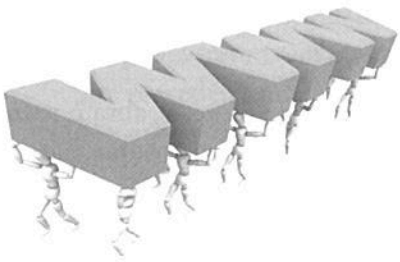

CAPÍTULO 4

O modelo CSS

Neste capítulo estudaremos as diretrizes que orientam a criação do modelo CSS. Veremos os conceitos que regem o efeito cascata e a herança CSS. Você aprenderá a determinar a ordem de cascata e a estabelecer a diferença entre regra CSS normal e importante. Veremos como calcular a especificidade de seletores e qual sua importância na ordem de cascata. Apresentaremos o modelo de formatação visual dos boxes e encerraremos com o detalhamento das propriedades CSS diretamente relacionadas à estrutura visual.

4.1 Efeito cascata

Você talvez já deva ter questionado a razão do termo cascata na terminologia das folhas de estilo. A dinâmica de aplicação das regras de estilo pode ser comparada ao fluir de uma cascata, como veremos adiante.

Folhas de estilo em cascata são originárias de três fontes distintas: *autor*, *usuário* e *agente de usuário* (por exemplo: o navegador).

Neste capítulo, a ênfase é para um modelo visual, assim vamos convencionar usar o termo navegador para caracterizar um agente de usuário gráfico.

O autor cria folhas de estilos e as serve incorporadas ou linkadas aos seus documentos, conforme vimos no Capítulo 3.

Ao usuário é facultado criar folhas de estilo personalizadas de acordo com suas preferências e necessidades. Os navegadores em geral fornecem funcionalidades para criação de folhas de estilo do usuário.

Existe uma folha de estilo interna no navegador, criada pelo fabricante, que aplica estilo-padrão aos diferentes elementos (X)HTML. Por exemplo: um conteúdo

marcado com o elemento `em` é apresentado em itálico e um conteúdo marcado com o elemento `h1` apresenta texto em negrito e fonte grande.

Para criar uma folha de estilo de usuário no Internet Explorer 6, abra o menu **Ferramentas > Opções da Internet... > Acessibilidade**. Na caixa de diálogo que se abre, há uma opção que permite linkar para uma folha de estilo criada pelo usuário.

No Firefox, crie sua folha de estilo e salve com o nome de `userContent.css`. Localize o diretório que contém o seu perfil e coloque a folha de estilo no subdiretório `chrome`. Reinicie o navegador para ver os efeitos da folha que você criou. Com algumas variações, o diretório do perfil, no Windows, está em `C:\Documents and Settings\Administrador\Dados de aplicativos\Mozilla\Firefox\Profiles`.

Suponha que o autor estilizou as fontes na cor vermelha, o usuário na cor azul, enquanto que o estilo-padrão para fontes no navegador é a cor preta. Qual das cores têm precedência? Para resolver conflitos tanto entre folhas de estilo quanto entre regras CSS, entra em cena o efeito cascata.

A cada regra de estilo é atribuído um peso. A precedência é para a regra com o maior peso.

4.1.1 Ordem da cascata

Para resolver conflitos entre regras de estilo, o navegador coloca em ordem ascendente as regras conflitantes e atribui precedência àquela que ocupa a posição mais alta na ordenação. Tal ordenação é chamada de ordem de cascata.

O procedimento para determinar a ordem de cascata é o seguinte:

1. Identificar as declarações de estilo que se aplicam a um mesmo seletor no documento. São as chamadas regras conflitantes.
2. Estabelecer a ordem ascendente de prioridade levando em conta a importância (`normal` e `important`- veremos adiante na Seção 4.1.2) e a origem (autor, usuário, navegador), conforme a seqüência a seguir:
   a) declarações-padrão do navegador;
   b) declarações normais do usuário;
   c) declarações normais do autor;
   d) declarações importantes do autor;
   e) declarações importantes do usuário.

3. Ordenar as regras com mesma importância e origem pela especificidade do seletor – veremos adiante na Seção 4.1.3. Maior especificidade resulta em mais alta precedência.

4. Finalmente, para regras com mesma importância, origem e especificidade, a precedência é para aquela que foi escrita por último na folha de estilo.

