esse é um cabeçalho 6</h6>

* O código a seguir mostra alguns títulos com algum conteúdo abaixo deles.

<h1>Elementos de cabeçalho</h1>

<h2>Sumário</h2>

<p>Algum texto aqui...</p>

<h2>Exemplos</h2>

<h3>Exemplo 1</h3>

<p>Algum texto aqui...</p>

<h3>Exemplo 2</h3>

<p>Algum texto aqui...</p>

<h2>Veja também</h2>

<p>Algum texto aqui...</p>

**Questões de acessibilidade**

Uma técnica de navegação comum para usuários de software de leitura de tela é pular de um título para outro para determinar rapidamente o conteúdo da página. Por isso, é importante não pular um ou mais níveis de título. Fazer isso pode criar confusão, pois a pessoa que está navegando dessa forma pode ficar se perguntando onde está o rumo que está faltando.

**Não fazer**

<h1>Esse é um cabeçalho 1</h1>

<h3>Esse é um cabeçalho 3</h3>

<h4>Esse é um cabeçalho 4</h4>

**Fazer**

<h1>Esse é um cabeçalho 1</h1>

<h2>Esse é um cabeçalho 2</h2>

<h3>Esse é um cabeçalho 3</h3>

Os títulos podem ser aninhados como subseções para refletir a organização do conteúdo da página. A maioria dos leitores de tela também pode gerar uma lista ordenada de todos os títulos em uma página, o que pode ajudar uma pessoa a determinar rapidamente a hierarquia do conteúdo:

1. **h1** Besouros
   1. **h2** Etimologia
   2. **h2** Distribuição e Diversidade
   3. **h2** Evolução
      1. **h3** Paleozóico tardio
      2. **h3** jurássico
      3. **h3** Cretáceo
      4. **h3** Cenozóico
   4. **h2** Morfologia Externa
      1. **h3** Cabeça
         1. **h4** Partes da boca
      2. **h3** Tórax
         1. **h4** Protórax
         2. **h4** Pterotórax
      3. **h3** Pernas
      4. **h3** Asas
      5. **h3** Abdômen

Quando os títulos são aninhados, os níveis dos títulos podem ser "ignorados" ao fechar uma subseção.

**Exemplos:**

* [Tags de título w3schools](https://curso.kenzie.com.br/w3schools.com/tags/tag_hn.asp)
* [Exemplo](https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_headers)

# Aula 07 - HTML Tags de divisão de conteúdo

## Tag article

A tag <article> representa uma composição completa ou independente em uma página. Pode ser uma postagem num fórum, um artigo de revista ou jornal, um post de um blog ou mesmo um comentário de um usuário.

Quando as tags <article> são aninhados, os elementos internos representam artigos que estão relacionados ao conteúdo do artigo externo. Por exemplo, os comentários do post de um blog, podem ser implementados com a tag article.

### Este exemplo mostra uma postagem de blog usando a tag article:

<article>

<header>

<h1>A Primeira Regra da Vida</h1>

<p><time>3 dias atrás</time></p>

</header>

<p>Se houver um microfone perto de você, assuma-o</p>

<footer>

<a href="#">Mostrar comentários...</a>

</footer>

</article>

Você pode ver um pouco mais sobre a tag <article> nesse vídeo:

[Tag article do HTML 5](https://www.youtube.com/watch?v=rUmkpuRNTgM)

## Tag section

A tag section representa uma seção genérica de um página. Uma seção, neste contexto, é um agrupamento temático de conteúdo, por exemplo, uma seção de serviços.

**Exemplos**

Exemplos de seções, seriam capítulos de um livro, ou seções numeradas de uma tese. Uma página de web pode ser dividida em seções, como por exemplo, uma para a introdução, uma para posts e outra para informações de contato.

No exemplo a seguir, vemos um artigo(parte de uma página da web) sobre maçãs, contendo duas seções curtas.

<article>

<p>A maçã é o fruto da macieira.</p>

<section>

<h1>Maça vermelha</h1>

<p>Essas maças vermelhas brilhantes são as mais comumente encontradas na maioria dos supermercados.</p>

</section>

<section>

<h1>Maça-verde</h1>

<p>Essas maças verdes e suculentas fazem um ótimo recheio para tortas de maça.</p>

</section>

</article>

Aqui está um programa de graduação com duas seções, uma para a lista de formandos e outra para a descrição da cerimônia.

<!DOCTYPE Html>

<html Lang=En>

<head>

<title>Cerimônia de Graduação Verão 2022</title>

</head>

<body>

<h1>Graduação</h1>

<section>

<h1>Cerimônia</h1>

<p>Procissão de Abertura</p>

<p>Discurso do Orador da Turma</p>

<p>Discurso do Presidente da Turma</p>

<p>Apresentação dos Diplomas</p>

<p>Discurso de Fechamento do Diretor</p>

</section>

<section>

<h1>Graduados</h1>

<ul>

<li>Molly Carpenter</li>

<li>Anastasia Luccio</li>

<li>Ebenezar McCoy</li>

<li>Karrin Murphy</li>

<li>Thomas Raith</li>

<li>Susan Rodriguez</li>

</ul>

</section>

</body>

</html>

### Exemplo de uso simples

**Antes**

<div>

<h2>Cabeçalho</h2>

<p>Amontoado de conteúdos maneiros</p>

</div>

**Depois**

<section>

<h2>Cabeçalho</h2>

<p>Amontoado de conteúdos maneiros</p>

</section>

Conforme mencionado acima, <section> é um elemento de corte genérico e só deve ser usado se não houver um elemento mais específico para representá-lo. Por exemplo, um menu de navegação deve ser envolvido em uma tag <nav>, mas uma lista de resultados de pesquisa ou uma exibição de mapa e seus controles não possuem elementos específicos e podem ser colocados dentro de uma <section>.

**Considere também estes casos:**

Se o conteúdo do elemento representa uma unidade atômica autônoma de conteúdo como, por exemplo, uma postagem de blog, um comentário ou um artigo de jornal), o <article> seria uma escolha melhor.

Você pode ver um pouco mais sobre a tag <section> nesse vídeo:

[Tag section do HTML 5](https://www.youtube.com/watch?v=Y_lv0Zl09Eo)

#### **Acesse o**[**exercício**](https://codepen.io/KenzieAcademyBR/pen/QWGRmmQ)

# Aula 01 - Introdução ao CSS

Neste módulo, você irá aprender a usar o CSS para manipular a aparência do conteúdo das suas páginas web.

## O que realmente é CSS?

CSS (Folha de Estilo em Cascata) é o código que você usa para dar estilo à sua página.

O CSS não é uma linguagem de programação mas sim uma linguagem de estilização. Isso significa que o CSS permite aplicar estilos seletivamente a elementos em documentos HTML. Por exemplo, para selecionar todos os elementos parágrafo de uma página HTML e tornar o texto dentro deles vermelho, você escreveria este CSS:

p {

color: red;

}

### Anatomia de um conjunto de regras CSS

Vamos dar uma olhada no CSS acima com mais detalhes:



Toda essa estrutura é chamada de conjunto de regras, mas geralmente usamos o termo "regra", por ser mais curto. Note os nomes das partes individuais:

#### **Seletor(Selector)**

É, nesse caso, o nome do elemento HTML. Ele seleciona o(s) elemento(s) a serem estilizados(elementos <p>). Para dar estilo a um outro elemento, é só mudar o seletor.

#### **Propriedades(Property)**

Forma pela qual você altera um determinado estilo de um elemento HTML. No CSS, você escolhe quais propriedades você deseja alterar.

#### **Valor da propriedade(Property value)**

À direita da propriedade, depois dos dois pontos, nós temos o valor de propriedade, que escolhe uma dentre muitas aparências possíveis para uma determinada propriedade.

#### **Declaração(Declaration)**

Uma regra simples como color: red; especificando quais das propriedades do elemento você quer estilizar.

Note outras partes importantes da sintaxe:

* Cada linha de comando deve ser envolvida em chaves({})
* Dentro de cada declaração, você deve usar dois pontos(:) para separar a propriedade de seus valores
* Dentro de cada conjunto de regras, você deve usar um ponto e vírgula(;) para separar cada declaração da próxima

Então para modificar múltiplos valores de propriedades de uma vez, você deve escrevê-los separados por ponto e vírgula, desse modo:

p {

color: red;

width: 500px;

border: 1px solid black;

}

### Selecionando múltiplos elementos

Você também pode selecionar vários tipos de elementos e aplicar um único conjunto de regras a todos eles. Inclua múltiplos seletores separados por vírgulas. Por exemplo:

p, li, h1 {

color: red;

}

## Para saber mais

* Leia: [O que realmente é CSS?](https://developer.mozilla.org/pt-BR/docs/Aprender/Getting_started_with_the_web/CSS_basico)
* Leia: [Como CSS é estruturado](https://developer.mozilla.org/pt-BR/docs/Learn/CSS/First_steps/Como_CSS_e_estruturado)

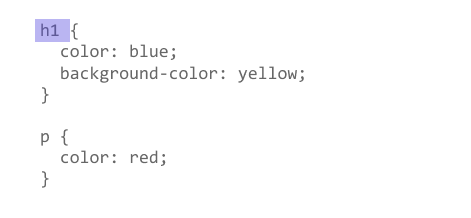
# Aula 02 - Seletores no CSS

Neste módulo, nós vamos nos aprofundar um pouco mais nos seletores de CSS.

