



CAPÍTULO 5

CSS Avançado

"Com o conhecimento nossas dúvidas aumentam" -- Johann Goethe

Desde o surgimento do CSS, os desenvolvedores front-end utilizam diversas técnicas para alterar a exibição dos elementos no navegador. Mesmo assim algumas coisas eram impossíveis de se conseguir utilizando somente CSS. Conhecendo o comportamento dos navegadores ao exibir um elemento (ou um conjunto de elementos) e como as propriedades do CSS agem ao modificar um elemento é possível obter resultados impressionantes.

O CSS3 agora permite coisas antes impossíveis como elementos com cor ou fundo gradiente, sombras e cantos arredondados. Antes só era possível atingir esses resultados com o uso de imagens e às vezes até com um pouco de JavaScript.

A redução do uso de imagens traz grandes vantagens quanto à performance e quantidade de tráfego de dados necessária para a exibição de uma página.

5.1 SELETORES AVANÇADOS

Os seletores CSS mais comuns e tradicionais são os que já vimos: por ID, por classes e por descendência.

No entanto, há muitos outros seletores novos que vão entrando na especificação e que são bastante úteis. Já vimos alguns, como os seletores de atributo que usamos anteriormente. Vejamos outros.

Seletor de irmãos

Veja o seguinte HTML, que simula um texto com vários parágrafos, títulos e subtítulos no meio do documento:

```
<article>
  <h1>Título</h1>
  <p>Início</p>

  <h2>Subtítulo</h2>
  <p>Texto</p>
  <p>Mais texto</p>
</article>
```

Como faremos se quisermos estilizar de uma certa maneira todos os parágrafos após o subtítulo?

O seletor de **irmãos** (*siblings*) (~) serve pra isso! Ele vem do CSS3 e funciona em todos os navegadores modernos (e no IE7 pra frente).

```
h2 ~ p {
  font-style: italic;
}
```

Isso indica que queremos selecionar *todos* os **p** que foram precedidos por algum **h2** e são irmãos do subtítulo (ou seja, estão sob a mesma tag pai). No HTML anterior, serão selecionados os dois últimos parágrafos (*Texto* e *Mais texto*).

Seletor de irmão adjacente

Ainda com o HTML anterior, o que fazer se quisermos selecionar apenas o parágrafo imediatamente seguinte ao subtítulo? Ou seja, é um **p** irmão do

h2 mas que aparece logo na sequência.

Fazemos isso com o seletor de irmão adjacente - *adjacent sibling*:

```
h2 + p {  
    font-variant: small-caps;  
}
```

Nesse caso, apenas o parágrafo *Texto* será selecionado. É um irmão de h2 e aparece logo depois do mesmo.

Esse seletor faz parte da especificação CSS2.1 e tem bom suporte nos navegadores modernos, incluindo o IE7.

Seletor de filho direto

Se tivermos o seguinte HTML com títulos e seções de um artigo:

```
<article>  
  <h1>Título principal</h1>  
  
  <section>  
    <h1>Título da seção</h1>  
  </section>  
</article>
```

Queremos deixar o título principal de outra cor. Como fazer? O seletor de nome de tag simples não vai resolver:

```
/* vai pegar todos os h1 da página */  
h1 {  
    color: blue;  
}
```

Tentar o seletor de hierarquia também não vai ajudar:

```
/* vai pegar todos os h1 do article, incluindo o de dentro da :  
article h1 {
```

```
    color: blue;
}
```

Entra aí o **seletor de filho direto** (>) do CSS2.1 e suportado desde o IE7 também.

```
/* vai pegar só o h1 principal, filho direto de article e não o
article > h1 {
    color: blue;
}
```

Negação

Imagine o seguinte HTML com vários parágrafos simples:

```
<p>Texto</p>
<p>Outro texto</p>
<p>Texto especial</p>
<p>Mais texto</p>
```

Queremos fazer todos os parágrafos de cor cinza, exceto o que tem o texto especial. Precisamos destacá-lo de alguma forma no HTML para depois selecioná-lo no CSS. Uma classe ou ID resolve:

```
<p>Texto</p>
<p>Outro texto</p>
<p class="especial">Texto especial</p>
<p>Mais texto</p>
```

Mas como escrever o CSS? Queremos mudar a cor dos parágrafos que *não têm a classe especial*. Um jeito seria mudar a cor de todos e sobrescrever o especial depois:

```
p {
    color: gray;
}

p.especial {
    color: black; /* restaura cor do especial */
```

```
}
```

No CSS3, há uma outra forma, usando o **seletor de negação**. Ele nos permite escrever um seletor que pega elementos que não batem naquela regra.

```
p:not(.especial) {  
  color: gray;  
}
```

Isso diz que queremos todos os parágrafos que não têm a classe especial. A sintaxe do **:not()** recebe como argumento algum outro seletor simples (como classes, IDs ou tags).

Essa propriedade do CSS3 possui suporte mais limitado no IE, somente a partir da versão 9 (nos outros navegadores não há problemas).

Agora é a melhor hora de aprender algo novo

alura

Se você está gostando dessa apostila, certamente vai aproveitar os **cursos online** que lançamos na plataforma **Alura**. Você estuda a qualquer momento com a **qualidade** Caelum. Programação, Mobile, Design, Infra, Front-End e Business! Ex-aluno da Caelum tem 15% de desconto, siga o link!

[Conheça a Alura Cursos Online.](#)

5.2 PSEUDO-CLASSES

Pegue o seguinte HTML de uma lista de elementos:

```
<ul>  
  <li>Primeiro item</li>  
  <li>Segundo item</li>
```

```
<li>Terceiro item</li>
<li>Quarto item</li>
</ul>
```

E se quisermos estilizar elementos específicos dessa lista? Por exemplo, o primeiro elemento deve ter cor vermelha e o último, azul. Com esse HTML simples, usando apenas os seletores que vimos até agora, será bem difícil.

A solução mais comum seria adicionar classes ou IDs no HTML para selecioná-los depois no CSS:

```
<ul>
  <li class="primeiro">Primeiro item</li>
  <li>Segundo item</li>
  <li>Terceiro item</li>
  <li class="ultimo">Quarto item</li>
</ul>
```

Agora é fácil usar cores diferentes para o primeiro e último itens da lista.

Mas essa técnica exigiu alteração no HTML e exige que lembremos de colocar a classe correta, no ponto correto, toda vez que fizermos mudanças nos itens da lista.

O CSS tem um recurso chamado de **pseudo-classes** que são como classes CSS já pré-definidas para nós. É como se o navegador já colocasse certas classes por padrão em certos elementos, cobrindo situações comuns como essa de selecionar o primeiro ou o último elemento.