Este procedimento resulta na ordem de cascata crescente, conforme mostrado a seguir:

1. Folha de estilo padrão do navegador.

2. Folha de estilo do usuário.

3. Folha de estilo do autor.

   a) estilo externo (importado ou linkado);

   b) estilo incorporado (definido na seção `head` do documento);

   c) estilo inline (dentro de um elemento (X)HTML).

4. Declarações do autor com `!important`.

5. Declarações do usuário com `!important`.

4.1.2 Declarações normais e importantes

Por padrão, as regras de estilo escritas pelo autor têm precedência sobre as regras de estilo do usuário. Contudo, as CSS prevêem uma funcionalidade capaz de alterar esta precedência, a qual consiste em declarar a regra como importante e cuja sintaxe é mostrada no exemplo a seguir:

```
p {color: #000 !important}
```

Uma regra de estilo declarada `!important` tem precedência sobre uma regra normal. A diretiva `!important` permite ao usuário sobrescrever os estilos do autor, e esta é uma possibilidade poderosa para incrementar a acessibilidade. Ao autor é facultado o uso da diretiva `!important`; contudo, uma regra do usuário declarada com esta diretiva tem precedência sobre uma regra do autor também com esta diretiva.

4.1.3 Especificidade de um seletor

Vimos que um dos fatores determinantes da ordem de cascata é a especificidade do seletor. Existem diferentes sintaxes de escrita para um seletor, capazes de fazer com que ele case com um só elemento na marcação. Considere os códigos a seguir:

- **HTML**

```
<p class="diferente">Um parágrafo da classe diferente</p>
```

- **CSS**

```
p.diferente { font-size: 12px; }
p { font-size: 16px; }
```

As duas regras de estilo são aplicáveis à marcação sugerida no exemplo. Na primeira usamos a sintaxe, escrevendo o seletor composto pelo elemento e sua classe, enquanto na segunda por somente o elemento. Qual delas tem a precedência e será aplicada?

A precedência é do seletor mais específico ou, como o próprio nome diz, aquele que se refere mais especificamente ao elemento. O seletor tipo `p` é genérico (aplica-se a todos os parágrafos do documento) e o seletor classe `p.diferente` é mais específico (aplica-se somente aos parágrafos cujo atributo classe tem o valor `diferente`).

Denomina-se seletor tipo ao seletor cujo alvo é um determinado tipo de elemento na marcação. São aqueles representados por uma tag (X)HTML, tais como os seletores `body`, `h1`, `p`, `div`, `table` etc.

4.1.3.1 Cálculo da especificidade

O cálculo da especificidade de um seletor é feito segundo o seguinte roteiro:

- Conte a quantidade de atributos `id` no seletor.
- Conte a quantidade de outros atributos e pseudoclasses no seletor.
- Conte a quantidade de elementos no seletor.
- Escreva o resultado de cada contagem da esquerda para a direita, obedecendo à ordem de contagem mostrada (`id`, outros atributos, elemento) para obter um número.
- O número obtido representa a especificidade do seletor. Quanto maior o número, mais específico é o seletor.
- Ignore os pseudo-elementos (`:first-line`, `:first-letter`, `:before`, `:after`).

Considere a seguinte regra CSS:

```
p span.diferente {color: red;}
```

O seletor é `p span.diferente`, que tem como alvo os elementos `span` marcados com a classe `diferente` e contidos em parágrafos. A especificidade deste seletor é:

- Quantidade de seletores `id` = 0.
- Quantidade de seletores atributo = 1 (atributo classe = `diferente`).
- Quantidade de seletores elemento = 2 (`p` e `span`).
- Especificidade: 012 – 12 é a especificidade do seletor.

Outros exemplos para você conferir:

Seletor	Especificidade
`li`	001
`ul li a`	003
`ul li.corrente a`	013
`*`	000
`table#principal tr th`	103
`p:first-letter`	001
`#nav a.um`	111

Quando o estilo é aplicado inline com uso do atributo `style` na tag do elemento, a especificidade é a maior possível e tem precedência sobre qualquer outro seletor. Estilos inline têm especificidade igual a 1.000 (mil).

4.2 Modelo de formatação visual dos boxes

Modelo de formatação visual é a maneira como os agentes de usuário processam a árvore do documento para apresentação em mídia visual.

Nesta seção abordaremos especificamente a mídia tela do monitor e o agente de usuário navegador. A maioria dos conceitos aqui estudados é aplicada por extensão às demais mídias visuais e seus agentes.