## Seletores CSS

Em CSS, os seletores são usados ​​para direcionar os elementos HTML em nossas páginas da web que queremos estilizar. Há uma grande variedade de seletores CSS disponíveis, permitindo uma precisão refinada ao selecionar os elementos a serem estilizados.

Um seletor CSS é a primeira parte de uma regra CSS. É um padrão de elementos e outros termos que informam ao navegador quais elementos HTML devem ser selecionados para que os valores de propriedade dentro da regra sejam aplicados a eles.



## Tipos de seletores

Existem alguns agrupamentos diferentes de seletores e saber qual tipo de seletor você pode precisar o ajudará a encontrar a ferramenta certa para o trabalho.

### Seletores por nome de elemento

Seleciona elementos através do seu nome:

/\* Todos os elementos <a> \*/

a {

color: red;

}

### Seletores por classe

Perceba que para selecionar um elemento através da sua classe, você precisa adidionar um ponto(.) antes do nome da classe:

.titulo{

/\* declarações aqui \*/

}

### Seletores por ID

Assim como na classe você precisa de um ponto, para utilizar o seletor por ID, você precisa adicionar uma cerquilha(#) antes do nome do ID:

#tituloUnico{

/\* declarações aqui \*/

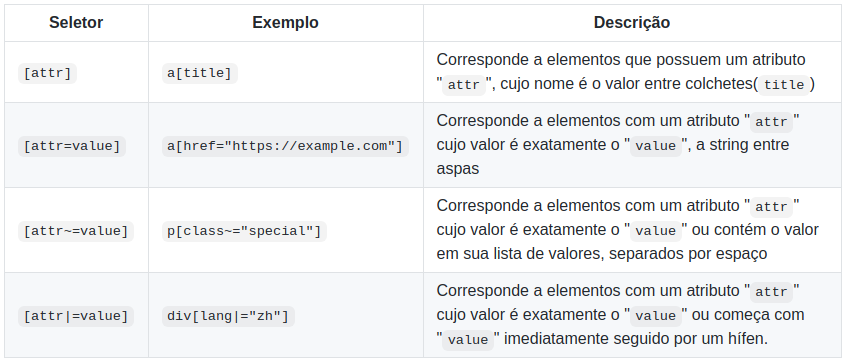
}

### Seletores por atributo

Os elementos HTML podem ter atributos que fornecem mais detalhes sobre o elemento que está sendo marcado. Em CSS, você pode usar seletores de atributo para estilizar elementos com determinados atributos. Esta lição mostrará como usar esses seletores.

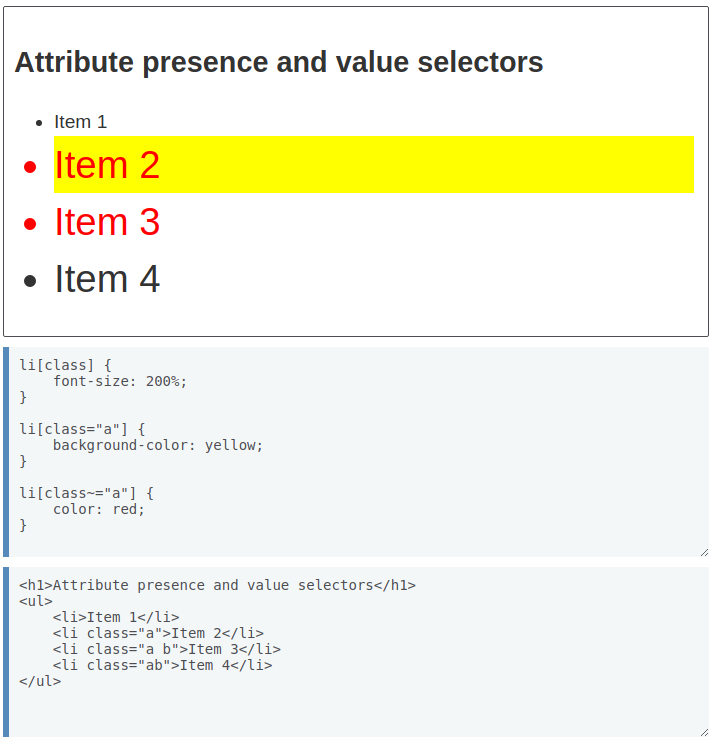
#### **Seletores de presença e valor**

Esses seletores permitem a seleção de um elemento com base na presença de um atributo sozinho(por exemplo, href) ou mesmo através do atributo juntamente com o valor desse.



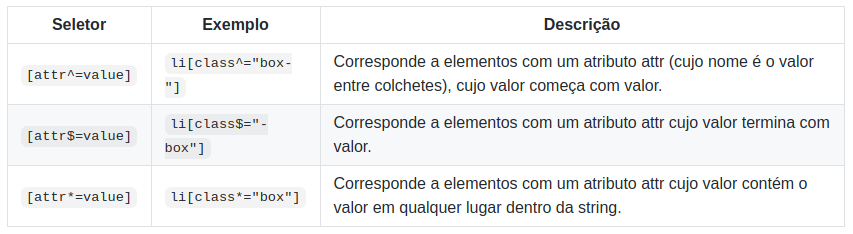
No exemplo abaixo você pode ver esses seletores sendo usados:

* Usando li[class] podemos combinar qualquer seletor com um atributo de classe. Isso corresponde a todos os itens da lista, exceto o primeiro.
* li[class="a"] corresponde a um seletor com uma classe de a, mas não a um seletor com uma classe de a com outra classe separada por espaço como parte do valor. Ele seleciona o segundo item da lista.
* li[class~="a"] corresponderá a uma classe de a, mas também a um valor que contém a classe de a como parte de uma lista separada por espaços em branco. Ele seleciona o segundo e o terceiro itens da lista.



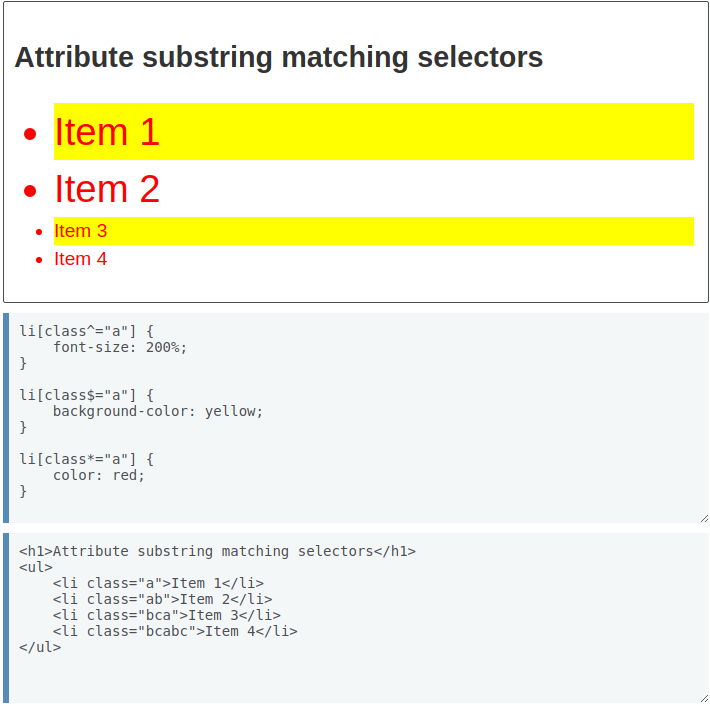
#### **Seletores de substring**

Esses seletores permitem uma correspondência mais avançada de substrings dentro do valor do seu atributo. Por exemplo, se você tivesse classes de box-warning e box-error e quisesse combinar tudo que começou com a string "box-", você poderia usar [class^="box-"] para selecionar os dois(ou [class|="box"] como descrito abaixo)



O próximo exemplo mostra o uso desses seletores:

* li[class^="a"] corresponde a qualquer valor de atributo que comece com a, portanto, corresponde aos dois primeiros itens da lista.
* li[class$="a"] corresponde a qualquer valor de atributo que termina com a, portanto, corresponde ao primeiro e terceiro item da lista.
* li[class\*="a"] corresponde a qualquer valor de atributo onde a apareça em qualquer lugar na string, portanto, corresponde a todos os itens da nossa lista



#### **Seletores de substring**

Se você deseja combinar valores de atributo sem distinção entre maiúsculas e minúsculas, você pode usar o valor i antes do colchete de fechamento. Este sinalizador informa ao navegador para corresponder caracteres ASCII sem distinção entre maiúsculas e minúsculas. Sem o sinalizador, os valores serão correspondidos de acordo com a distinção entre maiúsculas e minúsculas do idioma do documento - no caso do HTML, será sensível a maiúsculas e minúsculas.

