

DESVENDANDO OS RECURSOS DO

CSS

Aprenda passo a passo a criar folhas de estilos

GUIA PRÁTICO DO CSS

- Síntaxe básica, inserção de imagem, formatação, cores
- Como criar listas e tabelas no CSS
- Exemplos de Mapa de Imagem e Image Replacement
- Tudo sobre criação e uso de Classes

CSS AVANÇADO

- Classificação e enquadramento
- Como tornar um elemento invisível
- Posicionamento estático, fixo, relativo e absoluto
- Exemplo de texto com sombra e com contorno

RECURSOS ADICIONAIS

- Criação de menus sofisticados
- Exemplo de criação de formulários
- Uso do CSS no Dreamweaver
- Criação de arquivos CSS externos

RENATA HIROMI MINAMI MIYAGUSKU

DESVENDANDO OS RECURSOS DO

CSS

Aprenda passo a passo a criar folhas de estilos

GUIA PRÁTICO DO CSS

- Sintaxe básica, inserção de imagem, formatação, cores
- Como criar listas e tabelas no CSS
- Exemplos de Mapa de Imagem e Image Replacement
- Tudo sobre criação e uso de Classes

CSS AVANÇADO

- Classificação e enquadramento
- Como tornar um elemento invisível
- Posicionamento estático, fixo, relativo e absoluto
- Exemplo de texto com sombra e com contorno

RECURSOS ADICIONAIS

- Criação de menus sofisticados
- Exemplo de criação de formulários
- Uso do CSS no Dreamweaver
- Criação de arquivos CSS externos



© 2007 by Digerati Books

Todos os direitos reservados e protegidos pela Lei 9.610 de 19/02/1998. Nenhuma parte deste livro, sem autorização prévia por escrito da editora, poderá ser reproduzida ou transmitida sejam quais forem os meios empregados: eletrônicos, mecânicos, fotográficos, gravação ou quaisquer outros.

Diretor Editorial

Luis Matos

Assistência Editorial

Carolina Evangelista

Preparação dos originais

Jeferson Ferreira

Projeto Gráfico

Daniele Fátima

Revisão

Guilherme Laurito Summa

Diagramação

Daniele Fátima

Fabiana Pedrozo

Capa

Daniel Brito

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

M658d Miyagusku, Renata.

Desvendando os Recursos do CSS / Renata
Miyagusku. – São Paulo: Digerati Books, 2007.
112 p.

ISBN 978-85-60480-55-5

1. Cascading Style Sheets (CSS).
2. Webdesign. I. Título.

CDD 005.72

Universo dos Livros Editora Ltda.

Rua Tito, 1.609

CEP 05051-001 • São Paulo/SP

Telefone: (11) 3648-9090 • Fax: (11) 3648-9083

www.universodoslivros.com.br

e-mail: editor@universodoslivros.com.br

Conselho Administrativo: Alessandro Gerardi, Alessio Fon Melozo, Luis
Afonso G. Neira, Luis Matos e William Nakamura.

Introdução

Muito se fala a respeito de programação Web e a maioria das pessoas que entra neste universo, baseando-se em exemplos e tutoriais de todas as procedências (alguns muito confiáveis, outros nem tanto), acaba tornando-se, de certa forma, “desenvolvedores”. É mínimo o número de pessoas que tem conhecimento a respeito dos chamados *Web Standards*, ou Padrões Web. Esse padrão foi desenvolvido (e ainda é, já que se trata de um processo contínuo) pelo W3C (*World Wide Web Consortium*), que tem como site o www.w3.org. Trata-se de um consórcio de empresas de tecnologia cujo principal objetivo é desenvolver e promover padrões mundiais de programação que permitam que uma aplicação ou site sejam acessados em qualquer lugar, a partir dos mais diversos dispositivos (computadores, PDAs, celulares etc.), sem que haja perda de características estéticas e funcionais. Fazem parte dessa padronização as linguagens HTML, XHTML, XML e CSS, entre outras.

Este livro oferecerá ao leitor um curso completo de CSS (*Cascading Style Sheets*, ou Folhas de Estilo em Cascata) baseado nos padrões W3C. Ele garante que a base teórica para o desenvolvimento das folhas de estilo está de acordo com o padrão mundial de interoperabilidade. Portanto, o leitor, ao colocar em prática os conceitos aqui apresentados, programará sites com layouts flexíveis às mais diversas plataformas, baseadas em códigos legíveis e bem estruturados.

O conteúdo do livro é composto por um primeiro capítulo com conceitos básicos, que prepararão o leitor para o desenvolvimento das folhas de estilo propriamente ditas. Esses conceitos incluem a definição do que são as CSS, qual suas estruturas básicas e quais são os tipos de folhas de estilo existentes. Além disso, serão fornecidas referências a respeito de unidades de medidas e cores, que servirão de consulta ao longo de todas as implementações.

O segundo capítulo contém uma relação dos principais elementos e seus atributos, ou seja, quais características podem ser alteradas por meio das

folhas de estilo, e como isso é feito. Além das respectivas definições, cada item vem acompanhado de exemplos que facilitam o entendimento de como funciona a estilização dos elementos mediante a formatação de seus atributos.

O terceiro capítulo aborda e exemplifica a criação de classes, cuja função principal é diferenciar folhas de estilos para um mesmo tipo de elemento. Utilizando classes, é possível criar, por exemplo, dois ou mais tipos de cabeçalhos sem que um estilo interfira no outro.

O quarto capítulo é composto por tópicos avançados relacionados às CSS que tornarão o layout dos seus sites muito mais elaborados e atraentes. Esses recursos incluem classificação e enquadramento para modificar não apenas o aspecto, mas também a disposição dos elementos em uma página, as diferenças entre posicionamento absoluto e relativo, e como utilizá-las, definição e exemplos de pseudo-classes e pseudo-elementos, filtros de transparência, criação de menus e formulários de diversos tipos.

O capítulo cinco fornece informações a respeito da diferenciação de estilos em relação a tipos de mídia (dispositivos de saída) distintos, isto é, o leitor aprenderá a definir, para uma única página, estilos diferentes para visualizá-la na tela do computador, ou para imprimi-la, visto que na maioria dos casos uma única formatação não se conforma a diferentes saídas.

Para finalizar, o sexto capítulo é composto por tópicos relacionados à utilização do Adobe Dreamweaver, que, apesar de ser um programa específico para desenvolvimento de documentos HTML, com o tempo se aperfeiçoou e tem se mostrado uma ferramenta muito útil para gerar, editar e organizar folhas de estilo, tanto em arquivos CSS como arquivos embutidos em páginas HTML.

Pré-requisitos

Este livro é voltado inteiramente ao desenvolvimento das folhas de estilo em cascata (CSS), e, por isso, é fundamental que o leitor já tenha pelo menos um conhecimento básico da linguagem HTML, como definição e hierarquia das tags, criação de páginas etc. Pelo fato de as CSS possuírem relação direta com essa linguagem, praticamente todos os exemplos demonstrados neste livro envolverão as duas linguagens.

Capítulo 1

Conceitos básicos sobre CSS

Este capítulo tem como principal objetivo fornecer uma visão geral a respeito das *Cascading Style Sheets*, incluindo elementos como definição, principais funções e vantagens de uso na criação de sites esteticamente padronizados, fácil manutenção e alta flexibilidade de layouts.

Além disso, será dado um breve resumo sobre o padrão Tableless e sua relação com as folhas de estilo no que se refere aos padrões de execução.

Será demonstrada, ainda neste capítulo, a sintaxe básica das CSS, bem como suas formas de implementação e uso, além de outros conceitos básicos que serão utilizados ao longo de todo o livro, como unidades de medidas, cores etc. Enfim, todas as informações fundamentais que servirão de ponto de partida para iniciarmos o desenvolvimento das folhas de estilo.

Definição

As CSS surgiram como uma solução às deficiências e limitações que a linguagem HTML começou a apresentar já há algum tempo. Com o aparecimento de *sites* mais complexos, repletos de recursos e informações – levando em conta, sempre, o aspecto estético das páginas –, houve como consequência negativa a formação de arquivos HTML compostos por informações e formatações misturadas. Isso tornou a manutenção de tais arquivos um tanto complicada e demorada, o que, na maioria das vezes, tornava as páginas inconsistentes e sem padrão.

A função principal das CSS é, justamente, extrair a formatação de uma página do código HTML, separando-a do conteúdo propriamente dito (informações). Além de aumentar o nível de organização, isso indica que elas podem definir, de antemão, a formatação de todos os elementos de uma ou várias páginas. Com isso, torna-se muito mais fácil manter um padrão de fontes, cores e estilos, na medida em que será mais prático modificar tais atributos, pois tal atividade será executada a partir do tipo de elemento ou classe (e não da informação). Toda alteração nos elementos e atributos das CSS modificam toda a página que a utiliza, dispensando a alteração tag a tag. Essa estilização também inclui mudança de propriedades em função de eventos.

Um layout definido em uma folha de estilo pode ser aplicado com referência ao arquivo que contém as formatações, assim como os estilos podem ser definidos no cabeçalho do arquivo HTML que o utilizará (por meio da tag `<style>`). Porém, neste último caso, a diferença para a formatação mesclada ao conteúdo praticamente não existe, e perdem-se muitas das vantagens deste tipo de implementação. Ela deve ser utilizada somente em casos de formatações específicas para um único elemento.

Utilizando um arquivo CSS externo à sua página, é possível, por exemplo, possuir vários arquivos CSS diferentes, e, quando se deseja modificar o layout de uma página, basta alterar a chamada ao arquivo CSS.

Resumindo, o uso das folhas de estilo (principalmente as externas) fornece não só a certeza de que todo o seu site estará padronizado e formatado por igual, mas também uma enorme economia de trabalho. Uma vez que um arquivo CSS puder ser reaproveitado para definir o layout de várias páginas ao mesmo tempo, dispensa-se a repetição de todo o código em cada arquivo HTML.

Padrão Tableless

Como o próprio nome descreve, trata-se de um padrão de desenvolvimento de páginas com layouts não baseados em tabelas. Isso significa projetar layout e conteúdo separadamente, independentes um do outro. Algumas pessoas confundem o conceito, dizendo que o padrão condena qualquer uso de tabelas, mas não se trata exatamente disso. Esse padrão defende o uso de tabelas apenas para tabular dados, ou seja, para o propósito original da tag criada. Para os seguidores desse padrão, separar o layout do conteúdo significa desenvolver páginas mais leves e mais flexíveis, na medida em que o conteúdo livre se adaptará a qualquer browser, plataforma etc.

Esse conceito vai ao encontro das CSS, visto que o objetivo principal das folhas de estilo é justamente possibilitar a separação de formatação e conteúdo, e, assim, fornecer maior flexibilidade às páginas HTML.

Sintaxe básica

Basicamente, a sintaxe das CSS consiste em:

```
elemento { atributo: valor;  
atributo1: valor;  
.  
.  
.  
}
```

Onde elemento corresponde ao elemento da linguagem HTML (parágrafos, cabeçalhos, componentes de listas e tabelas etc). Entre chaves, serão definidos os atributos desse elemento e seus respectivos valores.

As CSS não são *case sensitive*, ou seja, não diferenciam letras maiúsculas de minúsculas. Essa característica permite que cada desenvolvedor estabeleça seu padrão de nomenclaturas, desde que esses nomes não sejam compostos (se forem compostos, utilize hífen ou *underline*). Lembre-se de que é importante ter bom senso: escolha nomes intuitivos (que lembrem sua função ou tipo de objeto que referencia) na criação de estilos para manter o documento legível para futuras manipulações, além de manter o código indentado para uma melhor visualização da hierarquia dos elementos e atributos. Caso o leitor queira inserir comentários dentro do código, basta delimitá-los com as marcações `/*` e `*/`.

É importante lembrar que nas CSS os atributos (propriedades dos elementos) possuem identificadores com nomes únicos, isto é, os nomes compostos são sempre ligados por hifens. Alguns nomes são iguais aos atributos das tags, mas também serão encontradas diferenças de nomenclatura (bgcolor e Background-color).

Em relação aos elementos, existe a possibilidade de criar um estilo com um ID próprio e único. Sua sintaxe seria a seguinte, na definição do estilo:

```
#texto { color: #000000 }
```

Já no arquivo HTML:

```
<div id="texto">  
Texto formatado pelo estilo identificado pelo ID #texto.  
</div>
```

É importante lembrar que esse ID só pode ser utilizado por um elemento: um único elemento empregará um estilo único, não sendo possível aplicar este ID a outros elementos.

Tipos de CSS

O termo “em cascata” deve-se ao fato de haver diversas folhas de estilo, com uma sucedendo a outra, em diferentes níveis, e todas ligadas a um mesmo documento HTML. Porém, somente a definição da formatação mais interna (mais próxima ao código HTML) será respeitada. As CSS podem ser definidas de três formas: diretamente no código HTML, em uma seção própria para definições das formatações no *header* do documento, ou em um arquivo CSS externo. Se houver uma formatação de um determinado elemento dentro do documento HTML, e este também fizer referência a um arquivo CSS externo, por exemplo, será respeitada a formatação definida no documento HTML.

A seguir, serão detalhadas e exemplificadas essas três formas de utilizar as CSS em uma página HTML.

Arquivo CSS externo

Criar um arquivo CSS para definir o estilo de uma página é um modo prático de desvincular o layout do conteúdo de uma página e tornar possível o reaproveitamento de uma série de formatações, para serem usadas em outros sites. A única preocupação que se deve tomar é, uma vez que um arquivo CSS é chamado em várias páginas distintas, alterá-lo significa modificar o layout de todas as páginas que o utiliza.

Para criar um arquivo CSS, basta definir os elementos e suas formatações em um editor de textos, e salvá-lo com a extensão `.css`. Veja exemplos de definições:

```
h1 { color: blue }  
body { background-image: imagem.gif }  
ul { list-style-type: circle }
```

A primeira linha do código define a cor do cabeçalho `h1`. A segunda define a imagem do plano de fundo, ao passo que a terceira define o tipo do marcador da lista.

Salvando essa sequência em um arquivo chamado **formato.css**, ele será chamado como atributo da tag `<link>`, dentro da tag `<head>`, na página HTML, com as informações de cabeçalho da página:

```
<head>
    <link rel="stylesheet" type="text/css"
href="formato.css"/>
</head>
```

CSS no *header* da página HTML

Também é possível colocar essas definições de formatação no próprio arquivo HTML. Basta utilizar a tag `<style>`, que será chamada dentro da tag `<head>`. Este será o tipo de CSS mais utilizado no livro:

```
<head>
    <style type="text/css">
h1 { font-color: blue }
body { background-image: imagem.gif }
ul { list-style-type: circle }
    </style>
</head>
```

CSS *inline*

Este modo de definir a formatação dos elementos de uma página não difere muito da formatação com tags HTML, na medida em que não poupa o trabalho do desenvolvedor de adequar elemento a elemento em uma página. Essa forma de utilizar as CSS só se mostra útil quando se trata de uma formatação específica para aquele conteúdo (assim, ignora-se a formatação CSS anterior), além de eliminar o aninhamento de tags (que torna o código extenso e difícil de visualizar), já que a formatação é feita em uma tag apenas. Veja um exemplo, que define o alinhamento de um cabeçalho:

```
<body>
<h1 style="text-align: center">Cabeçalho</h1>
.
.
.
</body>
```

Quando utilizamos várias CSS em uma única página, principalmente de tipos diferentes, o estilo a ser seguido será o mais detalhado, ou o que

estiver mais próximo do código HTML formatado por elas. Em outras palavras, se tivermos uma folha de estilo em um arquivo CSS com a formatação de um elemento, e, no arquivo HTML que o referencia existir uma estilização diferente para o mesmo elemento, será seguido esse estilo, desde que ele tenha igual ou maior detalhamento. No entanto, é possível definir a prioridade de um estilo em relação aos outros com o uso da declaração !important após a definição do valor do atributo, independente de localização da estilização:

```
<style type="text/css">
h1 { font-color: blue !important; }
</style>
```

No exemplo anterior, mesmo que o estilo definido esteja em um arquivo externo e exista uma formatação para a tag <h1> no código HTML, é o estilo com a marcação !important que será obedecido.

Se tratar-se de apenas uma página HTML, dê preferência à definição das folhas de estilo em arquivos CSS separados, ao invés de incluí-las no código HTML, principalmente quando um site é composto por várias páginas e se deseja manter uma padronização de layout. Arquivos externos significam folhas de estilos reaproveitáveis: é possível montar “bibliotecas” de arquivos CSS para os mais diversos fins. Além disso, o fato de essas definições estarem desvinculadas do conteúdo das páginas, não só resultam em um código mais “limpo” e legível, mas também facilitam futuras manutenções e alterações.

As tags *<div>* e **

A tag `<div>`, embora pertença ao HTML, tem papel importante na atribuição de estilos na medida em que divide um documento em seções, que, por sua vez, poderão conter elementos formatados por estilos que pertencem somente àquela seção.

As seções definidas por meio da tag `<div>` comportam-se de diversas maneiras; tudo depende de como ela é utilizada. Tags separadas servem para criar seções distintas que não se comunicam entre si, isto é, os estilos definidos em cada uma delas não se misturam:

```
<div>
    .
    .
</div>
<div>
    .
    .
</div>
```

Já as tags aninhadas permitem que o estilo aplicado em uma seção faça parte também da seção interna a ela, criando o efeito cascata:

```
<div>
    .
    Seção 1
    .
<div>
    .
    Seção 2
    .
</div>
</div>
```

No esquema anterior, a Seção 2 está dentro da Seção 1. Isso significa que os estilos definidos na primeira seção serão visíveis para a segunda; pelo menos desde que ela não possua definição de estilos para o mesmo elemento, visto que nas folhas de estilo será considerado o estilo mais próximo ao código HTML. Porém, o contrário não acontece.

A tag `<div>` possui dois atributos que se relacionam com as CSS:

- `class`: serve para fazer referência a uma classe definida na seção de estilos, mais precisamente no *header* da página HTML;

- style: por meio desse atributo, é possível definir um estilo dentro da própria tag <div>.

A seguir, um exemplo genérico de como esses atributos são utilizados:

```
<html>
<header>
<style type="text/css">
div.classe { color:blue }
</style>
</header>
<body>
  <h1>Texto sem estilo.</h1>
  <div class="classe">
    <h1>Texto com fonte azul.</h1>
    <div style="text-align: center">
      <h1>Texto com fonte azul e alinhamento central.</h1>
    </div>
  </div>
</body>
</html>
```

Essa página exibe três cabeçalhos, sendo que o primeiro está fora da seção definida pela tag <div>, uma seção que utiliza a classe “classe”, que formata o texto com a cor azul. Por fim, um estilo definido na própria tag, que centraliza o *header* (seguindo a formatação da cor, já que a seção está dentro da anterior). Veja o resultado no browser:

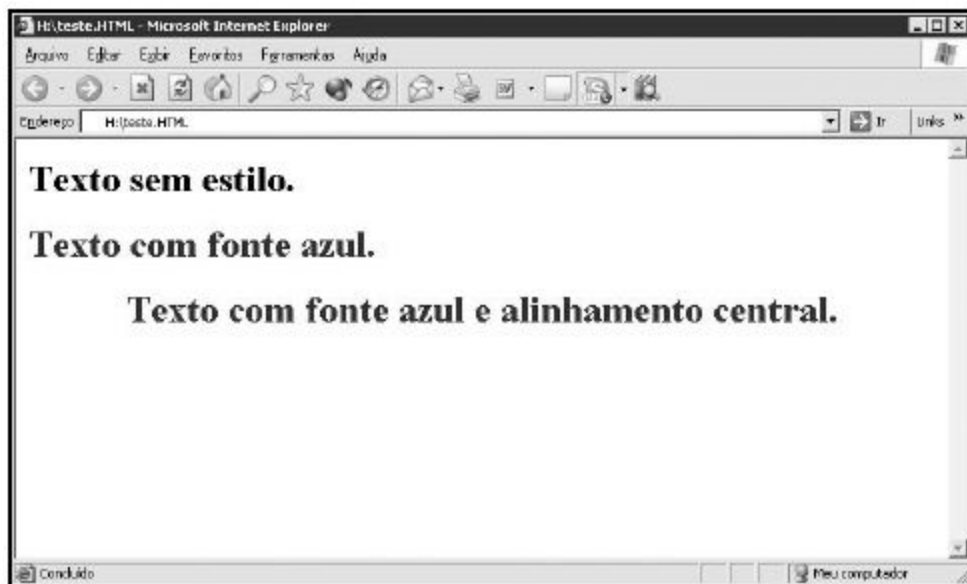


Figura 1.1.

Semelhante à <div>, a tag também possui a função de criar seções, porém, sem a capacidade de definir estilos. Em geral, a tag apenas “isola” um elemento, isto é, ela é utilizada em trechos pequenos de código (ao contrário da <div>, que pode delimitar grandes trechos de código em uma única seção).

