

O MODELO DE CAIXAS



M02C16

O MODELO DE CAIXAS

Entender o modelo das caixas é um dos primeiros passos para construir interfaces web e começar a dar forma aos seus sites. Nos últimos quatro capítulos, estudamos a essência das folhas de estilo e já sabemos criar seletores, identificá-los e personalizá-los. Agora chegou a hora de colocarmos tudo isso em caixas configuráveis e vamos começar a desenhar nossas primeiras páginas.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.

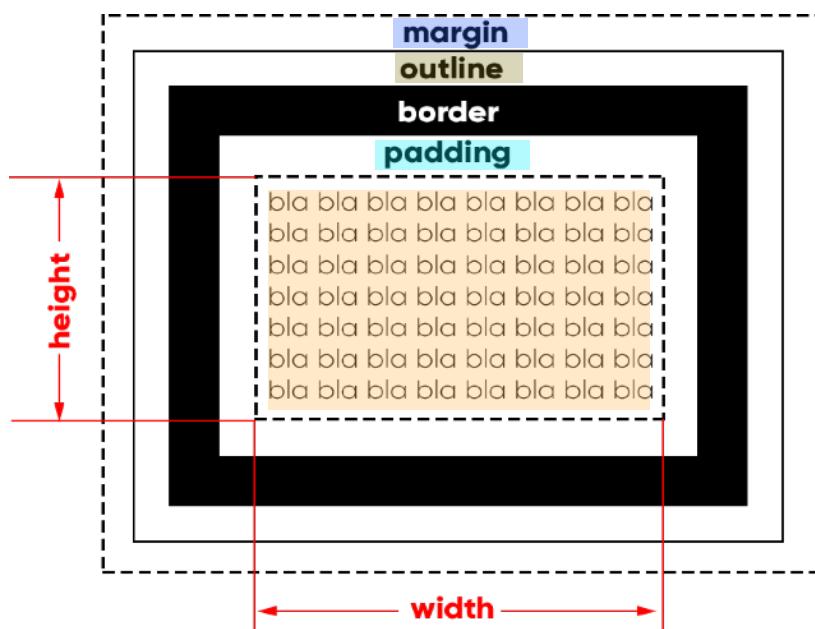
O que é uma caixa?

De forma simples e objetiva, baseado em um conceito chamado “**box model**”, a grande maioria dos elementos HTML que temos no nosso site são como caixas. Elas são **containers** que armazenam conteúdos ou até mesmo outras caixas.



Anatomia de uma caixa

Vamos analisar como uma caixa vai ser apresentada por todos os navegadores. Olhe atentamente o diagrama a seguir, que é exatamente o já citado **modelo de caixa**:



Tudo começa a partir do **conteúdo** (content), que representamos acima com o bla bla bla... Por padrão, toda caixa é composta apenas pelo conteúdo e não possui padding, nem border, nem outline e nem margin. Uma exceção curiosa é o elemento <body> que já vem com uma margin de 8px.

Todo conteúdo possui uma **largura** (width) e uma **altura** (height) e a esse conjunto de propriedades, damos o nome de **box-size** (tamanho da caixa). O tamanho da caixa não inclui as medidas de padding, border, outline e margin.

Depois do conteúdo e de seu tamanho, vamos nos focar na **borda** que fica em volta dele. Ela pode ter uma **espessura**, uma **cor** e um **formato**.

Entre a borda e o conteúdo - da borda para dentro - temos o **preenchimento** (padding) e da borda para fora, temos a **margem** (margin).

Entre a margem e a borda, podemos determinar o **contorno** (outline) que é muito pouco utilizado, mas existe. Ele é um traçado visual que podemos criar fora da borda e o cálculo da sua espessura faz parte da margem estabelecida.

Vamos criar um exemplo simples para exemplificar todos esses componentes, configurando as propriedades do modelo de caixa de um título `<h1>`. Acompanhe o trecho de código a partir das definições de estilo.