O modelo de formatação visual implica que cada elemento da árvore do documento crie um ou mais *boxes* (caixas) de acordo com o *Box Model* que veremos na seção seguinte. São características próprias dos boxes: dimensões, posição, tipo, relacionamento com outros boxes e informações externas, tais como dimensões intrínsecas de imagens. Nesta seção estudaremos os fundamentos básicos do modelo visual relacionados aos tipos de boxes.

4.2.1 Container

Também denominado de bloco de conteúdo, trata-se de um box retangular que contém outros boxes chamados de descendentes. A conceituação de container é importante para o entendimento dos mecanismos de cálculo de dimensões e posicionamento de boxes descendentes. Um box descendente não precisa ter necessariamente suas dimensões confinadas aos limites do container, podendo ultrapassá-lo. Este comportamento é chamado de *overflow*.

4.2.2 Elementos nível de bloco e boxes bloco

Elementos nível de bloco são aqueles elementos da marcação (X)HTML formatados visualmente como blocos de conteúdos. São exemplos: os parágrafos `p` e os cabeçalhos `h1` – `h6`.

4.2.2.1 Box bloco anônimo

Considere o código a seguir:

```
<div>
   Texto qualquer contido diretamente na DIV
   <p>Texto de um parágrafo contido na DIV</p>
</div>
```

Admitindo que os elementos `div` e `p` são níveis de bloco, qual será o comportamento do texto contido diretamente na `div`? Inline (veremos adiante na Seção 4.2.3) ou de bloco? Observe o esquema mostrado na Figura 4.1:

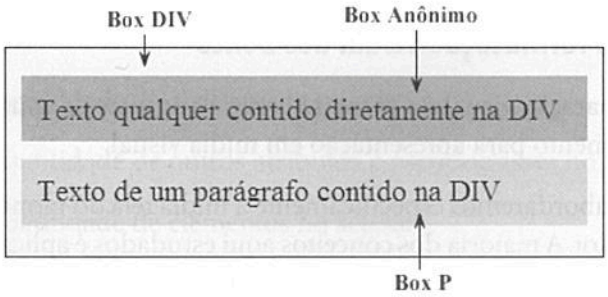

Figura 4.1 – Box bloco anônimo.

Quando inserirmos em um container (no nosso exemplo a `div`) um box bloco (o parágrafo, em nosso exemplo), estaremos forçando os conteúdos inseridos diretamente no container a se comportarem como box bloco, que são denominados *boxes blocos anônimos*.

4.2.3 Elementos inline e boxes inline

Elementos inline são aqueles elementos da marcação (X)HTML que não formam novos blocos de conteúdo. O conteúdo é distribuído em linha. São exemplos: o elemento para enfatizar `strong` e as imagens.

4.2.3.1 Box inline anônimo

Considere o código a seguir:

```
<p>Texto <strong>enfatizado</strong> mais texto</p>
```

O elemento `p` criou um box bloco contendo três boxes inline. O box para *enfatizado*, que foi gerado pelo elemento `strong`, e os boxes para *Texto* e *mais texto*, gerados por um elemento nível de bloco, o parágrafo. Estes dois últimos são chamados de *boxes inline anônimos*.

4.3 Box Model

O Box Model CSS descreve os boxes criados pelos elementos contidos na árvore do documento e apresentados segundo o modelo de formatação visual. O Box Model CSS é mostrado no diagrama da Figura 4.2:

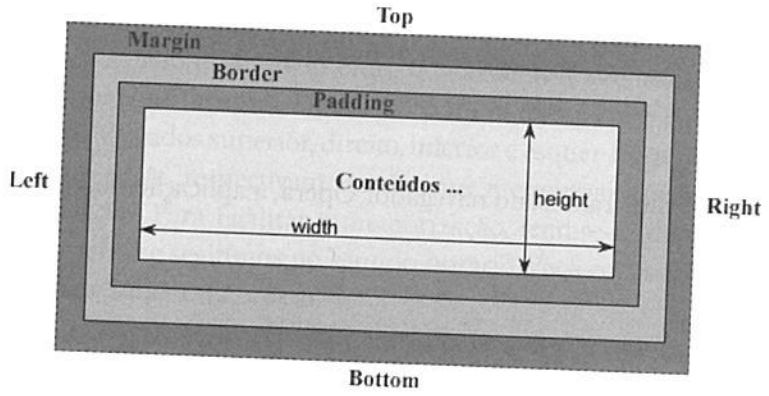

Figura 4.2 – Diagrama do Box Model CSS.