Há duas pseudo-classes do CSS3 que representam exatamente o primeiro elemento filho de outro (**first-child**) e o último elemento filho (**last-child**). Essas classes já estão definidas, não precisamos aplicá-las em nosso HTML e podemos voltar para o HTML inicial:

```
<ul>
  <li>Primeiro item</li>
  <li>Segundo item</li>
```

```
<li>Terceiro item</li>
<li>Quarto item</li>
</ul>
```

No CSS, podemos usar pseudo-classes quase da mesma forma que usaríamos nossas classes normais. Repare que para diferenciar um tipo do outro, mudou-se o operador de ponto para dois pontos:

```
li:first-child {
  color: red;
}

li:last-child {
  color: blue;
}
```

O suporte a esses seletores é completo nos navegadores modernos. O `first-child` vem desde o IE7, Firefox 3 e Chrome 4. E (estranhamente), o `last-child` só a partir do IE9 mas desde o Firefox 1 e Chrome 1.

nth-child

Um seletor ainda mais genérico do CSS3 é o `:nth-child()` que nos permite passar o índice do elemento. Por exemplo, podemos pegar o terceiro item com:

```
li:nth-child(3) { color: yellow; }
```

Porém, o mais interessante é que o `nth-child` pode receber uma expressão aritmética para indicar quais índices selecionar. É fácil fazer uma lista zebra, com itens ímpares de uma cor e pares de outra:

```
/* elementos pares */
li:nth-child(2n) { color: green; }

/* elementos ímpares */
```

```
li:nth-child(2n+1) { color: blue; }
```

O suporte existe a partir do IE9, Firefox 3.5 e Chrome 1.

Pseudo classes de estado

Queremos mudar a cor de um link quando o usuário passa o mouse por cima. Ou seja, queremos mudar seu visual a partir de um evento do usuário (no caso, passar o mouse em cima).

Uma solução ingênua seria criar um código JavaScript que adiciona uma classe nos links quando o evento de mouseover acontece (e remove a classe no mouseout).

Entretanto, o CSS possui excelentes **pseudo-classes que representam estados dos elementos** e, em especial, uma que representa o momento que o usuário está com o mouse em cima do elemento, a **:hover**.

É como se o navegador aplicasse uma classe chamada **hover** automaticamente quando o usuário passa o mouse em cima do elemento e depois retirasse a classe quando ele sai. Tudo sem precisarmos controlar isso com JavaScript.

```
/* seleciona o link no exato momento em que passamos o mouse p  
a:hover {  
    background-color:#ff00ff;  
}
```

Podemos usar hover em todo tipo de elemento, não apenas links. Mas os links ainda têm outras pseudo-classes que podem ser úteis:

```
/* seleciona todas as âncoras que têm o atributo "href", ou se  
a:link {  
    background-color:#ff0000;  
}
```



```
/* seleciona todos os links cujo valor de "href" é um endereço
a:visited {
    background-color:#00ff00;
}

/* seleciona o link no exato momento em que clicamos nele */
a:active {
    background-color:#0000ff;
}
```

5.3 PSEUDO ELEMENTOS

Pseudo-classes nos ajudam a selecionar elementos com certas características sem termos que colocar uma classe manualmente neles. Porém, o que fazer quando precisamos selecionar certo tipo de conteúdo que nem elemento tem?

Exemplo: temos um texto num parágrafo:

```
<p>A Caelum tem os melhores cursos!</p>
```

Queremos dar um estilo de revista ao nosso texto e estilizar apenas a *primeira letra* da frase com uma fonte maior. Como fazer para selecionar essa letra? A solução ingênua seria colocar um elemento ao redor da letra para podermos selecioná-la no CSS:

```
<p><span>A</span> Caelum tem os melhores cursos!</p>
```

HTML confuso e difícil de manter.

O CSS apresenta então a ideia de **pseudo-elementos**. São elementos que não existem no documento mas podem ser selecionados pelo CSS. É como se houvesse um elemento lá!

Podemos voltar o HTML inicial:

```
<p>A Caelum tem os melhores cursos!</p>
```

E no CSS:

```
p::first-letter {  
    font-size: 200%;  
}
```

Temos ainda outro pseudo-elemento para selecionar a primeira linha apenas em um texto grande:

```
p::first-line {  
    font-style: italic;  
}
```

Novos conteúdos

Há ainda um outro tipo de pseudo-elemento mais poderoso que nos permite gerar conteúdo novo via CSS. Imagine uma lista de links que queremos, visualmente, colocar entre colchetes:

```
[ Link 1 ]  
[ Link 2 ]  
[ Link 3 ]
```

Podemos, claro, apenas escrever os tais colchetes no HTML. Mas será que o conteúdo é semântico? Queremos que esses colchetes sejam indexados pelo Google? Queremos que sejam lidos como parte do texto pelos leitores de tela?

Talvez não. Pode ser um conteúdo apenas visual. Podemos gerá-lo com CSS usando os pseudo-elementos **after** e **before**.

O HTML seria simples:

```
<a href="...">Link1</a>  
<a href="...">Link2</a>  
<a href="...">Link3</a>
```

E no CSS:

```
a::before {
```

```
    content: '[';
}

a::after {
    content: ' ]';
}
```

Ou ainda, imagine que queremos colocar a mensagem (**externo**) ao lado de todos os links externos da nossa página. Usando pseudo-elementos e seletores de atributo, conseguimos:

```
a[href^=http://]::after {
    content: ' (externo)';
}
```

Isso pega todos os elementos `<a>` que começam com `http://` e coloca a palavra (**externo**) depois.

5.4 EXERCÍCIOS: SELETORES, PSEUDO-CLASSES E PSEUDO-ELEMENTOS

1.

Vamos alterar nossa página de *Sobre*, a **sobre.html**. Queremos que as primeiras letras dos parágrafos fiquem em negrito.

Adicione no arquivo **sobre.css** o seletor com pseudo-elemento `::first-letter` para isso.

```
p::first-letter {
    font-weight: bold;
}
```

Teste a página no navegador!

2.

Repare que os parágrafos nessa página *Sobre* têm uma indentação no início. Agora queremos **remover apenas a indentação do primeiro parágrafo** da página.

Poderíamos colocar uma classe no HTML. Ou, melhor ainda, sabendo que esse é o primeiro parágrafo (<p>) depois do título (<h1>), usar o **seletor de irmão adjacente**.

Acrescente ao **sobre.css**:

```
h1 + p {  
    text-indent: 0;  
}
```

Teste novamente no navegador.

3.

Podemos ainda usar o pseudo-elemento `::first-line` para alterar o visual da primeira linha de todos os parágrafos. Por exemplo, transformando-a em **small-caps** usando a propriedade `font-variant`:

```
p::first-line {  
    font-variant: small-caps;  
}
```

Teste de novo no navegador.