Unidades de medida

As CSS permitem o uso de uma grande variedade de unidades de medida, que servem para definir a dimensão dos mais diversos elementos (largura de bordas de tabela, tamanho de fonte etc.). A seguir, serão listadas as unidades de medida válidas nas CSS, com exemplos de como utilizá-las:

Unidade de medida	Descrição
%	Indica valor percentual. Exemplo: 10%.
in	Indica valor em polegadas. Exemplo: 2in.
cm	Indica valor em centímetros. Exemplo: 5cm.
mm	Indica valor em milímetros. Exemplo: 15mm.
em	1em equivale ao tamanho atual de uma fonte. Isso significa que 2em é igual a duas vezes o tamanho da fonte atual. Exemplo: se um elemento é exibido com fonte 10pt, a medida 2em equivalerá à fonte 20pt. Esta unidade de medida é muito útil em CSS, pois se adapta automaticamente à configuração de fonte que cada browser utiliza.
ex	Um ex equivale à medida x-height de uma fonte (aproximadamente metade do tamanho de uma fonte). Exemplo: 2ex.

pt	Indica valor em pontos (1pt = 1/72in). Exemplo: 4pt.
pc	Indica valor em picas (1pc = 12pt). Exemplo: 2pc.
px	Indica valor em pixels (um ponto da tela do computador). Exemplo: 3px.

Tabela 1.1.

Observação: devido à variação na resolução dos monitores, definida por cada pessoa, torna-se difícil prever com qual configuração de vídeo as páginas serão acessadas. A utilização de unidades de medida, como polegadas e centímetros, principalmente para valores maiores, pode causar a visualização incorreta de um site no que diz respeito a imagens e elementos que ultrapassam o espaço disponível. Portanto, o uso de valores percentuais pode ser bastante interessante, pois isso calculará as medidas em função do espaço disponível, mantendo a proporção dos elementos.

Definição das cores

Cores são atributos essenciais para qualquer elemento de uma página HTML na medida em que conferem destaque e diferenciação. As CSS mostram-se muito úteis e funcionais também na formatação de cores, tanto para definição quanto para alteração, em função de eventos. Independentemente de onde são utilizadas, as cores dos elementos de uma página podem ser definidas de quatro formas.

Nome da cor

Existem aproximadamente 150 nomes de cores válidos que podem ser utilizados. É a maneira mais fácil de definir uma cor, porém, isso não garante que todos os browsers aceitarão tal definição. A tabela de nomes de cores será fornecida neste capítulo.

Valores RGB

No sistema RGB (*Red*, *Green* e *Blue*), as cores são obtidas a partir da mistura dessas três cores, e suas proporções podem variar de 0 a 255. Quando é atribuído valor zero às três cores, ou seja, na ausência desses três tons, obtemos a cor preta. Já utilizando o valor máximo dos três tons, obtemos o branco. Ainda neste capítulo, forneceremos uma tabela de cores básicas com seus respectivos valores RGB, além de uma tabela completa com várias combinações de cores que resultam em novas tonalidades.

Percentual RGB

Trata-se de uma variação do sistema RGB que utiliza valores percentuais para balancear o nível de cada uma das três cores. A cor preta, que no sistema RGB tradicional é definida por `rgb(0,0,0)`, com valores percentuais seria `rgb(0%,0%,0%)`. O valor 255 equivale a 100%.

Valor hexadecimal

Uma cor tem sua representação em escala hexadecimal mediante um código composto pelo caractere “#”, seguido por uma seqüência de seis

caracteres alfanuméricos (0-9 e A-F). Cada par de caracteres corresponde às cores vermelho, verde e azul. Se a representação da cor preta no sistema RGB é feita com o formato `rgb(0,0,0)`, no hexadecimal será `#000000`.

Quando um valor hexadecimal for composto por pares de valores idênticos, é possível “abreviar” esses valores, omitindo o par e indicando-o apenas uma vez. Veja os exemplos seguintes:

`#000000` = `#000`

`#FF00FF` = `#F0F`

`#778899` = `#789`

Sintaxe de definição de cores nas CSS

A tabela a seguir mostra, de forma sintática, as definições fornecidas até agora:

Definição da cor	Descrição
nome _ cor	Nome da cor. Exemplo: white.
rgb(n,n,n)	Valor RGB. Exemplo: vermelho é igual a <code>rgb(255,0,0)</code> .
rgb(n%, n%, n%)	Valor RGB em percentuais. Exemplo: verde é igual a <code>rgb(0%,100%,0%)</code> .
#rrggbb ou #rgb	Número hexadecimal. Exemplo: preto é igual a <code>#000000</code> ou <code>#000</code> .

Tabela 1.2.

Tabelas de cores

Neste item, serão fornecidos os valores das cores básicas nos sistemas RGB e hexadecimal, e, também, os nomes e valores hexadecimais para as mais diversas variações de tonalidades obtidas a partir das cores básicas.

Código RGB	Código hexadecimal	Cor
rgb (0,0,0)	#000000	Preto
rgb(255,0,0)	#FF0000	Vermelho
rgb(0,255,0)	#00FF00	Verde
rgb(0,0,255)	#0000FF	Azul
rgb(255,255,0)	#FFFF00	Amarelo
rgb(0,255,255)	#00FFFF	Azul claro
rgb(255,0,255)	#FF00FF	Lilás
rgb (192,192,192)	#C0C0C0	Cinza
rgb(255,255,255)	#FFFFFF	Branco

Tabela 1.3: Cores básicas.

Nome da cor (em inglês)	Código hexadecimal
AliceBlue	#F0F8FF
AntiqueWhite	#FAEBD7
Aqua	#00FFFF
Aquamarine	#7FFFD4
Azure	#F0FFFF
Beige	#F5F5DC
Bisque	#FFE4C4
Black	#000000
BlanchedAlmond	#FFEBCD
Blue	#0000FF
BlueViolet	#8A2BE2

Brown	#A52A2A
BurlyWood	#DEB887
CadetBlue	#5F9EA0
Chartreuse	#7FFF00
Chocolate	#D2691E
Coral	#FF7F50
CornflowerBlue	#6495ED
Cornsilk	#FFF8DC
Crimson	#DC143C
Cyan	#00FFFF
DarkBlue	#00008B
DarkCyan	#008B8B
DarkGoldenRod	#B8860B

DarkGray	#A9A9A9
DarkGrey	#A9A9A9
DarkGreen	#006400
DarkKhaki	#BDB76B
DarkMagenta	#8B008B
DarkOliveGreen	#556B2F
DarkOrange	#FF8C00
DarkOrchid	#9932CC
DarkRed	#8B0000
DarkSalmon	#E9967A
DarkSeaGreen	#8FBC8F
DarkSlateBlue	#483D8B
DarkSlateGray	#2F4F4F
DarkSlateGrey	#2F4F4F

DarkTurquoise	#00CED1
DarkViolet	#9400D3
DeepPink	#FF1493
DeepSkyBlue	#00BFFF
DimGray	#696969
DimGrey	#696969
DodgerBlue	#1E90FF
FireBrick	#B22222
FloralWhite	#FFFAF0
ForestGreen	#228B22
Fuchsia	#FF00FF
Gainsboro	#DCDCDC
GhostWhite	#F8F8FF

Gold	#FFD700
GoldenRod	#DAA520
Gray	#808080
Grey	#808080
Green	#008000
GreenYellow	#ADFF2F
HoneyDew	#F0FFF0
HotPink	#FF69B4
IndianRed	#CD5C5C
Indigo	#4B0082
Ivory	#FFFFFF
Khaki	#F0E68C
Lavender	#E6E6FA
LavenderBlush	#FFF0F5

LawnGreen	#7CFC00
LemonChiffon	#FFFACD
LightBlue	#ADD8E6
LightCoral	#F08080
LightCyan	#E0FFFF
LightGoldenRodYellow	#FAFAD2
LightGray	#D3D3D3
LightGrey	#D3D3D3
LightGreen	#90EE90
LightPink	#FFB6C1
LightSalmon	#FFA07A
LightSeaGreen	#20B2AA
LightSkyBlue	#87CEFA

LightSlateGray	#778899
LightSlateGrey	#778899
LightSteelBlue	#B0C4DE
LightYellow	#FFFFE0
Lime	#00FF00
LimeGreen	#32CD32
Linen	#FAF0E6
Magenta	#FF00FF
Maroon	#800000
MediumAquaMarine	#66CDAA
MediumBlue	#0000CD
MediumOrchid	#BA55D3
MediumPurple	#9370D8
MediumSeaGreen	#3CB371

MediumSlateBlue	#7B68EE
MediumSpringGreen	#00FA9A
MediumTurquoise	#48D1CC
MediumVioletRed	#C71585
MidnightBlue	#191970
MintCream	#F5FFFA
MistyRose	#FFE4E1
Moccasin	#FFE4B5
NavajoWhite	#FFDEAD
Navy	#000080
OldLace	#FDF5E6
Olive	#808000
OliveDrab	#6B8E23

Orange	#FFA500
OrangeRed	#FF4500
Orchid	#DA70D6
PaleGoldenRod	#EEE8AA
PaleGreen	#98FB98
PaleTurquoise	#AFEEEE
PaleVioletRed	#D87093
PapayaWhip	#FFEFD5
PeachPuff	#FFDAB9
Peru	#CD853F
Pink	#FFC0CB
Plum	#DDA0DD
PowderBlue	#B0E0E6
Purple	#800080

Red	#FF0000
RosyBrown	#BC8F8F
RoyalBlue	#4169E1
SaddleBrown	#8B4513
Salmon	#FA8072
SandyBrown	#F4A460
SeaGreen	#2E8B57
SeaShell	#FFF5EE
Sienna	#A0522D
Silver	#C0C0C0
SkyBlue	#87CEEB
SlateBlue	#6A5ACD
SlateGray	#708090

SlateGrey	#708090
Snow	#FFFAFA
SpringGreen	#00FF7F
SteelBlue	#4682B4
Tan	#D2B48C
Teal	#008080
Thistle	#D8BFD8
Tomato	#FF6347
Turquoise	#40E0D0
Violet	#EE82EE
Wheat	#F5DEB3
White	#FFFFFF
WhiteSmoke	#F5F5F5
Yellow	#FFFF00

YellowGreen	#9ACD32
-------------	---------

Tabela 1.4: Cores complexas.

Capítulo 2

Elementos e atributos

Um elemento é um objeto inserido em uma página HTML, e pode ser um texto, uma imagem, um link etc. Cada elemento possui uma série de atributos que definem características como aparência (cor, tamanho e estilo, por exemplo), posicionamento na página e outros aspectos. As CSS extraem essas definições de formato do código HTML e as colocam em uma seção própria para estilização, deixando apenas as informações/estruturas que serão formatadas por elas.

Existem duas formas de definir vários atributos para um único elemento. Veja a seguir os exemplos de folhas de estilo para um parágrafo:

```
p {  
    color: blue;  
    font-style: italic;  
    font-size: 20px;  
    font-family: verdana  
}
```

Ou:

```
p { color: blue;  
    font: italic 20px verdana  
}
```

Note que para formatar um parágrafo existem atributos relacionados à fonte, como estilo, a própria fonte e seu tamanho. Todas as propriedades pertencem a um mesmo “atributo-pai”: font. Quando isso acontece, é possível agrupar todos os atributos de uma só vez, listando cada

formatação, lado a lado, sem vírgulas. Esse agrupamento de atributos só pode ser feito para o caso de pertencerem à mesma “família”. Com isso, é importante lembrar que o atributo de cor, por exemplo, não pertence à “família” font e, portanto, não pode ser incluído na definição da fonte.

Neste capítulo, demonstraremos, por meio de definições e exemplos, os principais atributos para a criação de folhas de estilo e em quais objetos eles podem ser aplicados.

Background

Os atributos de background das folhas de estilo permitem que sejam alteradas todas as propriedades relacionadas à cor de fundo de um elemento, definir uma imagem como plano de fundo, configurar se essa imagem se repetirá vertical e horizontalmente e qual será o seu posicionamento na página HTML:

Atributo	Descrição	Valor
background	Define todos os atributos do background (fundo) da página (cor, imagem) em apenas uma declaração.	background-attachment background-color background-image
background-attachment	Define se a imagem de fundo permanece fixa ou é paginada com a barra de rolagem.	scroll fixed
background-color	Define a cor de fundo de um elemento.	Nome ou código hexadecimal da cor.
background-image	Define uma imagem como plano de fundo.	Nome do arquivo de imagem.
background-position	Define a posição inicial de uma imagem no plano de fundo.	top left top center

		top right center left center center center right bottom left bottom center bottom right x% y% (x e y = 0 a 100) xpos ypos
background-repeat	Define se uma imagem será repetida no plano de fundo, e como isso será feito.	repeat repeat-x repeat-y no-repeat

Tabela 2.1.

Exemplo de formatação de background

O exemplo a seguir define o estilo do fundo de uma página cuja cor será preta, e um cabeçalho com cor de fundo amarela:

```
<style type="text/css">
body { background-color: #000000 }
h1 { background-color: yellow }
</style>
```

Agora, em código HTML:

```
<body>
<h1>Cabeçalho Com Fundo Amarelo</h1>
</body>
```

Este será o resultado:



Figura 2.1.

Texto

Os atributos relacionados aos elementos de texto referem-se tanto à formatação quanto ao posicionamento e à direção:

Atributo	Descrição	Valor
color	Define a cor de um texto.	Cor desejada.
direction	Define a direção do texto.	ltr (esquerda para direita) rtl (direita para esquerda)
line-height	Define a distância entre as linhas.	Normal. Número desejado. Distância desejada. Porcentagem.
letter-spacing	Aumenta ou diminui o espaço entre os caracteres.	Normal Length
text-align	Define o alinhamento do texto.	left right center justify
text-decoration	Adiciona ornamentação ao texto.	none underline overline

		line-through blink
text-indent	Endenta a primeira linha de um texto em um elemento.	Espaçamento desejado. Porcentagem.
text-transform	Controla o formato das letras de um elemento (maiúscula, minúscula, etc.).	none capitalize uppercase lowercase
white-space	Define como os espaços em branco dentro de um elemento são tratados.	normal pre nowrap
word-spacing	Aumenta ou diminui o espaço entre palavras.	Normal. Espaço desejado.

Tabela 2.2.

Exemplo de formatação de texto

Este exemplo de definição de estilo formatará a cor da fonte e o espaçamento entre as letras:

```
<style type="text/css">
p { font-family: verdana;
    letter-spacing: 0.4cm;
    color: rgb(0,0,255) }
</style>
```

A seguir, o código HTML e o layout da página:

```
<body>  
<p>Espa&ccedil;o entre letras.</p>  
</body>
```

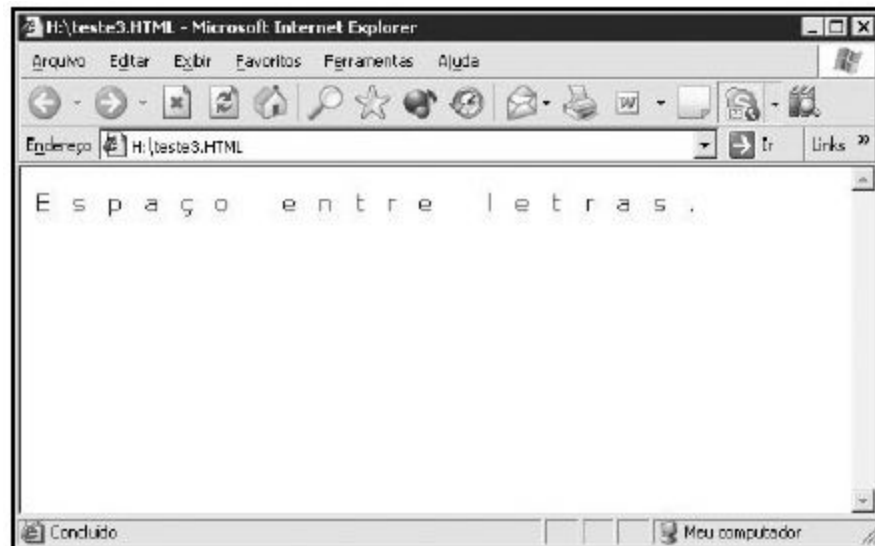


Figura 2.2.

Fonte

Folhas de estilo referentes à fonte dos textos, em geral, podem ser definidas para toda a página ou para elementos separadamente, porém, os atributos, listados a seguir, são os mesmos:

Atributo	Descrição	Valor
font	Define todos os atributos de fonte (estilo, tamanho etc.) em apenas uma declaração.	font-family font-size font-stretch font-style font-variant font-weight
font-family	Define uma lista de fontes preferenciais para a página, podendo ser o nome da fonte ou o nome genérico da família da fonte.	Nome da fonte
font-size	Define o tamanho da fonte.	xx-small x-small small medium large x-large xx-large smaller larger Tamanho desejado. Porcentagem.

font-stretch	Condensa ou expande a <i>font-family</i> atual.	normal wider narrower ultra-condensed extra-condensed condensed semi-condensed semi-expanded expanded extra-expanded ultra-expanded
font-style	Define o estilo da fonte.	normal italic oblique
font-variant	Exibe texto com fonte normal ou <i>smallcaps</i> (letras minúsculas convertidas em maiúsculas).	normal small-caps
font-weight	Define a espessura de uma fonte.	normal bold bolder lighter 100 200 300 400 500

		600
		700
		800
		900

Tabela 2.3.

Exemplo de formatação de fonte

O exemplo seguinte define um padrão de fonte para toda a página e, portanto, o estilo é definido para o elemento body. Caso algum elemento tenha alguma característica de fonte diferente, um estilo deve ser criado especialmente para ele:

```
<style type="text/css">
body { font-family: verdana }
h2 { font-style: italic }
p { font-style: normal }
</style>
```

Código HTML e o layout da página:

```
<body>
<h2>Cabeçalho com fonte Verdana itálico</h2>
<p>Parágrafo com fonte Verdana normal.</p>
</body>
```




Figura 2.3.

Bordas

Os atributos listados a seguir exercem a função de definir e formatar as bordas ao redor de um elemento. Essa formatação pode ser feita de uma só vez por meio do “atributo-pai”, ou separadamente, dependendo da necessidade da estilização:

Atributo	Descrição	Valor
border	Define todos os atributos das bordas (largura, estilo, cor) em apenas uma declaração.	border-width border-style border-color
border-bottom	Define todos os atributos da borda inferior em apenas uma declaração.	border-bottom-width border-style border-color
border-bottom-color	Define a cor da borda inferior.	border-color
border-bottom-style	Define o estilo da borda inferior.	border-width border-style border-color
border-bottom-width	Define a largura da borda inferior.	border-bottom-width border-style border-color
border-	Define a cor da borda.	border-color

color		
border-left	Define todos os atributos da borda esquerda em apenas uma declaração.	border-style thin medium thick Largura desejada. Nome ou código hexadecimal da cor. border-left-width border-style border-color
border-left-color	Define a cor da borda esquerda.	border-color
border-left-style	Define o estilo da borda esquerda.	border-style
border-left-width	Define a largura da borda esquerda.	thin medium thick Largura desejada.
border-right	Define todos os atributos da borda direita em apenas uma declaração.	border-right-width border-style border-color

border-right-color	Define a cor da borda direita.	border-color
border-right-style	Define o estilo da borda direita.	border-style
border-right-width	Define a largura da borda direita.	thin medium thick Largura desejada.
border-spacing	Define a distância que separa as bordas das células	Distância desejada.
border-style	Define o estilo das quatro bordas em apenas uma declaração.	none hidden dotted dashed solid double groove ridge inset outset
border-top	Define todos os atributos da borda superior em apenas uma declaração.	border-top-width border-style border-color

border-top-color	Define a cor da borda superior.	border-color
border-top-style	Define o estilo da borda superior.	border-style
border-top-width	Define a largura da borda superior.	thin medium thick Largura desejada.
border-width	Define a largura das quatro bordas em apenas uma declaração.	thin medium thick Largura desejada.

Tabela 2.4.

Exemplo de formatação de bordas em parágrafos

O exemplo a seguir demonstrará como definir os estilos das bordas de um parágrafo:

```
<style type="text/css">
p.estilo1 { border-style: double;
            border-width: thick }
p.estilo2 { border-style: double solid }
</style>
```

Note que no estilo2, o estilo da borda é definido duas vezes (double e solid). No caso de definir o estilo das quatro bordas de uma só vez, a ordem a ser seguida será: superior, direita, inferior e esquerda. Caso sejam definidos dois estilos apenas (como no exemplo), será assumido o estilo double para superior e inferior, e solid para direita e esquerda. Veja o código HTML:

```
<body>  
<p class="estilo1">Estilo de Borda 1.</p>  
<p class="estilo2">Estilo de Borda 2.</p>  
</body>
```

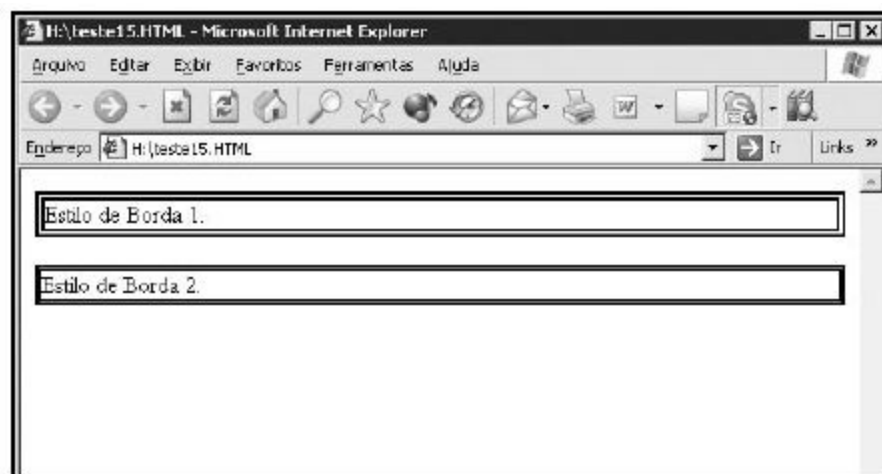


Figura 2.4.

