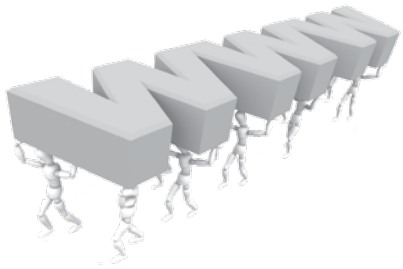


CAPÍTULO 1

Ferramentas básicas de desenvolvimento

Este capítulo destina-se aos iniciantes. Nele apresentaremos as ferramentas mínimas necessárias ao desenvolvimento de sites, comentando e mostrando os softwares relacionados que são instalados por padrão nos ambientes de desenvolvimento mais populares. Serão também mostrados alguns editores (X)HTML, editores de imagens e navegadores, disponíveis para download na internet e de uso gratuito. Se você é um desenvolvedor experiente, poderá ir para o Capítulo 2, sem prejuízo do aprendizado.

1.1 Introdução

Para desenvolver um site, você não precisa adquirir ferramentas de edição sofisticadas e dispendiosas. Particularmente, os editores do tipo WYSIWYG, que são tão bem-aceitos, procurados e usados, são uma ferramenta de desenvolvimento não-recomendada para o início do aprendizado. Tais ferramentas devem ser usadas apenas quando o desenvolvedor tiver total conhecimento e controle da maneira como geram o código.

A sigla WYSIWYG para a expressão What You See Is What You Get é empregada para designar interfaces de produção em que o resultado final do desenvolvimento é semelhante àquele mostrado na interface do software durante a sua criação. A maioria dos editores de texto é desse tipo, como, por exemplo, o Microsoft Word. Nesses editores os textos, paginação, formatação e apresentação do documento são mostrados na tela durante a fase de criação, de maneira idêntica ao resultado final, quando o documento for impresso ou apresentado nas mídias para as quais as criações do editor se destina.

Sites são criados com o uso de editores (X)HTML. O exemplo clássico – e o mais conhecido dos editores – é o Adobe Dreamweaver, uma ferramenta poderosa quando usada de maneira adequada. Editores (X)HTML do tipo WYSIWYG, como é o Drea-

A sintaxe da regra CSS *não é* sensível ao tamanho de caixa da fonte (você pode usar letras minúsculas ou maiúsculas, indiferentemente) e múltiplos espaços são tratados como espaço simples. Usar ou não espaços entre os componentes da regra CSS fica a critério do desenvolvedor. Todas as regras CSS mostradas a seguir são válidas:

```
h1 { border: 1px solid red; }
h1 {border:1px solid red;}
h1{ border:    1px    solid    red;}
H1 {  border: 1px solid RED;}
h1{ BORDER: 1px solid red;  }
```

Eliminar o espaço entre o seletor e o sinal de abrir chaves pode causar confusão em alguns navegadores e deve ser evitado.

Tratando-se de linguagem de programação, sempre que houver mais de uma forma válida de escrever o código o desenvolvedor deve escolher uma delas e adotá-la como seu padrão pessoal. Isto torna o código consistente e facilita a manutenção. Com as folhas de estilo aplica-se esta prática, e você pode escolher qualquer das formas mostradas anteriormente ou, mesmo, outras variações para escrever suas folhas de estilo. Contudo, as duas formas de uso mais difundidas são as mostradas no primeiro e no segundo itens do exemplo anterior. A primeira adota um espaço em branco junto ao sinal de chaves ({ }). A segunda, um espaço somente para separar o seletor da declaração. A forma estendida de declarar as propriedades em linhas distintas é escrita conforme mostrado a seguir, sendo a endentação a critério de desenvolvedor.

```
h1 {
   border: 1px solid red;
   }
```

Um componente facultativo, mas de grande utilidade na escrita de folhas de estilo, é o sinal para inserir comentários. À semelhança de qualquer linguagem de programação, existe um sinal próprio para marcar comentários no código de estilos, conforme mostrado nos exemplos a seguir:

Comentário em uma linha:

```
/* Este é um comentário em uma linha */
```

Bloco de comentário:

```
/*  Este é um bloco de comentário em linhas
   diferentes contendo muitas informações
   sobre uma trecho da folha de estilos */
```