A MIRROR FASHION É A MAIOR EMPRESA COMÉRCIO ELETRÔNICO NO SEGMENTO DE MODA EM TODO O MUNDO. FUNDADA EM 1932, possui filiais em 124 países, sendo líder de mercado com mais de 90% de participação em 118 deles.

NOSSO CENTRO DE DISTRIBUIÇÃO FICA EM [JACAREZINHO, NO PARANÁ](#). DE LÁ, SAEM 48 AVIÕES QUE DISTRIBUEM NOSSOS produtos às casas do mundo todo. Nosso centro de distribuição:

4.

Vamos voltar a mexer na página **index.html** do nosso site.

Temos o menu superior (**.menu-opcoes**) que é uma lista de links. Podemos melhorar sua usabilidade alterando seus estados quando o usuário passar o mouse (**:hover**) e quando clicar no item (**:active**).

Adicione ao arquivo **estilos.css**:

```
.menu-opcoes a:hover {  
    color: #007dc6;  
}  
  
.menu-opcoes a:active {  
    color: #867dc6;  
}
```

Teste o menu passando o mouse e clicando nas opções (segure um pouco o clique para ver melhor o efeito).

5.

O **hover** é útil em vários cenários. Um interessante é fazer um menu que abre e fecha em puro CSS.

Temos já o nosso **.menu-departamentos** na esquerda da página com várias categorias de produtos. Queremos adicionar *subcategorias* que aparecem apenas quando o usuário passar o mouse.



Hoje, o menu é um simples `` com vários itens (``) com links dentro:

```
<li><a href="#">Blusas e Camisas</a></li>
```

Vamos adicionar no **index.html** uma **sublista** de opções **dentro** do `` de *Blusas e Camisas*, dessa forma:

```
<li>
  <a href="#">Blusas e Camisas</a>
  <ul>
    <li><a href="#">Manga curta</a></li>
    <li><a href="#">Manga comprida</a></li>
    <li><a href="#">Camisa social</a></li>
    <li><a href="#">Camisa casual</a></li>
  </ul>
</li>
```

Por padrão, queremos que essa sublista esteja escondida (**display: none**). Quando o usuário passar o mouse (**:hover**), queremos exibi-la (**display: block**).

Altere o arquivo **estilos.css** com essa lógica.

```
.menu-departamentos li ul {
  display: none;
}

.menu-departamentos li:hover ul {
  display: block;
}

.menu-departamentos ul ul li {
  background-color: #dcdcdc;
}
```

Teste a página e a funcionalidade do menu.

6.

Para ajudar a diferenciar os links dos submenus, queremos colocar um **traço na frente**. Podemos alterar o HTML colocando os traços - algo visual

e não semântico -, ou podemos **gerar esse traço** via CSS com pseudo-elementos.

Use o `::before` para injetar um conteúdo novo via propriedade `content` no CSS:

```
.menu-departamentos li li a::before {  
  content: '- '  
}
```

Teste a página.

Veja os exercícios opcionais a seguir com outras possibilidades.

Editora Casa do Código com livros de uma forma diferente



Editoras tradicionais pouco ligam para ebooks e novas tecnologias. Não dominam tecnicamente o assunto para revisar os livros a fundo. Não têm anos de experiência em didáticas com cursos.

Conheça a **Casa do Código**, uma editora diferente, com curadoria da **Caelum** e obsessão por livros de qualidade a preços justos.

[Casa do Código, ebook com preço de ebook.](#)

5.5 EXERCÍCIOS OPCIONAIS

1.

A propriedade `content` tem muitas variações. Uma variação simples, mas útil, é usar caracteres *unicode* para injetar símbolos mais interessantes.

Altere o seletor que fizemos no exercício anterior:

```
.menu-departamentos li li a::before {  
  content: '\272A';  
  padding-right: 5px;  
}
```

Para os títulos dos painéis podemos adicionar um *unicode* diferente:

```
.painel h2::before {  
  content: '\2756';  
  padding-right: 5px;  
  opacity: 0.4;  
}
```

Repare que usamos também a propriedade **opacity** para deixar esse elemento mais transparente e sutil.

Mais opções do Unicode

Consulte em uma tabela Unicode outros caracteres e seu código **hex** correspondente.

<http://www.alanwood.net/unicode/dingbats.html>

<http://www.alanwood.net/unicode/#links>

2.

(avançado) Volte à página **sobre.html**, abra-a no navegador.

Em um exercício anterior, colocamos as primeiras linhas em *small-caps* usando o seletor **p::first-line**. Repare que todos os parágrafos foram afetados.

E se quiséssemos que apenas *parágrafos de início de seção* fossem

afetados? Podemos pensar assim: queremos alterar todos os parágrafos que não estão no meio do texto, ou seja, não são precedidos por outro parágrafo (mas sim precedidos por títulos, figuras etc).

Com o seletor `:not()` do CSS3, conseguimos (**alterar o que fizemos no exercício anterior**):

```
:not(p) + p::first-line {  
  font-variant: small-caps;  
}
```

Isso significa: **selecione as primeiras linhas dos parágrafos que não são precedidos por outros parágrafos.**

5.6 CSS3: BORDER-RADIUS

Uma das novidades mais celebradas do CSS3 foi a adição de bordas arredondadas via CSS. Até então, a única forma de obter esse efeito era usando imagens, o que deixava a página mais carregada e dificultava a manutenção.

Com o CSS3 há o suporte a propriedade `border-radius` que recebe o tamanho do raio de arredondamento das bordas. Por exemplo:

```
div {  
  border-radius: 5px;  
}
```



Podemos também passar valores diferentes para cantos diferentes do elemento:

```
/* todas as bordas arredondadas com um raio de 15px */  
.a {  
  border-radius: 15px;  
}
```

```
/* borda superior esquerda e inferior direita com 5px  
borda superior direita e inferior esquerda com 20px */  
.b {  
  border-radius: 5px 20px;  
}
```

```
/* borda superior esquerda com 5px  
borda superior direita e inferior esquerda com 20px  
borda inferior direita com 50px */  
.c {  
  border-radius: 5px 20px 50px;  
}
```

```
/* borda superior esquerda com 5px  
borda superior direita com 20px  
borda inferior direita com 50px  
borda inferior esquerda com 100px */  
.d {  
  border-radius: 5px 20px 50px 100px;  
}
```



5.7 CSS3: TEXT-SHADOW

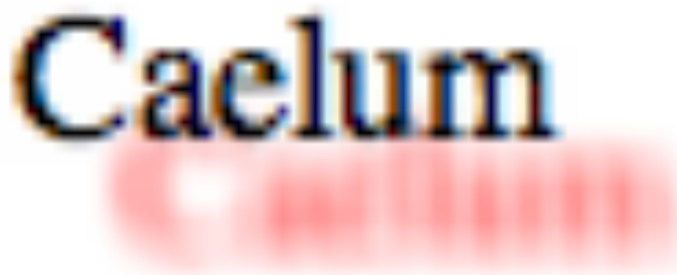
Outro efeito do CSS3 é o suporte a sombras em textos com `text-shadow`. Sua sintaxe recebe o deslocamento da sombra e sua cor:

```
p {  
  text-shadow: 10px 10px red;  
}
```

The image shows the word "Caelum" in a blue serif font. Below it, there is a red shadow of the same text, shifted 10 pixels to the right and 10 pixels down. The shadow is sharp and has a distinct red color.