Margens

A margem de um elemento caracteriza-se como o espaço que o contorna. Por meio dos atributos definidos a seguir, é possível determinar a largura desse espaço igualmente para todos os quatro lados, ou fazê-lo um a um para o caso de margens com medidas diferentes:

Atributos	Descrição	Valor
margin	Define todos os atributos da margem em apenas uma declaração.	margin-top margin-right margin-bottom margin-left
margin-bottom	Define a margem inferior de um elemento.	Auto. Largura desejada. Porcentagem.
margin-left	Define a margem esquerda de um elemento.	Auto. Largura desejada. Porcentagem.
margin-right	Define a margem direita de um elemento.	Auto. Largura desejada. Porcentagem.
margin-top	Define a margem superior de um elemento.	Auto. Largura

		desejada. Porcentagem.
--	--	---------------------------

Tabela 2.5.

Os valores podem ser definidos em centímetros ou em percentuais, sendo que, nesse último caso, como já dito anteriormente, o valor será relativo ao espaço disponível na página. Trata-se de uma unidade de medida bastante conveniente, pois, com esta proporção, em relação à página, evita-se que a largura “estoure” o espaço disponível, dependendo da resolução do monitor de quem acessa (algo que não se pode prever).

Exemplo de formatação de margens

O exemplo seguinte demonstra três parágrafos, sendo o primeiro sem estilização de margem, o segundo com estilização de todas as margens feitas de uma só vez, e o terceiro somente com estilização da margem esquerda:

```
<style type="text/css">
p.todas { margin: 20% 2cm 20% 2cm }
p.esq { margin-left: 10% }
</style>
```

Veja o código HTML e como a página será exibida:

```
<body>
<p>Parágrafo sem margens definidas.</p>
<p class="todas">Parágrafo com margens definidas.</p>
<p class="esq">Parágrafo com margem esquerda definida.</p>
</body>
```

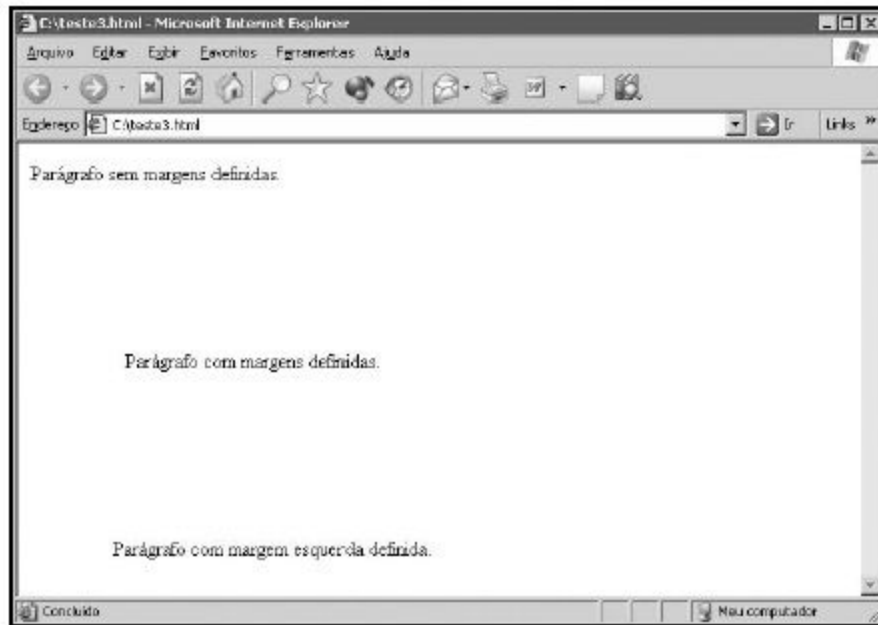



Figura 2.5.

Padding

Padding pode ser definido como o espaço que existe entre um elemento e as bordas da célula que o contém. Assim, uma folha de estilo de padding definirá esse espaçamento. Isso pode ser feito de uma só vez ou separadamente para cada lado da borda:

Atributo	Descrição	Valor
padding	Define todos os atributos de padding em apenas uma declaração.	padding-top padding-right padding-bottom padding-left
padding-bottom	Define o padding inferior de um elemento.	Largura desejada. Porcentagem.
padding-left	Define o padding esquerdo de um elemento.	Largura desejada. Porcentagem.
padding-right	Define o padding direito de um elemento.	Largura desejada. Porcentagem.
padding-top	Define o padding superior de um elemento.	Largura desejada. Porcentagem.

Tabela 2.6.

Exemplo de estilização de paddings

O exemplo a seguir demonstra como a estilização do padding nas células pode ser feita:

```
<style type="text/css">
td.celula1 { padding: 1cm }
td.celula2 { padding-top: 10%;
              Padding-left: 2cm; }
</style>
```

Na segunda linha, os quatro lados são definidos. Depois, ocorre a definição do topo em porcentagem. Por último, define-se o lado esquerdo em centímetros.

A seguir, o código HTML:

```
<body>
<table border="1">
<tr>
<td class="celula1">
Espa&ccedil;os definidos de uma s&ocute; vez.
</td>
</tr>
</table>
<br>
<table border="1">
<tr>
<td class="celula2">
Espa&ccedil;o superior e esquerdo definidos.
</td>
</tr>
</table>
</body>
```

O código anterior resultará na seguinte página:

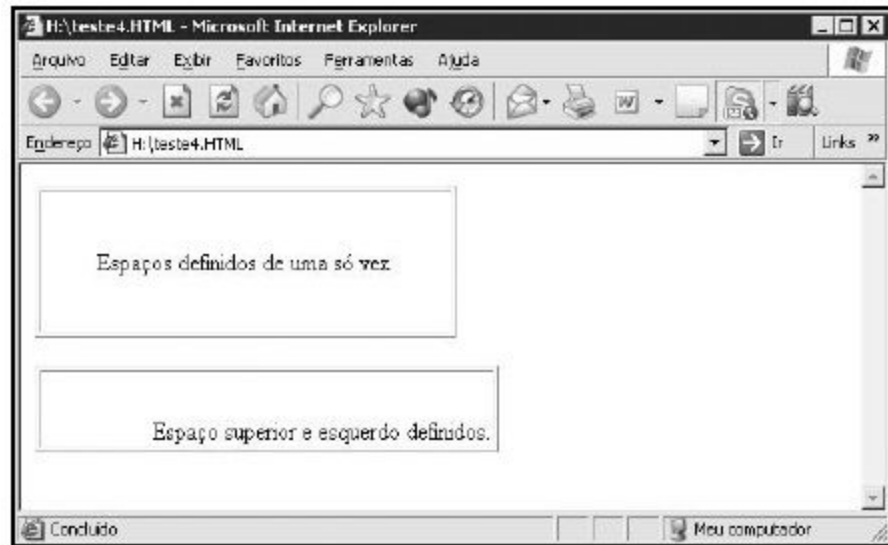


Figura 2.6.

Lista

Os atributos relacionados à formatação de listas nas folhas de estilo referem-se à definição do tipo de marcador utilizado nos itens, além de permitirem também que se defina uma imagem como marcador:

Atributo	Descrição	Valor
list-style	Define todos os atributos das listas (tipo, posição, imagem) em apenas uma declaração.	list-style-type list-style-position list-style-image
list-style-image	Define uma imagem como marcador de lista.	Nome do arquivo de imagem.
list-style-position	Define onde o marcador do item da lista será posicionado em relação à lista.	inside outside
list-style-type	Define o tipo de marcador para os itens da lista.	none disc circle square decimal decimal-leading-zero lower-roman

	upper-roman lower-alpha upper-alpha lower-greek lower-latin upper-latin hebrew armenian georgian cjk-ideographic hiragana katakana hiragana-iroha katakana-iroha
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Tabela 2.7.

É importante lembrar que, no caso de utilizarmos uma imagem (em geral, no formato .gif) como marcador de item de lista, o arquivo da imagem deve ter seu diretório informado na folha de estilo que o referencia, caso esteja em local diferente, independente do tipo de folha utilizada. Ou seja, a imagem utilizada como marcador sempre deverá ter o seu diretório devidamente informado no atributo `list-style-image`, dentro de `url()`. Veja dois exemplos:

- Se a página HTML contém as folhas de estilo que chamam o arquivo do marcador em sua própria seção de *header* e está no diretório *C:*, por exemplo, caso a imagem esteja no mesmo diretório, basta indicar o nome do arquivo. Se as imagens estiverem, por exemplo, em *C:/imagens*, o caminho completo deverá ser indicado:

```
<style type="text/css">
list-style-image: url("marcador.gif")
</style>
```

Ou:

```
<style type="text/css">
list-style-image: url("../imagens/marcador.gif")
</style>
```

- Se a página HTML, localizada em C:, chama um arquivo CSS em C:/CSS, vale a regra já descrita: se o arquivo da imagem do marcador não estiver no mesmo local da folha de estilo, o diretório deve ser indicado:

```
<style type="text/css">
    list-style-image: url("marcador.gif")
</style>
```

Ou:

```
<style type="text/css">
    list-style-image: url("../imagens/marcador.gif")
</style>
```

E, na página HTML, considerando que o arquivo CSS está em C:/CSS:

```
<head>
<link rel="stylesheet" type="text/css" href="../css/estilo.
css" />
</head>
```

Exemplo de definição de imagem como marcador de lista

O exemplo seguinte cria uma lista utilizando uma imagem como marcador de item:

```
<style type="text/css">
ul { list-style: outside url("marcador.gif")
}
</style>
```

E agora o código HTML, com a visualização da página logo em seguida:

```
<body>
<ul>
<li>Lápis</li>
<li>Caneta</li>
<li>Borracha</li>
<li>Régua</li>
</ul>
</body>
```

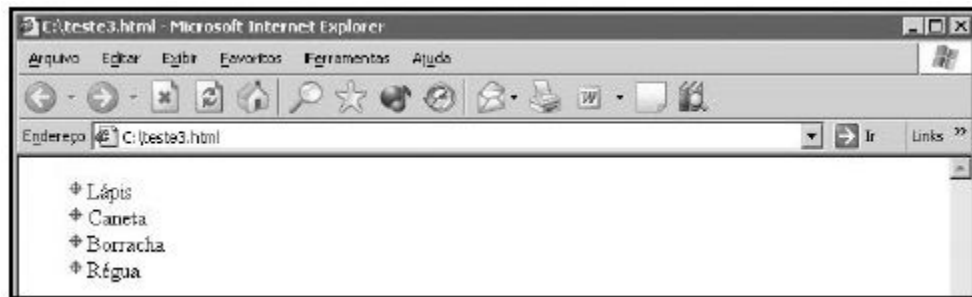


Figura 2.7.

Tabela

As tabelas podem ter diversos atributos formatados com as folhas de estilo, como estilo de borda de célula, espaçamento entre células, exibição ou não de células vazias etc., que permitem a criação de diversos estilos de tabelas, para os mais variados tipos de tabulação de informações:

Atributo	Descrição	Valor
border-collapse	Define se as bordas de uma tabela são contraídas em bordas simples ou separadas, como no padrão HTML.	collapse separate
border-spacing	Define a distância que separa as bordas das células (válido somente para o modelo de bordas separadas).	Distância desejada.
caption-side	Define a posição do título da tabela.	top bottom left right
empty-cells	Define se as células vazias serão exibidas ou não em uma tabela (válido somente para o modelo de bordas separadas).	show hide
table-layout	Define o algoritmo utilizado para exibir as células, linhas e colunas de uma tabela.	auto fixed

Tabela 2.8.

Exemplo de formatação de tabela

O exemplo a seguir mostra uma tabela simples, demonstrando a relação aluno/nota/faltas. Ela terá suas bordas contraídas (apenas uma linha, diferente do padrão de borda dupla do HTML) e o alinhamento das notas e faltas, valores numéricos, ficará à direita da célula. Para isso, foi criada uma classe (esse assunto será mais bem detalhado no próximo capítulo) apenas para armazenar essas informações de tipo numérico, já que não queremos esse tipo de alinhamento para o nome do aluno:

```
<style type="text/css">
body { font-family: arial }
table { border-collapse: collapse;
        border-width: 1px }
table td { border-top-width: 2px }
table td.numerico { text-align: right }
</style>
```

Agora, o código HTML:

```
<body>
<table border="1">
<caption>Notas e Faltas</caption>
<th>Aluno(a)</th>
<th>Nota</th>
<th>Falta</th>
<tr>
<td>Maria</td>
<td class="numerico">8.5</td>
<td class="numerico">2</td>
</tr>
<tr>
<td>Jos&eacute;</td>
<td class="numerico">8.0</td>
<td class="numerico">1</td>
</tr>
</table>
</body>
```



Figura 2.8.

Exemplo de tabela com linhas de cores alternadas

Agora, criaremos folhas de estilo para formatar uma tabela que possuirá linhas com cores alternadas. Para isso, serão criados dois estilos para o elemento `<tr>`, a linha da tabela, e um deles será feito por meio de uma classe. Essa classe será referenciada em linhas alternadas da tabela. A seguir, acompanhe a sintaxe da folha de estilo e do código HTML:

```
<style type="text/css">
table.tabela tr td {background: #B0E0E6;}
table.tabela tr.alterna td {background: #F0F8FF;}
</style>
```

Agora, o código HTML:

```
<body>
<table border="1px" cellpadding="5px" class="tabela">
<caption>T&iacute;tulo</caption>
<tr>
<td>Lin1/Col1 </td>
<td>Lin1/Col2</td>
<td>Lin1/Col3</td>
</tr>
<tr class="alterna">
```

```

<td>Lin2/Col1 </td> <td>Lin2/Col2</td>
<td>Lin2/Col3</td>
</tr>
<tr>
<td>Lin3/Col1 </td>
<td>Lin3/Col2</td>
<td>Lin3/Col3</td>
</tr>
<tr class=»alterna»>
<td>Lin4/Col1 </td>
<td>Lin4/Col2</td>
<td>Lin4/Col3</td>
</tr>
<tr>
<td>Lin5/Col1 </td>
<td>Lin5/Col2</td>
<td>Lin5/Col3</td>
</tr>
</table>
</body>

```

Agora, o resultado final:

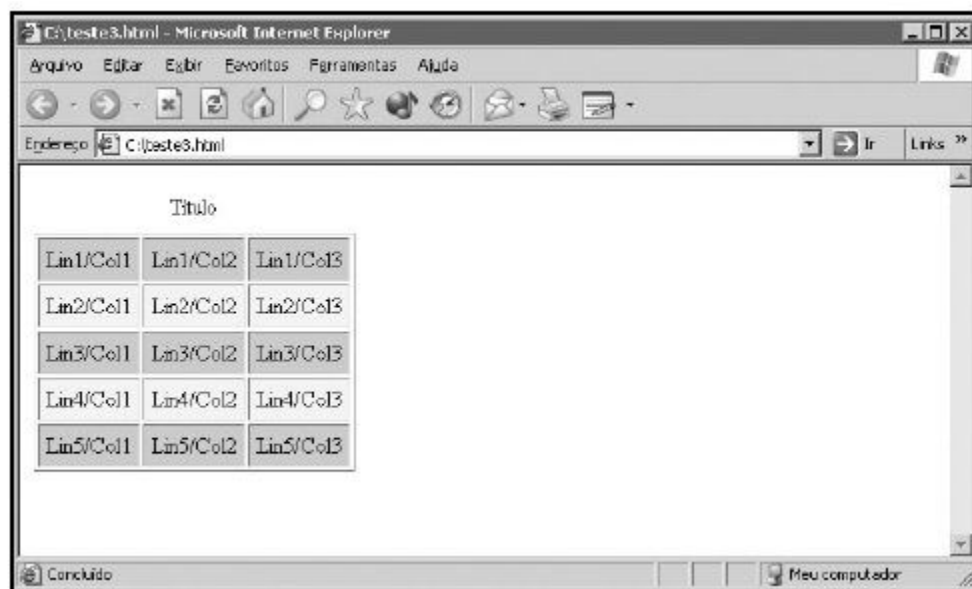


Figura 2.9.

Imagens

Imagens não são consideradas elementos no HTML, mas são atributos de elementos. Isto é, podem servir de papel de parede, marcador de item de lista, link etc. Do mesmo modo, nas CSS, elas não são estilizadas, mas existe uma infinidade de recursos que podem ser aplicados em imagens, e que incrementam o visual da página HTML.

Exemplo de mapa de imagem com CSS

Um mapa de imagem consiste em definir pontos (ou coordenadas, já que os pontos são compostos pela combinação de linhas e colunas, representando referências a partir do topo e da esquerda do início da imagem) em uma imagem e atribuir a eles eventos ou links.

Para compreender como é feito um mapa de imagens, é importante definir alguns conceitos sobre como as imagens são interpretadas e como suas medidas são utilizadas para esse mapeamento. Observe a figura e em seguida as definições de cada medida e coordenada:

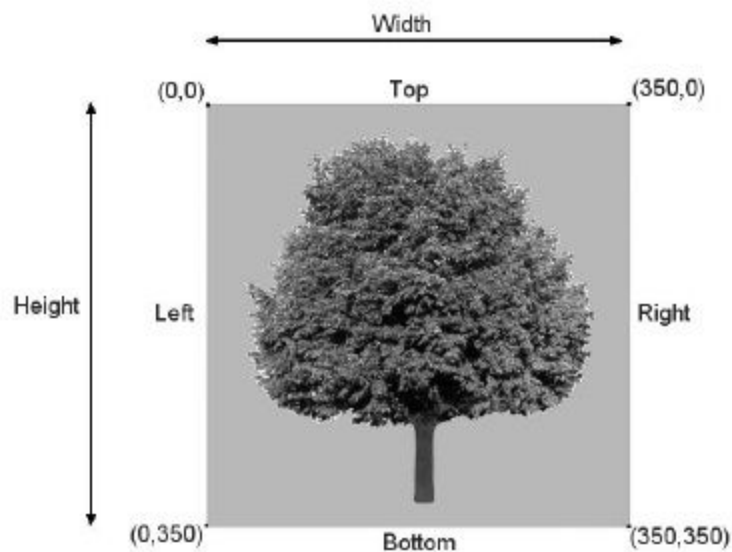


Figura 2.10.

O tamanho de uma imagem, devido a sua composição, é mais facilmente calculado em pixels. No exemplo anterior, o desenho da árvore possui tamanho de 350×350 pixels. A seguir, detalharemos melhor cada uma das medidas:

- *Top*: é o topo (ou parte superior) da imagem. No conceito de coordenadas “coluna, linha”, seria a linha=0;
- *Left*: extremidade esquerda da imagem. No conceito de coordenadas “coluna, linha”, seria a coluna=0;
- *Right*: extremidade direita da imagem. No conceito de coordenadas “coluna, linha”, e tomando como exemplo a imagem anterior, que possui tamanho 350×350, seria a coluna=350;
- *Bottom*: o “fundo” (ou parte inferior) da imagem. No conceito de coordenadas “coluna, linha”, e tomando como exemplo a imagem anterior, que possui tamanho 350×350, seria a linha=350;
- *Height*: altura total da imagem, que no caso possui 350 pixels;
- *Width*: largura total da imagem, que no caso possui 350 pixels.

Partindo desses princípios, relacionaremos cada um destes detalhes com alguns atributos que são utilizados em conjunto com imagens nas folhas de estilo:

- *height*: define a altura da área em que a imagem será exibida (se a altura for menor que a altura real da imagem, esta será exibida “cortada”);
- *width*: define a largura da área em que a imagem será exibida (se a largura for menor que a largura real da imagem, esta será exibida “cortada”);
- *top*: posição inicial horizontal em que a imagem será posicionada;
- *left*: posição inicial vertical em que a imagem será posicionada;
- *position*: utilizando a posição relativa (*position: relative*), uma imagem será posicionada em um local definido pelos atributos *top* e *left* em relação à página. No caso da posição ser absoluta (*position: absolute*), se algum outro elemento foi posicionado anteriormente na página, a imagem considerará a posição desse outro elemento (*top* e *left*) como as coordenadas iniciais.

Neste capítulo foram demonstrados apenas os aspectos teóricos dos mapas de imagens. Um exemplo mais detalhado será desenvolvido no capítulo sobre menus, no qual será implementado um menu com mapa de imagens.

Exemplo de image replacement

Atualmente, existem diversas páginas HTML que utilizam imagens no lugar de títulos e isso é mais que justificável, pois, em geral, os títulos de páginas vêm acompanhados por logotipos e os recursos gráficos de uma imagem são muito mais sofisticados que os dos textos. A substituição de imagem por texto consiste em um recurso bastante útil e provê consistência a um site, visto que prevê as mais diversas configurações de browser por onde uma página pode ser acessada.

É praticamente impossível prever (diferente de identificar) a configuração de todos os browsers que acessarão sua página e quais recursos estão habilitados ou não. As possibilidades vão de desabilitação de imagens ou de folhas de estilo, até a utilização de um recurso de leitura de tela (programas específicos para deficientes visuais, tradutores de conteúdo etc.). Prevenir-se contra a ocorrência de alguma exceção que pode alterar negativamente o layout de sua página também faz parte de uma implementação bem feita.

