CSS3

Para começar a dar mais poder às CSS, criando estilos personalizados, precisamos aprender a utilizar os seletores de id (#) e de class (.) de maneira eficiente. Ao criar nosso conteúdo em HTML, podemos **identificar** um determinado elemento único com um id, ou **agrupar** elementos múltiplos que tenham características semelhantes com um class. Vamos olhar um exemplo simples de documento HTML, com propriedades identificadoras:

```
22  <body>
23      <h1 id="titulo-principal">Aprenda Desenvolvimento Web</h1>
24      <h1>Aprenda HTML</h1>
25      <h2 class="topico">Semântica Web</h2>
26      <p class="texto">Lorem ipsum dolor sit, amet consectetur adipisicing elit.
        Optio, quibusdam? Tempore sequi commodi natus, vitae enim sed molestias. Rem,
        voluptas illum. Ut, debitis error quas explicabo corporis officia fugit.
        Doloribus.</p>
27      <h1>Aprenda CSS</h1>
28      <h2 class="topico">Estilos Web</h2>
29      <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Dolores, repellat
        dicta. In distinctio quasi nesciunt blanditiis. Quas expedita et architecto
        alias laudantium, facere neque quo facilis! Ut est perferendis sed!</p>
30  </body>
```

Olhando atentamente para a **linha 23**, vimos que o <h1> principal se diferencia dos demais que estão nas **linhas 24 e 27**, pois ele possui um id.

Agora foque sua atenção nas **linhas 25 e 28** e perceba que os dois títulos <h2> possuem o mesmo class.

Um id vai **identificar** um elemento único dentro da página atual. Essa identificação vai nos permitir criar um estilo especial para um elemento isolado. Já um class vai identificar uma **classe** à qual um ou mais elementos pertençam, compartilhando características em comum a todos os que façam parte desse grupo.

SE LIGA NA REGRA: Em um mesmo documento HTML, só podemos usar um id para um único elemento, o que significa que não podemos ter dois elementos com um mesmo id dentro de uma mesma página. Porém, podemos atribuir um mesmo class para vários elementos que possuam essa mesma característica.

Em seguida, baseado no código HTML visto anteriormente, vamos criar um estilo local criando uma área <style> dentro da seção <head> do documento atual.

```
7     <style>
8         body {
9             font: normal 1em Arial, Helvetica, sans-serif;
10        }
11
12        h1 {
13            font-size: 2em;
14            color: darkgreen;
15        }
16
17        h2 {
18            color: green;
19        }
20    </style>
```

O que fizemos foi criar configurações simples de estilo baseadas nos elementos body, h1 e h2 do nosso HTML. Até o momento, não fizemos nenhuma configuração personalizada para nenhum seletor. O resultado é bem previsível:

Perceba no resultado ao lado que os dois primeiros títulos possuem apresentações idênticas, mesmo que o primeiro deles possua um id.

Isso aconteceu porque não fizemos nenhuma configuração específica para ele. Não adianta colocar apenas id ou class em um elemento e não personalizar o resultado visual usando seletores personalizados.

Agora vamos adicionar algumas declarações e seletores personalizados, ainda dentro da área <style> que criamos:

No código ao lado, criamos seletores com simbologias novas. Isso porque não são seletores para elementos HTML, e sim seletores para id e class. Na **linha 21**, criamos um seletor personalizado para o identificador titulo-principal, que está atribuído ao h1. Você deve ter percebido que colocamos um símbolo # antes do id.

Já as **linhas 27 e 31** são seletores que receberão declarações para as classes topico e texto. Todos os seletores personalizados para uma class são identificados por um ponto que vem antes do nome da classe

```
21      #titulo-principal {
22          background-color: darkgreen;
23          color: white;
24          text-align: center;
25      }
26
27      .topico {
28          text-align: right;
29      }
30
31      .texto {
32          text-align: justify;
33      }
```

MAIS UMA REGRA: Todo id em HTML é identificado nas CSS por um símbolo #. Toda class em HTML é identificada nas CSS por um ponto.