Ou ainda pode receber um grau de espalhamento (blur):

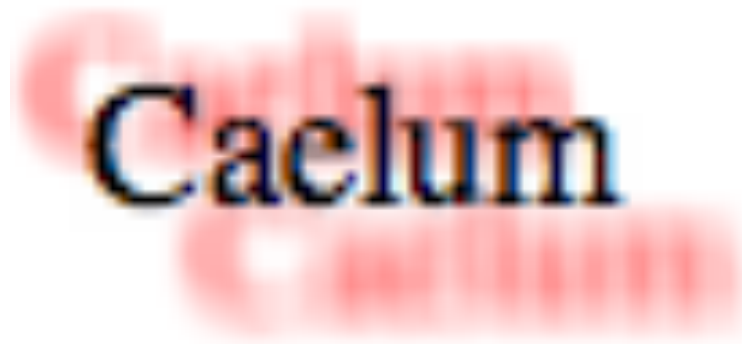
```
p {  
  text-shadow: 10px 10px 5px red;  
}
```

The image shows the word "Caelum" in a blue serif font. Below it, there is a red shadow of the same text, shifted 10 pixels to the right and 10 pixels down. The shadow is blurred, giving it a soft, out-of-focus appearance.

É possível até passar mais de uma sombra ao mesmo tempo para o mesmo elemento:

```
p {  
  text-shadow: 10px 10px 5px red, -5px -5px 4px red;  
}
```

}



Já conhece os cursos online Alura?

alura

A **Alura** oferece centenas de **cursos online** em sua plataforma exclusiva de ensino que favorece o aprendizado com a **qualidade** reconhecida da Caelum. Você pode escolher um curso nas áreas de Programação, Front-end, Mobile, Design & UX, Infra e Business, com um plano que dá acesso a todos os cursos. Ex-aluno da Caelum tem 15% de desconto neste link!

[Conheça os cursos online Alura.](#)

5.8 CSS3: BOX-SHADOW

Além de sombras em texto, podemos colocar sombras em qualquer elemento com **box-shadow**. A sintaxe é parecida com a do **text-shadow**:

```
box-shadow: 20px 20px black;
```



Podemos ainda passar um terceiro valor com o blur:

```
box-shadow: 20px 20px 20px black;
```



Diferentemente do `text-shadow`, o `box-shadow` suporta ainda mais

um valor que faz a sombra aumentar ou diminuir:

```
box-shadow: 20px 20px 20px 30px black;
```



Por fim, é possível usar a keyword **inset** para uma borda interna ao elemento:

```
box-shadow: inset 0 0 40px black;
```



5.9 OPACIDADE E RGBA

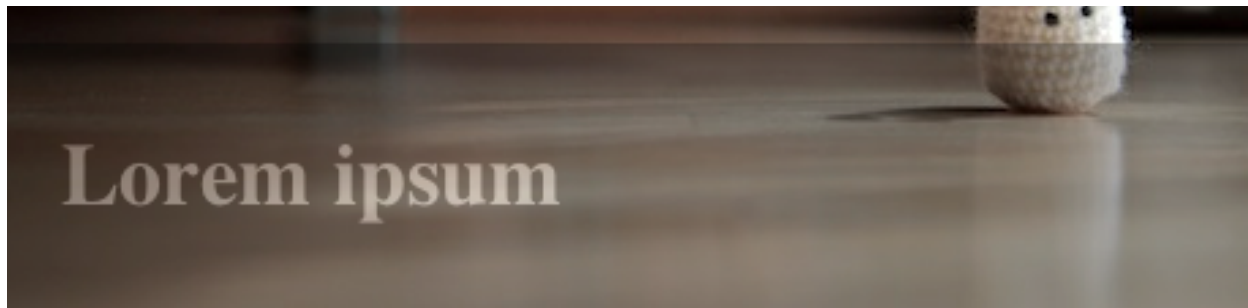
Desde o CSS2 é possível mudar a opacidade de um elemento para que ele seja mais transparente com o uso da propriedade **opacity**.

Veja esse exemplo com um parágrafo por cima de uma imagem:



Colocamos um fundo preto e cor branca no texto. E se quisermos um efeito legal de transparência para deixar a imagem mostrar mais?





É simples com a **opacity** que recebe um valor decimal entre 0 e 1:

```
p {  
  opacity: 0.3;  
}
```

Repare como todo o elemento fica transparente, o fundo e o texto. E se quiséssemos o texto em branco opaco e o fundo com transparência? No CSS3, podemos usar outra técnica, a de definir uma cor de fundo com valor de opacidade específica.

No CSS3, além das cores hex normais (**#ffffff** pro branco), podemos criar cores a partir de seu valor RGB, passando o valor de cada canal (Red, Green, Blue) separadamente (valor entre 0 e 255):

```
/* todos equivalentes */  
color: white;  
color: #ffffff;  
color: rgb(255, 255, 255);
```

Porém, existe uma função chamada RGBA que recebe um quarto argumento, o chamado canal Alpha. Na prática, seria como a opacidade daquela cor (um valor entre 0 e 1):

```
/* branco com 80% de opacidade */  
color: rgba(255,255,255, 0.8);
```

No exemplo da foto, queríamos apenas o fundo do texto em preto transluzente (o texto não). Em vez do opacity, podemos fazer então:

```
p {
```



```
background-color: rgba(0,0,0,0.3);  
color: white;  
}
```



5.10 PREFIXOS

Muitos recursos do HTML5 e do CSS3 ainda são experimentais. Isso quer dizer que foram incluídos no rascunho da especificação mas ainda não são 100% oficiais. As especificações ainda estão em aberto e vai demorar algum tempo até elas serem fechadas.

Existem recursos mais estáveis e menos estáveis. Alguns já estão bastante maduros e há bastante tempo na spec, e não são esperadas mais mudanças. Por outro lado, alguns são bem recentes e talvez ainda possa haver mudança até a aprovação final da especificação.