Pensando nisso, existe um modo de definir uma imagem como título de uma página e, caso ocorra algum erro e a imagem não puder ser exibida, o seu espaço será ocupado por um texto correspondente. No exemplo seguinte, temos um título definido pela tag `<h1>` que será substituído por uma imagem. Por meio da estilização da tag ``, que agrupa elementos em uma página, define-se que o título não será mostrado e a imagem ocupará seu lugar. Porém, se o browser estiver configurado para desconsiderar folhas de estilo ou imagens, a estilização será ignorada e o texto será exibido. Veja a definição do estilo:

```
h1#titulo {
    background-image: url(titulo.gif);
} h1#titulo span {
    display: none;
}
```

Agora, em HTML:

```
<body>
<h1 id="topo">
<span>Titulo em Formato Texto</span>
</h1>
</body>
```


Capítulo 3

Classes

Além da formatação dos elementos, as CSS permitem a definição de classes, que são estruturas de estilização que recebem um nome próprio. Com essa denominação, é possível criar estilos diferentes para um mesmo tipo de elemento, ou formatar um determinado atributo para diversos elementos que o possuam entre suas propriedades. Observe a sintaxe a seguir:

```
elemento.nome _ classe { atributo: valor }
```

Agora, sem um elemento definido:

```
.nome _ classe { atributo: valor }
```

No código HTML, o nome da classe deve ser declarado:

```
<elemento class="nome _ classe"> ... </elemento>
```

Exemplo de criação e uso de classes

Suponha que seu site possua dois tipos de parágrafos, para usos distintos. Se for definida apenas uma folha de estilo para a tag <p>, não haverá como diferenciá-las e a formatação terá de ser feita separadamente em cada elemento. Considerando esse caso, vamos definir que existem parágrafos alinhados à esquerda e parágrafos centralizados, sendo que estes também possuirão características definidas pela estilização do parágrafo de modo

genérico. As definições e seu respectivo uso seriam para classes em cada estilo:

```
<style type="text/css">
p { color: blue;
    font: italic 20px verdana }
p.esquerda { text-align: left }
p.centro { text-align: center }
</style>
```

Agora, o código HTML e o resultado na página:

```
<body>
<p class="centro">
Parágrafo alinhado no centro.
</p>
<p class="esquerda">
Parágrafo alinhado à esquerda.
</p>
</body>
```

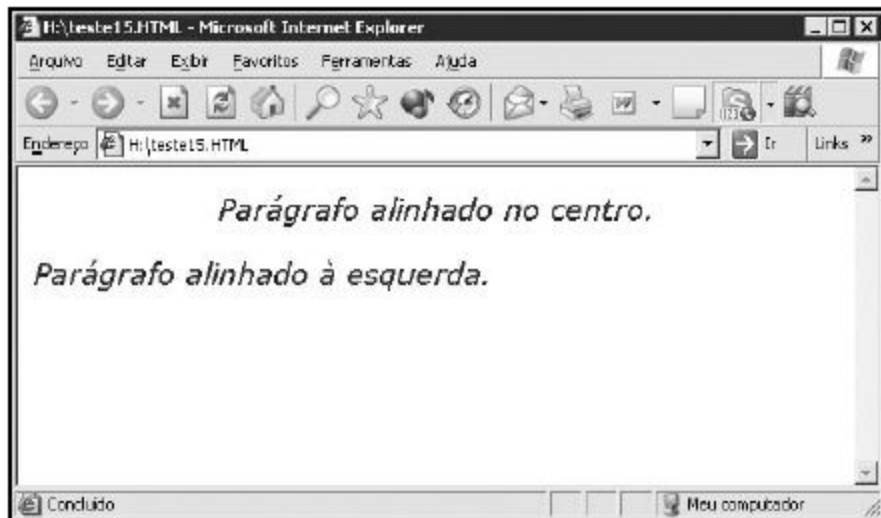


Figura 3.1.

A vantagem de não associar a classe a um elemento específico está na possibilidade de atribuí-la a mais de um tipo de elemento. Se as classes “centro” e “esquerda” fossem criadas apenas como:

```
.centro { color: blue;
          font: italic 20px verdana;
text-align: center }
```

Isto é, se fossem criadas sem o elemento, elas poderiam ser utilizadas, por exemplo, para formatar um elemento cabeçalho (<h1> a <h6>), pois eles possuem os mesmos atributos do elemento <p>. Assim, seria possível implementar:

```
<body>
<p class="centro">
Parágrafo utilizando a classe gcentro h.
</p>
<h1 class="esquerda">
Cabeçalho utilizando a classe gcentro h.
</h1>
</body>
```

Capítulo 4

CSS avançado

Neste capítulo, serão abordados assuntos de caráter avançado sobre CSS, incluindo classificação, enquadramento e posicionamento de elementos em uma página, utilização de pseudoclasses e pseudo-elementos, alteração do nível de transparência em uma imagem (e onde é possível aplicar tal efeito), criação de botões, menus e formulários e, por fim, configuração do layout de uma página de acordo com o dispositivo de saída (tela, impressora etc.). Todos esses assuntos serão acompanhados de exemplos que facilitam a compreensão de como se dá o funcionamento das CSS para cada um dos casos.

Classificação e enquadramento

As folhas de estilo possuem atributos de classificação e enquadramento que podem ser aplicados aos mais diversos elementos. A seguir, temos uma referência rápida de cada um deles. Em seguida, os elementos serão detalhados e exemplificados:

Atributo	Descrição	Valor
clear	Define qual(is) lado(s) de um elemento não permite(m) a presença de outros elementos flutuantes.	left right both none
cursor	Define o tipo de cursor que será exibido.	url auto crosshair default pointer move e- resize ne-resize nw- resize n-resize se-resize sw- resize s-resize w-resize text wait help

display	Define se/como um elemento será exibido.	none inline block list-item run-in compact marker table inline- table table- row- group table- header- group table- footer- group table- row table- column- group table- column table- cell table- caption
float	Define onde um elemento será exibido em relação a outro elemento.	left right none

visibility	Define se um elemento será visível ou não em uma página.	visible hidden collapse (para tabelas)
------------	----------------------------------------------------------	-------------------------------------------------

Tabela 4.1.

Alterando o tipo do cursor

O atributo cursor permite que alteremos o tipo de cursor a ser exibido em uma página. Trata-se de um recurso estético que auxilia na diferenciação do tipo de elemento que o mouse indica.

O exemplo a seguir mostra como alterar o tipo do cursor. Os estilos foram aplicados diretamente no elemento, no body da página, porque cada um terá um estilo diferente, sem repetição:

```
<html>
<head>
<style type="text/css">
body { font-family: sans-serif }
</style>
</head>
<body>
<p>Posicione o mouse sobre cada palavra e observe os tipos
de cursor:</p>
<span style="cursor:auto">
- Auto</span><br />
<span style="cursor:crosshair">
- Crosshair</span><br />
<span style="cursor:default">
- Default</span><br />
<span style="cursor:pointer">
- Pointer</span><br />
<span style="cursor:move">
- Move</span><br />
<span style="cursor:e-resize">
- e-resize</span><br />
<span style="cursor:ne-resize">
```

```

- ne-resize</span><br />
<span style="cursor:nw-resize">
- nw-resize</span><br />
<span style="cursor:n-resize">
- n-resize</span><br />
<span style="cursor:se-resize">
- se-resize</span><br />
<span style="cursor:sw-resize">
- sw-resize</span><br />
<span style="cursor:s-resize">
- s-resize</span><br />
<span style="cursor:w-resize">
- w-resize</span><br />
<span style="cursor:text">
- Text</span><br />
<span style="cursor:wait">
- Wait</span><br />
<span style="cursor:help">
- Help</span>
</body>
</html>

```

Caso o leitor queira definir páginas de estilo para esses tipos de cursor no início da página, bastaria criar classes para cada tipo e depois definir o elemento com o atributo class. Veja o exemplo:

```

<html>
<head>
<style type="text/css">
body { font-family: sans-serif }
.cursor _ auto { cursor: auto }
.cursor _ crosshair { cursor: crosshair }
.cursor _ default { cursor: default }
.cursor _ pointer { cursor: pointer }
.cursor _ move { cursor: move }
</style>
</head>
<body>
<p>Posicione o mouse sobre cada palavra e observe os tipos
de cursor:</p>
<span class="cursor _ auto">
- Auto</span><br />
<span class="cursor _ crosshair">
- Crosshair</span><br />
<span class="cursor _ default">
- Default</span><br />
<span class="cursor _ pointer">

```



```
- Pointer</span><br />
<span class="cursor _ move">
- Move</span><br />
</body>
</html>
```

Tornando um elemento invisível

O atributo `visibility` permite ocultar ou não um elemento em uma página. Veja um exemplo com três parágrafos em que um deles é definido como invisível:

```
<style type="text/css">
body { font-family: sans-serif }
p.invisivel { visibility: hidden }
</style>
```

Agora, o código HTML e a visualização da página:

```
<body>
<p>Parágrafo Visível 1</p>
<p class="invisivel">Parágrafo Invisível</p>
<p>Parágrafo Visível 2</p>
</body>
```

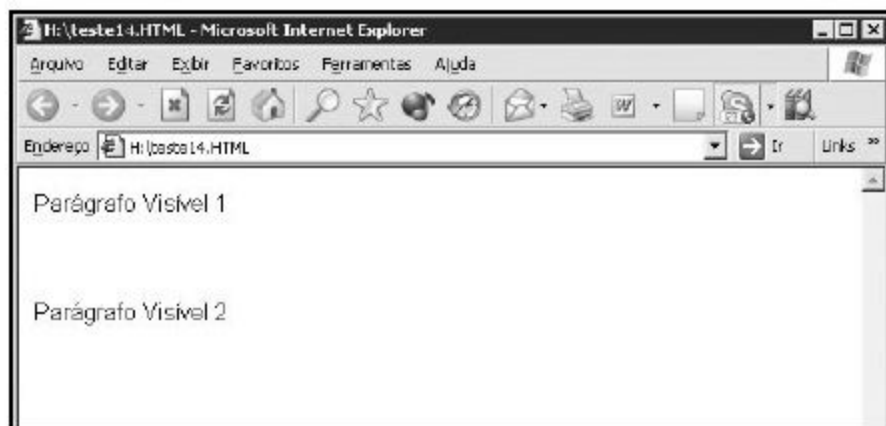


Figura 4.1.

Note que o elemento invisível, embora não apareça, ocupa espaço na página. Então, se ele nunca for utilizado/exibido, não há necessidade de

declará-lo invisível.

Alterando o modo de exibição

As CSS possuem um atributo chamado display, que altera o formato de um elemento. Ele pode ser exibido como parte de uma lista, de uma tabela ou fazer parte de um bloco de elementos, tendo a característica de poder ser alterado dinamicamente por meio da linguagem Javascript.

A seguir, o leitor acompanhará a estilização de dois textos que, por estarem delimitados pela tag (com o estilo display: list-item), serão divididos por uma quebra de linha, pois se comportarão como itens de uma lista:

```
<html>
<head>
<style type="text/css">
body { font-family: sans-serif }
span { display: list-item }
</style>
</head>
<body>
<span>Este texto será posicionado e exibido</span>
<span>como itens de uma lista.</span>
</body>
</html>
```

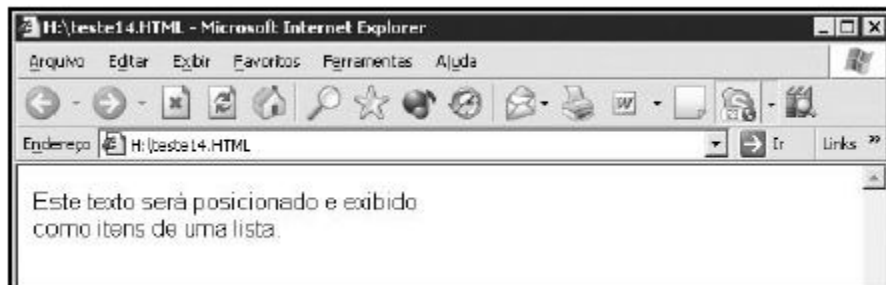


Figura 4.2.

Se a estilização for retirada, ou se for utilizado o valor inline, os dois textos ficarão na mesma linha:

```

<html>
<head>
<style type="text/css">
body { font-family: sans-serif }
span { display: inline }
</style>
</head>
<body>
<span>Este texto será posicionado e exibido</span>
<span>como um texto inline.</span>
</body>
</html>

```

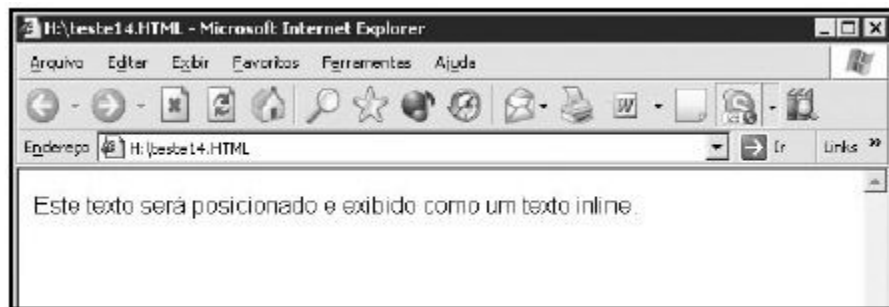


Figura 4.3.

O atributo float

O atributo float é um dos mais interessantes em CSS, pois com ele é possível definir onde um elemento será exibido em relação a outro, sem que uma posição fixa seja definida, ou seja, como se este elemento estivesse flutuando na página. Diferente do atributo position (melhor detalhado mais adiante), que define a posição exata de um elemento em relação à página que o exibe, ou a outro elemento, de forma fixa, o float definirá a disposição dos elementos, podendo criar colunas, fazer um texto contornar uma imagem etc., de forma flexível (como se ele estivesse realmente flutuando na tela) em relação a seu conteúdo.

O exemplo a seguir cria estilos para duas colunas que ocuparão, respectivamente, 30% e 70% da largura da página:

```

<html>
<head>

```

```

<style type="text/css">
div.coluna1 {
    float: left;
    width: 30%;
    color: blue;
}
div.coluna2 {
    float: left;
    width: 70%;
    color: red;
}
</style>
</head>
<body>
<div class="coluna1">
<h3>Coluna 1</h3>
<p>Esta coluna utiliza o atributo float e ocupa 30% do espaço
total da página, que contém duas colunas.</p>
</div>
<div class="coluna2">
<h3>Coluna 2</h3>
<p>Esta segunda coluna ocupa 70% da tela e também utiliza
o atributo float: left e é posicionada ao lado da Coluna 1.
</p>
</div>
</body>
</html>

```

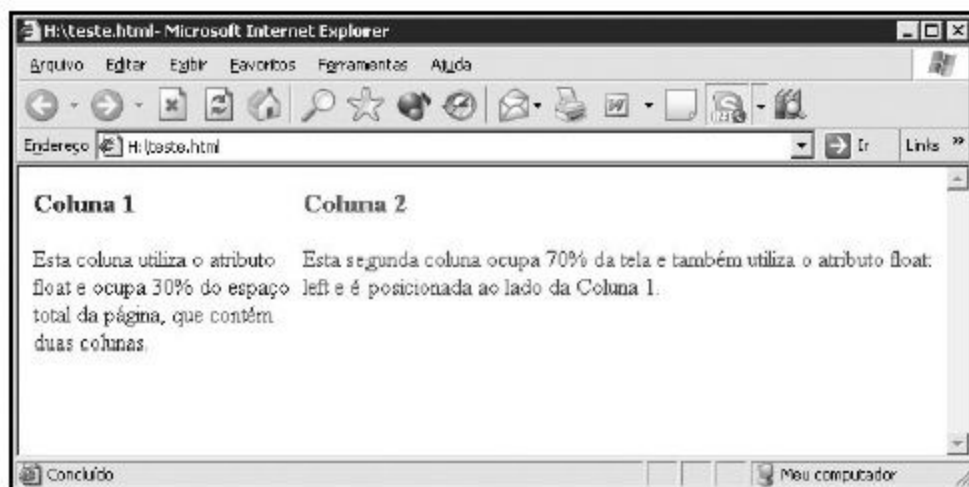


Figura 4.4.

Observe que o atributo float pode ser extremamente interessante para a criação de páginas com várias divisões/colunas, sem ter de utilizar tabelas, e com a vantagem da flexibilidade com que o conteúdo pode ser exibido.

O atributo clear

Complementando a função do atributo float, conheceremos, agora, um pouco do atributo clear, que possui a propriedade de definir quais lados de um elemento ficará limpo, sem a presença de nenhum outro elemento. Por padrão, todo elemento flutuante inserido em uma página será posicionado imediatamente ao lado do elemento anterior. Isso significa que esse atributo só terá efeito a partir do segundo elemento colocado na página. Se deseja abrir um espaço em branco à direita de um elemento, basta definir o seu float com o valor right.

Para exemplificar o funcionamento desse atributo, vamos retomar o exemplo anterior, incluindo-o na classe coluna2:

```
<html>
<head>
<style type="text/css">
div.coluna1 {
    float: left;
    width: 30%;
    color: blue;
}
div.coluna2 {
    float: left;
    width: 70%;
    color: red;
    clear: left;
}
</style>
</head>
<body>
<div class="coluna1">
<h3>Coluna 1</h3>
<p>Esta coluna utiliza o atributo float e ocupa 30% do espaço
total da página, que contém duas colunas.</p>
</div>
<div class="coluna2">
<h3>Coluna 2</h3>
<p>Esta segunda coluna ocupa 70% da tela e também utiliza
o atributo float: left e é posicionada ao lado da Coluna 1.
```

```
</p>  
</div>  
</body>  
</html>
```

Agora, o resultado com a coluna2 abaixo da coluna1, visto que foi definido que a coluna2 teria seu lado esquerdo “limpo”:

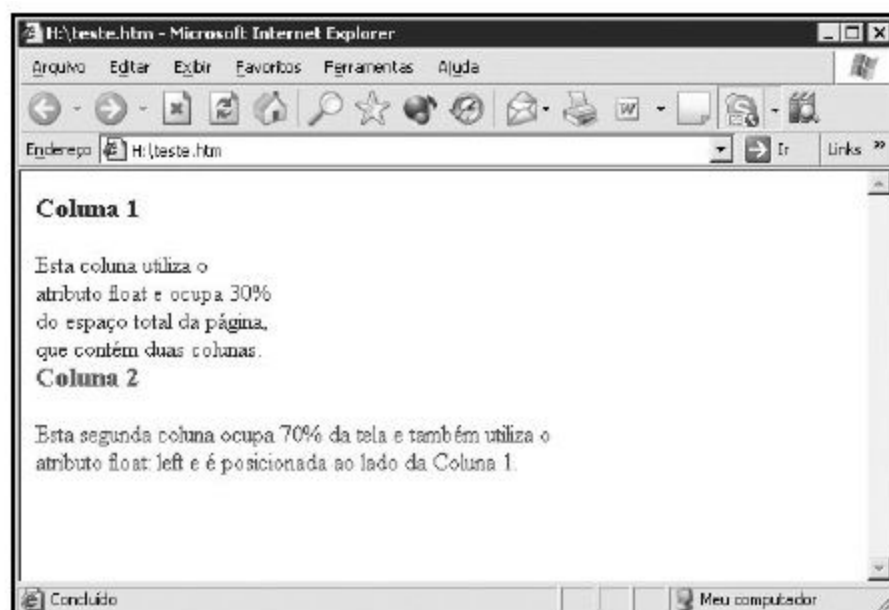


Figura 4.5.

Posicionamento

Os atributos de posicionamento de elementos definem não apenas como e onde um elemento se posicionará, mas também em relação a que ele fará isso, qual será seu próprio formato e como se comportará caso ultrapasse sua própria área. A seguir, fornecemos a definição de cada atributo juntamente com os valores por eles aceitos. Em seguida, detalharemos algumas particularidades desses atributos:

Atributo	Descrição	Valor
bottom	Define a distância em que a margem inferior de um elemento está acima/abaixo em relação à margem inferior de seu “elemento-pai”.	auto Porcentagem. Distância (valor).
clip	Define a forma de um elemento. O elemento é reduzido à esse formato e só então é exibido.	shape auto
left	Define a distância que a margem esquerda de um elemento está à esquerda/direita em relação à margem esquerda de seu “elemento-pai”.	auto Porcentagem. Distância (valor).
overflow	Define o que acontece com o conteúdo de um elemento caso ele ultrapasse sua área.	visible hidden scroll auto
position	Posiciona um elemento em posição	static

	estática, relativa, absoluta ou fixa.	relative absolute fixed
right	Define a distância que a margem direita de um elemento está à esquerda/direita em relação à margem direita de seu “elemento-pai”.	auto Porcentagem. Distância (valor).
top	Define a distância que a margem superior de um elemento está acima/abaixo em relação à margem superior de seu “elemento-pai”.	auto Porcentagem. Distância (valor).
vertical-align	Define o alinhamento vertical de um elemento.	baseline sub super top text-top middle bottom text-bottom Porcentagem. Distância (valor).
z-index	Define a posição de um elemento dentro de uma fila de elementos.	auto Número.

Tabela 4.2.

Posições: estática x fixa

A posição estática consiste na posição padrão de um elemento em relação às coordenadas da página. Quando o atributo `position` é definido como `static`, são ignorados os atributos `top`, `bottom`, `left` e `right`. Ao contrário, a posição fixa é justamente definida por esses quatro atributos, isto é, o elemento se posicionará na janela do browser de acordo com as coordenadas especificadas por eles, desconsiderando paginação com a barra de rolagem.