No exemplo abaixo, o primeiro seletor corresponderá a um valor que começa com um - ele corresponde apenas ao primeiro item da lista porque os outros dois itens da lista começam com um A maiúsculo. O segundo seletor usa o sinalizador que não diferencia maiúsculas de minúsculas e, portanto, corresponde a todos os itens da lista.



## Para saber mais

* Leia: [Seletores CSS](https://developer.mozilla.org/pt-BR/docs/Learn/CSS/Building_blocks/Selectors)

Aula 03 - Unidades de medida

**Videoaula Kenzie Academy**

Para o seguinte vídeo, vocês vai precisar de cerca de 25 minutos

**Visão geral**

CSS tem várias opções de unidades para se usar ao determinar o tamanho de alguma propriedade no CSS. Aprender todas as suas opções para unidades CSS pode ser fundamental para estilizar de uma melhor forma e o layout tenha uma ótima aparência em qualquer tela.

**O que é uma unidade de medida?**

Uma unidade CSS determina o tamanho de uma propriedade que você está definindo para um elemento ou conteúdo. Por exemplo, se você deseja definir a margem de propriedade de um parágrafo, você deve dar a ela um valor específico. Este valor inclui a unidade CSS.

Vamos ver um exemplo:

p {

margin: 20px;

}

Nesse caso, margin é uma propriedade, e 20px é o valor, onde px é a unidade de medida.

Embora seja comum ver unidades como o px sendo usadas, a grande questão é: "Qual é a melhor unidade para se usar aqui?"

Aqui estão algumas considerações a serem feitas ao escolher um tipo de unidade e exemplos de casos de uso:

**Unidades absolutas vs. Unidades relativas**

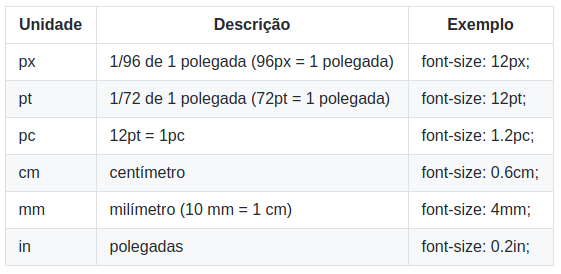
Ao considerar todas as opções para quais unidades usar, é importante considerar as duas categorias de unidades: absolutas e relativas.

Unidades absolutas

As unidades que são “absolutas” têm o mesmo tamanho, independentemente do elemento pai ou do tamanho da janela. Isso significa que uma propriedade definida com um valor que tem uma unidade absoluta terá esse tamanho quando vista em um dispositivo móvel ou computador.

As unidades absolutas podem ser úteis ao trabalhar em um projeto onde a capacidade de resposta não está sendo considerada. Por exemplo, aplicativos de desktop que não podem ser redimensionados podem ser estilizados para as dimensões padrão. Se a janela não escalar, você também não precisa que o conteúdo escale.

**Dica: unidades absolutas podem ser menos favoráveis ​​para sites responsivos porque não escalam quando o tamanho da tela muda.**



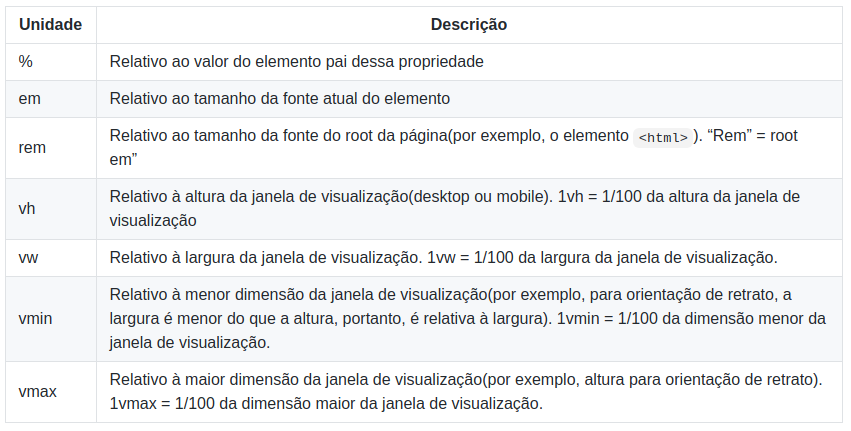
Pixels (px) são normalmente a unidade absoluta mais popular para telas. Centímetros, milímetros e polegadas são mais comuns para impressão e você pode nem saber que eram opções!

Unidades relativas

As unidades relativas são úteis para estilizar sites responsivos porque são dimensionadas em relação ao pai ou ao tamanho da janela(dependendo da unidade).

**Como regra geral, as unidades relativas podem ser usadas como padrão para sites responsivos. Isso pode ajudá-lo a evitar a necessidade de atualizar estilos para diferentes tamanhos de tela.**

As unidades relativas podem ser um pouco mais difíceis do que as unidades absolutas para determinar qual usar, então vamos examinar suas opções em detalhes.



Nem sempre está claro qual dessas opções é melhor usar para cada tipo de propriedade CSS. Por exemplo, % é geralmente mais apropriado para propriedades relacionadas ao layout, como largura, do que para tamanho da fonte.

Aqui estão alguns exemplos de quando você usaria cada unidade relativa:

* %: Você deseja que um elemento filho tenha 10% da largura do pai como margem para que nunca preencha todo o elemento pai. Se o tamanho do pai mudar, a margem também será atualizada.

.filho {

margin: 10%;

}

* em: Você deseja que a fonte de um elemento filho tenha metade do tamanho da fonte de seu pai(por exemplo, o parágrafo sob o título de uma seção).

.filho {

font-size: 0.5em;

}

* rem: O tamanho da fonte deve ser o dobro do tamanho da fonte do elemento raiz. Pode ser assim que você dimensiona seus cabeçalhos, porque todos eles devem ter o mesmo tamanho, independentemente do contêiner pai.

header {

font-size: 2rem;

}

* vh: Sua página deve sempre ter a altura da janela de visualização.

body {

height: 100vh;

}

* vw: Você tem uma seção com texto que deve ter a metade da largura da janela de visualização.

section {

width: 50vw;

}

* vmin: Você tem uma imagem que deve ser sempre tão larga quanto a menor dimensão da janela de visualização. Em um dispositivo móvel no modo retrato, a imagem será tão ampla quanto a largura da janela de visualização.

img {

width: 100vmin;

}

* vmax: Você não se importa se uma imagem for cortada porque você deseja que ela preencha completamente a dimensão maior da janela de visualização. Por exemplo, se a imagem de um padrão for usada como plano de fundo.

img {

width: 100vmax;

}

No geral, quando e como você escolhe suas unidades CSS se resumem a algumas perguntas:

* Eu quero que o meu estilo seja dimensionado quando o tamanho da janela de visualização mudar?
* Se eu quiser que ele seja escalonado, ao que eu quero que ele seja relativo?

Depois de responder a essas perguntas, é muito mais fácil definir qual unidade usar!

**Para saber mais**

* Leia: [Conheça as Unidades de CSS](https://www.turbosite.com.br/blog/conheca-as-unidades-de-css-css-rem-vh-vw-vmin-vmax-ex-ch/)

Aula 04 - Aplicando cor a elementos com CSS

Nessa aula vamos ver um pouco sobre as cores no css, propriedades como color, background, que podem ser útil nas pŕoximas atividades, e também páginas para seleção de paleta de cores.

**Coisas que podem ter cor**

No nível do elemento, tudo em HTML pode ter uma cor aplicada a ele. Em vez disso, vamos olhar para as coisas em termos dos tipos de coisas desenhadas nos elementos, como texto e bordas e assim por diante. Para cada um, veremos uma lista das propriedades CSS que aplicam cores a eles.

**Propriedades**

Sempre que um elemento é renderizado, essas propriedades são usadas para determinar a cor do texto, seu plano de fundo e quaisquer decorações no texto.

color

Propriedade utilizada para definir a cor de um texto

A cor a ser usada ao desenhar o texto e quaisquer decorações de texto, como a adição de sublinhados ou sobrelinhados e assim por diante.

background-color

A cor de fundo de um elemento, seja ele do tipo em linha ou em bloco.

text-shadow

Define um efeito de sombra para o texto. Entre as opções de configuração para a sombra, está a cor dessa sombra.

border

Essa propriedade é utilizada para definir uma borda par um elemento, dentre as suas configuração está a cor dessa borda.

opacity

Propriedade utilizar para regular a transparência de um elemento, ou seja, deixá-lo mais opaco ou mais transparente.