Os navegadores desejam implementar os novos recursos antes mesmo da especificação terminar, para que os desenvolvedores possam começar a usar as novas propriedades e experimentá-las na prática. Entretanto, como a sintaxe final depois de aprovada pode ser diferente, os navegadores implementam as novas propriedades instáveis usando nomes provisórios.

A boa prática é pegar o nome da propriedade e colocar um **prefixo** específico do fabricante na frente. Quando a especificação ficar estável, tira-se esse prefixo e suporta-se a sintaxe oficial.

As bordas arredondadas, por exemplo, hoje são suportadas em todos os navegadores modernos com o nome normal da propriedade a **border-radius**. Mas até o Firefox 3.6, por exemplo, o suporte da Mozilla era experimental e a propriedade era chamada de **-moz-border-radius** nesse navegador. No Chrome até a versão 3, precisávamos usar o prefixo deles com **-webkit-border-radius**.

Os prefixos dos fabricantes mais famosos são:

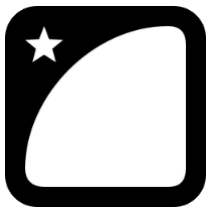
- **-webkit-**: navegadores Webkit (Chrome, Safari, iOS, Android)
- **-moz-**: Firefox (Mozilla)
- **-ms-**: Internet Explorer (Microsoft)
- **-o-**: Opera

É preciso consultar tabelas de compatibilidade para saber qual navegador suporta qual propriedade e se é necessário usar prefixos para certas versões. Se quisermos o máximo de compatibilidade, precisamos colocar vários prefixos ao mesmo tempo:

```
p {  
  /* Chrome até versão 3, Safari até versão 4 */  
  -webkit-border-radius: 5px;  
  
  /* Firefox até versão 3.6 */  
  -moz-border-radius: 5px;  
  /* Todas as versões modernas dos navegadores,  
  incluindo IE e Opera que nunca precisaram de  
  prefixo pra isso */  
  border-radius: 5px;  
}
```

Nos casos de CSS3 que tratamos até agora (border-radius, text-shadow, box-shadow, rgba), todos os navegadores modernos já suportam sem uso de prefixos. Você só precisará deles se quiser suportar versões antigas (nesse caso, consulte as documentações para saber o que é necessário e quando).

Você pode também fazer o curso WD-43 dessa apostila na Caelum



Querendo aprender ainda mais sobre? Esclarecer dúvidas dos exercícios? Ouvir explicações detalhadas com um instrutor?

A Caelum oferece o **curso WD-43** presencial nas cidades de São Paulo, Rio de Janeiro e Brasília, além de turmas incompany.

[Consulte as vantagens do curso *Desenvolvimento Web com HTML, CSS e JavaScript*](#)

5.11 CSS3: GRADIENTES

O CSS3 traz também suporte à declaração de gradientes sem que precisemos trabalhar com imagens. Além de ser mais simples, a página fica mais leve e a renderização fica melhor por se adaptar a todo tipo de resolução.

Existe suporte a gradientes lineares e radiais, inclusive com múltiplas paradas. A sintaxe básica é:

```
.linear {  
  background: linear-gradient(white, blue);  
}
```

```
.radial {  
  background: radial-gradient(white, blue);  
}
```

```
}
```

Podemos ainda usar gradientes com angulações diferentes e diversas paradas de cores:

```
.gradiente {  
  background: linear-gradient(45deg, #f0f9ff 0%, #cbefff 47%, #
```

Prefixos

A especificação de gradientes já está em sua versão final e já é implementada sem prefixos em todos os browsers.

Mas, enquanto a especificação ainda estava em rascunho, antes de seu final, uma sintaxe diferente nos gradientes era usada. Isso quer dizer que versões mais antigas dos navegadores que suportavam gradientes esperam uma sintaxe diferente.

Como as especificações em rascunho são implementadas prefixadas nos navegadores, é fácil colocar essas regras com sintaxe diferente para esses navegadores. O problema é que o código fica grande e difícil de manter.

A versão atual da especificação suporta um primeiro argumento que indica a direção do gradiente:

```
.linear {  
  background: linear-gradient(to bottom, white, blue);  
}
```

O código anterior indica um gradiente do branco para o azul vindo de cima **para baixo**. Mas na versão suportada antes do rascunho dos gradientes, o mesmo código era escrito com **top** ao invés de **to bottom**:

```
.linear {  
  background: -webkit-linear-gradient(top, white, blue);  
  background: -moz-linear-gradient(top, white, blue);  
  background: -o-linear-gradient(top, white, blue);  
}
```

Pra piorar, versões bem mais antigas do WebKit - notadamente o Chrome até versão 9 e o Safari até versão 5 -, usavam uma sintaxe ainda mais antiga e complicada:

```
.linear {  
  background: -webkit-gradient(linear, left top, left bottom,  
    color-stop(0%, white), color-stop(100%, blue));  
}
```

Se quiser o máximo de compatibilidade, você terá que incluir todas essas variantes no código CSS.

Gambiarras pro IE antigo

O IE só suporta gradientes CSS3 a partir da versão 10, mas desde o IE6 era possível fazer gradientes simples usando um CSS proprietário da Microsoft:


```
.linear {  
  filter: progid:DXImageTransform.Microsoft.gradient(  
    startColorstr='#ffffff', endColorstr='#0000ff',GradientType  
  )  
}
```

O CSS acima faz um gradiente linear do topo para baixo, como nos outros exemplos, funcionar no IE6 até IE9. O IE10 já não aceita mais essa gambiarra e exige que você use a sintaxe oficial do CSS3 que vimos no início.

Geração de gradientes

É comum também a configuração de um **background** simples e sólido antes do gradiente, a ser usado pelos navegadores mais antigos. Como eles não entendem gradientes, usarão a cor sólida e terão um design adequado e usável. Os navegadores mais novos vão ler a regra do gradiente e ignorar a cor sólida (progressive enhancement):

```
.gradiente {  
  background: #cbebff;
```



```
background: linear-gradient(45deg, #f0f9ff 0%, #cbefff 47%, #
```

```
}
```

Uma ferramenta bastante útil e recomendada é o **Ultimate CSS Gradient Generator** da ColorZilla. Ela nos permite criar gradientes CSS3 visualmente como num editor de imagens. Além disso, já implementa todos os hacks e prefixos para navegadores diferentes. Há até uma opção que permite que enviemos uma imagem com o design de um gradiente e ele identifica as cores e gera o código correto.

<http://www.colorzilla.com/gradient-editor/>

Recomendamos fortemente o uso dessa ferramenta para gerar gradientes CSS3.

5.12 PROGRESSIVE ENHANCEMENT

Nesse ponto, você pode estar pensando nos navegadores antigos. Bordas redondas, por exemplo, só funcionam no IE a partir da versão 9. Até o IE8 não há como fazer isso com CSS (nem com prefixos).