Posições: relativa x absoluta

A posição relativa define que um elemento será posicionado em uma coordenada definida em relação à sua posição normal/original. Já a posição absoluta define esse posicionamento em relação à seção a que o elemento pertence. Se o objeto está dentro de uma seção (definida pela tag `<div>`) interna a outra seção, os valores das coordenadas definidas terão como origem as posições iniciais da “seção-pai”.

O exemplo a seguir demonstra bem o funcionamento dessas duas posições.

Exemplo: texto com sombras

Este exemplo utiliza as posições relativa e absoluta dentro de tags `<div>`, que produzirão o efeito de texto com sombra.

Note que a primeira classe criada, `sombra1`, possui posição relativa. Como se trata do primeiro elemento a ser posicionado na tela, ele também poderia ter posição absoluta, pois as coordenadas seriam definidas em função do topo da página. Já para a segunda sombra, `sombra2`, que será deslocada quatro pixels da esquerda para a direita, e de cima para baixo, a posição será absoluta devido ao fato de estarmos utilizando a tag `<div>` aninhada. Isso faz com que a `sombra2` tome como ponto de partida não o topo da página, mas a posição em que a `sombra1` foi posicionada. O mesmo ocorre com a classe `texto`, abaixo da seção da `sombra2`, e, por isso, considerará a posição desta. Veja a definição dos estilos:

```
<style type="text/css">

div.sombra1 {
position:relative;
color:powderblue;
font: 50px "arial black";
}

div.sombra2 {
position:absolute;
top:4px;
left:4px;
color:deepskyblue;
font: 50px "arial black";
}

div.texto {
position:absolute;
top:4px;
left:4px;
color:royalblue;
font: 50px "arial black";
}
</style>
```

Agora, em código HTML:

```
<body>
<div class="sombra1">Texto Com Duas Sombras
<div class="sombra2">Texto Com Duas Sombras
<div class="texto">Texto Com Duas Sombras
</div>
</div>
</div>
</body>
```

O código anterior resultará no seguinte efeito:



Figura 4.6.

Faça testes alterando o valor do atributo position e o aninhamento das tags <div>. Observe como os textos serão posicionados quando as referências são modificadas.

Exemplo: texto com contorno

Com esse exemplo, reforçaremos o conceito de posicionamento em função de posições absolutas. Utilizando quatro elementos de texto em posições diferentes, criaremos um contorno ao redor de um quinto elemento, que possuirá cor diferente dos demais. Veja a definição dos estilos:

```
<style type="text/css">

div.contorno1 {
position:absolute;
left:2px;
top:2px;
color:blue;
font: 50px "arial black";
}

div.contorno2 {
position:absolute;
left:2px;
top:0px;
color:blue;
font: 50px "arial black";
}

div.contorno3 {
position:absolute;
left:-2px;
top:2px;
color:blue;
font: 50px "arial black";
}

div.contorno4 {
position:absolute;
left:2px;
top:0px;
color:blue;
```

```
font: 50px "arial black";  
}  
  
div.texto {  
position: absolute;  
left: -1px;  
top: -1px;  
color: white;  
font: 50px "arial black";  
}  
</style>
```

Agora, em código HTML:

```
<body>  
<div class="contorno1">Texto Com Contorno  
  
<div class="contorno2">Texto Com Contorno  
<div class="contorno3">Texto Com Contorno  
<div class="contorno4">Texto Com Contorno  
<div class="texto">Texto Com Contorno  
</div>  
</div>  
</div>  
</body>
```

Note que na criação dos estilos são definidas quatro classes cujas posições variam em dois pontos na vertical e na horizontal. Essas coordenadas formam um quadrado. À primeira vista, elas podem não fazer sentido, mas considerando que cada elemento é posicionado em função do último elemento criado (já que estarão em seções aninhadas), torna-se mais fácil a compreensão. A classe contorno1 é posicionada em um lugar qualquer; foi escolhida a coordenada (2,2) para facilitar a comparação com as outras posições. A classe contorno2 será posicionada dois pontos à esquerda de contorno1, mantendo-se na posição horizontal. A classe contorno3 será posicionada em função de contorno2: dois pontos à esquerda e dois pontos abaixo. Seguindo a mesma lógica, a classe contorno4 será posicionada em função da classe contorno3.

Feito isso, vamos inserir um texto da cor branca na posição central (horizontal e verticalmente) em relação às quatro classes criadas. Essa classe texto será posicionada em função da posição de contorno4. Como as classes anteriores tiveram suas posições variando em dois pontos, o texto será posicionado um ponto à esquerda e um ponto acima de contorno4

(recuo de um ponto em cada direção) para ficar centralizado. O resultado será o seguinte:



Figura 4.7.

Pseudoclasses

As pseudoclasses consistem em uma combinação de classes e eventos, na medida em que acrescenta ou altera o estilo de determinados elementos em função de um evento que ocorre com ele. A seguir, temos a sintaxe básica de como utilizar uma pseudoclassee quais são elas, com suas respectivas descrições:

```
elemento:pseudo-classe { atributo: valor }
```

Ou:

```
elemento.classe:pseudo-classe { atributo: valor }
```

Pseudoclassee	Descrição
:first-child	Acrescenta estilo ao primeiro elemento “filho” de outro elemento.
:link	Acrescenta estilo a um link não visitado.
:visited	Acrescenta estilo a um link visitado.
:hover	Acrescenta estilo a um elemento quando o mouse é posicionado sobre ele.
:active	Acrescenta estilo a um elemento ativo/selecionado.
:focus	Acrescenta estilo a um elemento quando ele estiver com foco.
:lang	Define o idioma a ser utilizado em um determinado

elemento.

Tabela 4.3.

Como se pode perceber, as pseudoclasses são mais voltadas a links. A utilização delas nos poupa da necessidade de utilizar scripts de outras linguagens para determinar o que acontece em determinados eventos.

Exemplo: estilização de link

Demonstraremos dois casos de mudança de estilo em função dos eventos que as subclasses representam: mudança de cor e tamanho de fonte quando o ponteiro do mouse é posicionado sobre um link, e mudança de cor quando este é clicado:

```
<style type="text/css">
a:link {color: #6A5ACD}
a:visited {color: #FF6347}

a.cor:hover {color: #9ACD32}

a.tamanho:hover {font-size: 120%}
</style>
```

Na segunda linha, vemos a estilização genérica para todos os tipos de link. Na quarta, vê-se a estilização de mudança de cor da fonte para links da classe cor. Na penúltima linha, vemos a mudança de tamanho de fonte somente para links da classe tamanho. Este é o código HTML que utiliza as folhas de estilo definidas anteriormente:

```
<body>
<p><b><a class="cor" href="#">
Link com mudan&ccedil;a de cor.
</a></b></p>
<p><b><a class="tamanho" href="#">
Link com mudan&ccedil;a de tamanho de fonte.
</a></b></p>
</body>
```

Simulando um botão

A partir do elemento link, e utilizando as pseudoclasses apresentadas, é possível criar uma classe que simula um botão que, ao pressionar do mouse, tem sua aparência alterada (como se fosse pressionado).

O primeiro passo é definir o botão quanto à forma, tamanho, cores etc. Isso será feito por meio da classe botao (sem pseudoclasse). Em seguida, modificando os atributos das bordas, criaremos dois estilos para as pseudoclasses :visited e :hover, que apenas terão as cores invertidas para os pares top/left e bottom/right. Essa estilização será feita da seguinte maneira:

```
<style type="text/css">
a.botao {
    font-family: verdana;
    font-size: 12pt;
    font-weight: bold;
    padding: 5px;
    background-color: #DCDCDC;
    color: #666666;
    text-decoration: none;
}
a.botao:visited {
    border-top: 1px solid #CCCCCC;
    border-bottom: 2px solid #666666;
    border-left: 1px solid #CCCCCC;
    border-right: 2px solid #666666;
}
a.botao:hover {
    border-top: 2px solid #666666;
    border-bottom: 1px solid #CCCCCC;
    border-left: 2px solid #666666;
    border-right: 1px solid #CCCCCC;
}
</style>
```

No código HTML, observe que apenas definimos um link da classe botao, juntamente com a página que será aberta quando o botão é clicado, e com o texto dele. Perceba que essa classe, uma vez criada, poderá ser reaproveitada para criar diversos botões com links e textos diferentes:

```
<body>
<a href="#" class="botao">Clique aqui!</a>
```


</body>

Como o que parece ser um botão é definido pelo atributo padding, formatado em função do conteúdo do link (no caso, o texto **Clique aqui!**), você pode colocar qualquer texto no botão que ele o tomará como referência para o seu tamanho. Experimente alterar o texto do botão e veja: ele tem o tamanho alterado proporcionalmente.

Pseudo-elementos

Um pseudo-elemento consiste em uma referência que permite formatar apenas uma parte de um elemento. Observe:

Pseudo-elemento	Descrição
:first-letter	Acrescenta estilo à primeira letra de um texto.
:first-line	Acrescenta estilo à primeira linha de um texto.
:before	Insere conteúdo antes de um elemento (não funciona no IE).
:after	Insere conteúdo depois de um elemento (não funciona no IE).

Tabela 4.4.

Exemplo de: first-letter

O exemplo que segue consiste em exibir um texto cuja primeira letra possuirá uma formatação diferente das restantes. Veja uma folha de estilo:

```
<style type=»text/css»>
p { font-size: 20pt }
p:first-letter { color:blue;
                  font-size:30pt }
</style>
```

Agora, em código HTML:

```
<body>  
<p>Primeira letra de um texto</p>  
</body>
```

Incluindo esse código em uma página HTML, temos o seguinte resultado:

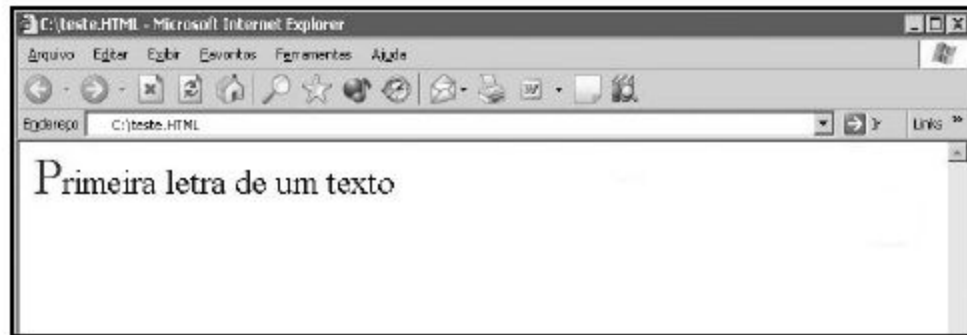


Figura 4.8.

Transparência com CSS

As folhas de estilo possuem filtros que permitem alterar o nível de transparência/opacidade de um elemento (principalmente imagens). Isso é feito através dos atributos seguintes:

Atributo	Descrição	Valor
<code>-moz-opacity:n</code>	Define nível de transparência no Mozilla.	n = 0.0 a 1.0
<code>filter:alpha(opacity=n)</code>	Define nível de transparência no IE.	n = 0 a 100

Tabela 4.5.

Para garantir que o efeito será visível em todos os browsers, os dois atributos podem ser utilizados juntos, pois cada um deles interpretará apenas o atributo válido para ele.

A seguir, temos um exemplo de exibição de uma imagem cujo nível de transparência é alterado quando o mouse é posicionado sobre ela (com a utilização dos eventos `onmouseover` e `onmouseout`).

Em primeiro lugar, o nível de transparência do elemento imagem é definido na seção de estilos:

```
<style type="text/css">
img
{ -moz-opacity:0.4;
  filter:alpha(opacity=40)
}
</style>
```

No corpo da página, uma imagem é inserida e ela terá 40% de opacidade (quanto menor o valor, mais transparente o elemento será), de acordo com o estilo previamente definido. Dentro da tag de imagem haverá dois eventos

que modificarão o nível de opacidade e transparência: quando o cursor do mouse é posicionado sobre a imagem, o nível de opacidade é 100, e quando o cursor é retirado daquela área, a imagem volta a ter transparência, como mostra o código:

```
<body>  
  
  
</body>
```

A diferença de transparência será vista como no exemplo seguinte:

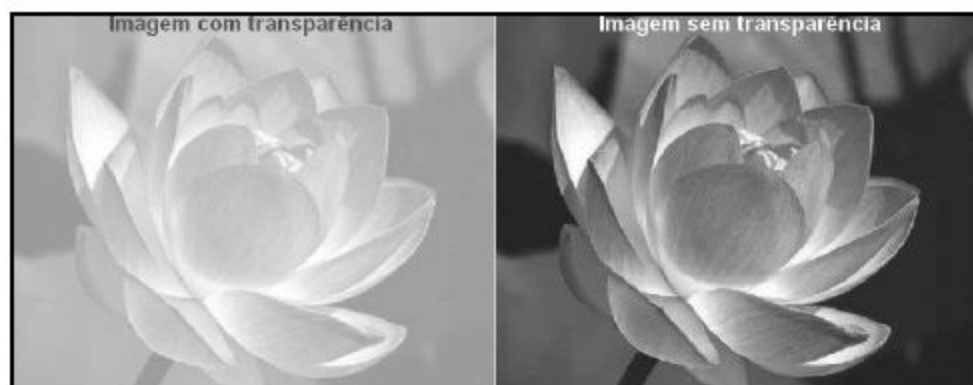


Figura 4.9.

Menus

As folhas de estilo permitem a estilização de links que criam efeitos bastante interessantes em menus horizontais e verticais. Em HTML, o mais comum é que os menus sejam criados a partir de listas. Isso se dá pelo fato de ser um elemento o que mantém os itens organizados, sua estrutura é preservada caso o browser não suporte CSS ou imagens, oferece facilidade na manutenção e inclusão de novos itens etc. Este tópico fornecerá vários exemplos de menus baseados em listas e, também, menus horizontais formatados com a utilização de parágrafos, todos com recursos extras que darão ao seu menu um aspecto mais profissional.

Exemplo de menu vertical

O exemplo a seguir define o estilo de um menu vertical simples, cujas opções têm sua cor alterada quando o ponteiro do mouse é posicionado sobre elas:

```
<style type="text/css">      1
#menu {
width: 10em;
margin: 0;
font: 15px Verdana;        2
background: gray;
border: 1px solid yellow;
}
#menu li {
list-style: none;          3
margin: 1em 1em;
}
#menu li a {
text-decoration: none;     4
color: white;
}
#menu li a:hover {
background: yellow;        5
color: gray;
}
#menu li a:active {
background: gray;          6
color: yellow;
}
</style>
```

No código anterior, definimos os seguintes pontos:

- Bloco 1: delimita área de estilo;
- Bloco 2: define atributos do menu, como largura, margem, fonte, cores etc;
- Bloco 3: define o estilo da estrutura dos itens do menu, como o espaçamento entre itens;
- Bloco 4: define atributos do texto dos itens do menu (sem sublinhado e na cor branca);
- Bloco 5: define atributos do item quando o mouse for sobre ele posicionado;
- Bloco 6: define atributos do item ativo do menu.

O código HTML do menu apenas insere os itens e o id menu faz a ligação com os estilos criados:

```
<body>
<ul id="menu">
<li>
<a href="#" title="Voltar ao início">Home</a>
</li>
<li>
<a href="#" title="Cadastros Gerais">Cadastros</a>
</li>
<li>
<a href="#" title="Emitir Relatórios">Relat&ocirc;rios</a>
</li>
<li>
<a href="#" title="Configurar site">Configura&ccedil;&otilde;es</a>
</li>
<li>
<a href="#" title="Fale conosco">Contato</a>
</li>
</ul>
</body>
```

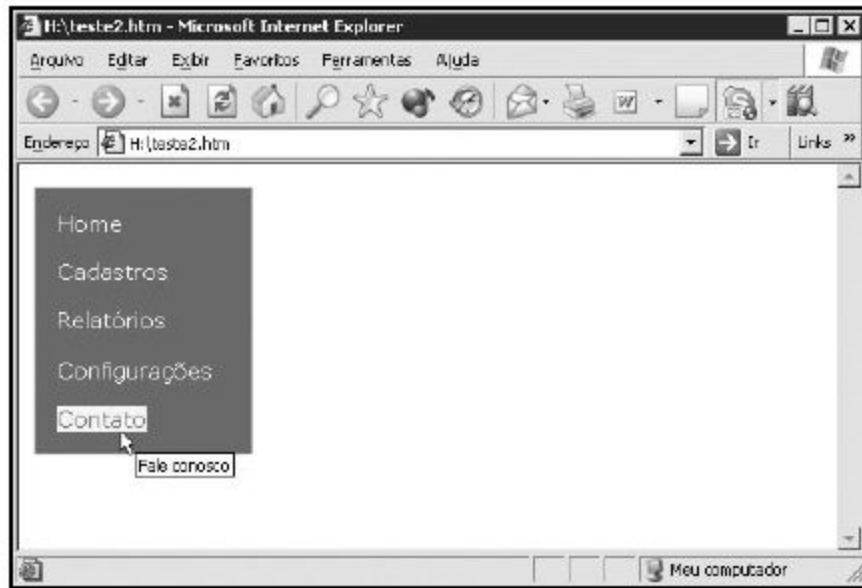


Figura 4.10.

Observação: no atributo href não foi definida a página ou parte da página que deve ser aberta/chamada, mas o menu pode ser testado sem problemas.

Exemplo de menu horizontal

No caso do menu horizontal, ao invés de utilizarmos listas (que são verticais), faremos o uso de parágrafos (que são horizontais) para fixar os links do menu. Uma classe chamada menu é criada para o parágrafo para garantir que o estilo será válido somente para o menu. Primeiramente é definido o estilo do menu e, com o uso da pseudoclasse: hover, definimos a cor de fundo que a opção do menu terá quando o ponteiro do mouse for posicionado sobre ela.

A seguir, veja a definição da folha de estilo que formatará o menu, e sua chamada no código HTML:

```
<style type="text/css">
p.menu a {
```



```

color: #000000;
background: #90EE90;
font: 12px Verdana;
text-decoration: none;
text-align: center;
border: 1px solid #000000;
padding: 1px 5px;
margin-right: 1px;
}
p.menu a:hover {
background: #32CD32;
}
</style>

```

1
2

No código anterior, temos:

- No bloco 1, a folha de estilo a ser definida será feita para os links a, delimitados pelos parágrafos p;
- No bloco 2, quando o mouse é posicionado sobre o item do menu, a cor do plano de fundo é alterada.

Agora, em código HTML:

```

<body>
<p class="menu">
<a href="#" >Home</a>
<a href="#">Quem Somos</a>
<a href="#">Hist&ocirc;rico</a>
<a href="#">Contato</a>
<a href="#">Mapa do Site</a>
</p>
</body>

```

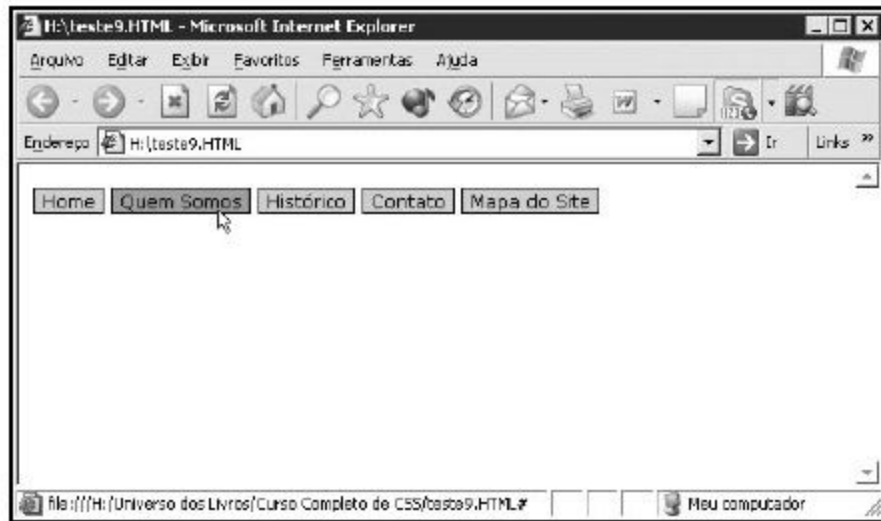


Figura 4.11.

Exemplo de menu com *tooltip*

Um menu com *tooltip* consiste em um conjunto de opções, e cada uma delas, ao ter o cursor do mouse posicionado sobre ela, mostrará uma caixa de texto com uma breve definição sobre o que se trata aquele link.

```
<style type="text/css">
ul#menutip {
width: 150px;                1
list-style: none;
font: 15px Arial;
}
ul#menutip li{              2
position: relative;
}
ul#menutip a {
display: block;
text-align: left;
padding: 4px 8px;
margin-bottom: 1px;        3
text-decoration: none;
color: #000000;
background: #00BFFF;
}
ul#menutip a:hover {
```

```

color: #FFFFFF;                                4
background: #00DFFF;
}
ul#menutip a span {
display: none;                                5
}
ul#menutip a:hover span {
display: block;
position: absolute;
top: 0;
left: 140px;
width: 130px;
padding: 4px;                                6
margin-left: 2px;
color: #000000;
background: #00DFFF;
font-size: 10px;
text-align: left;
border: 1px solid #000000;
}
</style>

```

No código anterior, temos:

- Bloco 1: a largura do menu (no caso, da lista definida pela tag), estilo, tipo e tamanho da fonte;
- Bloco 2: posição relativa para os itens do menu;
- Bloco 3: folha de estilo para os links que serão colocados internamente à lista;
- Bloco 4: alteração de cor de fonte e de background do item quando o ponteiro do mouse for posicionado sobre ele;
- Bloco 5: área ligada a cada item do menu, que não será exibida até que o mouse se posicione sobre ele;
- Bloco 6: quando o ponteiro do mouse é posicionado sobre o item, a área será estilizada para ser exibida com a indicação ao lado dele. Esta área será colocada em posição absoluta para ser posicionada ao lado do menu (caso contrário, ele seria mostrado sobre os itens, pois consideraria a posição em relação à página, não ao menu).