Ao adicionar os seletores personalizados ao nosso estilo local, o resultado fica bem diferente:

Aprenda Desenvolvimento Web **Aprenda HTML** **Semântica Web** Lorem ipsum dolor sit, amet consectetur adipisicing elit. Optio, quibusdam? Tempore sequi commodi natus, vitae enim sed molestias. Rem, voluptas illum. Ut, debitis error quas explicabo corporis officia fugit. Doloribus. **Aprenda CSS** **Estilos Web** Lorem ipsum dolor sit, amet consectetur adipisicing elit. Dolores, repellat dicta. In distinctio quasi nesciunt blanditiis. Quas expedita et architecto alias laudantium, facere neque quo facilis! Ut est perferendis sed!	**Aprenda Desenvolvimento Web** **Aprenda HTML** **Semântica Web** Lorem ipsum dolor sit, amet consectetur adipisicing elit. Optio, quibusdam? Tempore sequi commodi natus, vitae enim sed molestias. Rem, voluptas illum. Ut, debitis error quas explicabo corporis officia fugit. Doloribus. **Aprenda CSS** **Estilos Web** Lorem ipsum dolor sit, amet consectetur adipisicing elit. Dolores, repellat dicta. In distinctio quasi nesciunt blanditiis. Quas expedita et architecto alias laudantium, facere neque quo facilis! Ut est perferendis sed!

Analise os dois resultados acima e perceba as diferenças. Note que apenas o título h1 identificado como titulo-principal ganhou uma formatação diferente e os demais títulos de mesmo nível se mantiveram como antes. Inclusive, compare o resultado diferente obtido entre o primeiro e o segundo parágrafos. Apenas o primeiro ficou justificado. Isso foi por conta da aplicação de uma class apenas ao primeiro (volte lá no código HTML inicial e confira que o segundo parágrafo não possui class).

Propriedades herdadas

Discutimos três técnicas usadas para aplicar estilos aos nossos documentos HTML: **inline**, **local** e **externa** os estilos de cada serão combinados nessa ordem:

1º lugar: CSS externo;
2º lugar: CSS interno/local;
3º lugar: CSS inline.

Sendo assim, caso haja duplicidade nas configurações, o que fica valendo no final é a propriedade especificada nas configurações inline. Cada camada de aplicação de estilos vai adicionando propriedades novas e sobrescrevendo as configurações do nível anterior. Quando nós aplicamos configurações personalizadas de id ou de class, também vai existir uma combinação de configurações, como você já deve ter percebido ao analisar o exemplo que apresentamos nas páginas anteriores.

Voltando ao exemplo dado anteriormente (que vou replicar aqui para facilitar o entendimento), logo no início do corpo, tínhamos dois elementos h1:

Logo em seguida, na área de estilo, criamos uma configuração especial para os elementos h1:

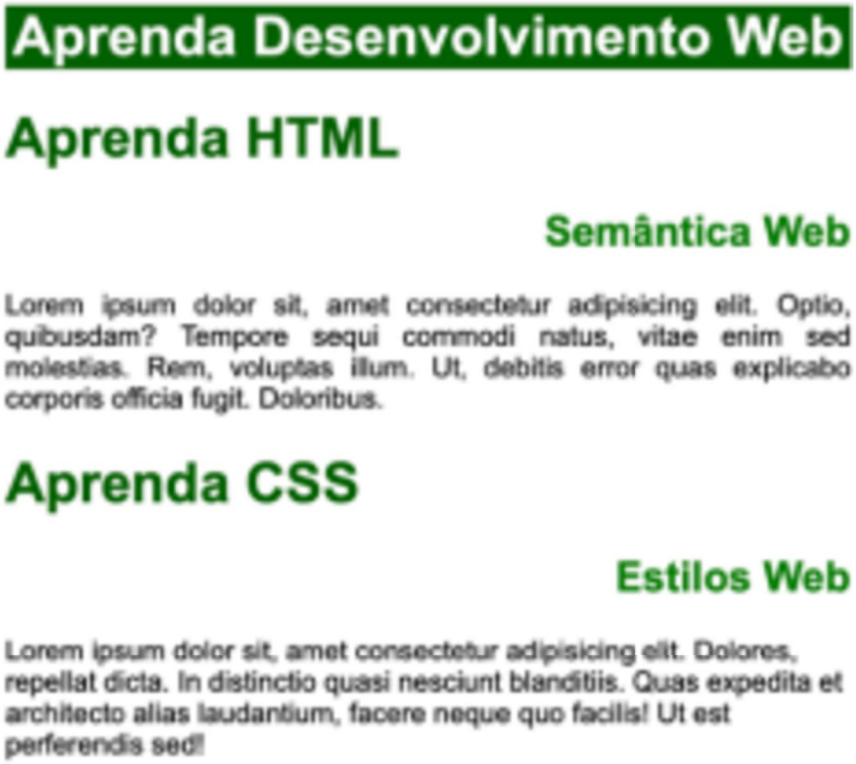
```
22  <body>
23      <h1 id="titulo-principal">Aprenda Desenvolvimento Web</h1>
24      <h1>Aprenda HTML</h1>
25      <h2 class="topico">Semântica Web</h2>
26      <p class="texto">Lorem ipsum dolor sit, amet consectetur adipisicing elit.
        Optio, quibusdam? Tempore sequi commodi natus, vitae enim sed molestias. Rem,
        voluptas illum. Ut, debitis error quas explicabo corporis officia fugit.
        Doloribus.</p>
27      <h1>Aprenda CSS</h1>
28      <h2 class="topico">Estilos Web</h2>
29      <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Dolores, repellat
        dicta. In distinctio quasi nesciunt blanditiis. Quas expedita et architecto
        alias laudantium, facere neque quo facilis! Ut est perferendis sed!</p>
30  </body>
```