O que fazer então? Muitos usuários no Brasil ainda usam IE8 e até versões mais antigas como IE7 e talvez IE6.

O melhor é não usar esses recursos modernos por causa dos usuários de navegadores antigos? **Não!**

Não vamos sacrificar a experiência da maioria dos usuários de navegadores modernos por causa do cada vez menor número de pessoas usando navegadores antigos, mas também não queremos esquecer os usuários de navegadores antigos e deixá-los sem suporte.

Progressive enhancement e graceful degradation

A ideia é fazer seu site funcionar em qualquer navegador, sem prejudicar os navegadores mais antigos e sem deixar de usar os novos recursos em navegadores novos. Graceful degradation foi o nome da primeira técnica a

pregar isso; o objetivo era montar seu site voltado aos navegadores modernos e fazer com que ele degradasse "graciosamente", removendo funcionalidades não suportadas.

A abordagem mais recente, chamada **progressive enhancement** tem uma ideia parecida mas ao contrário: comece desenvolvendo as funcionalidades normalmente e vá acrescentando pequenas melhorias mesmo que só funcionem nos navegadores modernos.

Com CSS3, podemos usar progressive enhancement. Não é possível ainda ter um site que dependa hoje totalmente do CSS3. Mas é possível desenvolver seu site e acrescentar melhorias progressivamente usando CSS3.

Ou seja, faça um layout que fique usável com bordas quadradas mas use bordas redondas para deixá-lo melhor, mais bonito, nos navegadores mais modernos.

Saiba mais no blog da Caelum:

<http://blog.caelum.com.br/css3-e-progressive-enhancement/>

5.13 EXERCÍCIOS: VISUAL CSS3

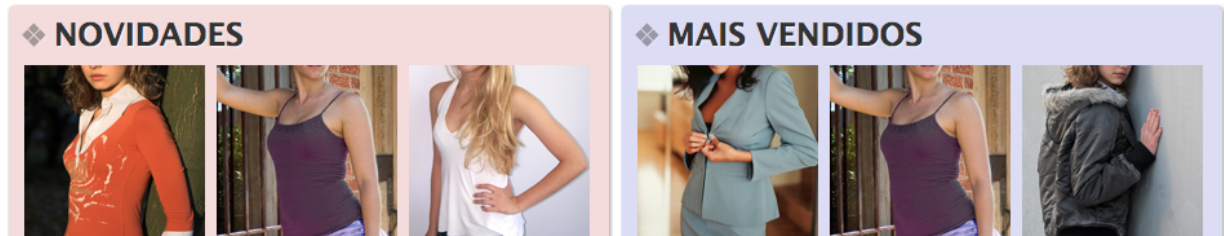
1.

Dê um ar mais atual para nossa home colocando alguns efeitos nos painéis.

Use `border-radius` e `box-shadow` no painel em si para destacá-lo mais. E use um `text-shadow` sutil para deixar o título do painel mais destacado.

```
.painel {  
  border-radius: 4px;  
  box-shadow: 1px 1px 4px #999;  
}
```

```
.painel h2 {  
  text-shadow: 3px 3px 2px #fff;  
}
```



Veja o resultado no navegador.

2.

O **box-shadow** também aceita a criação de bordas internas aos elementos além da borda externa. Basta usar a opção **inset**, **altere** no seletor que criamos no exercício anterior:

```
.painel {  
  box-shadow: inset 1px 1px 4px #999;  
}
```

Teste na sombra dos painéis que fizemos antes.

3.

O **border-radius** pode também ser configurado para bordas específicas apenas e até de tamanhos diferentes se quisermos.

```
.busca {  
  border-top-left-radius: 5px;  
  border-top-right-radius: 5px;  
}
```

4.

No CSS3, podemos usar cores com **canal Alpha** para translucência usando a sintaxe do RGBA. **Altere** a sombra do título do painel para branco

com 80% de opacidade.

```
.painel h2 {  
  text-shadow: 3px 3px 2px rgba(255,255,255,0.8);  
}
```

5.

Use gradientes nos painéis de produtos na Home. Não retire o `background-color` que já está no seletor, ele serve para os navegadores que não suportam o `gradient`. Adicionar o `gradient` a baixo do `background-color` existente.

O painel novidade, por exemplo, poderia ter:

```
.novidades {  
  background: linear-gradient(#f5dcdc, #bebef4);  
}
```

E o painel de mais vendidos:

```
.mais-vendidos {  
  background: linear-gradient(#dcdcf5, #f4bebe);  
}
```

Prefixos no exercício

Usamos no exercício a versão oficial da especificação sem prefixos que funciona nas últimas versões do Firefox, Chrome, Opera, Safari e Internet Explorer.

Note que não estamos suportando versões antigas desses navegadores de propósito. Se quiser, você pode adicionar as outras variantes de prefixos para suportá-los. Ou usar uma ferramenta como o Collorzilla Gradient Generator para automatizar:

<http://www.colorzilla.com/gradient-editor/>

Consulte o suporte nos browsers aqui: <http://caniuse.com/css-gradients>

Seus livros de tecnologia parecem do século passado?



Conheça a **Casa do Código**, uma **nova** editora, com autores de destaque no mercado, foco em **ebooks** (PDF, epub, mobi), preços **imbatíveis** e assuntos **atuais**.

Com a curadoria da **Caelum** e excelentes autores, é uma abordagem **diferente** para livros de tecnologia no

Brasil.

[Casa do Código, Livros de Tecnologia.](#)

5.14 CSS3 TRANSITIONS

Com as transitions, conseguimos animar o processo de mudança de algum valor do CSS.

Por exemplo: temos um elemento na posição **top:10px** e, quando passarmos o mouse em cima (:hover), queremos que o elemento vá para **top:30px**. O CSS básico é:

```
#teste {  
  position: relative;  
  top: 10px;  
}
```

```
#teste:hover {  
  top: 30px;  
}
```

Isso funciona, mas o elemento é deslocado de uma vez quando passamos o mouse. E se quisermos algo mais sutil? Uma animação desse valor mudando lentamente, mostrando o elemento se deslocando na tela? Usamos CSS3 Transitions.

Sua sintaxe possui vários recursos, mas seu uso é mais simples, para esse nosso caso, seria apenas:

```
#teste:hover {  
  transition: top 2s;  
}
```

Isso indica que queremos animar a propriedade `top` durante 2 segundos.