Observe no código HTML seguinte que foi criada uma lista não ordenada (com a tag), cujos itens () são links que, internamente, possuem áreas do tipo , que, por sua vez, possuirão a descrições de cada

opção, exibidas apenas quando o ponteiro do mouse for posicionado naquele item:

```
<body>
<ul id="menutip">
<li>
<a href="#">Home
<span> Retornar &agrave; P&acute;gina Inicial.</span>
</a>
</li>

<li>
<a href="#">Quem Somos
<span>Conhe&ccedil;a a nossa equipe, nossos objetivos e
compromissos
sociais.</span>
</a>
</li>

<li>
<a href="#">Hist&oacute;rico
<span>Acompanhe o hist&oacute;rico de atividades de nossa
empresa.</span>
</a>
</li>

<li>
<a href="#">Mapa do site
<span>Mapa com o conte&uacute;do completo do site.</span>
</a>
</li>

<li>
<a href="#">Contato
<span>Envie suas d&uacute;vidas e sugest&otilde;es.</span>
</a>
</li>
</ul>
</body>
```

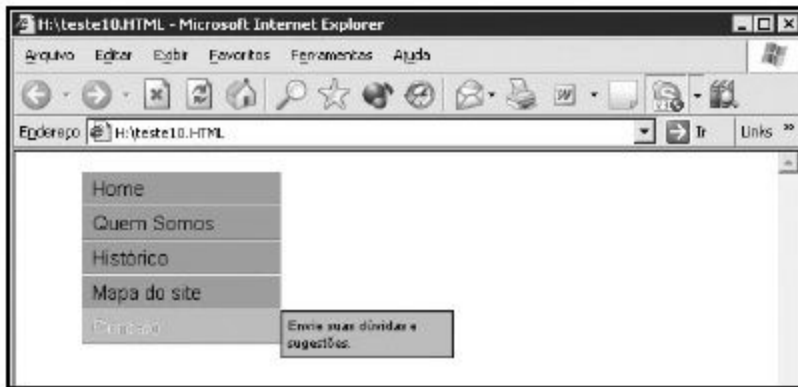


Figura 4.12.

Exemplo de menu com *imagemap*

Muitas vezes, desejamos variar o formato dos menus dos sites com o objetivo de obter layouts mais criativos e originais. Uma alternativa bastante interessante é utilizar *imagemaps*, ou mapas de imagens. Essa estrutura consiste em um mapeamento de coordenadas (coluna x linha), que tem como base uma imagem. A esses pontos é possível atribuir links que levarão o usuário a outras páginas HTML:

```
<style type="text/css">
ul {
position: relative;
list-style: none;
padding: 0;
margin: 0;
width: 390px;
height: 373px;
background: url(brasil _ map.jpg) no-repeat;
}

a {
position: absolute;
display: block;
text-decoration: none;
text-align: center;
color: #F00000;
border: 4px solid #F00000;
}
```

1

2

```

a.menu _ no{
top:95px;
left:129px;
}
a.menu _ ne{
top:130px;
left:300px;
}
a.menu _ ce{
top:188px;
left:198px;
}
a.menu _ se{
top:240px;
left:260px;
}
a.menu _ su{
top:305px;
left:205px;
}

a span { display: none }

a.menu _ no:hover span, a.menu _ ne:hover span,
a.menu _ ce:hover span, a.menu _ se:hover span,
a.menu _ su:hover span {
width:120px;
display:block;
position:relative;
font:11px arial;
padding:5px;
border:1px solid #F00000;
background:#FFFFFF;
color:#000000;
text-decoration:none;
}

a:hover {
border: none;
top: 250px;
left: 20px;
}
</style>

```

No código anterior, temos:

- Bloco 1: lista cuja imagem de background servirá de base para os itens do menu. A largura e altura da imagem devem ser informados para o correto posicionamento dos links;
- Bloco 2: aparência dos links (no caso, os quadradinhos vermelhos), incluindo cor, ausência de sublinhado, alinhamento central etc;
- Bloco 3: posiciona cada link no mapa. Como os links possuem localizações específicas (e não genéricas), cada um deve ter uma classe própria com a coordenada de posicionamento;
- Bloco 4: esconde caixas de texto com descrição dos links;
- Bloco 5: folha de estilo genérica para formatação de todas as caixas de texto definidas na área . Essas caixas de texto, ocultas em seu estado Inicial, serão exibidas quando o ponteiro do mouse for posicionado sobre os quadradinhos;
- Bloco 6: caixa de texto com descrição do link na posição indicada.

A seguir, o código HTML com o endereço a ser chamado por cada link, e o texto que descreve/define cada um deles:

```
<body>
<ul>
<a href="#" class="menu _ no">
<span>Informa&ccedil;&otilde;es sobre a regi&atilde;o Nor-
te</span></a>
<a href="#" class="menu _ ne">
<span>Informa&ccedil;&otilde;es sobre a regi&atilde;o Nor-
deste</span></a>
<a href="#" class="menu _ ce">
<span>Informa&ccedil;&otilde;es sobre a regi&atilde;o Centro-
Oeste</span></a>
<a href="#" class="menu _ se">
<span>Informa&ccedil;&otilde;es sobre a regi&atilde;o Su-
deste</span></a>
<a href="#" class="menu _ su">
<span>Informa&ccedil;&otilde;es sobre a regi&atilde;o Sul</
span></a>
</ul>
</body>
```

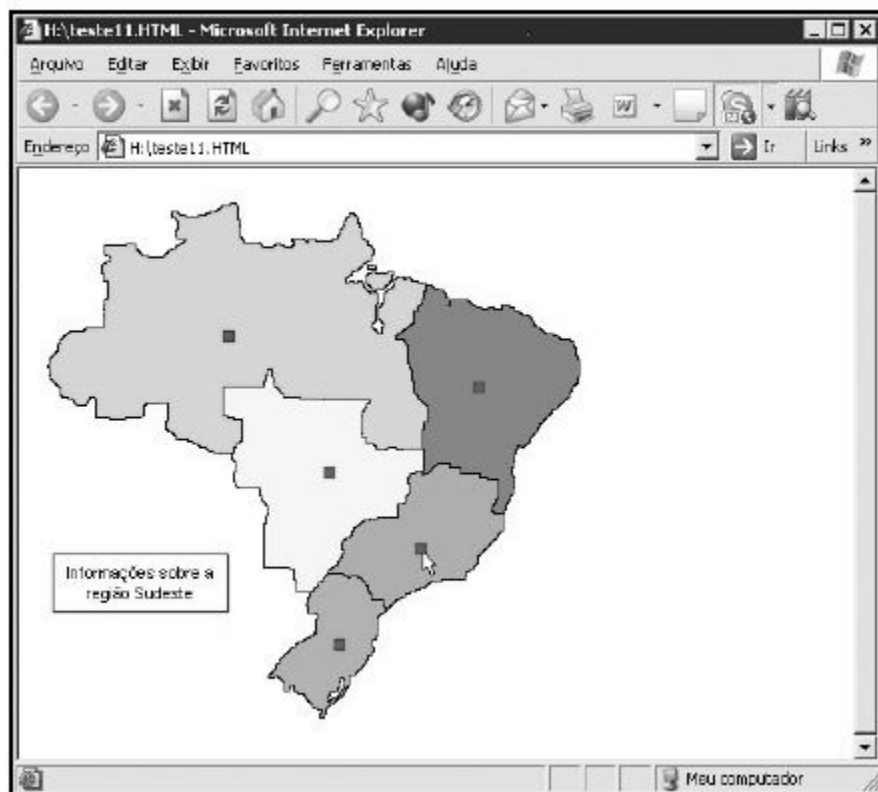


Figura 4.13.

Formulários

Este tópico tem como objetivo demonstrar a formatação e a padronização da aparência de formulários. O foco dele está em como as folhas de estilo podem ser aplicadas a estas estruturas sem considerar o funcionamento interno delas (validação de campos, funções de botões etc.), que, em geral, é implementado com a linguagem Javascript.

Exemplo de formulário

O exemplo a seguir gera folhas de estilo para a formatação de títulos, campos de *input* e botões. As classes criadas para cada objeto tornam-se extremamente úteis para a padronização do layout de formulários, permitindo também a inclusão de novos objetos em páginas que já a utilizam e o reaproveitamento para novas páginas:

```
<style type="text/css">
#mainform {
background: #FFFFFF;
border: 1px solid #000000;
font: 12px arial;
border-collapse: collapse;
color: #7B68EE;
}
#mainform th {
background: #DCDCDC;
padding: 3px;
font: bold 15px arial;
border-bottom: 1px solid #7B68EE;
}
#mainform td {
padding: 3px;
}
#mainform input {
background: #DCDCDC;
border: 1px solid #000000;
}
#mainform textarea {
border: 1px solid #000000;
background: #DCDCDC;
}
#mainform input.botao {
background: #DCDCDC;
```

1

2

3

4

5

6

```
color: #000000;  
border: 1px solid #000000;  
}  
</style>
```

6

No código anterior, temos:

- Bloco 1: folha de estilo com atributos gerais do formulário, que valem para todos os elementos internos a ele (*input*, botão etc.), como tipo e cor de fonte, tipo de borda etc;
- Bloco 2: folha de estilo específica para o cabeçalho da tabela que receberá os elementos do formulário;
- Bloco 3: folha de estilo para os itens da tabela, ou seja, os elementos de *input* do formulário;
- Bloco 4: folha de estilo com atributos do objeto de *input* do formulário;
- Bloco 5: folha de estilo com atributos do objeto *textarea* do formulário;
- Bloco 6: folha de estilo com atributos do objeto *botao* do formulário.

Agora, em código HTML:

```
<body>  
<form name="formulario" action="" method="" >  
<table id="mainform">  
<tr>  
<th colspan="2">Fale Conosco</th>  
</tr>  
<tr>  
<td><label>Nome:</label>  
</td>  
<td><input name="nome" type="text" size="33"  
maxlength="100"></td>  
</tr>  
<tr>  
<td><label>E-mail:</label> </td>  
<td><input name="email" type="text" size="33"  
maxlength="100"></td>  
</tr>  
<tr>  
<td><label>  
Mensagem:</label></td>  
<td><textarea name="comentario" cols="25"
```

```

rows="7"></textarea></td>
</tr>
<tr>
<td><input name="submit" type="submit"
value="Enviar" class="botao">
</td>
</tr>
</table>
</form>
</body>

```

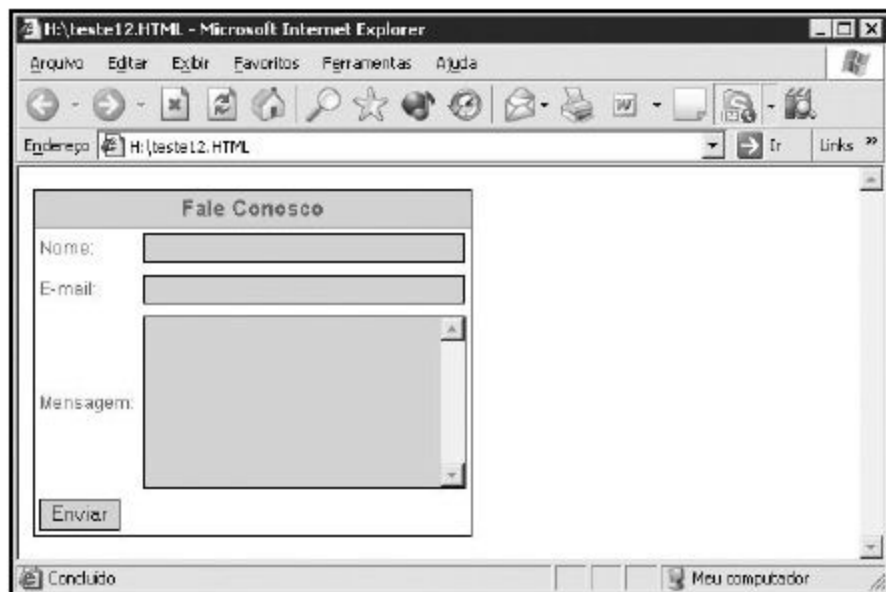


Figura 4.14.

O leitor deve ter percebido que, nesse exemplo, foi utilizada uma tabela para estruturar o formulário. Essa alternativa poderia ser considerada contrária ao padrão Tableless, já que está utilizando uma tabela para um objetivo diferente do que é proposto para esse tipo de elemento. Porém, a tabela não está sendo utilizada para estruturar o layout de uma página inteira (e conseqüentemente torná-la inflexível o suficiente para provocar algum problema de visualização em relação ao browser que a acessa), mas sim de apenas um elemento (<form>), que pode ser posicionado em diversos pontos da página e cujo objetivo é justamente manter fixa a estrutura do formulário.

Capítulo 5

Tela ou impressão?

É comum acontecer de uma página possuir um ótimo layout, porém, na hora de imprimi-la, o resultado é insatisfatório, seja pelo tamanho da fonte, que não se conforma ao tamanho do papel, seja pela disposição da imagem etc. Até mesmo para evitar desperdício, existe o interesse de imprimir somente o texto e desprezar a imagem que serve de background.

As CSS possuem uma espécie de “regra de estilo” definida como `@media`, cuja função é delimitar as folhas utilizadas para visualização de uma página na tela do computador (screen) ou para impressão (print). Veja sua estrutura:

```
<style>
@media tipo-de-mídia {
    Definição do(s) estilo(s)
    .
    .
}
</style>
```

Veja um exemplo que define atributos distintos para cada tipo de mídia (tela ou impressora) e também como proceder caso existam atributos comuns para elas:

```
<style>

@media screen {
p { font-family:verdana; font-size:15px }
}
```

```
@media print {  
p { font-family:serif; font-size:10px }  
}  
  
@media screen,print {  
p { font-style:italic }  
}  
</style>
```

No caso demonstrado, tanto para a visualização da página na tela do computador quanto para a impressão, os parágrafos seriam formatados em itálico. Porém, a fonte seria menor para a mídia impressa.

Capítulo 6

CSS e Dreamweaver

O Dreamweaver, editor de páginas HTML da Adobe, além de ser uma ferramenta para a criação de páginas, também possui suporte à criação e organização de folhas de estilo, entre outros tipos de documentos (como Javascript, XML, PHP etc.). O objetivo deste capítulo é fornecer uma visão geral de como o leitor pode se beneficiar dos recursos do Dreamweaver para trabalhar com folhas de estilo, considerando-o uma alternativa para a implementação delas. Portanto, serão apresentadas apenas as funções relacionadas às CSS.

Em relação ao desenvolvimento de páginas, não é aconselhado que se dependa única e exclusivamente de programas editores, nos quais o usuário projeta a página em função do layout (sem saber como isso é feito em HTML) e não do código. No caso das CSS, a função do Dreamweaver muda um pouco e ele pode ser visto mais como uma ferramenta complementar do que fundamental, pois o código das CSS terá de ser manipulado diretamente. O aspecto positivo é que o programa auxilia nessa implementação mediante menus *dropdown*, que listam os elementos, atributos e mapas de cores a serem utilizados na estilização e painéis que organizam e permitem a edição dessas folhas.

O download da versão trial do Adobe Dreamweaver CS3 pode ser feito no site www.adobe.com/products/dreamweaver/. Os requisitos mínimos de sistema para a instalação do software, para o Windows, são:

- Processador: Intel Pentium 4 ou compatível;

- Sistema operacional: Windows XP SP2, ou superior, Windows Vista Home Premium, Business, Enterprise, Ultimate;
- Memória RAM: 512 MB;
- Disco rígido: 1.3 GB de espaço livre;
- Drive de CD ou DVD-ROM;
- Acesso à Internet para ativação do software. Para o Macintosh:
- Processador: G4, G5, ou Intel-based Mac;
- Sistema operacional: Mac OS 10.4.8;
- Memória RAM: 512 MB;
- Disco rígido: 1.7 GB de espaço livre;
- Drive de CD ou DVD-ROM;
- Acesso à Internet para ativação do software.

Interface



Figura 6.1: Tela inicial.

Por meio da tela inicial, é possível abrir um arquivo recém-criado, criar um novo documento de um determinado tipo, ou a partir de um template existente. Isso vale tanto para as páginas HTML quanto para as CSS (ou outro tipo de arquivo).

A seguir, exemplificaremos como o Dreamweaver pode ser utilizado para a criação de arquivos CSS e, também, como auxiliar na estilização de elementos em páginas HTML.

Criando arquivos CSS externos

Ao selecionar a opção para criar um novo arquivo CSS em branco, ou baseado em um template a partir da tela inicial, uma área de trabalho para uma nova folha de estilo externa será aberta, criando um arquivo para ser chamado na seção de estilos da página HTML.

A área de trabalho exibida a seguir é o local em que todos os estilos serão definidos:

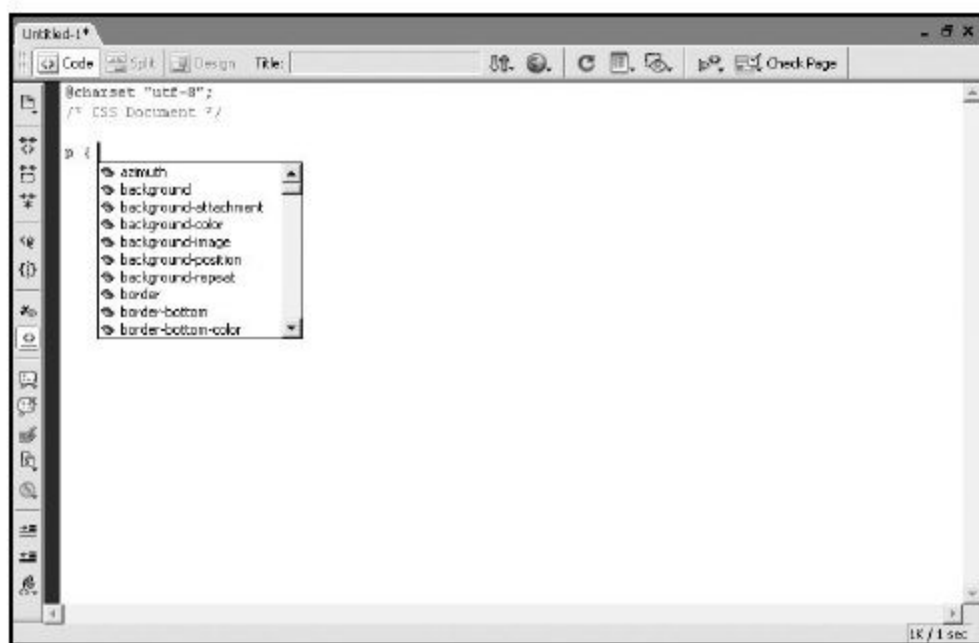


Figura 6.2: Área de trabalho.

Na medida em que são digitados os nomes dos elementos, atributos ou valores, o Dreamweaver apresenta opções para completar automaticamente o código, facilitando o processo de definição dos estilos.


Terminada a definição dos estilos, salve o arquivo com a extensão .css. Ele já poderá ser utilizado como folha de estilo para uma ou mais páginas HTML, como já foi ensinado no início deste livro.

Organizando as folhas de estilo em uma página HTML

Para compreendermos melhor como o Dreamweaver identifica e organiza as folhas de estilo, tomaremos como exemplo a criação de novas páginas com poucos elementos, para facilitar o entendimento. A partir do momento que esses conceitos básicos forem assimilados, a edição de folhas de estilo existentes será bem mais intuitiva e não apresentará dificuldades.

O primeiro passo é abrir o Dreamweaver e criar um novo arquivo HTML. Salve o arquivo com o nome e no diretório que você desejar para dar início a alguns testes no aplicativo envolvendo CSS.