No diagrama destacamos uma área mais interior, denominada área dos conteúdos, cujas dimensões (largura e altura) são definidas pelas propriedades CSS `width` e `height`. Segue-se uma área chamada de enchimento, cuja espessura é definida pela propriedade CSS `padding`. Em volta do enchimento, temos uma borda cuja espessura, cor e tipo são definidos por `border`. Finalmente, um espaço denominado margem com espessura definida pela propriedade `margin`.

A área da margem é sempre transparente. As dimensões da área de conteúdos dependem de uma série de fatores, entre eles definição explícita de dimensões, natureza do conteúdo e tipo de conteúdo. A propriedade CSS `background` define o fundo a ser aplicado nas áreas de conteúdos, de enchimento e da borda.

Considere a marcação (X)HTML e a estilização mostradas nos códigos a seguir:

- **HTML**

```
...
<div>
   <p>Texto contido no parágrafo. Lorem...</p>
</div>
...
```

- **CSS**

```
body {margin:0; padding:0; font-size:20px;}
div {
   width: 440px;
   background: #999;
   margin: 45px 40px 5px 50px;
   padding: 15px 30px 10px 0;
   }
p {
   text-align: justify;
   background: #ccc;
   margin: 20px 20px 20px 40px;
   padding: 15px 30px 10px 0;
   }
```

Na Figura 4.3 mostramos, no navegador Opera, a aplicação do Box Model.

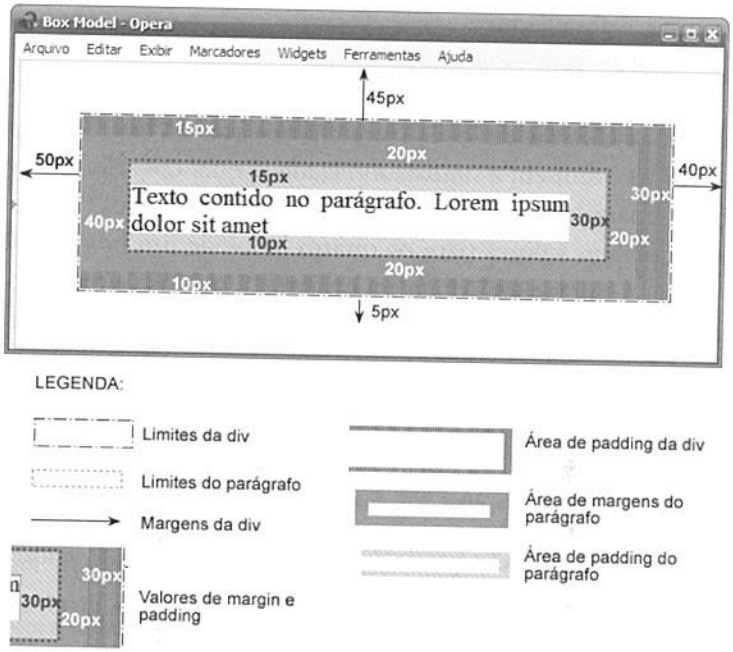

Figura 4.3 – Box Model no Opera.

4.4 Propriedades CSS para o Box Model

Veremos, a seguir, as regras para aplicação das propriedades CSS `margin`, `border` e `padding`. O box criado no modelo é um quadrilátero onde cada um dos lados é identificado, conforme mostra a Figura 4.2, por um termo em inglês designativo da posição do lado. Os lados superior, direito, inferior e esquerdo são identificados por `top`, `right`, `bottom` e `left`, respectivamente. Procure memorizar a ordem em que estão nomeados os lados. Para facilitar a memorização, lembre-se de que começamos com o lado superior e seguimos no sentido horário. Você constatará, logo adiante, a importância de saber esta ordem.