Por padrão, a animação é linear, mas temos outros tipos para animações mais suaves:

- `linear` - velocidade constante na animação;
- `ease` - redução gradual na velocidade da animação;
- `ease-in` - aumento gradual na velocidade da animação;
- `ease-in-out` - aumento gradual, depois redução gradual na velocidade da animação;
- `cubic-bezier(x1,y1,x2,y2)` - curva de velocidade para animação customizada (avançado)

```
#teste:hover {  
  transition: top 2s ease;  
}
```

Para explorar o comportamento dos tipos de animações disponíveis, e como criar uma curva de velocidade customizada para sua animação, existe uma ferramenta que auxilia a criação do `cubic-bezier`:

<http://www.roblaplaca.com/examples/bezierBuilder/>

Podemos ainda usar mais de uma propriedade ao mesmo tempo,

incluindo cores!

```
#teste {  
  position: relative;  
  top: 10px;  
  color: white;  
}  
  
#teste:hover {  
  top: 30px;  
  color: red;  
  transition: top 2s, color 1s ease;  
}
```

Se quisermos a mesma animação, mesmo tempo, mesmo efeito para todas as propriedades, podemos usar o atalho **all** (que já é o valor padrão, inclusive):

```
#teste:hover {  
  transition: all 2s ease;  
}
```

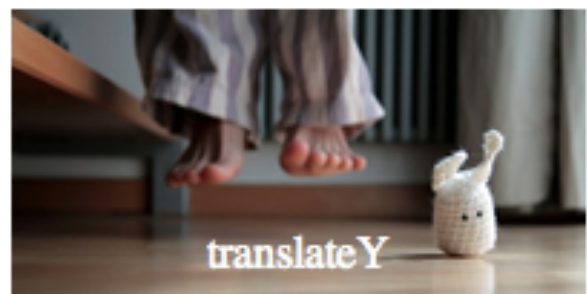
5.15 CSS3 TRANSFORMS

Com essa nova especificação, agora é possível alterar propriedades visuais dos elementos antes impossíveis. Por exemplo, agora podemos alterar o ângulo de um elemento, mostrá-lo em uma escala maior ou menor que seu tamanho padrão ou alterar a posição do elemento sem sofrer interferência de sua estrutura.

Translate

```
.header {  
  /* Move o elemento no eixo horizontal */  
  transform: translateX(50px);  
}  
  
#main {
```

```
/* Move o elemento no eixo vertical */  
transform: translateY(-20px);  
}  
  
footer {  
  /* Move o elemento nos dois eixos (X, Y) */  
  transform: translate(40px, -20px);  
}
```



Rotate

```
#menu-departamentos {  
  transform: rotate(-10deg);  
}
```

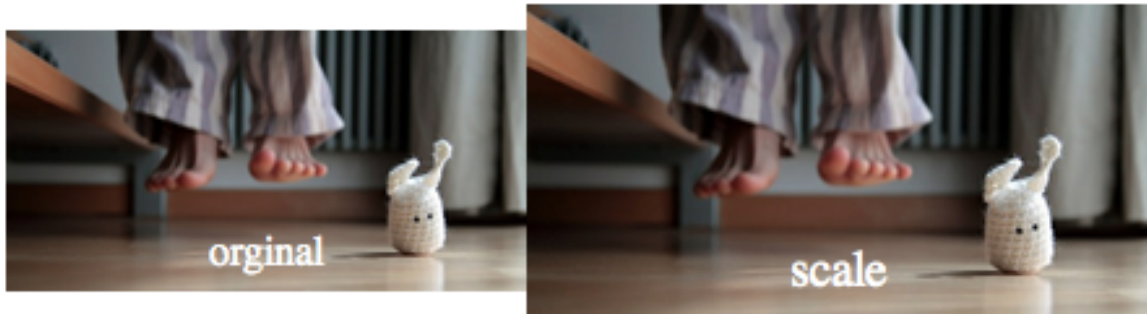


Scale

```
#novidades li {  
  /* Alterar a escala total do elemento */
```

```
transform: scale(1.2);
}

#mais-vendidos li {
  /* Alterar a escala vertical e horizontal do elemento */
  transform: scale(1, 0.6);
}
```



Skew

```
footer {
  /* Distorcer o elemento no eixo horizontal */
  transform: skewX(10deg);
}

#social {
  /* Distorcer o elemento no eixo vertical */
  transform: skewY(10deg);
}
```



É possível combinar várias transformações no mesmo elemento, basta declarar uma depois da outra:

```
html {  
  transform: rotate(-30deg) scale(0.4);  
}
```

Prefixos nos exercícios

No exercício, optamos por usar sintaxes que funcionam em todos os browsers mas apenas em suas últimas versões. Isso quer dizer que usamos tudo sem prefixos. Se quiser suportar versões mais antigas, adicione mais prefixos.

Consulte a compatibilidade e prefixos:

<http://caniuse.com/css-transitions>

<http://caniuse.com/transforms2d>

5.16 EXERCÍCIOS: CSS3 TRANSFORM E TRANSITION

1.

Quando o usuário passar o mouse em algum produto dos painéis de destaque, mostre uma sombra por trás com **box-shadow**. Use também uma transição com **transition** para que essa sombra apareça suavemente:

```
.painel li:hover {  
  box-shadow: 0 0 5px #333;  
  transition: box-shadow 0.7s;  
}
```

Teste o resultado no navegador.

2.

Altere a regra anterior para também colocar agora um fundo branco no elemento. Anime esse fundo também, fazendo um efeito tipo *fade*.

Na regra `transition` de antes, podemos indicar que todas as propriedades devem ser animadas; para isso, podemos usar a keyword `all` ou simplesmente omitir esse argumento.

```
.painel li:hover {  
  background-color: rgba(255,255,255,0.8);  
  box-shadow: 0 0 5px #333;  
  transition: 0.7s;  
}
```

3.

Mais coisas de CSS3! Ainda quando passar o mouse em cima do item do painel, queremos aumentar o elemento em 20%, dando uma espécie de zoom.

Use *CSS transform* pra isso, com `scale`. **Adicione** na regra anterior (sem remover o que já tínhamos):

```
.painel li:hover {  
  transform: scale(1.2);  
}
```

Teste e repare como o `scale` também é animado suavemente. Isso porque nossa transição estava com `all`.

4.

Altere a regra anterior do `transform` pra também fazer o elemento rotacionar suavemente em 5 graus no sentido anti-horário:

```
.painel li:hover {  
  transform: scale(1.2) rotate(-5deg);  
}
```




5.

(opcional) Faça os elementos ímpares girarem em sentido anti-horário e os pares no sentido horário!

No exercício anterior, fizemos todos girarem anti-horário. Vamos sobrescrever essa regra para os elementos pares usando o seletor `:nth-child`:

```
.painel li:nth-child(2n):hover {  
  transform: scale(1.2) rotate(5deg);  
}
```

6.

(opcional) Repare como a animação ocorre apenas quando passamos o mouse em cima, mas quando tiramos, a volta do efeito não é animada.