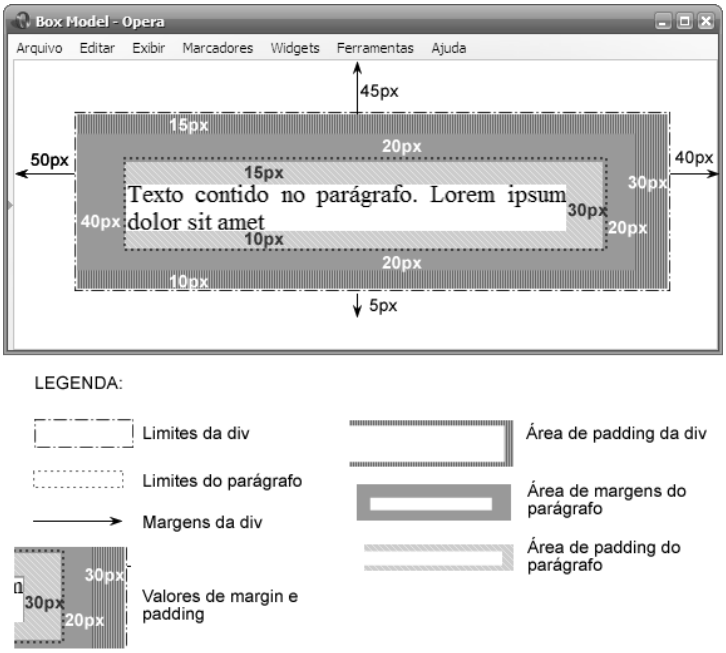


Figura 4.3 – Box Model no Opera.

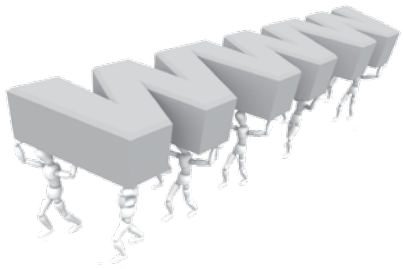
4.4 Propriedades CSS para o Box Model

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;
```



CAPÍTULO 5

Seletores

Neste capítulo veremos os diferentes tipos de seletores CSS. Conhecer os seletores e saber combiná-los de diferentes maneiras coloca nas mãos do autor uma poderosa ferramenta de estilização. No Capítulo 3 tivemos uma idéia da versatilidade dos seletores ao estudarmos o seletor classe e o seletor `id`. Entretanto, as variações não se limitam àqueles dois tipos, indo bem mais adiante, e você verá que o conjunto de seletores possíveis abrange uma extensa lista. Conhecer e saber empregar seletores ajuda a manter o código (X)HTML bem mais limpo, evitando o uso desnecessário de elementos e atributos extras na marcação.

5.1 Definições

Em [C3 S3.1.4.3] e [C3 S3.1.4.4], definimos seletor classe e seletor `id`. Caso você não se lembre destes seletores, volte àquelas seções e faça uma revisão antes de prosseguir. Os seletores se classificam em dois grandes grupos: simples e compostos.

5.1.1 Seletor simples

Seletor simples é aquele constituído de um só elemento, podendo ou não estar associado com uma classe, um `id` ou uma pseudoclasse. São exemplos de seletores simples: `p`, `p.um`, `p#principal`.

5.1.1.1 Seletor universal

É o seletor que casa com todos os elementos contidos no documento. Todas as instâncias de todos os elementos são alvos do seletor. Este seletor é representado pelo sinal gráfico asterisco (*).

Exemplo:

```
* {color: red;}
```

Não havendo padronização, nada garante que uma determinada fonte escolhida pelo autor esteja disponível na máquina do usuário.

Uma má escolha de fontes é mais difícil de acontecer do que uma escolha inadequada de fonte. Estudaremos alguns fundamentos básicos de tipografia que servirão como guia teórico para orientar a escolha tipográfica para seu site.

6.1.1 Termos tipográficos

Como mostrado na Figura 6.1, usaremos a terminologia em inglês para nomear os termos tipográficos, mas faremos uma referência à tradução onde for conveniente.

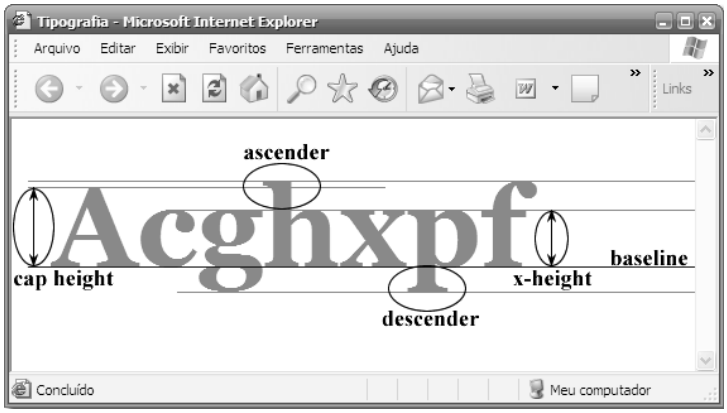


Figura 6.1 – Termos tipográficos.

No Capítulo 3, ao estudarmos medidas CSS de comprimento, vimos a unidade `ex` e agora vamos fazer uma revisão daquele conceito com atenção ao comprimento `x-height` mostrado na Figura 6.1. Este comprimento é igual a `1ex`.

Não só as letras do alfabeto têm suas peculiaridades tipográficas. Os números dividem-se em dois grupos, conforme a disposição corrida dos caracteres numéricos como mostrado na Figura 6.2.

Na primeira linha da Figura 6.2, todos os números têm a mesma altura e o mesmo alinhamento, além de fazerem parte do grupo chamado de *nonranging* (palavra inglesa oriunda do verbo *to range,* que significa percorrer), e os da segunda linha são do grupo denominado *ranging* ou *old-style* (estilo antigo).

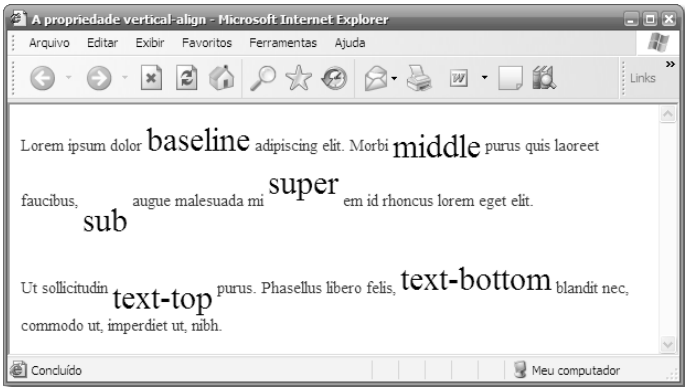


Figura 6.12 – Propriedade `vertical-align` em textos.

O uso mais comum para a propriedade `vertical-align` é o controle do posicionamento vertical de imagens inline. Na Figura 6.13 o efeito da aplicação das regras de estilo já mostradas quando aplicadas às imagens.

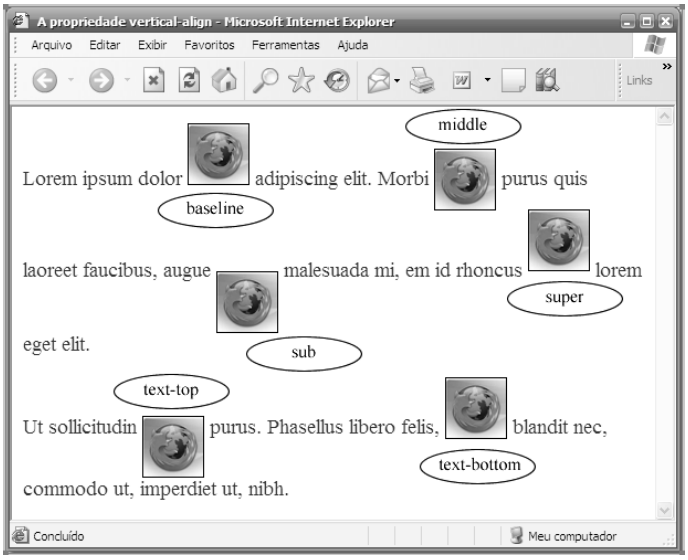


Figura 6.13 – Propriedade `vertical-align` em imagens.

Você pode declarar um valor em porcentagem para a propriedade `vertical-align`. O cálculo da porcentagem é feito tomando como base o valor definido para `line-height`. Um valor negativo causa um deslocamento vertical para baixo e positivo para cima.

Considere a seguinte regra CSS:

```
h1 {color: #6699FF;}
```

Observe que cada um dos pares que representam RGB é formado por números iguais. Neste caso, a sintaxe CSS permite abreviar a grafia para:

```
h1 {color: #69F;}
```

Temos, então, uma forma alternativa com três caracteres hexadecimais para representar uma cor. É de boa prática adotar esta forma abreviada sempre que o formato do número hexadecimal para a cor permitir (os três pares com números iguais entre si).

Esteja ciente de que não se pode abreviar para três caracteres cores como: #808080 e #FFCC63.

7.1.2 Cor RGB

Outra notação menos usada e conhecida é a RGB, que define a cor por uma lista de três números colocados entre parênteses e separados por vírgula, precedida da sigla `rgb`, como mostrado nos seguintes exemplos:

```
h3 {color: rgb(125, 222, 90);}
```

ou:

```
h4 {color: rgb(30%, 25%, 70%);}
```

Observe que são duas as formas de escrever o valor da propridade na regra CSS. Na primeira usamos um número entre 0 e 255 e, na segunda, um valor em porcentagem de 0 a 100%. Tal como na notação hexadecimal, cada número representa a quantidade de red, green e blue (vermelho, verde e azul) que entra na composição da cor.

Não é permitido na sintaxe CSS misturar número com porcentagem para compor uma cor.

7.1.3 Cor por palavra-chave

Podemos usar uma palavra-chave para definir uma cor. As palavras-chave válidas são: `aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `orange`, `purple`, `red`, `silver`, `teal`, `white` e `yellow`. Cada uma destas palavras é o nome de uma cor em inglês, por exemplo, *red* = *vermelha*, *blue* = *azul,* e assim por diante.

A Figura 7.5 mostra o posicionamento da imagem de fundo com uso de palavras-chave.

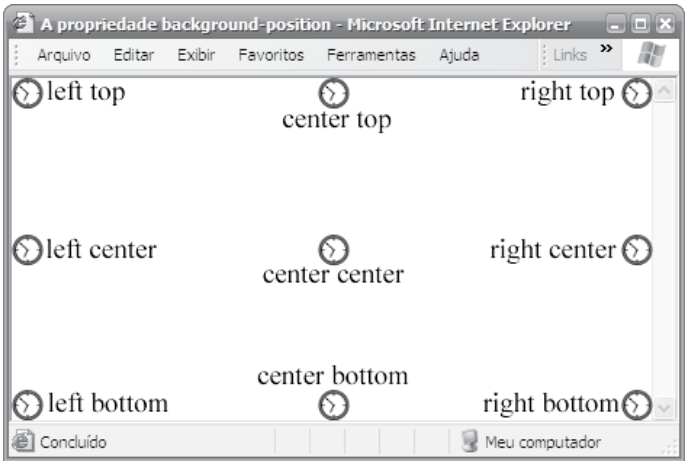


Figura 7.5 – Background-position – Palavra-chave.

Ao declarar valores usando palavras-chave conforme mostrado na Figura 7.5, ficamos restritos a nove posições para a imagem.

Valores outros que não palavras-chave permitem colocar a imagem em qualquer posição na tela. Porcentagem é outro valor válido, mas, ao contrário das palavras-chave, a ordem de declaração das porcentagens é importante e `25% 80%` é uma posição diferente de `80% 25%`. O primeiro valor declarado refere-se ao afastamento a partir do limite esquerdo do elemento onde a imagem está posicionada (valores positivos resultam em afastamento para a direita e valores negativos para a esquerda) e o segundo valor ao afastamento, a partir do limite superior do elemento (valores positivos resultam em afastamento para baixo e valores negativos para cima). Na Figura 7.6 você tem alguns exemplos para exemplificar este tipo de posicionamento.

O exemplo mostrado na figura é ilustrativo. O ponto de referência tomado na imagem para posicionamento depende de como o valor é declarado. Maiores detalhes serão vistos adiante.

O valor 0% na figura tem por finalidade única uniformizar a apresentação do exemplo. Como dissemos no Capítulo 3, não há necessidade de especificar a unidade da medida quando seu valor é 0 (zero).

Entendendo o posicionamento do ícone:

- Em nosso exemplo, o ícone tem as dimensões de `27` x `27px`.
- Decidimos posicioná-lo afastado do lado esquerdo de uma distância igual a `5px`.
- Espaçamos da mesma distância o início do texto do cabeçalho à direita do ícone.
- O alinhamento vertical do ícone será no centro.
- Precisaremos de `5px` + `27px` + `5px` = `37px` de espaço livre à esquerda (`padding-left: 37px;`) para posicionar o ícone.
- O fundo ocupa a área de padding do elemento. Assim, precisamos posicionar o ícone `5px`, afastado do limite esquerdo, ou seja, sua coordenada x é igual a `5px`. Daí vem a regra CSS `background-position: 5px center`, conforme foi visto no Capítulo 7. Podemos omitir a posição `center` nesta declaração, pois ela será assumida automaticamente.

Na Figura 8.10 vemos o detalhe do posicionamento do ícone.

Figura 8.10 – Posicionamento do ícone.

Flutuado

```
h1 {
   border: 10px solid #069;
   background-color: #daf6d1;
   width: 6em;
   height: 2.5em;
   float: left;
   margin: 0 0.5em 0 0;
   text-align: center;
   }
```

Neste exemplo definimos largura e altura para o elemento cabeçalho de modo a confiná-lo em um box com aquelas dimensões. Observe que a definição das dimen-

Em HTML, o fechamento da tag `li` é facultativo. Será praxe fecharmos esta e todas as tags, pois estamos baseados em XHTML, cuja sintaxe não permite tags abertas.

9.1.2 Listas ordenadas

Para marcar uma lista na qual é importante a ordenação da informação, usamos o elemento `ol`. Cada item da lista é marcado com o elemento `li`, conforme mostrado no seguinte exemplo:

```
<ol>
   <li>Bata tudo no liquidificador</li>
   <li>Misture o fermento lentamente</li>
   <li>Asse em forno pré-aquecido</li>
</ol>
```

Listas ordenadas normalmente são renderizadas pelo navegador com uma marca seqüencial, antes do texto de cada item, indicando a ordenação.

9.1.3 Listas de definição

Listas de definição são aquelas constituídas por uma série de pares termo/definição(ões) e para marcá-las usamos o elemento `dl`. Cada grupo de informação que se repete na lista é marcado com um elemento `dt`, definindo o termo da lista e um ou mais elementos `dd` para as definições do termo, tal como mostrado no seguinte exemplo:

```
<dl>
   <dt>Baixo custo</dt>
      <dd>Significativa redução de preço em relação ao modelo anterior</dd>
   <dt>Fácil de usar</dt>
      <dd>Painel automático com dicas de operação</dd>
   <dt>Segurança</dt>
      <dd>Proteção contra choque elétrico</dd>
      <dd>Revestido com material inflamável</dd>
</dl>
```

Listas de definição normalmente são renderizadas pelo navegador com recuos de texto (endentações) para destacar os termos das definições.

Na Figura 9.1 a renderização-padrão dos três tipos de listas mostrados.