**Valores**

As cores no CSS podem ser aplicadas tanto através de uma palavra chave, ou seja, o nome da cor, quanto através de valores numéricos, utilizando as unidades hexadecimal ou rgb, há também outras formas, porém essas são as mais comuns.

Vamos utilizar como exemplo a cor **preta**:

* **Palava chave:** black
* **Hexadecimal:** #000000
* **RGB:** rgb(0, 0, 0)

**Para saber mais**

* Leia: [Propriedade color do CSS - MDN](https://developer.mozilla.org/pt-BR/docs/Web/CSS/color)
* Leia: [opacity - MDN](https://developer.mozilla.org/pt-BR/docs/Web/CSS/opacity)
* Leia(em inglês): [CSS Colors - W3Schools](https://www.w3schools.com/css/css_colors.asp)
* Leia(em inglês): [CSS RGB Colors - W3Schools](https://www.w3schools.com/css/css_colors_rgb.asp)
* Leia(em inglês): [CSS HEX Colors - W3Schools](https://www.w3schools.com/css/css_colors_hex.asp)

**Paletas**

* [Paletton](https://paletton.com/#uid=1000u0kllllaFw0g0qFqFg0w0aF)
* [Coolors](https://coolors.co/palettes/trending)
* [Adobe Color](https://color.adobe.com/create/color-wheel)

**Conteúdo opcional**

* Leia(em inglês): [Applying color to HTML elements using CSS](https://developer.mozilla.org/en-US/docs/Web/HTML/Applying_color)
* Leia(em inglês): [CSS Color Module Level 3](https://drafts.csswg.org/css-color-3/)
* Leia(em inglês): [color - MDN](https://developer.mozilla.org/en-US/docs/Web/CSS/color)
* Leia(em inglês): [opacity - MDN](https://developer.mozilla.org/en-US/docs/Web/CSS/opacity)

# Aula 05 - Fontes CSS

Nesse módulo nós vamos falar sobre a estilização fontes com css.

## Videoaula Kenzie Academy

Para o seguinte vídeo, você vai precisa de cerca de 10 minutos

## Estilizando fontes

Nós temos algumas opções de estilização de fontes, desde o seu tamanho até família dessa fontes. Vamos ver algumas propriedades e como utilizá-las!

### font-family

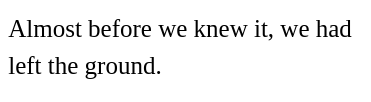
A propriedade font-family especifica uma lista de um ou mais nomes de famílias de fontes e / ou categorias de família para o elemento selecionado:

p {

font-family: Georgia, serif;

}

Esse código resultará no seguinte:

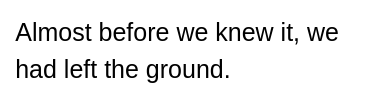


p {

font-family: "Gill Sans", sans-serif;

}

Esse código resultará no seguinte:



**As aspas são necessária quando o nome da família é composto, ou seja, tem mais um nome**

Os valores são separados por vírgulas para indicar que são alternativas. O navegador selecionará a primeira fonte da lista, se não encontrar a primeira ele utiliza a segunda e assim por diante.

Você deve sempre incluir pelo menos uma categoria de família em uma lista de famílias de fontes, uma vez que não há garantia de que qualquer fonte esteja disponível. Isso permite que o navegador selecione uma fonte alternativa quando necessário.

#### **Valores**

Há duas opções de valores, definir apenas a categoria de família da fonte ou definir uma fonte específica.

##### **Categorias de famílias**

Quando definimos apenas a categoria, o navegador aplica uma fonte dessa categoria que ela tenha por padrão em sua memória

Temos algumas opções de categorias de fontes. As principais são:

* serif

A fonte do tipo serif é aquela que tem uma 'perninha' no início e final da letra.

Alguns exemplos de fonte do tipo serif são Times New Roman e Georgia, como vimos no exemplo acima.

* sans-serif

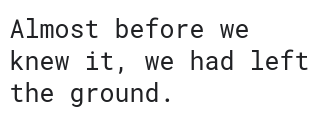
A fonte do tipo sans-serif é o contrário da serif, ou seja, 'sem serifa', como o próprio nome já mostra.

Alguns exemplos, são Arial, Roboto e Gill Sans, que vimos no exemplo acima.

* monospace

Na fonte to tipo monospace, todos os caracteres passam a ocupar exatamente o mesmo espaço. Muito utilizada em IDE's.

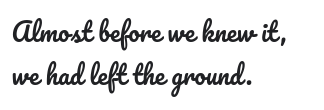
Um exemplo de fonte monospace é a Roboto Mono:



* cursive

A font do tipo cursive é aquela que possui um design mais sofisticado, com estilo de manuscrito.

Um exemplo de fonte cursive é a Pacifico:



### font-size

A propriedade CSS font-size define o tamanho da fonte.

#### **Valores**

Há algumas opções de valores para essa propriedade. A mais utilizada é um valor numérico, seguido de uma unidade de medida como px, em ou rem, que são as mais utilizadas se tratando de fontes.

### font-weight

A propriedade CSS font-weight define a espessura da fonte. Os pesos disponíveis dependem da família de fontes atualmente definida.

#### **Valores**

* normal

Espessura normal, equivalente aos valores 400 ou mesmo 500

* bold

Negrito. Equivalente aos valores 600 ou mesmo 700.

* lighter

Um peso de fonte relativo mais leve do que o elemento pai.

* bolder

Um peso de fonte relativo mais pesado que o elemento pai.

* valor numérico que varia de 100 a 900, sem a utilização de unidades de medida

### line-height

A propriedade CSS line-height define a altura de uma caixa de linha. É comumente usado para definir a distância entre as linhas de texto. Em elementos de nível de bloco, ele especifica a altura mínima das caixas de linha dentro do

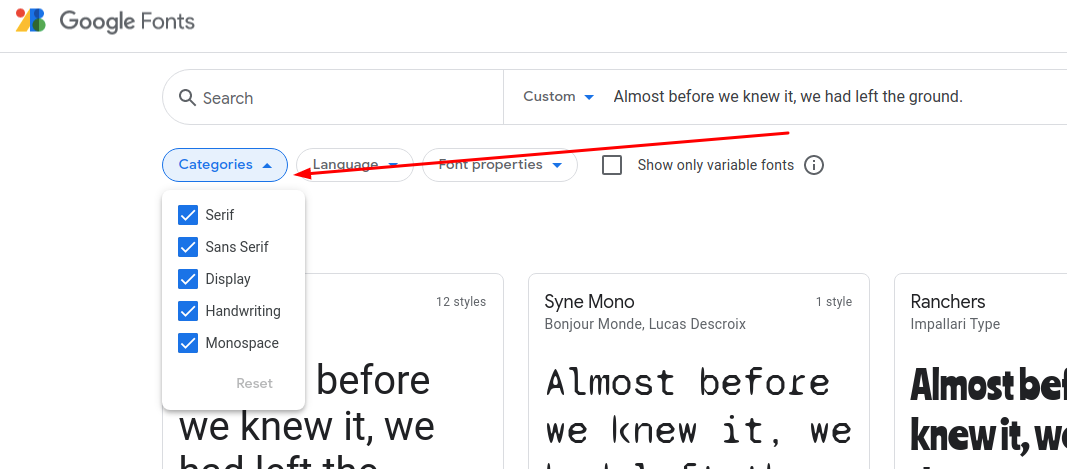
#### **Valores**

* Principalmente valores numéricos, seguido ou não de uma unidade de medida. Quado é aplicado um valor sem unidade, ele fica relativo ao tamanho da fonte do texto no qual está aplicando o line-height