Essas duas configurações de tamanho e cor da fonte serão aplicadas aos dois títulos, já que ambos são do tipo h1. Por fim, adicionamos uma configuração específica para o elemento que tem um id personalizado:

Dessa maneira, o primeiro h1 do código HTML (aquele que tem um id) ganhará mais três configurações - cor de fundo, cor da fonte e alinhamento do texto - graças a esse seletor extra. Note que na primeira configuração CSS dos elementos h1, dissemos que a cor era darkgreen.

Porém, quando aplicamos a segunda configuração de color no seletor #titulo principal, ela vai sobrescrever o valor da propriedade para white. As outras duas

Pseudo-classes e pseudo-elementos



Uma pseudo-classe CSS é uma palavra-chave adicionada às declarações de um seletor após um sinal de dois pontos e especificam um estado especial de um elemento. Existem várias pseudo-classes para estilos, podemos citar :hover, :visited, :active, :checked, :empty e :focus.

Já um pseudo-elemento CSS é uma palavra-chave adicionada às declarações de um seletor após dois sinais de dois pontos e permitem que você formate um pedaço específico do elemento referenciado. Os principais pseudo-elementos usados nas CSS são ::before, ::after, ::first-letter, ::first-line.

Vamos começar criando um exemplo bem simples e bastante usado para exemplificar o uso de pseudo-classes e pseudo-elementos: a personalização de links.

```
43  <body>
44      <h1>Personalizando um link</h1>
45      <p>
46          Para aprender HTML5 e CSS3, acesse o
47          <a href="https://gustavoguanabara.github.io">
48              GitHub do Guanabara
49          </a>
50      </p>
51      <p>
52          Se quiser conhecer mais sobre os padrões, visite o site do
53          <a href="https://www.w3c.br">Escritório do W3C no Brasil</a>
54      </p>
55      <p>Para acompanhar os cursos, acesse o
56          <a href="https://www.youtube.com/cursoemvideo/" class="especial">
57              canal do CursoemVideo
58          </a>
59      </p>
60  </body>
```

Começando pelo corpo do documento, contendo só o HTML, vamos criar um texto com um título e três parágrafos, com três links. Note que o último link recebeu a atribuição de uma classe específica. O resultado visual está sendo apresentado a seguir:

Vamos alterar essa apresentação padrão com nossas configurações de estilo. Crie a área <style> dentro de <head> e coloque os seguintes seletores.

```
 8          body {
 9              font: normal 1em Arial, Helvetica, sans-serif;
10          }
11
12          a {
13              font-weight: bold;
14              text-decoration: none;
15              color: red;
16          }
17
18          a:visited {
19              color: darkred;
20          }
21
22          a:hover {
23              text-decoration: underline;
24          }
```

Nas declarações acima, criamos três configurações para os links: a primeira (**linha 12**) para links inéditos (não visitados), colocando o texto em negrito, removendo o sublinhado e colocando-os em cor vermelha.

No segundo seletor para links (**linha 18**), configuramos as âncoras já visitadas (com a pseudo-classe :visited). Nele, apenas mudamos a cor para vermelho escuro.

Já na terceira declaração (**linha 22**), fizemos configurações para todos os links, quando passarmos o mouse por cima (pseudo-classe :hover) e o sublinhado volta a aparecer.