4.4.1 Propriedade `margin`

A propriedade `margin` define as dimensões das margens de um box. Você pode definir cada uma das quatro margens do box com valores diferentes, conforme mostrado a seguir:

```
margin-top: 20px;
margin-right: 30px;
margin-bottom: 5px;
margin-left: 10px;
```

ou usar a sintaxe abreviada:

```
margin: 20px 30px 5px 10px;
```

As propriedades CSS `font`, `background`, `margin`, `padding`, `border`, `border-width`, `border-style`, `border-color` e `outline` admitem a *sintaxe abreviada*, a qual consiste em declarar uma lista de valores separados por um espaço, conforme mostrado. Para as propriedades em que se definem valores para os quatro lados de um box, tais como `margin` e `padding`, é válida a sintaxe abreviada e a ordem em que os valores são escritos na lista correspondem aos lados superior, direito, inferior e esquerdo, respectivamente.

A declaração abreviada admite ainda as seguintes variações:

- Se as quatro margens são iguais, declare um valor:

```
margin: 20px; /* margem de 20px nos quatro lados */
```

- Se as quatro margens são iguais duas a duas, declare dois valores:

```
margin: 15px 10px;   /* margens superior e inferior de 15px e direita e
                        esquerda de 10px */
```

- Declare três valores:

```
margin: 20px 10px 15px; /* margem superior de 20px margens direita e esquerda de
                           10px e margem inferior de 15px */
```

4.4.2 Propriedade `padding`

A propriedade `padding` define as dimensões do enchimento (ou espessura) entre o conteúdo e a borda. Você pode definir cada um dos quatro enchimentos do box com valores diferentes, conforme mostrado a seguir:

```
padding-top: 20px;
padding-right: 30px;
padding-bottom: 5px;
padding-left: 10px;
```

ou usar a sintaxe abreviada:

```
padding: 20px 30px 5px 10px;
```

A sintaxe é idêntica àquela para a propriedade `margin` e a declaração abreviada também admite as variações mostradas para tal propriedade.

4.4.3 Propriedade `border`

A propriedade `border` define a espessura, o estilo e a cor das bordas do box. Cada uma destas três características da borda pode ser declarada separadamente para cada lado do box, conforme veremos em seguida.

border-width

Define a espessura da borda.

```
border-top-width: 2px;
border-right-width: 3px;
border-bottom-width: 4px
border-left-width: 1px;
```

ou usar a sintaxe abreviada:

```
border-width: 2px 3px 4px 1px;
```

Você define a espessura da borda declarando uma medida CSS de comprimento [C3 S3.2] ou usando as palavras-chave `thin`, `medium` e `thick`, obtendo as espessuras de bordas fina, média e grossa, conforme mostradas na Figura 4.4.

```
border-width: thin;
border-width: medium;
border-width: thick;
```

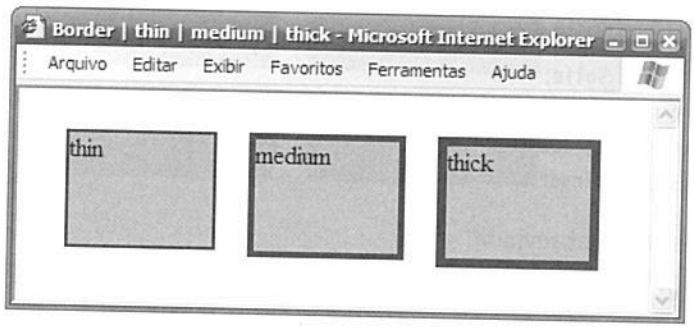

Figura 4.4 – Bordas por palavra-chave.

border-color

Define a cor da borda.

```
border-top-color: red;
border-right-color: yellow;
border-bottom-color: black;
border-left-color: cyan;
```

ou usar a sintaxe abreviada:

```
border-color: red yellow black cyan;
```

Para declarar a cor da borda, você usa um dos valores CSS para cores, conforme descrito no Capítulo 7, ou o valor `transparent`.

Convém notar que o navegador Internet Explorer 6 não oferece suporte para o valor transparente em bordas, renderizando bordas transparentes na cor preta. Na Figura 4.5 você vê renderização de uma `div` nos navegadores Firefox, Opera e IE6. A regra de estilo que define as bordas para a `div` é:

```
div {border: 10px solid transparent}
```