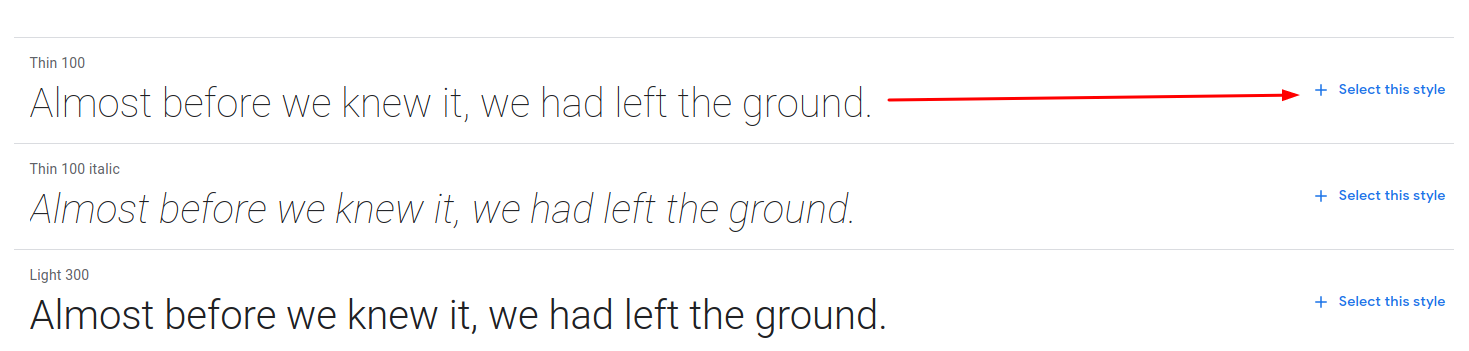
## Importando Fontes

### Como utilizar o Google Fonts

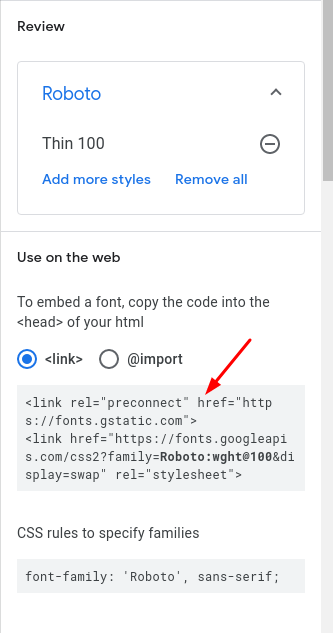
1. Acesse o website da Google Fonts, através da url [fonts.google.com](https://fonts.google.com/)
2. Pesquise pela fonte desejada ou procure por alguma rolando a página inicial. Você também pode filtrar pela categoria desejada.



1. Selecione a fonte desejada.
2. Selecione os estilos dessa fonte que deseja utilizar, clicando em "Select this style", ao lado do estilo.



1. Depois é só importar no seu website, utilizando o código disponibilizado pela plataforma, e então, aplica no seu CSS.



## Para saber mais

* Leia: [Estilizando Fontes com CSS](https://www.devmedia.com.br/estilizando-fontes-com-css/24226)

Aula 06 - Dimensionamento

Nessa aula vamos ver um pouco sobre o dimensionamento no css, e as principais propriedades utilizadas para dimensionar elementos, como width, height, max-width, min-width, max-height e min-height

**Propriedades**

width

A propriedade width define a largura de um elemento.

min-width

A propriedade min-width define uma largura mínima para um elemento. Impede que o valor usado da propriedade width se torne menor do que o valor especificado para largura mínima.

max-width

A propriedade max-width define a largura máxima de um elemento. Impede que o valor usado da propriedade width se torne maior do que o valor especificado para a largura máxima.

height

A propriedade height especifica a altura de um elemento.

min-height

A propriedade min-height define uma altura mínima para um elemento. Isso evita que o valor usado na propriedade height se torne menor do que o valor especificado para a altura mínima.

max-height

A propriedade max-height define a altura máxima de um elemento. Isso evita que o valor usado da propriedade height se torne maior do que o valor especificado para a altura máxima.

Valores

* Para as propriedades relacionadas a **largura**: Normalmente, valores numéricos, seguido de uma unidade de medida. A mais utilizada são px, % ou vw.
* Para as propriedades relacionadas a **altura**: Normalmente, valores numéricos, seguido de uma unidade de medida. A mais utilizada são px ou vh.

**Para saber mais**

* Leia: [Propriedade width](https://developer.mozilla.org/pt-BR/docs/Web/CSS/width)
* Leia: [Propriedade height](https://developer.mozilla.org/pt-BR/docs/Web/CSS/height)

Aula 07 - Posicionamento

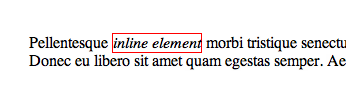
Nessa aula vamos ver um pouco sobre posicionamento utilizando a propriedade display.

**O display**

Todo elemento em uma página da web é uma caixa retangular. A propriedade display no CSS determina apenas como essa caixa se comporta. Existem alguns valores que são comumente usados:

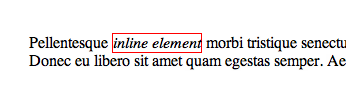
inline

Pense em elementos como <span>, <em> ou <b> e como envolver o texto nesses elementos em uma string de texto não interrompe o fluxo do texto.



O elemento <em> possui uma borda vermelha de 1px. Observe que ele está alinhado com o resto do texto.

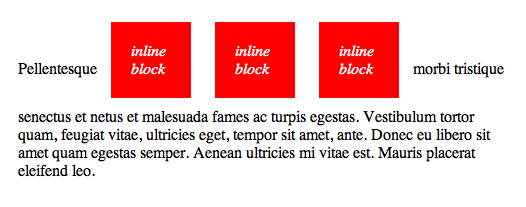
Um elemento inline irá aceita margin e padding, mas o elemento ainda fica embutido como você pode esperar. A margem e o preenchimento apenas afastarão outros elementos horizontalmente, não verticalmente.



Um elemento inline não aceita height e width. Ele simplesmente irá ignorar.

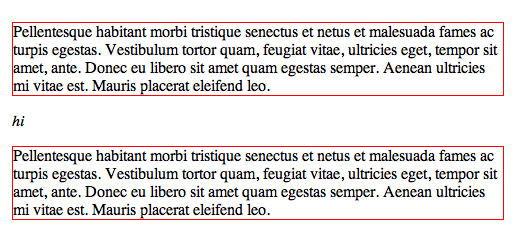
inline-block

Um elemento definido como inline-block é muito semelhante ao inline no sentido de que será definido em linha com o fluxo natural do texto (na "base da linha"). A diferença é que você pode definir uma largura e altura.



block

Vários elementos são definidos como block pela folha de estilo do navegador. Eles geralmente são elementos de contêiner, como <div>, <section> e <ul>. Também "blocos de texto" como <p> e <h1>. Os elementos de nível de bloco não ficam em linha, mas passam por eles. Por padrão(mesmo sem definir uma largura), eles ocupam todo o espaço horizontal disponível.



Para saber mais

* Leia: [Pare de chutar e aprenda como funciona o display: block](https://medium.com/collabcode/pare-de-chutar-e-aprenda-como-funciona-o-display-block-98480c987950)

Aula 01 - O box model (O modelo da caixa)

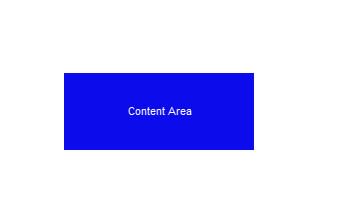
Todo iniciante deve primeiro começar com o básico. No caso do CSS, o básico é aprender o box model. Antes de prosseguir com o aprendizado de quaisquer outros conceitos CSS, este é o que você deve dominar primeiro!

Isso tende a causar confusão com os desenvolvedores iniciantes, então agora é o momento de esclarecer tudo. Esperançosamente, esta aula irá ajudá-lo a aprender todos os elementos básicos do modelo de caixa(box model) e como eles estão conectados.

Antes de mergulhar mais fundo, todos precisam entender que cada elemento no HTML é uma caixa retangular. Você provavelmente já ouviu isso várias vezes antes, mas este é um conceito importante que todo desenvolvedor deve estar ciente.

Quando um navegador renderiza(desenha) uma página da web, cada elemento, por exemplo, um pedaço de texto ou uma imagem, é desenhado como uma caixa retangular seguindo as regras do CSS Box Model.

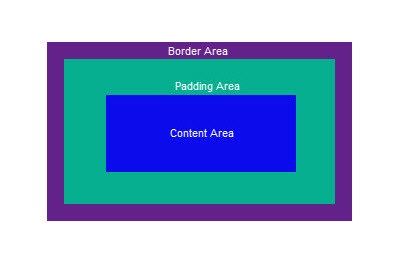
* No centro da caixa está o próprio conteúdo, que ocupa uma certa altura e largura. Esta região é conhecida como **área de conteúdo**.  
  O tamanho da área de conteúdo pode ser determinado automaticamente ou você pode definir explicitamente o tamanho da altura e da largura.



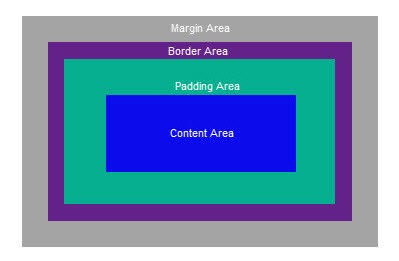
* Em torno da **área de conteúdo**, esta é uma região conhecida como **área de preenchimento(padding)**. O tamanho do preenchimento (padding) pode ser o mesmo ao redor, ou você pode definir individualmente para preenchimento superior, inferior, esquerdo e direito. Se você estiver usando um plano de fundo para o elemento, o plano de fundo se estenderá até a área de preenchimento.