```
    border: 1px solid #ccc;
    border-width: 0 1px 1px;
    }
#nav a:hover {
    text-decoration: none;
    background: #c7daec;
    color:#69c;
    }
#nav a.bullet {background:#69c url(bullet.gif) no-repeat right;}
#nav a.bullet:hover {background: #c7daec url(bullet-over.gif) no-repeat right;      }
#nav li ul {
    position: absolute;
    left: -1000em;
    width: 10em;
    margin: 0;
    }
#nav ul ul {margin: -2.1em 0 0 10.1em;}
#nav li:hover ul ul, #nav li.over ul ul {left: -1000em;}
#nav li:hover ul, #nav li li:hover ul, #nav li.over ul, #nav li li.over ul {
    left: auto;
}
```

Convém destacar as seguintes particularidades na folha de estilo:

- A declaração de largura (`10em`) e altura (`2em`) para o elemento `a` permitirá posicionar os submenus.
- As declarações `position: absolute;` e `left: -1000px;` escondem os submenus e a declaração `left: auto;` revela os submenus.
- As declarações de margens para `#nav ul ul` posicionam os submenus.
- A classe `over` é inserida por JavaScript, pois esta versão usa o mesmo código JavaScript da versão vertical.

A Figura 9.25 mostra o menu drop down na versão horizontal.

Figura 9.25 – Drop down horizontal.

Na Figura 10.13 é mostrada a renderização-padrão do código. Observar que os navegadores Internet Explorer 6 e Firefox estilizam de maneira diferente o controle desabilitado. Podemos obter estilização idêntica nos dois navegadores com uso das CSS, conforme veremos.

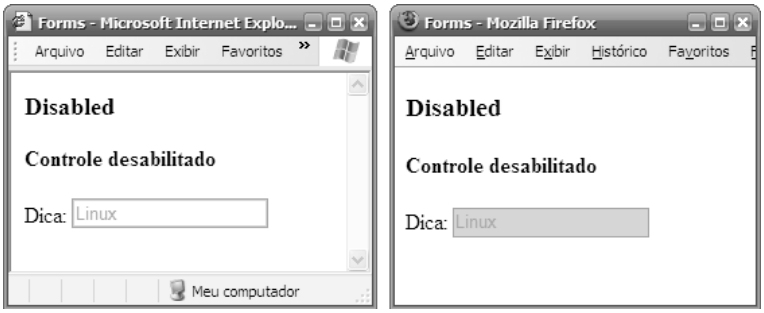


Figura 10.13 – Atributo `disabled`.

O atributo `readonly` põe o controle no modo somente para leitura e produz os seguintes efeitos em um elemento:

- Recebe o foco, mas não pode ser modificado pelo usuário.
- São incluídos normalmente na navegação por tabulação.
- Pode enviar o dado contido no controle para processamento quando o formulário for enviado.

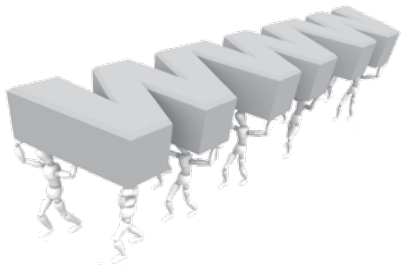
Os seguintes elementos suportam o atributo `readonly`: `input` e `textarea`.

A maneira como são renderizados os elementos em modo somente para leitura depende do agente de usuário. No exemplo seguinte, o elemento `input` está em modo somente para leitura. Em conseqüência, ele não pode receber dados de entrada do usuário, mas pode ter seu valor enviado para processamento.