Formatação de textos

No modo Design (clique no botão , escreva algum texto na área de trabalho, selecione-o e altere tipo de fonte, cor, tamanho etc. Utilize a barra de ferramentas na parte inferior da tela:

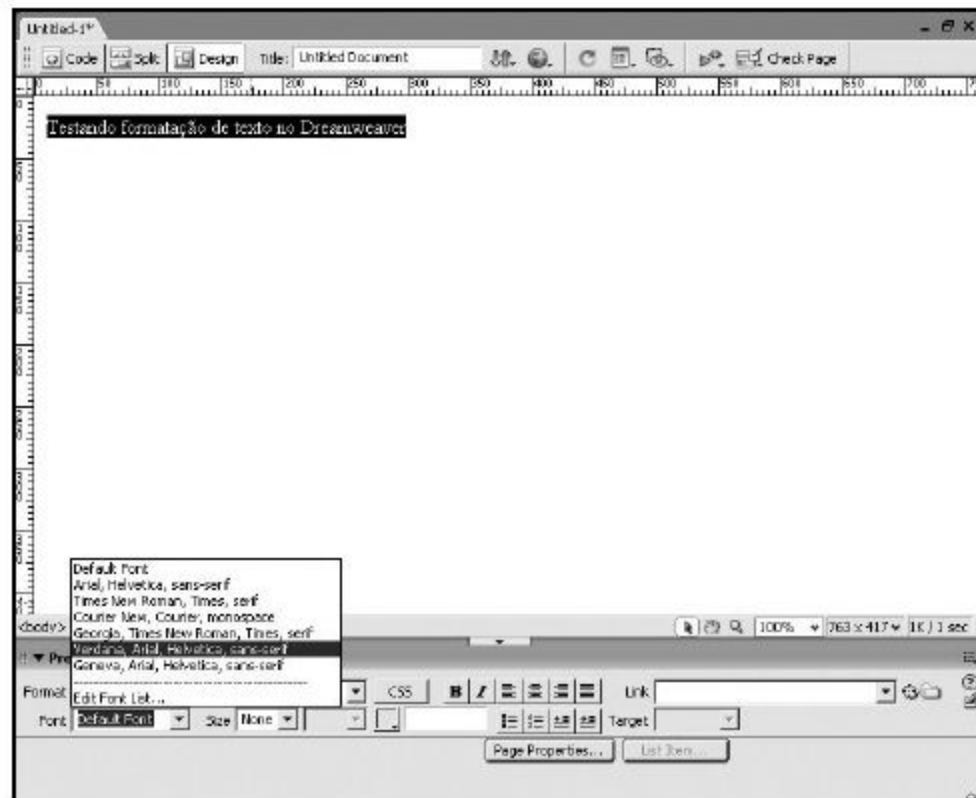


Figura 6.3: Alterando propriedades do texto.

Alteradas algumas propriedades do texto, observe que no campo **Style** surgiu o nome *Style1*, que corresponde ao conjunto de modificações que acabou de ser definido:

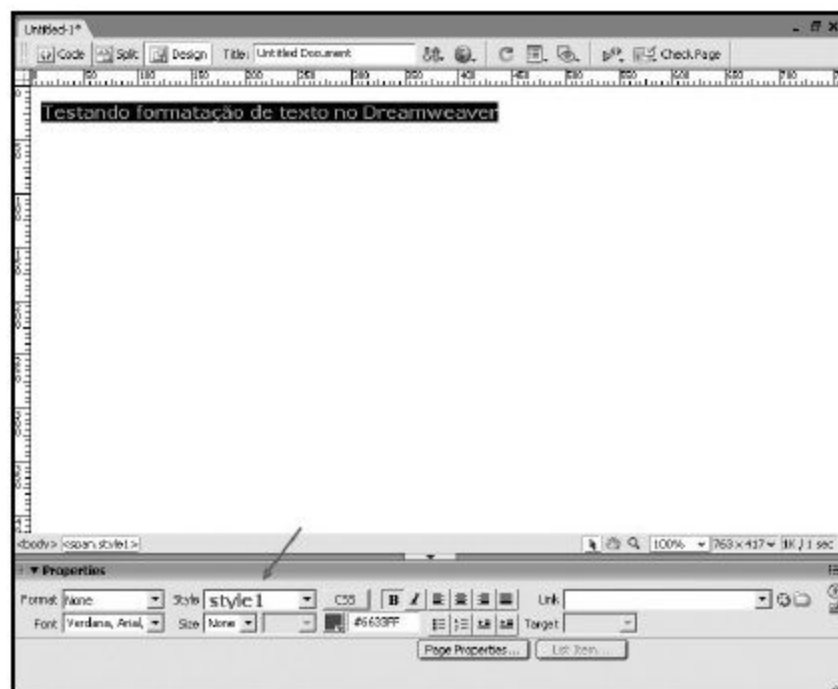


Figura 6.4: Criação de estilo.

Agora, altere o layout para o modo Code (clique no botão <dw_button_mode_code.jpg>) e observe o código gerado:

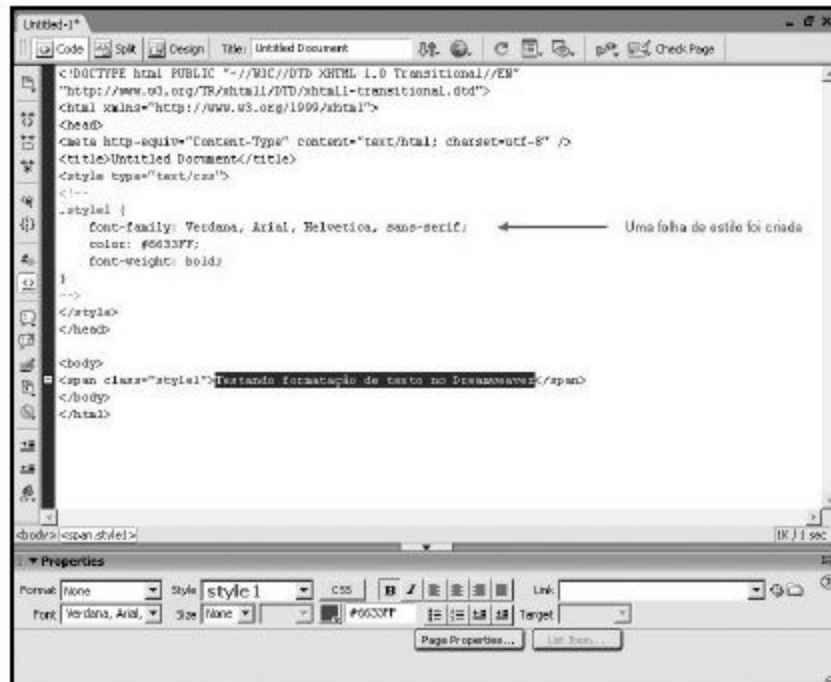


Figura 6.5: Código gerado.

Na figura anterior, note que uma CSS foi criada, devido à formatação feita no texto. Essa folha de estilo define uma classe `style1` que possui as propriedades de cor, fonte e estilo do texto. Na seção `<body>` da página, o texto é identificado como `class="style1"`, indicando que aquele texto deve ser formatado conforme define a folha de estilo.

Agora, insira outro texto abaixo do já escrito, modificando suas propriedades novamente. Observe que outro estilo (`style2`) é criado:

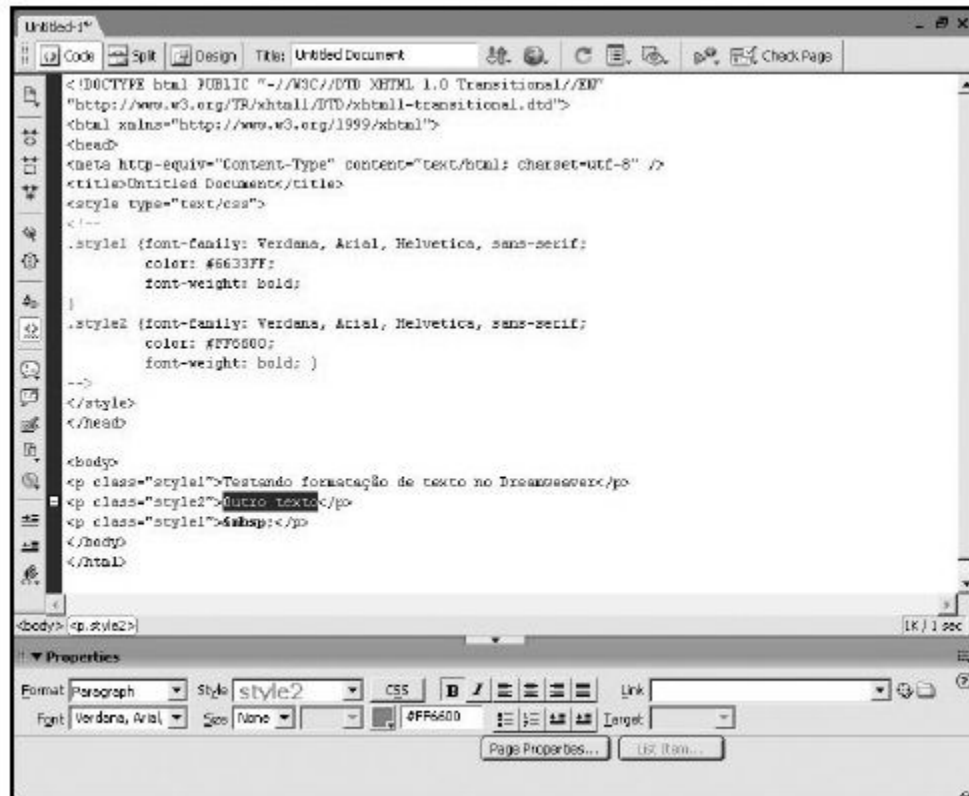


Figura 6.6: *Inserindo outro texto.*

Volte para o modo Code e verifique o código HTML gerado:

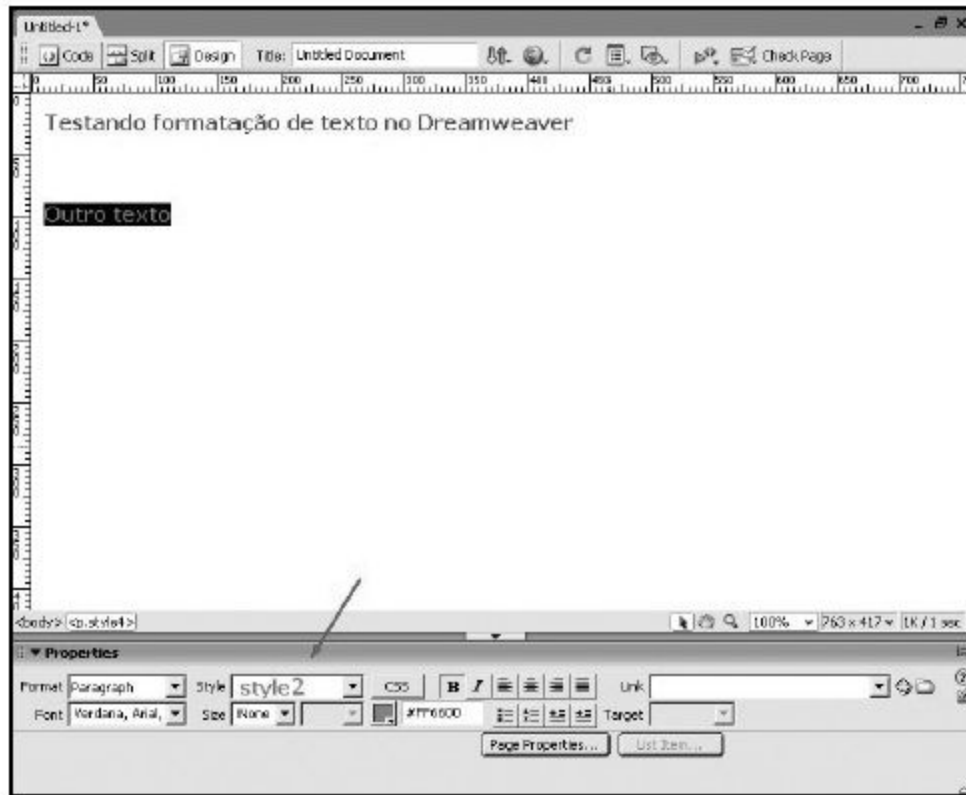


Figura 6.7: Código gerado.

Para cada texto inserido, uma nova classe é criada na folha de estilo com as propriedades de formatação.

Uma sugestão interessante a ser aplicada nesse caso seria otimizar o código gerado, ou seja, fazer com que exista uma “classe-pai” dessas duas classes que contenha atributos comuns entre elas. Nas classes style1 e style2 permaneceriam somente os atributos específicos; os atributos comuns seriam herdados da classe-pai.

Inserção de imagens

Agora, vamos verificar como o Dreamweaver se comporta em relação a arquivos de imagens. No mesmo documento em que os dois textos foram inseridos e formatados, localize, na barra de ferramentas superior, mais precisamente na aba **Common**, o botão para inserção de imagens:



Figura 6.8: Inserir imagem.

Uma nova janela é aberta para que você selecione uma imagem:

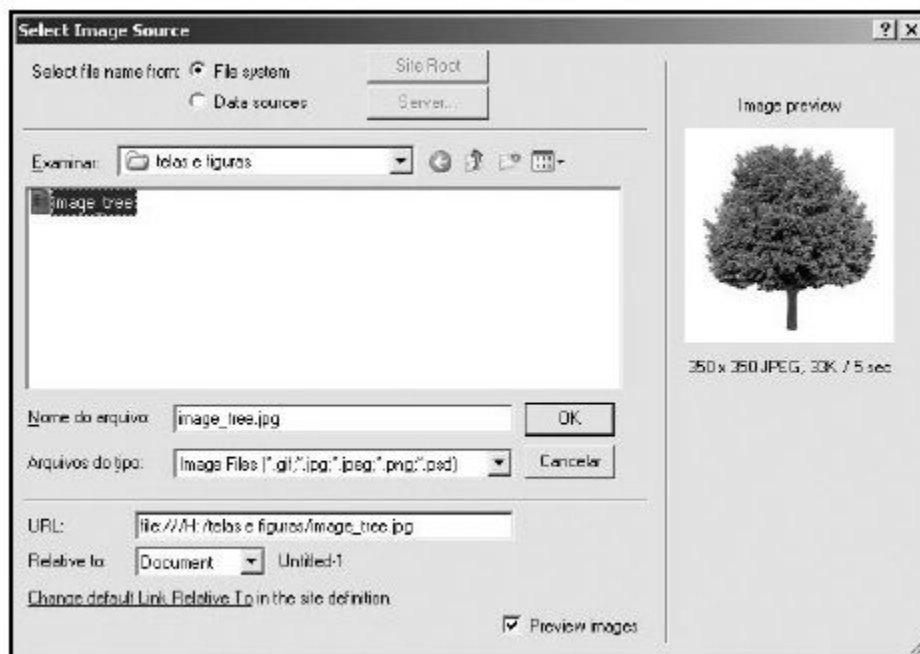


Figura 6.9: Selecionar imagem.

Outra janela será aberta para definir uma descrição para a imagem (atributo alt da tag), cujo nome aparecerá quando posicionamos o mouse sobre a imagem. No exemplo, o **Alternate text** recebe o texto *Árvore*:

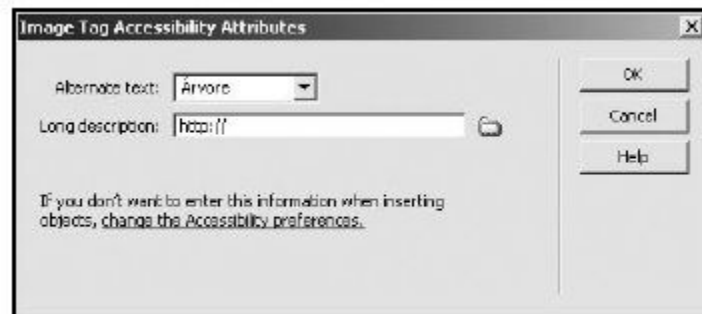


Figura 6.10: Definindo descrição da imagem.

A imagem é, então, inserida na página. Altere suas propriedades (alinhamento, borda etc.) e verifique como isso se refletiu no código HTML:



Figura 6.11: Imagem na página.

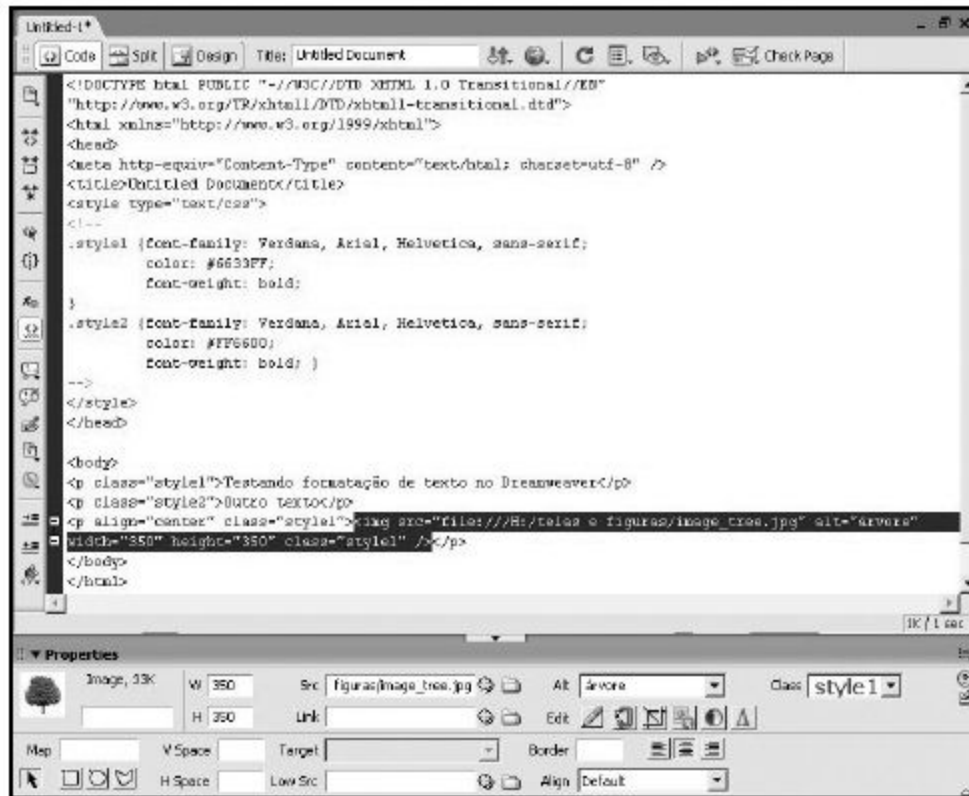


Figura 6.12: Código gerado.

Note que a imagem faz parte do estilo style1. Isso acontece, porque a imagem foi inserida em uma área na qual o estilo era esse. Observe a última imagem sobre formatação de texto: o código gerado mostra que, após a inserção do texto com style2, há uma linha em branco com style1; aí que a imagem foi inserida. Por ser uma imagem, não provocará nenhum impacto e o estilo poderia ser modificado para style2.

Veremos, a seguir, como esses estilos são reaproveitados com a criação de uma tabela.

Criação de tabela

De volta ao modo Design, posicione o cursor abaixo da imagem e insira uma tabela na página, clicando no botão indicado a seguir, na barra de ferramentas superior:

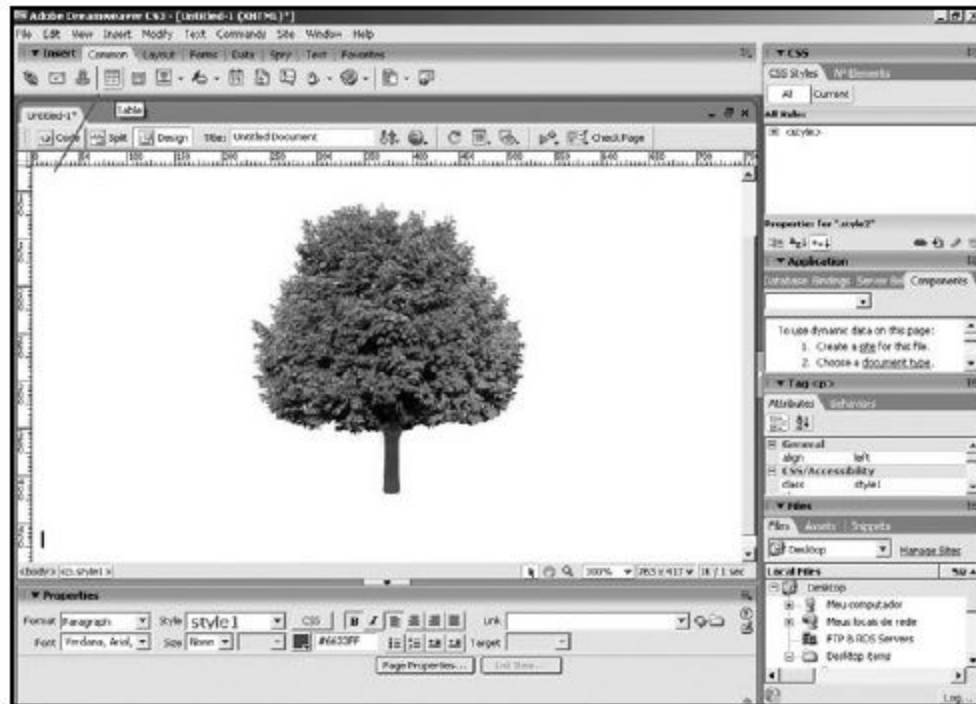


Figura 6.13: *Inserir tabela.*

Uma janela com as definições da tabela será exibida. Preencha as informações como na figura seguinte:



Figura 6.14: Definindo tabela.

Uma tabela será criada na página, segundo as definições anteriores:



Figura 6.15: Tabela criada.