* Em seguida, há uma área de **fronteira(borda)**. Isso cria uma borda ao redor do elemento e seu preenchimento. Você pode definir a espessura, cor e estilo da borda. As opções de estilo incluem nenhum, sólido, tracejado, pontilhado e vários outros.



* Finalmente, existe a área **marginal(mingin)**. Isso cria um espaço livre ao redor do elemento, preenchimento e borda. Novamente, você pode definir individualmente as margens superior, inferior, esquerda e direita. Sob certas circunstâncias, ocorre o colapso das margens e as margens entre os elementos adjacentes podem ser compartilhadas.



O Box model

* Leia(em inglês): [The box model](https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Box_model)

# Aula 02: Padding, margin e borda

## Padding e margin

Existem muitas maneiras de fornecer espaçamento entre os elementos e o conteúdo. Dois deles são preenchimento (padding) e margem.

## Padding (preenchimento)

O preenchimento (padding), conforme mencionado anteriormente no **Box Model**, é o espaçamento entre o conteúdo real e a borda da caixa.

Existem diferentes maneiras de escrever os valores da propriedade padding.

div {

padding: 10px; /\* Todos os Lados \*/

padding: 10px 5%; /\* Vertical Horizontal \*/

padding: 10px 20px 30em; /\* Top Horizontal Bottom \*/

padding: 10px 20px 30rem 40px;/\* acima 1em padding \*/

/\* direita 3px padding \*/

/\* abaixo 30px padding \*/

/\* esquerda 5px padding \*/

padding-left: 10px; /\* Esquerda \*/

padding-right: 10px; /\* Direita \*/

padding-top: 10px; /\* Top \*/

padding-bottom: 10px; /\* Bottom \*/

}

O padding também possui valores especiais que podem ser usados.

* %: porcentagem em relação à largura do contêiner
* initial: o padrão é 0
* inherit: valor de preenchimento do elemento que o contém
* unset: remove o valor

Mas temos que ficar atentos ao tipo de display dos nossos elementos, pois há diferença de aplicação para cada um deles

## (Inline | Block | Inline Block | Grid | Flex)

## Margin (margem)

A margem (margin), conforme mencionado anteriormente no Box Model, é o espaçamento entre um elemento e outro.

Existem diferentes maneiras de escrever os valores da propriedade margin.

div {

margin: 10px; /\* Todos os Lados \*/

margin: 10px 5%; /\* Vertical Horizontal \*/

margin: 10px 20px 30em; /\* Top Horizontal Bottom \*/

margin: 10px 20px 30rem 40px; /\* Top Direita Bottom Esquerda \*/

margin-left: 10px; /\* Esquerda \*/

margin-right: 10px; /\* Direita \*/

margin-top: 10px; /\* Top \*/

margin-bottom: 10px; /\* Bottom \*/

}

#### **Mais Exemplos**

div{

margin: 5%; /\* todos os lados: margem de 5% \*/

margin: 10px; /\* todos os lados: margem de 10px \*/

margin: 1.6em 20px; /\* topo e inferior: margem de 1.6em \*/

/\* esquerda e direita: margem de 20px \*/

margin: 10px 3% 1em; /\* topo: margem de 10px \*/

/\* esquerda e direita: margem de 3% \*/

/\* inferior: margem de 1em \*/

margin: 10px 3px 30px 5px; /\* topo: margem de 10px \*/

/\* direita: margem de 3px \*/

/\* inferior: margem de 30px \*/

/\* esquerda: margem de 5px \*/

margin: 2em auto; /\* topo e inferior: margem de 2em \*/

/\* caixa está horizontalmente centralizada \*/

margin: auto; /\* topo e inferior: margem de 0 \*/

} /\* caixa está horizontalmente centralizada \*/

**Referências**

* Leia: [Margin - w3schools](https://www.w3schools.com/cssref/pr_margin.asp).
* Leia: [Padding - w3schools](https://www.w3schools.com/cssref/pr_padding.asp).
* Leia: [Margin - mdn](https://developer.mozilla.org/pt-BR/docs/Web/CSS/margin).
* Leia: [Padding - mdn](https://developer.mozilla.org/pt-BR/docs/Web/CSS/padding).

## Border (borda)

A propriedade CSS **border** é usada para definir a borda de um elemento HTML.

A propriedade border é uma abreviação para três **subpropriedades** que definem o estilo, a cor e a largura de uma borda.

Exemplo:

border: 1px solid red;

border: (espessura) (tipo) (cor);

### Estilo da borda

Define o estilo da borda. O padrão é **none** (nenhuma borda).

border-style: none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset

### Cor da borda

Define a cor da borda. O padrão é o currentColor, definido pela propriedade color do elemento.

border-color: red;

### Bordas individuais em CSS

As subpropriedades e propriedades da borda também podem ser aplicadas a um lado individual de um elemento da web.

border-left: green;

border-top: pink;

border-right: blue;

border-bottom: skyblue;

* Leia: [Bordas - w3schools](https://www.w3schools.com/css/css_border.asp).
* Leia: [Bordas - mdn](https://developer.mozilla.org/pt-PT/docs/Web/CSS/border).

## Ferramentas

* Experimente!: [Border Generator](https://html-css-js.com/css/generator/border-outline/).
* Experimente!: [9elements](https://9elements.github.io/fancy-border-radius/#20.46.47.29--476.476).
* Experimente!: [CSS Border Generator](https://cssgenerator.org/border-css-generator.html).
* Experimente!: [Border-image generator](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Background_and_Borders/Border-image_generator).

Aula 03: Columns(múltiplas colunas de texto)

A propriedade Columns estende o modo de layout de bloco para permitir a fácil definição de múltiplas colunas de texto.

As pessoas têm problemas para ler texto se as linhas forem muito longas; se levar muito tempo para os olhos se moverem do final de uma linha para o início da seguinte, eles perderão o controle da linha em que estavam.  
Portanto, para aproveitar ao máximo uma tela grande, os autores devem ter colunas de texto de largura limitada colocadas lado a lado, como fazem os jornais.

**Exemplo:**



Infelizmente, isso era impossível de fazer com CSS e HTML sem forçar quebras de coluna em posições fixas, ou restringir severamente a marcação permitida no texto, ou usar scripts heróicos. Essa limitação é resolvida com a adição de novas propriedades CSS para estender o modo de layout de bloco tradicional.

**Usando colunas**

A **column-count** propriedade define o número de colunas para um número específico. Por exemplo,

<div id="col">

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit,

sed do eiusmod tempor incididunt ut labore et dolore magna

aliqua.

</p>

<p>

Ut enim ad minim veniam, quis nostrud exercitation ullamco

laboris nisi ut aliquip ex ea commodo consequat.

</p>

<p>

Duis aute irure dolor in reprehenderit in voluptate velit

esse cillum dolore eu fugiat nulla pariatur.

</p>

<p>

Excepteur sint occaecat cupidatat non proident, sunt in

culpa qui officia deserunt mollit anim id est laborum.

</p>

</div>

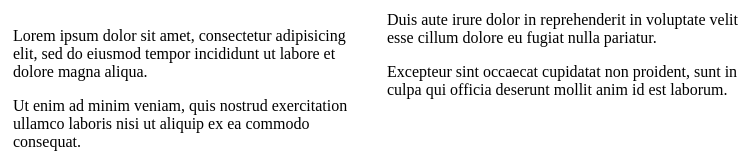
**CSS**

#col {

column-count: 2;

}

**Resultado**



**Outras propriedades**

* column-count
* column-gap
* column-rule-style
* column-rule-width
* column-rule-color
* column-rule
* column-span
* column-width

Veja mais nesse link

* Leia: [CSS columns Property - w3schools](https://www.w3schools.com/cssref/css3_pr_columns.asp)

Aula 04: Flexbox

Neste módulo, você aprenderá como usar o Modelo de Caixa Flexível, o Flexbox, no CSS para controlar o layout de elementos em uma página web.

Flexbox é um método de layout unidimensional para o layout de itens em linhas ou colunas. Os itens flexionam para preencher o espaço adicional e encolhem para caber em espaços menores.

Por muito tempo, as únicas ferramentas compatíveis entre navegadores confiáveis ​​disponíveis para a criação de layouts CSS eram coisas como float e position. Estes são bons e funcionam, mas em alguns aspectos também são bastante limitantes e frustrantes.

**Flex Container**

O Flex Container é a tag que envolve os itens flex, ao indicar display: flex, essa tag passa a ser um Flex Container.

**display**

Define o elemento como um flex container, tornando os seus filhos flex-itens.

display: flex;

/\*Torna o elemento um flex container

automaticamente transformando todos

os seus filhos diretos em flex itens.\*/

**flex-direction**

Define a direção dos flex itens. Por padrão ele é row (linha), por isso quando o display: flex; é adicionado, os elementos ficam em linha, um do lado do outro.