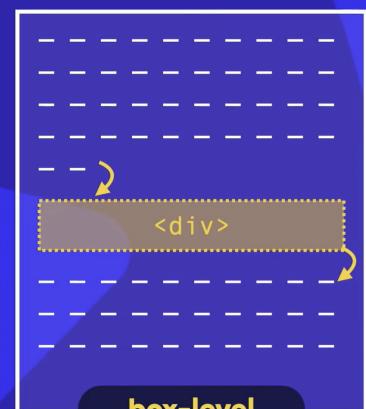
```
7 <style>
8   h1 {
9     width: 300px;
10    height: 50px;
11    background-color: lightgray;
12    border-width: 10px;
13    border-style: solid;
14    border-color: red;
15    padding: 20px;
16    outline-width: 30px;
17    outline-style: solid;
18    outline-color: blue;
19    margin: 50px;
20  }
21 </style>
22 </head>
23 <body>
24   <h1>Exemplo de Caixa</h1>
25 </body>
26 </html>
```

Todas as configurações serão aplicadas ao elemento `<h1>`, que é uma caixa e foi criado na **linha 24** do código acima. As **linhas 9 e 10** configuram o size da caixa (largura e altura, respectivamente) e fará com que ela tenha 300x50 pixels.

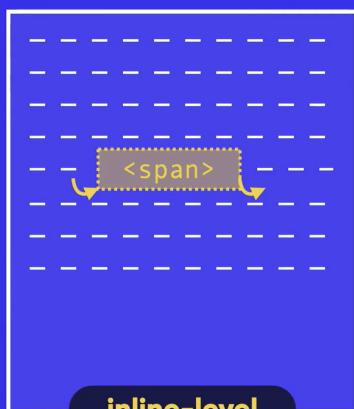
As **linhas de 12 a 14**, configuram uma borda sólida, vermelha e com 10 pixels de espessura.

A **linha 15** vai criar um espaço interno de preenchimento (da borda para dentro) de 20 pixels no elemento e a **linha 19** vai criar um espaço externo (da borda para fora) de 50 pixels.

Tipos de Caixa



box-level



inline-level

Tipos de Caixa

<div>
<h1>-<h6>
<p>
<main>
<header>
<nav>
<article>
<aside>
<footer>
<form>
<video>

box-level

<a>
<code>
<small>

<sup>-<sub>
<label>
<button>
<input>
<select>

inline-level

As **linhas de 16 a 18** vão usar parte da margem para criar um contorno azul, sólido e com 30 pixels de espessura.



TAMANHO TOTAL: Para calcular a largura e altura total de um elemento na tela, **some os tamanhos do conteúdo + preenchimento + borda + margem.** O contorno não vai entrar nessa conta, pois utiliza parte da medida da margem.

O resultado visual do código anterior será:



Olhando de perto, podemos analisar as medidas configuradas no código apresentado. As **medidas de height e width (300x50)** são medidas apenas pela parte pontilhada do conteúdo.

A border de 10px ficou em vermelho e o outline de 30px ficou em azul. O padding de 20px fica da borda para dentro e a margin de 50px fica da borda para fora.

Sendo assim, a medida total que essa caixa vai ocupar é de $50 + 10 + 20 + 300 + 20 + 10 + 50 = \textbf{460px de largura}$ e $50 + 10 + 20 + 50 + 20 + 10 + 50 = \textbf{210px de altura}.$



NOVIDADE DAS CSS3: Existe a nova propriedade **box-sizing** onde podemos definir que as dimensões **height** e **width** não são medidas apenas a partir do conteúdo (content-box) e sim pela borda (border-box).

Dá pra simplificar? Shorthand a ordem é fundamental

As configurações de borda e contorno também possuem *shorthands* para simplificar o código anterior. A ordem para as duas configurações é sempre a mesma para as duas shorthands: largura (-width), estilo (-style) e cor (-color).

MODO COMPLETO	SHORTHAND
<code>border-width: 10px; border-style: solid; border-color: red; outline-width: 30px; outline-style: solid; outline-color: blue;</code>	<code>border: 10px solid red; outline: 30px solid blue;</code>

Preenchimento e margem personalizados

Sentido horário começando do topo, sempre!