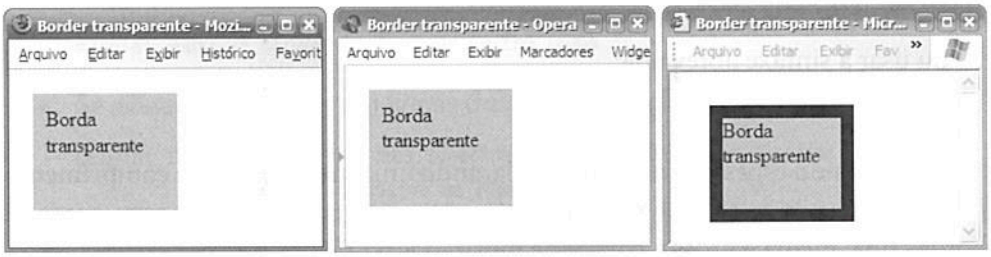

Figura 4.5 – Bordas transparentes.

border-style

Define o estilo da borda

```
border-top-style: solid;
border-right-style: ridge;
border-bottom-style: double;
border-left-style: inset;
```

ou usar a sintaxe abreviada:

```
border-style: solid ridge double inset;
```

Você pode aplicar nove estilos para bordas ou declarar o valor `none` para definir ausência de bordas. Pode parecer estranha e inútil a declaração `none`, mas, na prática, é muito usada para retirar bordas colocadas por padrão ou declaradas anteriormente em elementos específicos da marcação ou, ainda, para retirar a borda-padrão colocada em imagens que são links. Ver [C8 S8.2.3].

Estilo	Descrição sumária
`none`	Define espessura 0 para a borda.
`hidden`	O mesmo efeito de none, mas com precedência na resolução de bordas conflitantes.

Estilo	Descrição sumária (cont.)
`dotted`	Borda pontilhada.
`dashed`	Borda tracejada.
`solid`	Borda contínua ou sólida.
`double`	Borda constituída de duas linhas contínuas. A soma das espessuras das linhas com a do espaço que as separa e é igual ao valor de `border-width`.
`groove`	Borda com aparência entalhada.
`ridge`	Borda com aparência de ressalto.
`inset`	Borda em baixo relevo.
`outset`	Borda em alto relevo.

Nas figuras 4.6 e 4.7 são mostrados os aspectos dos diferentes estilos para bordas nos navegadores Firefox e Internet Explorer 6, respectivamente.

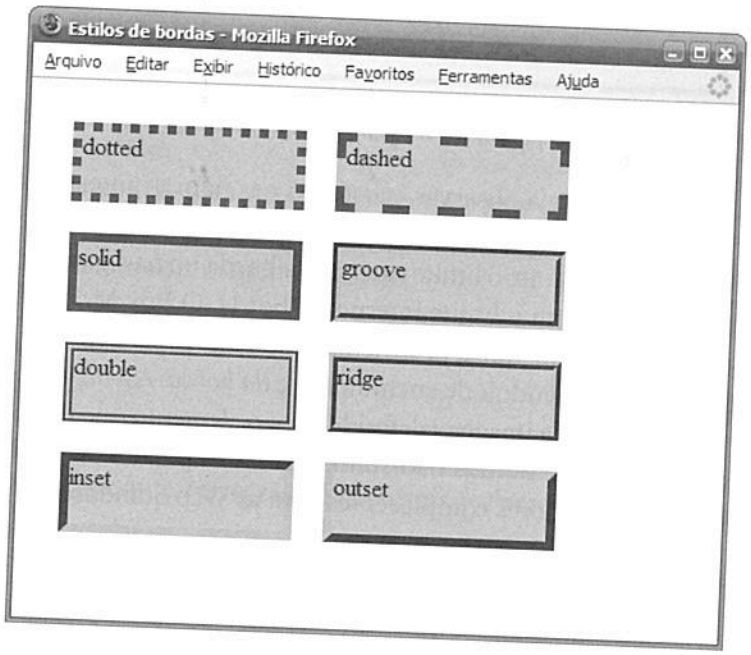

Figura 4.6 – Estilos de bordas no Firefox.

As principais diferenças entre os dois navegadores na apresentação das bordas são notadas nos estilos `dotted` e `dashed`. O navegador simula os efeitos para os quatro últimos estilos de borda mostrados, usando tons de cores. Se a cor definida for a preta, os efeitos não são observados no IE. Sendo colorida, as tonalidades adotadas para o efeito diferem nos dois navegadores.