```
<form action="">
   <label>Dica:
      <input readonly="readonly" value="Linux">
   </label>
</form>
```

Notar que em HTML basta escrever o atributo `readonly`, mas em XHTML é obrigatório indicar o seu valor, escrevendo `readonly ="readonly"` [C2 S2.2.3.6].

A Figura 10.14 mostra a renderização-padrão do código. Notar que os navegadores Internet Explorer 6 e Firefox estilizam de maneira diferente o controle desabilitado. Podemos obter estilização idêntica nos dois navegadores com uso das CSS, conforme veremos.



CAPÍTULO 11

Criando tabelas Web Standards

Tabelas são por excelência o identificador e o grande diferencial entre a marcação HTML ultrapassada e a moderna marcação de websites. Adaptando um dito popular bem-conhecido, é possível dizer: diga-me como usa as tabelas na sua marcação (X)HTML e lhe direi que espécie de desenvolvedor web você é.

Neste capítulo mostraremos quando e como devem ser usadas as tabelas em desenvolvimento de sites Web Standards. Desde simples tabelas contendo uma única linha de entrada até aquelas complexas de dupla entrada e duplos cabeçalhos. O emprego dos elementos de marcação para tabelas é estudado e analisado com detalhes nos códigos e exemplos apresentados.

11.1 Quando usar tabelas

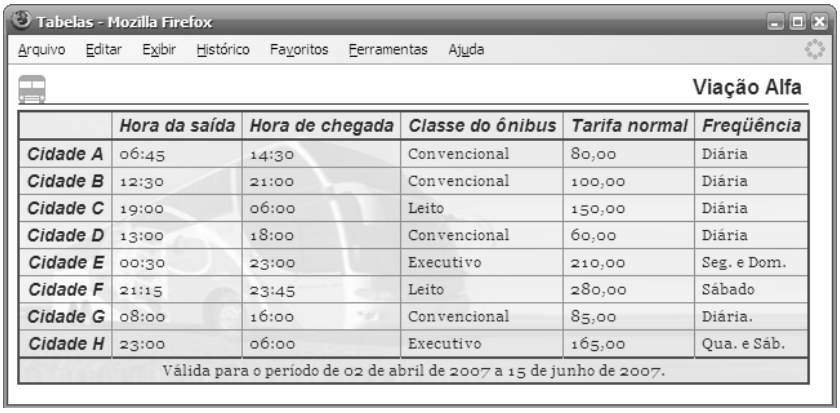
Infelizmente, um dos primeiros conceitos errôneos assimilados pela maioria daqueles que começam a estudar os padrões web é acreditar que as tabelas estão definitivamente banidas dos projetos criados para atender as Web Standards. O conceito tableless (do inglês: "sem tabelas") ganhou força na comunidade brasileira de desenvolvimento e foi interpretado literal e distorcidamente como se tabelas fossem proibidas. Tabelas estão previstas nas especificações da (X)HTML, são legais e devem ser usadas para os fins a que se destinam.

O elemento `table` foi introduzido com o HTML 2.0 em 1994 com a finalidade de apresentar dados tabulares como, por exemplo, tabelas de tempos, de informações tabulares sobre pesos, medidas, preços, tabelas de dados gerais etc.

Uma tabela é renderizada em um navegador gráfico, como uma espécie de grade, composta de colunas, linhas e células, podendo conter texto, imagem, link, script

```
tfoot tr td {
   text-align: center;
   border-top: 2px solid #069;
   }
tr td, tr th {
   padding: 2px 5px;
   font-size: 1.1em;
   border: 1px solid #69c;
   }
tr th {
   font: italic bold 1.4em Arial, Helvetica, sans-serif;
   }
```

A Figura 11.16 mostra o efeito desta nova folha de estilo aplicando as imagens de fundo nos elementos `table` e `caption`.



	Hora da saída	Hora de chegada	Classe do ônibus	Tarifa normal	Freqüência
Cidade A	06:45	14:30	Convencional	80,00	Diária
Cidade B	12:30	21:00	Convencional	100,00	Diária
Cidade C	19:00	06:00	Leito	150,00	Diária
Cidade D	13:00	18:00	Convencional	60,00	Diária
Cidade E	00:30	23:00	Executivo	210,00	Seg. e Dom.
Cidade F	21:15	23:45	Leito	280,00	Sábado
Cidade G	08:00	16:00	Convencional	85,00	Diária.
Cidade H	23:00	06:00	Executivo	165,00	Qua. e Sáb.

Figura 11.16 – Imagens de fundo.

Para inserir ícones nos cabeçalhos da tabela atribuímos uma classe para cada cabeçalho e definimos a imagem do ícone como fundo. As alterações na marcação são:

```
...
<thead>
      <tr>
         <td></td>
         <th scope="col" class="hora">Hora da saída</th>
         <th scope="col" class="hora">Hora de chegada</th>
         <th scope="col" class="classe">Classe do ônibus</th>
         <th scope="col" class="tarifa">Tarifa normal</th>
         <th scope="col" class="frequencia">Frequência</th>
      </tr>
   </thead>
...
```

Figura 12.8 – Imagem inline em parágrafo.

Para flutuar a imagem à esquerda ou à direita, aplicamos a seguinte regra CSS:

```
img {float: left;}
```

ou

```
img {float: right;}
```

Obtendo o efeito mostrado na Figura 12.9:

Figura 12.9 – Imagem float.

Observar que para elementos inline o comportamento é diferente daquele de blocos flutuados. A imagem flutuada foi deslocada para uma posição até tocar a borda do seu elemento container, no exemplo um parágrafo, e tanto o texto conteúdo do container quanto o elemento bloco que se segue no fluxo do documento, o segundo parágrafo, ocuparam uma posição em volta da imagem. Na prática, tal comportamento é usado para alinhar imagens à esquerda ou à direita de um texto.