Todo elemento de caixa possui *quatro valores* para padding e quatro para margin, sempre nessa mesma *ordem*: superior (-top), direita (-right), inferior (-bottom), esquerda (-left). Quando colocamos um único valor de dimensão para o preenchimento ou margem, esse mesmo valor é aplicado simetricamente a todas as direções, mas também podemos fazer códigos como:

MODO COMPLETO	SHORTHAND
<code>padding-top: 10px; padding-right: 15px; padding-bottom: 20px; padding-left: 25px; margin-top: 0px; margin-right: 10px; margin-bottom: 20px; margin-left: 30px;</code>	<code>padding: 10px 15px 20px 25px; margin: 0px 10px 20px 30px; padding: 10px 20px; margin: 0px 15px;</code>

Também existe a opção de indicar cada shorthand das propriedades de preenchimento e borda usando apenas **duas medidas**:

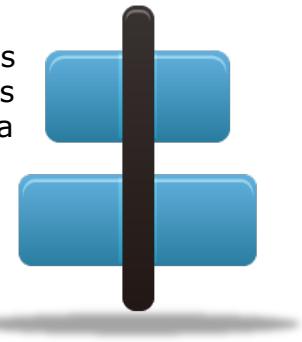
MODO COMPLETO	SHORTHAND
<pre>padding-top: 10px; padding-right: 20px; padding-bottom: 10px; padding-left: 20px; margin-top: 0px; margin-right: 15px; margin-bottom: 0px; margin-left: 15px;</pre>	<pre>padding: 10px 20px; margin: 0px 15px;</pre>

Essa simplificação só é possível quando as medidas -top e -bottom forem iguais entre si e o mesmo também ocorrer entre as medidas -right e -left.

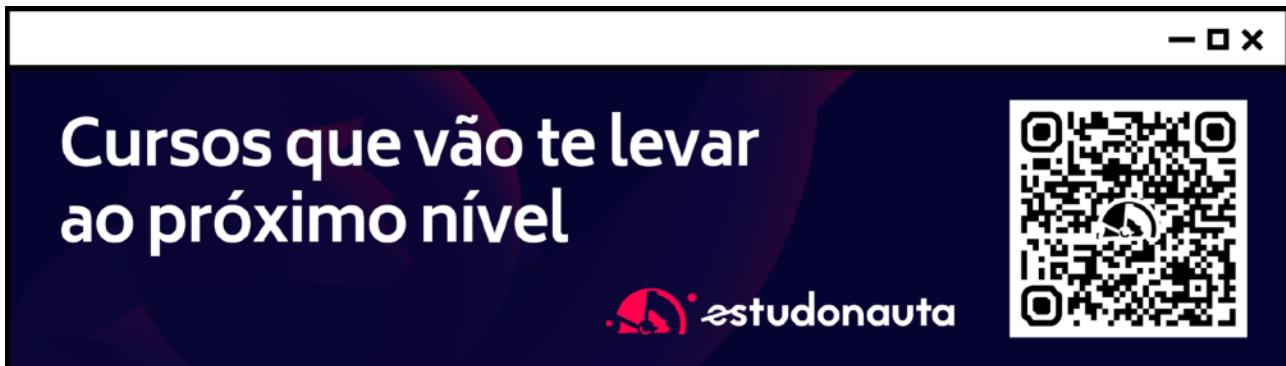
Margens no automático

Um recurso que também vai ser muito usado em nossos exercícios é a centralização de blocos. Para que isso seja feito, devemos pedir que o navegador calcule automaticamente as margens da esquerda e da direita para que o bloco seja colocado no meio do navegador, independente do tamanho da janela.

Para centralizar uma caixa, use a seguinte declaração no seu seletor:



```
margin: auto;
```



Cursos que vão te levar ao próximo nível

estudonauta

QR code

Tipos de Caixa

Dependendo do comportamento da caixa, podemos classificar um elemento em uma de duas categorias:

Caixa do tipo **block-level**

Um elemento dito *block-level* **sempre vai se iniciar em uma nova linha** e **vai ocupar a largura total do elemento** onde ele está contido. Se não estiver contido em nenhuma outra caixa, ele vai ocupar **100% da largura do <body>**.

O elemento *block-level* mais conhecido é o `<div>` e suas variações semânticas modernas da HTML5, como `<main>`, `<section>`, `<aside>`, etc.