O resultado visual está apresentado abaixo, compare com o resultado anterior e veja as diferenças:

Por fim, vamos adicionar algumas configurações relacionadas à classe especial, criada no documento HTML, para o terceiro link.

Na primeira declaração do código acima (**linha 26**), dizemos que o link da classe especial vai ter letra branca, fundo preto e perderá o sublinhado apenas quando movermos o mouse sobre ele.

Nas próximas declarações (**linhas 32 e 37**), vamos adicionar um símbolo » antes e outro símbolo « depois usando os pseudo-elementos ::before e ::after, respectivamente.

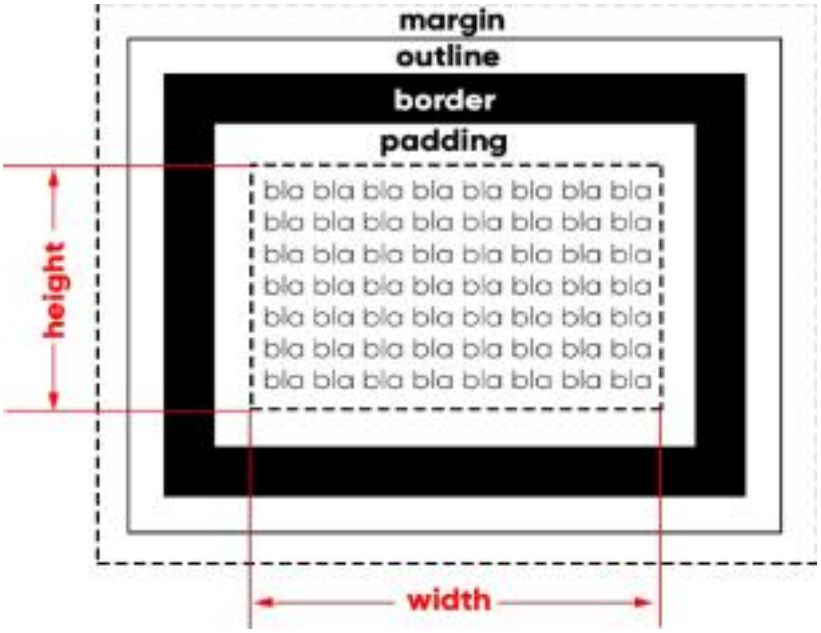
MODELO DE CAIXAS

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.

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.

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, somemos 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 = **460px de largura** e 50 + 10 + 20 + 50 + 20 + 10 + 50 = **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?

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).

`border-width: 10px;`
`border-style: solid;`
`border-color: red;`
`outline-width: 30px;`
`outline-style: solid;`
`outline-color: blue;`
	`border: 10px solid red;`
	`outline: 30px solid blue;`

Preenchimento e margem personalizados

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:

`padding-top: 10px;`
`padding-right: 15px;`
`padding-bottom: 20px;`
`padding-left: 25px;`
 	
`margin-top: 0px;`
`margin-right: 10px;`
`margin-bottom: 20px;`
`margin-left: 30px;`
	`padding: 10px 15px 20px 25px;`
	`margin: 0px 10px 20px 30px;`

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

`padding-top: 10px;`
`padding-right: 20px;`
`padding-bottom: 10px;`
`padding-left: 20px;`
 	
`margin-top: 0px;`
`margin-right: 15px;`
`margin-bottom: 0px;`
`margin-left: 15px;`
	`padding: 10px 20px;`
	`margin: 0px 15px;`

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;
```

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 e suas variações semânticas modernas da HTML5, como <main>, <section>, <aside>, etc.

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

<address> <article> <aside> <blockquote> <canvas> <dd> <dl> <dt> <fieldset> <figcaption>

<figure> <footer> <form> <h1> - <h6> <header> <hr> <main> <nav> <noscript> <p> <pre>

<section> <table> <tfoot> <video>

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:

<a> <abbr> <acronym> <b> <bdo>
 <button> <cite> <code> <dfn> <i>

<img> <input> <kbd> <label> <map> <object> <output> <q> <samp> <script> <select>

<small>   <sub> <textarea> <tt> <var>

Grouping Tags e Semantic Tags

A linguagem HTML padrão tinha apenas duas tags de agrupamento genérico: a 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 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.**#DevWeb** - Capítulo 16 *prof. Gustavo Guanabara* Página 10 de 17
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:

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.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:

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.

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.