```
#navegacao {
    width: 23%;
    float: left;
    background: #a2daec;
    }
#rodape {
    clear: both;
    height: 25px;
    background: #ffecd1;
    }
```

A Figura 13.8 mostra a rende rização do nosso layout líquido nas três resoluções mais usadas.

Figura 13.8 – Layout líquido.

Notar nesta solução que, independente do tamanho da janela do navegador, pelo fato de o layout ter sido definido com uma largura total de 85% e estar centralizado na tela, teremos espaços vazios de margens com largura de 7,5% para cada lado. Optamos ainda por manter os valores de `padding` em pixel e não porcentagens a fim de evitar quebra do layout em versões mais antigas do Internet Explorer.

Conforme vimos na Seção 13.1.2, em alguns casos, para previnir perda de legibilidade em linhas de texto muito curtas ou muito extensas, podemos definir uma largura mínima e outra máxima para nosso layout, acresentando as seguintes declarações CSS na folha de estilo:

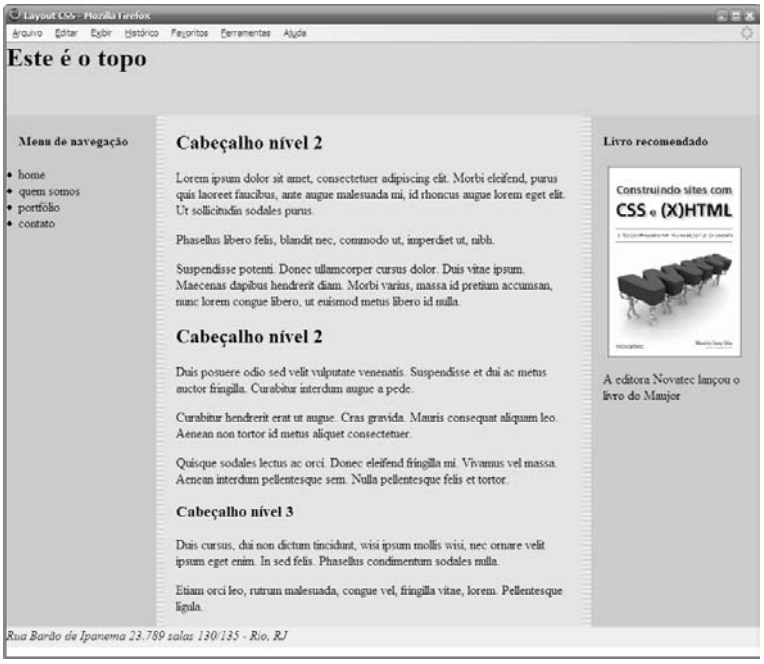


Figura 13.18 – Aplicação de faux columns no layout.

Contudo, temos um problema ainda a resolver. Observe na Figura 13.19 o que acontece quando os conteúdos das colunas laterais as fazem mais longas do que a coluna principal.

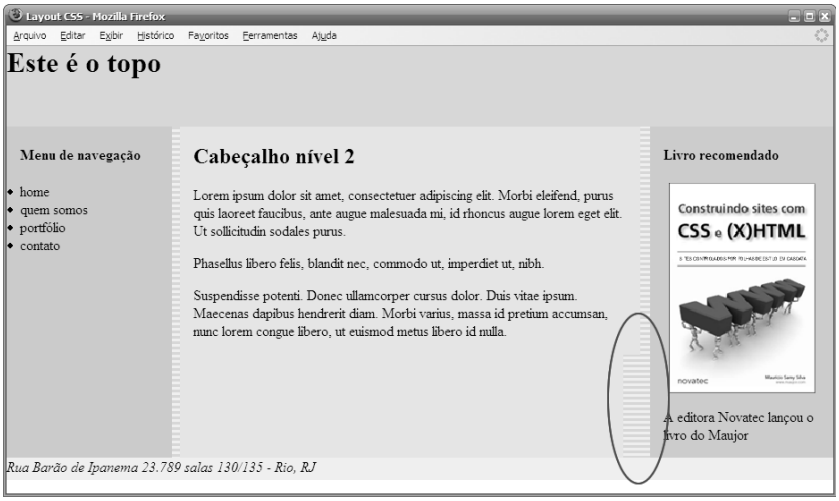


Figura 13.19 – Quebra na faux column.

Problemas causados por `haslayout` afetam designers e programadores de todos os níveis de experiência. É difícil prever os efeitos sobre a apresentação e o comportamento de boxes, assim como para os elementos contidos em tais boxes.

As conseqüências de um elemento ter ou não layout podem ser responsáveis, entre muitas outras, por:

- Muitos dos bugs de `floats` no IE.
- Tratamento diferenciado para propriedades básicas dos boxes.
- Sobreposição de margens entre um container e seus descendentes.
- Vários problemas relacionados à construção de listas.
- Diferenças no posicionamento de imagens de fundo.
- Diferenças entre navegadores quando do uso de scripts.

Você não tem idéia da causa de um bug no Internet Explorer? Abra o arquivo da folha de estilo e declare a regra CSS: `* {zoom: 1;}`. Esta regra dá layout para todos os elementos (* seletor universal) da página. Recarregue a página, e se o bug sumir é porque vinha sendo causado por um elemento que não tinha layout. Continue, dando layout a seções específicas da página, por exemplo: `#navegação * {zoom: 1;}`. Se o bug sumir, o elemento sem layout está na seção de navegação do site. E assim por diante.

Ingo Chao, um desenvolvedor residente na Alemanha, publica em seu site um relatório completo com o resultado do andamento das pesquisas sobre `haslayout` realizadas por uma equipe composta por seis experts efetivos e quatro colaboradores.

A matéria vem sendo atualizada sempre que novas descobertas são feitas, e encontra-se on-line em: `http://www.satzansatz.de/cssd/onhavinglayout.html`. Uma versão em português pode ser acessada em: `http://www.maujor.com/tutorial/haslayout.php`.

14.5 Comentários condicionais para o Internet Explorer