Na lista a seguir, coloquei alguns elementos HTML que são *block-level*:

<code><address></code>	<code><article></code>	<code><aside></code>	<code><blockquote></code>	<code><canvas></code>	<code><dd></code>
<code><div></code>	<code><dl></code>	<code><dt></code>	<code><fieldset></code>	<code><figcaption></code>	<code><figure></code>
<code><footer></code>	<code><form></code>	<code><h1> - <h6></code>	<code><header></code>	<code><hr></code>	<code></code>
<code><main></code>	<code><nav></code>	<code><noscript></code>	<code></code>	<code><p></code>	<code><pre></code>
<code><section></code>	<code><table></code>	<code><tfoot></code>	<code></code>	<code><video></code>	

Caixa do tipo **inline-level**

Um elemento do tipo *inline-level* **não vai começar em uma nova linha**, e sim no ponto exato onde foram definidos. E a **largura** dele vai ocupar apenas o **tamanho relativo ao seu conteúdo**.

Abaixo, listei alguns elementos *inline-level* usados pela HTML:

<code><a></code>	<code><abbr></code>	<code><acronym></code>	<code></code>	<code><bdo></code>	<code>
</code>
<code><button></code>	<code><cite></code>	<code><code></code>	<code><dfn></code>	<code></code>	<code><i></code>
<code></code>	<code><input></code>	<code><kbd></code>	<code><label></code>	<code><map></code>	<code><object></code>
<code><output></code>	<code><q></code>	<code><samp></code>	<code><script></code>	<code><select></code>	<code><small></code>
<code></code>	<code></code>	<code><sub></code>	<code><textarea></code>	<code><tt></code>	<code><var></code>
					<code><iframe></code>

Grouping Tags e Semantic Tags

A linguagem **HTML padrão** tinha apenas **duas tags de agrupamento genérico**: a `<div>` e a ``. A diferença básica entre elas é que a primeira é um elemento agrupador do tipo *block-level* e o segundo é *inline-level*. No mais, eles agem exatamente da mesma maneira, servindo para juntar vários outros elementos HTML.

Com o surgimento da **HTML5**, surgiram as **tags semânticas** de agrupamento. Isso não significa que as `<div>` e `` (agora chamadas de **não-semânticas**) deixaram de existir ou ficaram obsoletas, mas seu uso agora faz menos sentido, pois temos tags para dividir as partes do nosso documento HTML.

Vamos compreender a partir de agora os principais agregadores semânticos da HTML5.

Header

Cria áreas relativas a **cabeçalhos**. Pode ser o cabeçalho principal de um site ou até mesmo o cabeçalho de uma seção ou artigo. **Normalmente inclui títulos <h1> - <h6> e subtítulos.** Podem também conter menus de navegação.

Nav

Define uma área que possui os **links de navegação** pela estrutura de páginas que vão compor o website. Um `<nav>` pode estar dentro de um `<header>`.



Main

É um agrupador usado para delimitar o **conteúdo principal** do nosso site. Normalmente concentra as seções, artigos e conteúdos periféricos.

Section

Cria **seções** para sua página. Ela pode conter o conteúdo diretamente no seu corpo ou **dividir os conteúdos em artigos com conteúdos específicos**. Segundo a documentação oficial da W3C, “uma seção é um agrupamento temático de conteúdos, tipicamente com um cabeçalho”.

Article

Um **artigo** é um elemento que vai conter um **conteúdo que pode ser lido de forma independente e dizem respeito a um mesmo assunto**. Podemos usar um `<article>` para delimitar um **post** de blog ou fórum, uma **notícia**, etc.

Aside

Delimita um **conteúdo periférico** e complementar ao conteúdo principal de um artigo ou seção. Normalmente um conteúdo `<aside>` está posicionado ao lado de um determinado texto ou até mesmo no meio dele, exatamente como fizemos no bloco de texto apresentado anteriormente, falando sobre "MÚLTIPLOS NÍVEIS".