A mudança de row para column geralmente acontece quando estamos definindo os estilos em media queries para o mobile. Assim você garante que o conteúdo seja apresentado em coluna única.

flex-direction:row;

/\* Os itens ficam em linha\*/

flex-direction: row-reverse;

/\*Os itens ficam em linha reversa, ou seja 3, 2, 1.\*/

flex-direction: column;

/\*Os itens ficam em uma única coluna, um embaixo do outro.\*/

flex-direction: column-reverse;

/\*Os itens ficam em uma única coluna, um embaixo do outro, porém em ordem reversa: 3, 2 e 1.\*/

**flex-wrap**

Define se os itens devem quebrar ou não a linha. Por padrão eles não quebram linha, isso faz com que os flex itens sejam compactados além do limite do conteúdo.

Essa é geralmente uma propriedade que é quase sempre definida como flex-wrap: wrap; Pois assim quando um dos flex itens atinge o limite do conteúdo, o último item passa para a coluna debaixo e assim por diante.

flex-wrap: nowrap;

/\*Valor padrão, não permite a quebra de linha.\*/

flex-wrap: wrap;

/\* Quebra a linha assim que um dos flex itens não puder mais ser compactado.\*/

flex-wrap: wrap-reverse;

/\* Quebra a linha assim que um dos flex itens não puder mais ser compactado.

A quebra é na direção contrária, ou seja para a linha acima.\*/

**flex-flow**

O flex-flow é um atalho para as propriedades flex-direction e flex-wrap. Você não verá muito o seu uso, pois geralmente quando mudamos o flex-direction para column, mantemos o padrão do flex-wrap que é nowrap.

E quando mudamos o flex-wrap para wrap, mantemos o padrão do flex-direction que é row.

flex-flow: row nowrap;

/\*Coloca o conteúdo em linha e não permite a quebra de linha.\*/

flex-flow: row wrap;

/\*Coloca o conteúdo em linha e permite a quebra de linha.\*/

flex-flow: column nowrap;

/\*Coloca o conteúdo em coluna e não permite a quebra de linha.\*/

**justify-content**

Alinha os itens flex no container de acordo com a direção. A propriedade só funciona se os itens atuais não ocuparem todo o container. Isso significa que ao definir flex: 1; ou algo similar nos itens, a propriedade não terá mais função

Excelente propriedade para ser usada em casos que você deseja alinhar um item na ponta esquerda e outro na direita, como em um simples header com marca e navegação.

justify-content: flex-start;

/\*Alinha os itens ao início do container.\*/

justify-content: flex-end;

/\*Alinha os itens ao final do container.\*/

justify-content: center;

/\*Alinha os itens ao centro do container.\*/

justify-content: space-between;

/\*Cria um espaçamento igual entre os elementos.

Mantendo o primeiro grudado no início e o último no final.\*/

justify-content: space-around;

/\*Cria um espaçamento entre os elementos.

Os espaçamentos do meio são duas vezes maiores que o inicial e final.\*/

**align-items**

O align-items alinha os flex itens de acordo com o eixo do container. O alinhamento é diferente para quando os itens estão em colunas ou linhas.

Essa propriedade permite o tão sonhado alinhamento central no eixo vertical, algo que antes só era possível com diferentes hacks.

align-items: stretch;

/\*Valor padrão, ele que faz com que os flex itens cresçam igualmente.\*/

align-items: flex-start;

/\*Alinha os itens ao início.\*/

align-items: flex-end;

/\*Alinha os itens ao final.\*/

align-items: center;

/\*Alinha os itens ao centro.\*/

align-items: baseline;

/\*Alinha os itens de acordo com a linha base da tipografia.\*/

**align-content**

Alinha as linhas do container em relação ao eixo vertical. A propriedade só funciona se existir mais de uma linha de flex-itens. Para isso o flex-wrap precisa ser wrap.

Além disso o efeito dela apenas será visualizado caso o container seja maior que a soma das linhas dos itens. Isso significa que se você não definir height para o container, a propriedade não influencia no layout.

align-content: stretch;

/\*Valor padrão, ele que faz com que os flex itens cresçam igualmente na vertical.\*/

align-content: flex-start;

/\*Alinha todas as linhas de itens ao início.\*/

align-content: flex-end;

/\*Alinha todas as linhas de itens ao final.\*/

align-content: center;

/\*Alinha todas as linhas de itens ao centro.\*/

align-content: space-between;

/\*Cria um espaçamento igual entre as linhas.

Mantendo a primeira grudada no topo e a última no bottom.\*/

align-content: space-around;

/\*Cria um espaçamento entre as linhas.

Os espaçamentos do meio são duas vezes maiores que o top e bottom.\*/

**Para Saber mais sobre o flex, veja mais exemplos nesse link:**

* Conheça!: [Coding Imweb](https://coding.imweb.io/demo/flex/index.html)

**Flexbox**

* Leia: [Flexbox - MDN](https://developer.mozilla.org/pt-BR/docs/Learn/CSS/CSS_layout/Flexbox)
* Leia: [Flexbox - w3schools](https://www.w3schools.com/css/css3_flexbox.asp)
* Leia: [Flexbox - Flexbox guia completo](https://css-tricks.com/snippets/css/a-guide-to-flexbox/) **em inglês**

# Aula 05: Responsividade

O design responsivo da Web é a abordagem que sugere que o design e o desenvolvimento devem responder ao comportamento do usuário e ao ambiente com base no tamanho da tela, plataforma e orientação.

A prática consiste em uma mistura de grades e layouts flexíveis, imagens e um uso inteligente de **media queries** CSS. Conforme o usuário muda do laptop para o iPad, o site deve mudar automaticamente para acomodar a resolução.

É importante entender que o web design responsivo não é uma tecnologia separada - é um termo usado para descrever uma abordagem ao web design ou um conjunto de práticas recomendadas, usado para criar um layout que pode responder ao dispositivo que está sendo usado para visualizar o conteúdo.

## Defina a janela de visualização

As páginas otimizadas para uma variedade de dispositivos devem incluir uma meta tag viewport no cabeçalho do documento. Uma meta tag viewport fornece ao navegador instruções sobre como controlar as dimensões e o dimensionamento da página.

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

A propriedade width controla o tamanho da viewport. O valor desta propriedade pode ser definido com um número específico de píxels como por exemplo width=600 ou com um valor especial chamado device-width que representa a largura da onde o conteúdo está sendo apresentado em pixels de CSS considerando uma escala de 100%. (Ainda há as propriedades height e device-height, as quais podem ser úteis para páginas com elementos que mudam de posição baseado na altura da viewport.)

A propriedade initial-scale controla o nível de amplificação quando a página é carregada pela primeira vez. As propriedades maximum-scale, minimum-scale, e user-scalable controlam a permissão para o usuário efetuar aumento ou diminuição da página.

## Viewport

### Viewport - janela de visualização

A janela de visualização é a área visível do usuário em uma página da web.

A janela de visualização varia com o dispositivo e será menor em um telefone celular do que em uma tela de computador.

Antes dos tablets e telefones celulares, as páginas da web eram projetadas apenas para telas de computador, e era comum que as páginas tivessem um design estático e um tamanho fixo.

Então, quando começamos a navegar na Internet usando tablets e telefones celulares, as páginas da web de tamanho fixo eram muito grandes para caber na janela de visualização. Para corrigir isso, os navegadores nesses dispositivos reduziram a página da web inteira para caber na tela.

## Links alternativos

[Dev Media](https://www.devmedia.com.br/responsive-design-dicas-para-tornar-seu-site-acessivel-em-qualquer-resolucao/28316)

Aula 06: Media queries - Consulta de Mídias

Fazer um site com um layout adaptável é chamado de Web Design Responsivo. E as media query CSS são uma das partes mais importantes do design responsivo. Neste artigo, vamos dar uma olhada mais de perto nas media queries e como usá-las em CSS.

**O que é uma media query?**

Media query um recurso CSS3 que faz com que uma página da Web adapte seu layout a diferentes tamanhos de tela e tipos de mídia.

Sintaxe

@media media type and (condition: breakpoint) {

}

@media tipo de míedia && (condição: ponto de interrupção) {

}

Podemos direcionar diferentes tipos de mídia sob uma variedade de condições. Se a condição e / ou os tipos de mídia atenderem, as regras dentro da media query serão aplicadas, caso contrário, não.

@Regra de mídia

Começamos definindo media query a regra @media e depois incluímos regras CSS entre chaves. A regra @ media também é usada para especificar os tipos de mídia de destino.

@media () {

}

Parêntese

Dentro dos parênteses, definimos uma condição. Por exemplo, desejo aplicar um tamanho de fonte maior para dispositivos móveis. Para fazer isso, precisamos definir uma largura máxima que verifica a largura de um dispositivo:

.text {

font-size: 14px;

}

@media (max-width: 480px) {

.text {

font-size: 16px;

}

}

Normalmente, o tamanho do texto será de 14px. No entanto, como aplicamos uma media query, ela mudará para 16px quando um dispositivo tiver uma largura máxima de 480px ou menos.