Comentários condicionais é mais uma implementação proprietária da Microsoft, criada para servir de código ao navegador Internet Explorer nas versões 5 e superiores, rodando em ambiente Windows. Códigos servidos com comentários condicionais devem ser inseridos na marcação (X)HTM, tanto na seção `head` quanto na `body` do documento. Você não pode inserir comentários condicionais em arquivos de folhas de estilo. Observe o código:

pessoal, e a opção por uma delas deve ser fundamentada nas peculiaridades de cada projeto.

15.2 Uma imagem para múltiplas substituições

Se o seu projeto prevê o uso da técnica de substituição de textos por imagem em diferentes locais da página, considere a possibilidade e a conveniência de construir uma imagem para substituir diferentes textos. A vantagem de se optar por esta solução é o fato de que todas as imagens necessárias à substituição serão carregadas de uma só vez. Em mecanismos de navegação com rollover de imagens, não haverá atraso no aparecimento da imagem quando ocorrer a mudança de estado do link, pois as imagens necessárias ao rollover serão carregadas de uma só vez.

A título de exemplificação, vamos considerar que em uma página o designer projetou todos os cabeçalhos com uma fonte personalizada, e ao desenvolvedor não restou outra alternativa a não ser substituir os textos do cabeçalho por imagens. Os cabeçalhos serão de acordo com a fonte, segundo o modelo mostrado na Figura 15.3:

Figura 15.3 – Fonte do cabeçalho.

Construa uma imagem com todos os cabeçalhos conforme mostrado na Figura 15.4:

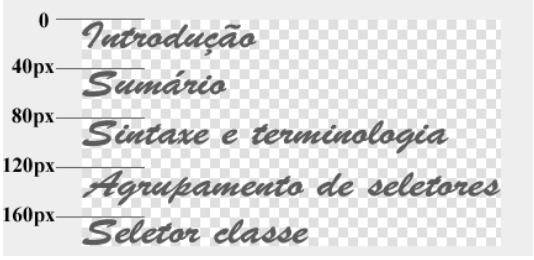
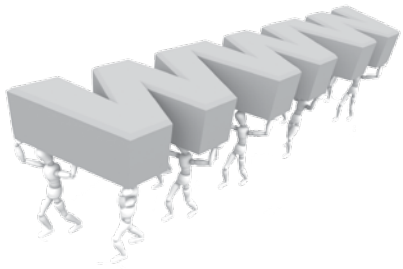


Figura 15.4 – Imagem dos cabeçalhos.

A imagem é uma gif transparente que foi gravada com o nome `cabecalhos.gif`. Notar que colocamos os cabeçalhos afastados verticalmente um do outro de um valos igual a `40px`. É importante adotar um espaçamento constante e preferencialmente múltiplos de 5 ou de 10, pois serão as coordenadas verticais para posicionar a imagem de fundo com a propriedade CSS `background`.



APÊNDICE C

Propriedades CSS 2.1

A tabela constante deste apêndice foi extraída do site do W3C e descreve as propriedades CSS 2.1 e seus valores possíveis. Informa ainda o valor inicial de cada propriedade, se é herdada ou não e a qual mídia aplica.

Na coluna dos valores das propriedades, o W3C adota uma notação que é explicada a seguir:

Notação	Significado
< >	tipo de valor ou referência para outra propriedade
[ ]	agrupamento
|	uma das alternativas deve ocorrer
||	uma ou mais alternativas podem ocorrer
*	0 ou mais
+	1 ou mais
?	0 ou 1
{x y}	mínimo de x e máximo de y podem ocorrer

Exemplos:

- <porcentagem> – A propriedade admite valores em porcentagem.
- `<border-width>` – A propriedade admite os mesmos valores da propriedade `border-width`.
- `scroll` | `fixed` | `inherit` – Deve ser declarado um dos valores para a propriedade.
- `<border-width>` || `<border-style>` || `'border-top-color'` – A propriedade admite declarar-se um, dois ou os três valores.