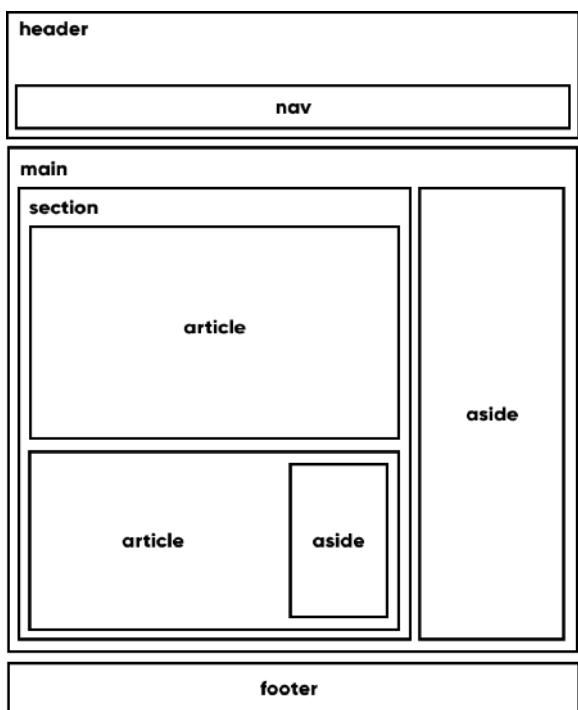
MÚLTIPLOS NÍVEIS: A sua criatividade e planejamento vai definir a estrutura do seu site. Sendo assim, é possível ter um ou mais `<article>` dentro de uma `<section>` ou até mesmo criar `<section>` dentro de um `<article>`. **Não existem limitações** quanto a isso.

Footer

Cria um **rodapé** para o **site inteiro**, **seção** ou **artigo**. É um conteúdo que **não faz parte diretamente do conteúdo nem é um conteúdo periférico** (o que caracterizaria um `<aside>`), mas possui informações sobre autoria do conteúdo, links adicionais, mapa do site, documentos relacionados.

A seguir, vou criar uma proposta de estrutura para um projeto de site. Não tome ela como a única possibilidade de criar o posicionamento de elementos de agrupamento semântico.

A screenshot of a website's footer. The background is purple. At the top, there is a window-like header with close, minimize, and maximize buttons. Below this, on the left, is the text "Soluções digitais para negócios" in large white font. On the right, there is a QR code with a small hostnet logo in the center. At the bottom, there is a logo for "hostnet" featuring a stylized cloud icon and the word "hostnet" in white.



```

9   <header>
10  <h1>Meu Site</h1>
11  <nav>link link link link...</nav>
12  </header>
13  <main>
14    <section>
15      <article>
16          <h2>Título</h2>
17          <p>Texto do artigo</p>
18      </article>
19      <article>
20          <h2>Título</h2>
21          <p>Texto do artigo</p>
22          <aside>
23              conteúdo periférico do artigo
24          </aside>
25      </article>
26  </section>
27  <aside>
28      conteúdo periférico do site
29  </aside>
30 </main>
31 <footer>
32     conteúdo do rodapé
33 </footer>

```

Analise o diagrama do lado esquerdo e o código do lado direito da imagem acima. Veja a hierarquia entre os elementos e quais deles estão dentro um do outro.

Sombras nas caixas

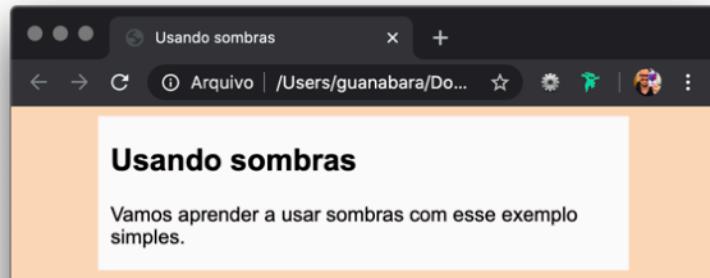
As sombras são muito úteis para dar volume, para dar a sensação de que as caixas estão ali realmente. Para exemplificar, vamos criar o seguinte código base:

```

7   <style>
8     body {
9       font: 1em Arial, Helvetica, sans-serif;
10      background-color: #peachpuff;
11    }
12    article#caixa {
13      background-color: #snow;
14      width: 400px;
15      margin: auto;
16      padding: 1px 10px;
17      /* sombra aqui */
18    }
19  </style>
20 </head>
21 <body>
22   <article id="caixa">
23     <h1>Usando sombras</h1>
24     <p>Vamos aprender a usar sombras com esse
25       exemplo simples. </p>
26   </article>
27 </body>
28 </html>

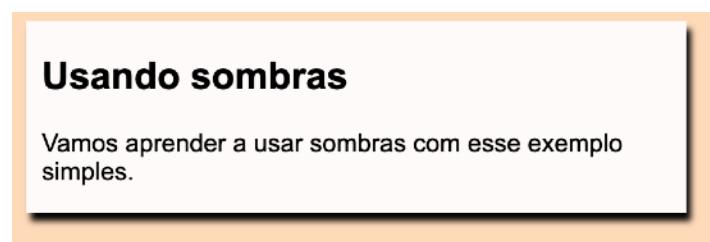
```