Podemos habilitar a animação na volta do elemento para o estado normal movendo as regras de transição para o `li` em si (e não só no `:hover`).

```
.painel li {  
  transition: 0.7s;  
}
```

7.

(opcional) Um terceiro argumento para a função de transição é a função de animação, que controla como o efeito executa de acordo com o tempo.

Por padrão, os efeitos são **lineares**, mas podemos obter resultados mais interessantes com outras opções como **ease**, **ease-in**, **ease-out** (e até o avançado **cubic-bezier()**).

Por exemplo, podemos **alterar** o que fizemos nos exercícios anteriores:

```
.painel li:hover {  
  transition: 0.7s ease-in;  
}  
  
.painel li {  
  transition: 0.7s ease-out;  
}
```

Agora é a melhor hora de aprender algo novo

alura

Se você está gostando dessa apostila, certamente vai aproveitar os **cursos online** que lançamos na plataforma **Alura**. Você estuda a qualquer momento com a **qualidade** Caelum. Programação, Mobile, Design, Infra, Front-End e Business! Ex-aluno da Caelum tem 15% de desconto, siga o link!

[Conheça a Alura Cursos Online.](#)

5.17 PARA SABER MAIS: ESPECIFICIDADE DE SELETORES CSS

Quando declaramos uma série de seletores CSS, é bem possível que nos deparemos com situações em que mais de um seletor esteja aplicando

propriedades visuais ao mesmo elemento do HTML. Nesse caso é necessário saber qual seletor tem precedência sobre os outros, a fim de resolver conflitos e garantir que as propriedades desejadas estejam sendo aplicadas aos elementos corretos.

O comportamento padrão dos seletores CSS, quando não há conflitos entre propriedades, é que as propriedades de mais de um seletor para um mesmo elemento sejam acumuladas. Veja no exemplo a seguir:

Estrutura HTML

```
<p>Texto do parágrafo em destaque</p>  
<p>Texto de um parágrafo comum</p>
```

Seletores CSS

```
p {  
  color: navy;  
}  
  
p {  
  font-size: 16px;  
}
```

No exemplo anterior, utilizamos o mesmo seletor duas vezes no CSS. Isso faz com que as propriedades sejam acumuladas em todos os elementos selecionados. No caso, todos os elementos da tag `p` em nosso documento serão exibidos da cor "navy" (azul marinho) e com a fonte no tamanho "16px".

Quando há conflito entre propriedades de seletores equivalentes, ou até mesmo em um mesmo seletor, é aplicada a propriedade declarada depois, como no exemplo a seguir:

```
p {  
  color: navy;  
  font-size: 12px;  
}
```

```
p {  
  font-size: 16px;  
}
```

Nesse caso, há conflito entre as propriedades `font-size` declaradas nos seletores. Como os seletores são equivalentes, os parágrafos serão exibidos com a fonte no tamanho "16px". A declaração anterior, com valor de "12px" é sobrescrita pela nova propriedade declarada mais abaixo no nosso CSS. A cor "navy" continua aplicada a todos os parágrafos do documento.

Especificidade de Seletores CSS

Seletores equivalentes têm suas propriedades sobrescritas conforme são declaradas. Mas o que acontece quando temos diferentes tipos de seletores?

Cada tipo de seletor tem um *peso* diferente quando o navegador os interpreta e aplica suas propriedades declaradas aos elementos de um documento. Existe uma maneira bem simples de saber como funcionam esses pesos porque eles fazem parte da especificação do CSS. Para ficar um pouco mais fácil é só pensarmos em uma regra simples: **quanto mais específico for o seletor, maior seu valor**. Por isso esse peso, ou valor, que cada seletor tem, recebe o nome de **especificidade**.

O seletor de tag, por exemplo `div {}`, é bem genérico. As propriedades declaradas nesse seletor serão aplicadas a todos os elementos `div` do nosso documento, não levando em consideração qualquer atributo que eles possam ter. Por esse motivo, o seletor de tag tem valor baixo de especificidade.

O seletor de classe, por exemplo `.destaque {}`, é um pouco mais específico, nós decidimos quais elementos têm determinado valor para esse atributo nos elementos do HTML, portanto o valor de especificidade do seletor de classe é maior do que o valor de especificidade do seletor de tag.

O seletor de id, por exemplo `#cabecalho {}`, é bem específico, pois só podemos ter um único elemento com determinado valor para o atributo id, então seu valor de especificidade é o maior entre os seletores que vimos até agora.

E quando temos seletores compostos ou combinados? Como calcular essa especificidade?

Podemos adicionar um ponto em cada posição do valor de um seletor para chegarmos ao seu valor de especificidade. Para isso vamos utilizar uma tabela para nos ajudar a conhecer esses valores, e a seguir vamos aplicar o cálculo a alguns seletores propostos.

Valor de Especificidade dos Seletores CSS		
Seletor de Id ex: <code>#rodape { font-size: 11pt; }</code>	Seletor de Classe ex: <code>.conteudo { width: 960px; }</code>	Seletor de Tag ex: <code>div { color: green; }</code>
1	1	1

Seguindo os valores descritos na tabela acima, podemos calcular o valor de especificidade para qualquer seletor CSS, por exemplo:

```
p {  
  /* valor de especificidade: 001 */  
  color: blue;  
}
```

```
.destaque {  
  /* valor de especificidade: 010 */  
  color: red;  
}
```

```
#cabecalho {  
  /* valor de especificidade: 100 */  
  color: green;
```

```
}
```

Nos seletores combinados e compostos, basta somar os valores em suas determinadas posições como nos exemplos a seguir:

```
#rodape p {  
  /* valor de especificidade: 101 */  
  font-size: 11px;  
}
```

```
#cabecalho .conteudo h1 {  
  /* valor de especificidade 111 */  
  color: green;  
}
```

```
.conteudo div p span {  
  /* valor de especificidade: 013 */  
  font-size: 13px;  
}
```

Quanto maior o valor da especificidade do seletor, maior a prioridade de seu valor, dessa maneira um seletor com valor de especificidade **013** sobrescreve as propriedades conflitantes para o mesmo elemento que um seletor com valor de especificidade **001**.

Essa é uma maneira simples de descobrir qual seletor tem suas propriedades aplicadas com maior prioridade. Por enquanto vimos somente esses três tipos de seletores CSS (tag, classe e id). No decorrer do curso vamos ver outros tipos mais avançados de seletores, e vamos complementando essa tabela para termos uma referência completa para esse cálculo.

CAPÍTULO ANTERIOR:

[Mais HTML e CSS](#)

PRÓXIMO CAPÍTULO:

[Web para dispositivos
móveis](#)

Você encontra a Caelum também em:

Blog
Caelum

Cursos
Online

Facebook

Newsletter

Casa do
Código

Twitter