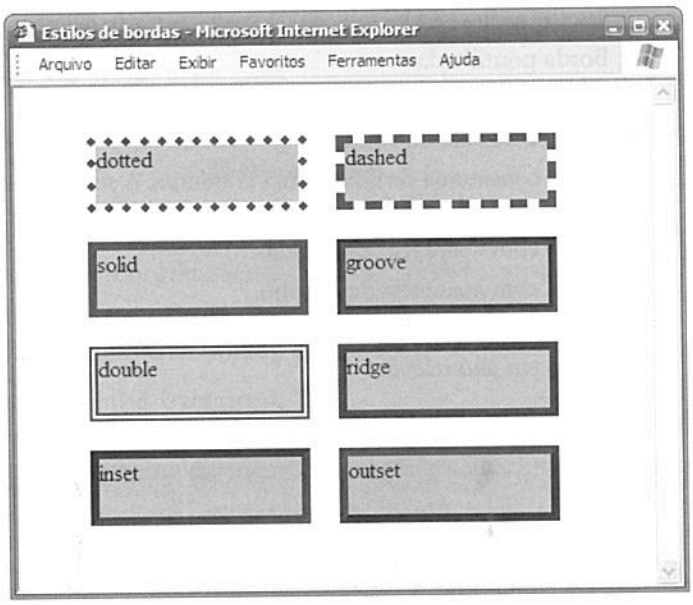

Figura 4.7 – Estilos de bordas no IE6.

Um detalhe importante a observar, mostrado nas figuras anteriores, é a cor de fundo para os estilos de bordas `dotted`, `dashed` e `double`. Observe que a cor cinza para o fundo dos boxes estende-se até o limite externo da borda no navegador Firefox. Para o IE6 o fundo coincide com o limite interno da borda. O Box Model, já estudado anteriormente [C4 S4.3], diz que a propriedade CSS `background` define o fundo a ser aplicado nas áreas de conteúdos, de enchimento e *da borda*. Assim sendo, qualquer fundo, seja uma cor ou uma imagem, definido para o elemento que recebeu a borda, deve ser visto por meio de bordas não-sólidas. É o comportamento adotado pelo Firefox e todos os navegadores complacentes com as Web Standards.

Além disto, convém notar, ainda, que para os estilos `dotted` e `dashed` a renderização é diferente. Os traços da borda `dashed` têm comprimentos diferentes e o formato da borda `dotted` é também diferente. O Navegador Internet Explorer 6 renderiza a borda pontilhada de `1px` como se fosse tracejada. A única maneira de servir uma borda pontilhada de `1px` para o IE6 é com uso de imagem.

border

Define abreviadamente a borda

```
border-top: 1px solid red;
border-right: 2px ridge yellow;
border-bottom: 4px double black;
border-left: 6px inset cyan;
```

ou usar a sintaxe abreviada para bordas que sejam iguais nos quatros lados:

```
border: 2px dotted white;
```

Ao usar a declaração abreviada `border`, não é obrigatório declarar os três valores, portanto as regras CSS mostradas a seguir são válidas:

```
border: 5px;
border: dotted;
border: red;
border: 2px double;
border: solid red;
border: 4px blue;
```

Os valores não-declarados são interpretados pelo navegador como sendo o valor inicial da propriedade. Os valores iniciais das três propriedades são:

```
border-width: medium;
border-style: none;
border-color: /* o mesmo valor da propriedade color do elemento
                 onde se aplica a borda. */
```

Como o valor inicial para o estilo da borda é `none` (nenhum), concluímos que declarações abreviadas que omitem o estilo da borda, como as mostradas a seguir, isoladamente não produzem nenhum efeito.

```
border: 5px;
border: red;
border: 4px blue;
```

Porém, se combinarmos aquelas declarações com outras, obteremos um método de estilização interessante. As regras CSS, a seguir, esclarecem o método proposto:

```
border: 4px blue;
border-style: solid dotted groove double;
```

ou, ainda:

```
border: solid blue;
border-width: 5px 8px 10px 2px;
```

As combinações de espessuras de bordas podem resultar em criações gráficas bastante interessantes. No site de uma designer inglesa há um desafio aberto aos visitantes, o qual consiste em criações gráficas com uso da propriedade CSS `border`. Há uma galeria com alguns trabalhos que pode ser visitada em: `http://www.tanfa.co.uk/css/borders/house.asp` Recomendo, também, uma visita à página demonstrativa de algumas formas construídas com a propriedade `border` no endereço: `http://www.infimum.dk/HTML/slantinfo.html`.