Olhando para o corpo da página, temos apenas um `<article>` (**linha 22**) com um breve conteúdo. A configuração de estilo (**linha 7** em diante) faz com que esse artigo seja configurado como uma pequena caixa centralizada. O resultado visual está apresentado a seguir:



Para criar uma sombra nessa caixa, vamos adicionar uma declaração especial na **linha 17**, substituindo o comentário que deixei lá.

```
box-shadow: 3px 5px 4px black;
```



Veja que uma sombra bem forte já pode ser percebida, assim que adicionamos a propriedade `box-shadow` e seus quatro valores. A ordem é sempre essa:

1. **Deslocamento horizontal** (*h-offset*): quanto a sombra vai andar para o lado direito (valores negativos causam deslocamento para a esquerda)
2. **Deslocamento vertical** (*v-offset*): quanto a sombra vai andar para baixo (valores negativos causam deslocamento para cima)
3. **Embaçamento** (*blur*): quanto a sombra vai se espalhar pelo fundo
4. **Cor** (*color*): cor da sombra. É possível usar transparência.



MUITO CUIDADO! Não exagere no uso de sombras, pois elas podem tornar o seu efeito visual muito pesado. Evite também usar sombras coloridas. Olhe ao seu redor e perceba que as sombras são sempre pretas. Use cores `rgba()` para obter uma transparência que cause efeitos mais suaves.

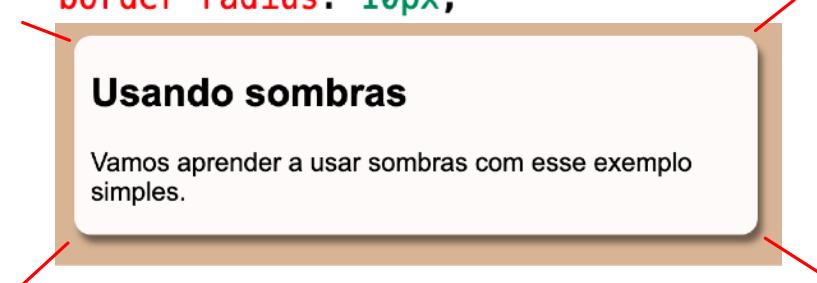
Bordas decoradas

As bordas das caixas não precisam ser sempre retangulares e podem ter alguns detalhes especiais. Vamos usar o mesmo exercício que estamos criando desde o item anterior onde aprendemos a usar sombras.

Vértices arredondados

Podemos arredondar os vértices usando uma declaração simples usando a propriedade `border-radius`. Adicione o seguinte comando ao seletor do artigo do exemplo que estamos criando:

```
border-radius: 10px;
```



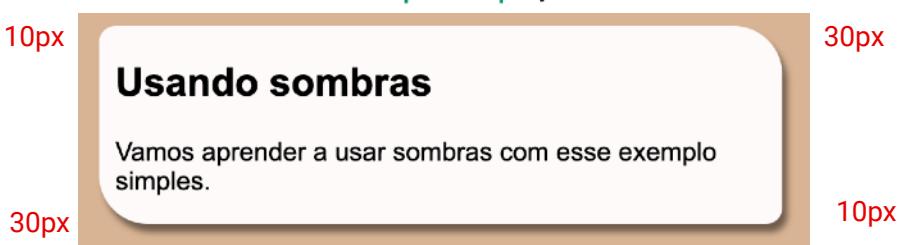
Na declaração acima, todos os vértices foram levemente arredondados (10px) de forma simétrica. Se for necessário, podemos indicar quatro medidas diferentes, uma para cada vértice. Olhe atentamente para o resultado abaixo e perceba que cada ponta está diferente.

```
border-radius: 10px 20px 30px 40px;
```



Assim como fizemos com as margens, também é possível indicar apenas dois valores, o que vai agir em vértices intercalados, partindo do canto superior esquerdo.

```
border-radius: 10px 30px;
```



Tenho desafios pra você!