Importante: Sempre coloque suas media queries no final de seu arquivo CSS.

Pontos de interrupção comuns: existe uma resolução padrão?

Uma das perguntas mais frequentes é “Qual ponto de interrupção devo usar?”. Há uma tonelada de dispositivos no mercado, então não podemos e não devemos definir pontos de interrupção fixos para cada um deles.

usados ​​na programação diária. Se você estiver usando uma estrutura CSS (como Bootstrap, Bulma, etc.), você também pode usar seus pontos de interrupção.

* 320px - 480px: Dispositivos móveis
* 481 px - 768 px: iPads, tablets
* 769 px - 1024 px: telas pequenas, laptops
* 1.025 px - 1.200 px: desktops, telas grandes
* 1201 px e mais - telas extragrandes, TV

Como eu disse acima, esses pontos de interrupção podem ser diferentes e não há um padrão definido exatamente, mas esses são alguns dos mais usados.

**Links de referencias:**

[Tableless](https://tableless.com.br/introducao-sobre-media-queries/) [Dev Media](https://www.devmedia.com.br/utilizando-css-media-queries/27085)

[MDN Web](https://developer.mozilla.org/pt-BR/docs/Web/CSS/Media_Queries/Using_media_queries)

Aula 07: Mobile first

Nessa aula nós vamos falar sobre o conceito de mobile first.

**Introdução**

Como o próprio nome já diz, mobile first, significa "primeiro o mobile". Ou seja, começar o desenvolvimento do front pelo layout para dispositivos móveis. E então, utilizar **media queries** para sobrescrever esse estilo em telas maiores, porém utilizando o min-width, ao invés do max-width.

**Responsividade X Mobile first**

É muito importante entender que fazer uma página responsiva e uma página em mobile first são duas coisas **muito diferentes**!

* Na **responsividade** você apenas adapta os elementos para que eles encaixem na tela sem quebrar a página.
* Com o **mobile first**, você vai ter dois layouts diferentes um para o desktop e outro para dispositivos móveis, podendo assim ter uma melhor disposição de elementos em cada uma das situações, de uma maneira que faça mais sentido para cada uma delas.

**Para saber mais**

* Leia: [Entenda o que é mobile first e conheça as suas principais vantagens](https://www.moblee.com.br/blog/mobile-first-principais-vantagens/)
* Leia: [Como escrever seu css para projetos mobile-first](https://www.todoespacoonline.com/w/2015/03/como-escrever-seu-css-para-projetos-mobile-first/)

Aula 08: Variáveis

Variáveis ​​CSS, mais precisamente conhecidas como propriedades personalizadas no CSS, são úteis para reduzir a repetição em CSS e também para efeitos de tempo de execução poderosos, como troca de tema e potencialmente estender futuros recursos CSS.

As propriedades personalizadas adicionam dois novos recursos à nossa caixa de ferramentas CSS:

* A capacidade de você atribuir valores arbitrários a uma propriedade com um nome escolhido pelo autor.
* A função var(), que permite que você possa usar esses valores em outras propriedades.

Aqui está um exemplo:

:root {

--main-color: #06c;

}

#foo h1 {

color: var(--main-color);

}

--main-color é uma propriedade customizada definida pelo desenvolvedor com um valor de # 06c. Observe que todas as propriedades personalizadas começam com dois travessões.

A função var() recupera e substitui a si mesma pelo valor da propriedade customizada, resultando em color: #06c;Contanto que a propriedade customizada seja definida em algum lugar em sua folha de estilo, ela deve estar disponível para a função var().

Você também pode utilizar as propriedades personalizadas dentro das media queries para ajudar no design responsivo. Um caso de uso pode ser expandir a margem em torno de seus principais elementos de corte à medida que o tamanho da tela aumenta:

:root {

--gutter: 4px;

}

section {

margin: var(--gutter);

}

@media (min-width: 600px) {

:root {

--gutter: 16px;

}

}

**Por que você usaria propriedades personalizadas de CSS nativas?**

* Você pode usá-los sem a necessidade de um pré-processador.
* Eles caem em cascata. Você pode definir uma variável dentro de qualquer seletor para definir ou substituir seu valor atual.
* Quando seus valores mudam (por exemplo, media query ou outro estado), o navegador redesenha conforme necessário .
* Você pode acessar e manipulá-los em JavaScript.

Material Adicional

* Leia: [Utilizando variáveis CSS](https://developer.mozilla.org/pt-BR/docs/Web/CSS/Using_CSS_custom_properties)
* Leia: [Medium](https://medium.com/devtranslate/como-usar-vari%C3%A1veis-css-29b5b980794c)

Aula: Breve guia de CSS

Esta aula é apenas um breve guia de css para te ajudar naquele momento em que você não lembra exatamente qual propriedade utilizar ou quais valores você pode passar para certa propriedade, entre outras situações.

Então aqui vão algumas referências(em inglês) para você ter sempre na mão quando bater uma dúvida.

*\*Itens 1 e 2 você pode fazer o download do pdf, o que eu recomendo, já que na página em si não está muito legível*

*\*\*Item 1 você também pode clicar nas imagens para visualizar em outra aba e em melhor qualidade*

1. [Complete CSS Cheat Sheet](https://websitesetup.org/css3-cheat-sheet/)
2. [CSS Cheat Sheet – A Complete Guide for Beginners and Professionals](https://www.hostinger.com/tutorials/css-cheat-sheet)
3. [Quick and Practical CSS Cheat Sheet](https://www.toptal.com/css/css-cheat-sheet)

Bônus

1. [W3Schools CSS Reference](https://www.w3schools.com/cssref/default.asp)
2. [W3Schools HTML Reference](https://www.w3schools.com/tags/default.asp)
   * *\*Também com comentários sobre tags depreciadas*

Referências HTML/CSS

Nessa aula, separamos algumas das principais referências sobre os principais e mais utilizados assuntos da sprint.

Se precisar relembrar de algum conceito, é só dar um pulo aqui.

**HTML Moderno**

Tags html:

1. Leia: [Elementos HTML](https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element)
2. Leia: [HTML elements reference](https://developer.mozilla.org/en-US/docs/Web/HTML/Element)(em inglês)
3. Leia: [Seções e estrutura de um documento HTML5](https://developer.mozilla.org/pt-BR/docs/Sections_and_Outlines_of_an_HTML5_document)

**CSS Básico**

Introdução:

1. Leia: [Como o CSS é estruturado](https://developer.mozilla.org/pt-BR/docs/Learn/CSS/First_steps/Como_CSS_e_estruturado)

Seletores:

1. Leia: [CSS selectors](https://developer.mozilla.org/pt-BR/docs/Learn/CSS/Building_blocks/Selectors)

Display:

1. Leia: [Display](https://developer.mozilla.org/en-US/docs/Web/CSS/display)(em inglês)
   * **Principais displays: block, inline-block, flex, grid e none**

Fontes:

 Leia: [fonts](https://www.w3schools.com/cssref/pr_font_font.asp" \t "_blanc)(em inglês)

* Principai propriedades: font-family, font-size, font-weight, font-style, font-variant, line-height

 Leia: [Como utilizar o Google Fonts](https://metring.com.br/como-utilizar-o-google-fonts)

Unidades de medida:

1. Leia: [CSS Units](https://www.w3schools.com/cssref/css_units.asp)(em inglês)

Dimensões:

1. Leia: [Dimensões - resumo](https://developer.mozilla.org/pt-BR/docs/Web/CSS/box_model)

Columns:

1. Leia: [CSS Multiple Columns](https://www.w3schools.com/css/css3_multiple_columns.asp)(em inglês)

**Avançando CSS**

Box Model:

1. Leia: [O Box Model](https://developer.mozilla.org/pt-BR/docs/Aprender/CSS/Construindo_blocos/The_box_model)

Margin e padding:

1. Leia: [CSS margin property](https://www.w3schools.com/cssref/pr_margin.asp)(em inglês)
2. Leia: [CSS padding property](https://www.w3schools.com/cssref/pr_padding.asp)(em inglês)

Bordas:

1. Leia: [CSS Borders](https://www.w3schools.com/css/css_border.asp)(em inglês)

**Avançando Posicionamento CSS**

Flexbox:

1. Leia: [Guia completo de flexbox](https://origamid.com/projetos/flexbox-guia-completo/)

**Responsividade CSS**

Responsividade:

1. Leia: [Design responsivo](https://developer.mozilla.org/pt-BR/docs/Learn/CSS/CSS_layout/Responsive_Design)
2. Leia: [Media queries](https://www.w3schools.com/css/css3_mediaqueries_ex.asp)(em inglês)
3. Leia: [CSS media queries](https://css-tricks.com/css-media-queries/)(em inglês)