Altere as propriedades da tabela, preenchendo suas células com texto. Note que não há estilo definido, e ela está com fonte Times New Roman:

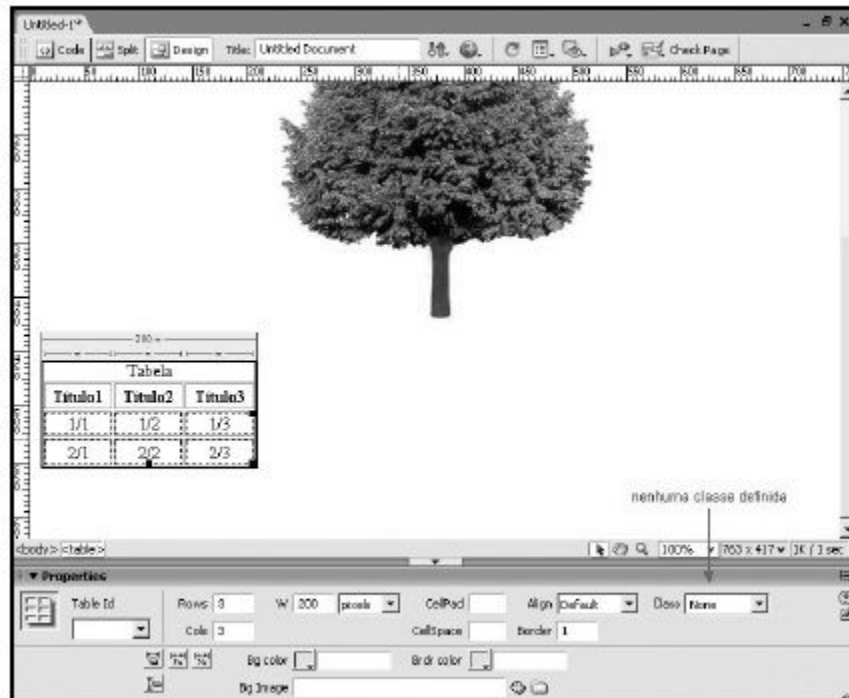


Figura 6.16: Modificando tabela.

Agora, defina um dos dois estilos existentes (style1 ou style2) para a tabela. Como os estilos já existentes referem-se às propriedades de textos, o conteúdo das células será alterado de acordo com esses estilos. Feito esse teste, observe também o código gerado no modo Code:

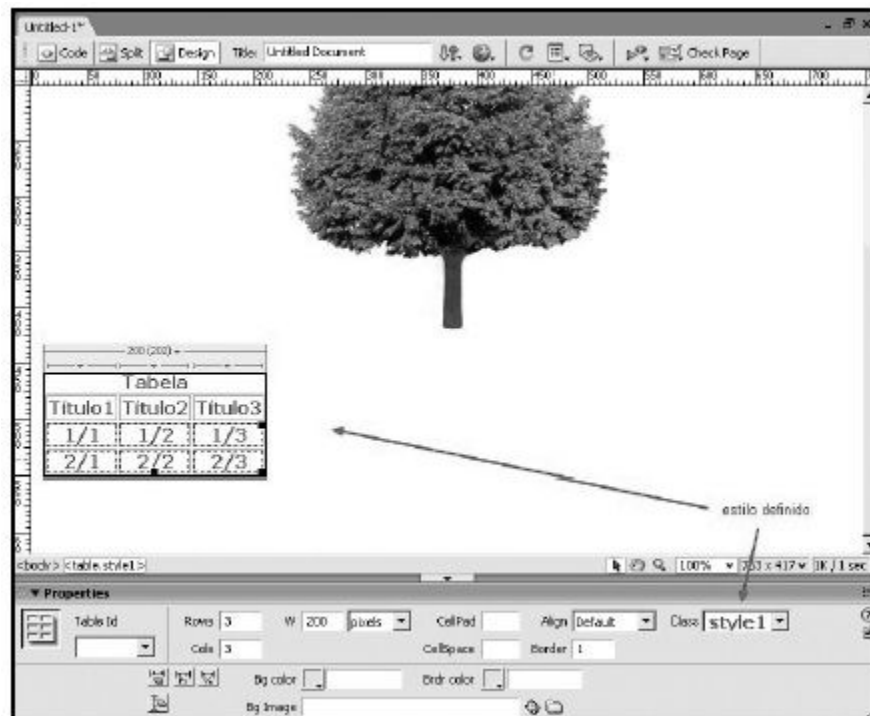


Figura 6.17: Tabela com estilo definido.

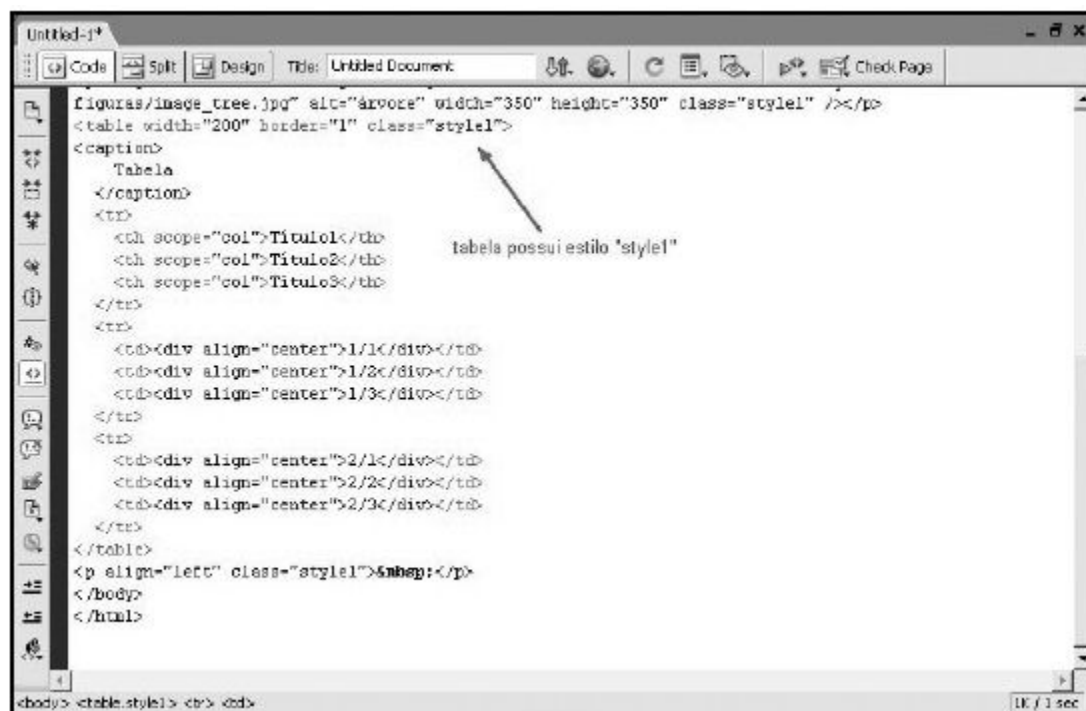


Figura 6.18: Código gerado.

Dessa forma, é possível reaproveitar e padronizar formatações, economizando linhas de código e garantindo formatações uniformes. No próximo item será mostrado como alterar essas CSS já existentes utilizando o Dreamweaver.

Alterando as folhas de estilo

Já vimos que o Dreamweaver cria estilos para cada elemento inserido em uma página, organizando-os em forma de folhas de estilo que permitem o reaproveitamento de estilos. Agora, veremos como ele organiza essas CSS (para esse arquivo recém-criado ou para páginas que já possuíam folhas de estilo) e como podemos alterá-las.

O primeiro passo é localizar o painel das CSS no grupo de painéis, posicionado na parte lateral direita da interface do programa. No caso de um arquivo CSS, será exibida uma lista dos elementos contidos. No caso de uma página HTML, serão listadas as folhas de estilo. Veja a seguir como o Dreamweaver organiza essas informações, dependendo do tipo de arquivo aberto:

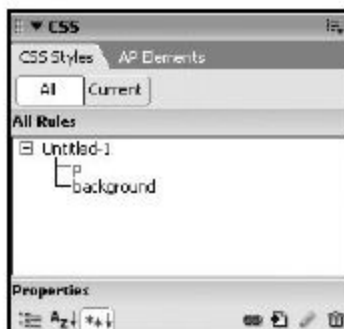


Figura 6.19: Painel das CSS com elementos.

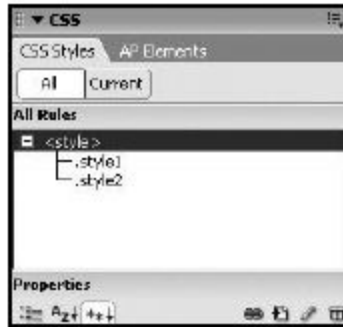


Figura 6.20: Painel das CSS com estilos.

Esse painel contém todas as folhas de estilo geradas ao longo da criação de uma página. Para acessar uma determinada folha de estilo, basta clicar duas vezes sobre ela. Uma janela será aberta, e nela é possível visualizar todos os atributos que podem ser formatados para aquele estilo:

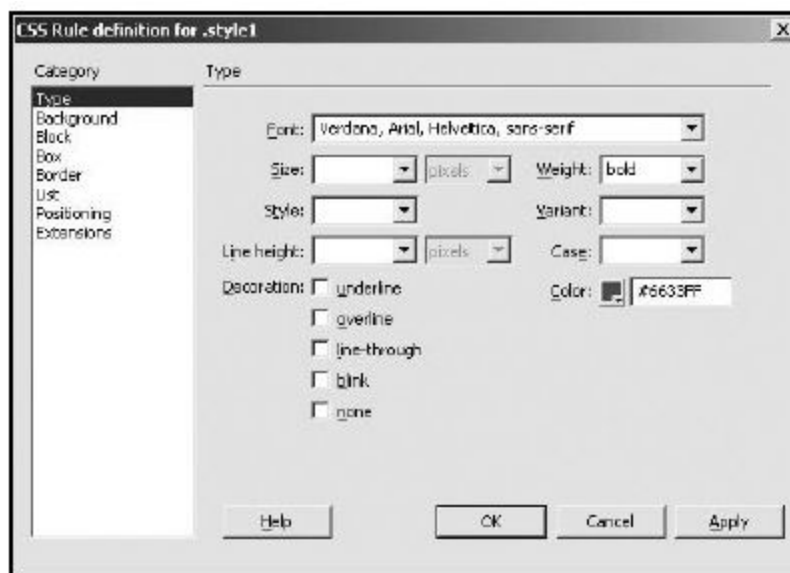


Figura 6.21: Atributos da folha de estilo.

Os atributos são subdivididos em categorias, como formatações de fonte, cor de fundo, bordas de tabelas e imagens, listas, posicionamento etc. Quando essas propriedades são modificadas, as alterações estendem-se a todos os objetos pertencentes à classe ligada àquele estilo.

Conclusão

Como foi afirmado inicialmente, as folhas de estilo surgiram como uma ótima solução no que diz respeito à organização e à construção de sites. Separando o conteúdo da formatação, torna-se muito mais fácil gerenciar sites compostos por várias páginas HTML. O processo de manutenção passa a ser muito mais prático e eficiente e o reaproveitamento de código torna mais rápida toda nova implementação. As vantagens não são vistas somente do lado das folhas de estilo, mas também no código HTML, que fica mais livre das formatações e, com isso, mais legível. Outra grande vantagem é que as folhas de estilo não apenas formatam informações, mas também incorporam recursos extras que o HTML não possui sozinho.

Mais do que apenas conhecer suas vantagens e aprender na prática como tudo isso é aplicado, considera-se muito importante ter consciência de que, como qualquer outra linguagem, as CSS devem seguir um padrão ao serem implementadas, garantindo portabilidade, acessibilidade e a garantia de que o código produzido seja de qualidade e boa performance.

Este livro foi elaborado com o objetivo de fornecer ao leitor uma base de informações para que ele tenha condições de desenvolver folhas de estilo rapidamente, mas sem se esquecer da importância de seguir os padrões mundiais das CSS. É importante prestar atenção à construção e consistência de todos os códigos. Isto significa, em outras palavras, ter consciência de que um site bem feito não se limita à boa aparência, mas trata, também, de como ele foi desenvolvido, permitindo que seu desenvolvimento seja contínuo, sem que isso interfira no conteúdo existente.

Apêndice 1

HTML básico

HTML Básico

HTML (Hypertext Markup Language) é uma das linguagens de programação mais utilizadas atualmente para a criação de *sites* e outras aplicações que utilizam os recursos da Web para divulgar produtos e serviços de maneira rápida e eficiente. Destaca-se entre as demais linguagens existentes no mercado pela sua ampla facilidade de manipular e estruturar páginas Web, definindo e separando por meio de marcações (chamadas tags) elementos como texto, título, imagem, tabela, lista e outros. Além disso, foi o primeiro código que surgiu para a programação de páginas Web e aplicações para Internet. Apesar da sua aparência sofisticada, as páginas produzidas em HTML não passam de documentos de texto simples, que podem ser produzidos usando somente um editor, como o Bloco de Notas do Windows. Então, que tal começar a testar alguns recursos e aprender a manipular os comandos desta linguagem, para produzir suas próprias páginas e trabalhos na Internet?

Um documento HTML é todo estruturado em setores, delimitados por elementos que chamamos de tags. As tags vêm, quase sempre, aos pares, havendo uma tag para indicar o início do setor e outra para indicar o final dele. As tags são indicadas entre os sinais de “<” e “>”, sendo que, para as tags de finalização de setor, coloca-se uma barra para a direita antes do nome da tag. O exemplo a seguir mostra as tags principais de um documento HTML:

<html>	Início do documento HTML.
<head>	Início do cabeçalho.
<title>	Início do título da página.
</title>	Fim do título da página.
</head>	Fim do cabeçalho.

<code><body></code>	Início corpo do documento.
<code></body></code>	Fim do corpo do documento.
<code></html></code>	Fim do documento HTML.

Abre-se o documento com a tag `<html>`. Logo em seguida abre-se o `<head>`, setor no qual podem ser colocados dados sobre a página, que serão usados em mecanismos de busca; e o título, que será colocado entre as tags `<title>` e `</title>`, tudo dentro do setor `<head></head>`. O título aparece na barra-título do browser do internauta. Entre `<body>` e `</body>` fica o conteúdo propriamente dito da página. Finalmente, fecha-se o documento com a tag de finalização `</html>`.

Muitas tags podem ter atributos, que fornecem valores adicionais para a formatação. Veja a tag `<body>`, por exemplo:

```
<body bgcolor="#FFFFFF" >
```

No caso, o atributo `bgcolor="#FFFFFF"` define a cor de fundo da página como branco (cujo código em notação hexadecimal é `#FFFFFF`).

Tags principais

A seguir, uma lista com os principais comandos HTML, que podem ser usados para formatar seu texto:

Formatação

 	Quebra de linha (não precisa ser finalizado).
<p>	Parágrafo.
	Negrito.
<u>	Sublinhado.
<i>	Itálico.
	Para enfatizar uma parte do texto.
	Determinar o tipo, cor e tamanho da fonte Ex: <fontface="Times" size= "4">.
<h1>	Padrão para títulos. Pode ir de <h1> a <h5>.
<hr>	Marca uma linha horizontal para dividir partes da página.
<div>	Tag de divisão lógica. Pode conter o atributo align. <p> e , por exemplo, devem estar inseridos

	dentro de <div>.
<basefont>	Usado no <head> para determinar a fonte-padrão da página.
<plaintext>	Serve para que a página mantenha a formatação do código, respeitando as quebras de linhas. O código que estiver dentro desta tag não será renderizado.
<pre>	O mesmo que o anterior, com a diferença de que o código continua sendo entendido pelo navegador.

Tabelas

<table> Indica uma tabela geral. Para que ela funcione, é necessário usar as tags de linhas e células, <tr>, <th> e <td>. A primeira serve para iniciar uma linha da tabela, enquanto as duas últimas iniciam células. <th> é usada em cabeçalhos e <td> nas demais células. Todas devem ser finalizadas. Veja um exemplo:

```
<table width=200
bgcolor="#CC4400"
border=1 cellpadding=10
cellspacing=5>
<tr><td>Célula da primeira
linha</td></tr>
<tr><td>Célula da segunda
linha</td></tr>
</table>
```

Esse código cria uma tabela com 200 pixels de largura, fundo verde, borda com a espessura de 1 pixel, espaçamento entre borda e conteúdo de 10 pixels e espaçamento entre as células de 5 pixels. A tabela tem duas linhas, cada uma contendo uma frase:

Célula da primeira linha
Célula da segunda linha

Links e imagens

`<a>` Tag usada para associar um link a um texto ou imagem. Seu principal atributo é **href**, no qual se deve informar o endereço alvo do link. O exemplo a seguir faz com que a palavra “Digerati” seja um link para o site <http://www.digerati.com.br>:

```
<a href="http://
www.digerati.com.br">Digerati</
a>
```

`` Tag usada para procurar uma imagem, com a expressão **src**, atribuída a um endereço. O exemplo abaixo indica que, naquele ponto da página, deve ser mostrada a imagem que se encontra no endereço indicado no atributo **src**:

```

```

1. Abra o Bloco de Notas e digite:

```
<html>
<head><title>Teste de
link</title>
</head>
<body>
<a href="teste2.html">Link
para o arquivo teste2.html</
a>
</body>
</html>
```

2. Salve o arquivo como *teste.html*. Abra um novo arquivo no Bloco de Notas e digite:

```
<html>
<head><title>Teste de link</
title>
```

```
</head>
<body>
Cheguei aqui a partir do
arquivo teste.html!
</body>
</html>
```

3. Salve esse novo arquivo no mesmo diretório em que você salvou o arquivo *teste.html*. Agora, rode o *teste.html* no seu browser. Clique no link. Você deverá ser levado ao arquivo *teste2.html*.

4. Agora, vamos fazer uma experiência com links. Mova o arquivo *teste2.html* para o diretório acima daquele em que ele se encontra. Rode o *teste.html* no browser e clique no link: ele não vai funcionar.

5. Vamos corrigir isso. Abra o *teste.html* com o Bloco de Notas e, no lugar de ``, digite ``. Agora vai funcionar. Por quê? Porque os dois pontos seguidos indicam “uma pasta acima”. Assim, o programa que está lendo o código (no caso, o navegador) entende: “suba para o nível acima do diretório atual e procure pelo arquivo *teste2.html*”. Isto é o que chamamos de “endereço relativo” – a localização do arquivo alvo é dada em relação à localização do arquivo atual.

Um endereço absoluto seria, por exemplo, *C:\Meus Documentos\Site\teste2.html*. A desvantagem deste método é óbvia: na hora em que você for colocar seus arquivos em um servidor na Internet, por exemplo, o browser dificilmente vai encontrar este arquivo.

Formulários

`<form>` Tag para a criação de formulários. Deve ter o atributo `method`, que pode ser `post` ou `get`, além de `action`, que aponta para o arquivo ou endereço para o qual devem ser enviados os dados inseridos pelo usuário. Há vários tipos de formulário: de múltipla opção, de opção única, de inserção de texto, entre outros. O tipo de formulário deve ser informado no atributo `type` da tag `<input>`. Você também pode inserir, no formulário, um botão que servirá, por exemplo, para efetuar a ação de enviar os dados para o endereço especificado. Veja um exemplo no código abaixo:

```
<form method="post"
action="http://
```

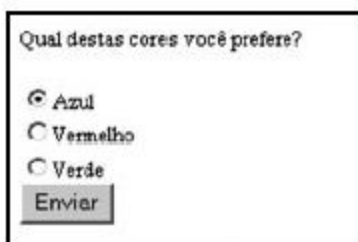


```
www.seudomínio.com/  
arquivo.extensão">  
<input type="text"  
name="nome _ da _ variavel"  
size="20" maxlength="20">  
<input type="submit"  
value="Enviar">  
</form>
```

Veja este outro exemplo:

```
<form  
action="resposta.php">  
<input type="radio" checked  
name="azul"  
value="azul">Azul<br>  
<input type="radio"  
name="vermelho" value="vermelho">Vermelho  
<br>  
<input type="radio"  
name="verde"  
value="verde">Verde<br>  
<input type="button"  
name="enviar"  
value="Enviar"></form>
```

Os elementos do tipo radio são como botões em série, cada qual representando uma opção, das quais só uma pode ser escolhida. O name serve como referência, e o value é o valor que vai ser jogado no destino especificado em action quando o usuário clicar no botão **Enviar**. Veja como fica esse código:

A imagem mostra uma caixa de formulário com o título "Qual destas cores você prefere?". Dentro da caixa, há três opções de radio button: "Azul" (selecionada), "Vermelho" e "Verde". Abaixo das opções, há um botão "Enviar".

Qual destas cores você prefere?

☒ Azul

☐ Vermelho

☐ Verde

Enviar

Comentários

Todo webdesigner e programador que se preze faz documentação em seus códigos. Pode parecer chato no começo, mas quando você tiver de

alterar um código depois de um mês, por exemplo, vai agradecer a si mesmo o fato de ter deixado explicações claras sobre o que faz cada bloco de código. Para isto é que existe, em toda linguagem, especificações para a inserção de comentários no código, que não interferem em seu funcionamento, ficando apenas como lembretes para o próprio autor. No HTML, a sintaxe para comentários tem duas formas:

- Comentários de uma linha:

```
<!--Isto é um comentário-->
```

- Comentários em várias linhas:

```
<!-- //Aqui começa o comentário  
//Mais uma linha de comentário  
//Última linha de comentário
```



Your gateway to knowledge and culture. Accessible for everyone.



z-library.se

singlelogin.re

go-to-zlibrary.se

single-login.ru



[Official Telegram channel](#)



[Z-Access](#)



<https://wikipedia.org/wiki/Z-Library>