Lá no repositório, além do material em PDF e dos códigos dos exercícios 100% disponíveis, também disponibilizamos alguns **desafios** que devem ser resolvidos. Esses desafios não incluem o código original e você deve tentar chegar à resposta sem copiar nenhum código.

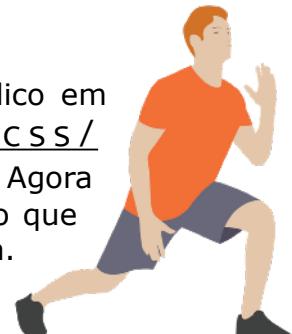
Com todo o conteúdo que vimos até essa aula, você já pode resolver o **desafio d010**. Acesse o repositório público, abra a área do curso de HTML+CSS e clique no link de acesso aos desafios. Manda ver! Só não fica pedindo a resposta! Você consegue resolver isso sozinho(a)!



Repositório em: <https://gustavoguanabara.github.io>

Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/exercicios/> e executar o **exercício 017** no seu computador. Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



Quer acompanhar tudo em vídeo?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo faz parte da playlist completa onde você encontra os **Módulos 1 e 2** do **Curso de HTML5 e CSS3**, completamente gravado com base nesse material.



Módulo 1 do curso: https://www.youtube.com/playlist?list=PLHz_AreHm4dkZ9-atkcmcBaMZdmLHft8n

Módulo 2 do curso: https://www.youtube.com/playlist?list=PLHz_AreHm4dlUpEXkY1AyVLQGcpSgVF8s

Teste seus conhecimentos

Terminou de ler esse capítulo e já acompanhou todos os vídeos e referências externas que indicamos? Pois agora, responda a essas 10 perguntas objetivas e marque em cada uma delas a única opção verdadeira. Aí sim, você vai poder comprovar que realmente entendeu o conteúdo.



1. A grande maioria dos elementos HTML são considerados como contêineres que podem guardar conteúdos ou outros contêineres. Este conceito é conhecido pelo termo em Inglês:

- A container box
- B box model
- C container content
- D box content

2. Todo contêiner possui uma propriedade chamada box-size, que é composto por duas medidas: largura e altura, representados respectivamente por:

- A width e height
- B height e width
- C width e weight
- D weight e height

3. Todos os elementos HTML que são caixas podem ter um espaço interno, que fica entre a borda e o conteúdo e se chama _____. Eles também podem ter um espaço externo, que fica além da borda e se chama _____. Qual das opções abaixo é a única que preenche as lacunas na ordem correta?

- A margin / padding
- B outline / padding
- C outline / margin
- D padding / margin

4. A shorthand border agrupa a seguir as propriedades, nesta ordem:

- A border-size + border-type + border-color
- B border-width + border-type + border-color
- C border-width + border-style + border-color
- D border-height + border-style + border-color

5. A shorthand padding, quando especificada com quatro parâmetros, configura as seguintes propriedades, na ordem:

- A padding-top, padding-right, padding-bottom, padding-left
- B padding-top, padding-left, padding-bottom, padding-right
- C padding-top, padding-bottom, padding-left, padding-right
- D padding-left, padding-right, padding-top, padding-bottom

6. De acordo com o comportamento de uma caixa, ela pode ser classificada como:

- A box-level ou inline-level
- B box-level ou block-level
- C block-level ou inline-level
- D container-level ou box-level

7. Qual dos elementos a seguir é o único que não é considerado grouping tag?

- A <aside>
- B <nav>
- C <head>
- D <footer>

8. "Delimita um conteúdo periférico e complementar ao conteúdo principal de um artigo ou seção". Esta descrição se encaixa melhor no conceito de qual elemento HTML5?

- A <main>
- B <article>
- C <session>
- D <aside>

9. Para aplicar sombra em um elemento HTML5 exibido como uma caixa, usamos qual propriedade?

- A shadow
- B block-shadow
- C box-shadow
- D shadow-box

10. Para arredondar os vértices de um elemento HTML5 exibido como uma caixa, usamos qual propriedade?

- A border-radius
- B box-radius
- C border-vertex
- D vertex-box

Suas anotações

Não guarde conhecimento. Ele é livre. Compartilhe o seu e veja ele se espalhando pelo mundo 