IBPINET
The Internet School
A·S·I·T
ADVANCED SCHOOL OF INTERNET TECHNOLOGY
PÓS-GRADUAÇÃO

web & design HTML avançado

Helder da Rocha
helder@ibpinet.net

WDHT-03-2000/04 A4

Web design
&
HTML
avançado

Helder da Rocha

Rev.: WDHT-03-2000/04 A4 (ASIT)

Capa, editoração eletrônica e revisão: o autor.

Código desta edição: **WDHT-03-2000/04** (4a. edição)
Formato: A4 – apostila (numeração de páginas por capítulo).
Responsável por esta tiragem: IBPINET São Paulo
Tiragem máxima desta edição: 80 cópias (turmas ASIT iniciadas até 07/2000).

R672s Rocha, Helder Lima Santos da, 1968-

"Web design e HTML avançado". 4a. edição. Coleção de textos independentes sobre Web design e HTML.
___ p. A4. Acompanha disquete ou CD de 3 ¼" ou *Web site* com código e exemplos.
Reprodução: IBPINET – São Paulo, SP, Março de 2000.

1. Web design – criação e apresentação de conteúdo interativo para a World Wide Web. 2. HTML – HyperText Markup Language (linguagem usada para criar páginas). 3. CSS – Cascading Style Sheets (linguagem usada para controlar forma). 4. World Wide Web (serviço de recuperação de informações da Internet) – Programação. 5. Internet (rede de computadores, meio de comunicação) – Programação. I. Título: Web design e HTML avançado – Web design – HTML – Além do HTML – Folhas de Estilo.

CDD 005.133

Conteúdo

2. HTML

Prefácio

Este material didático contém quatro textos selecionados e organizados especialmente para o curso "Web Design e HTML Avançado" que faz parte do currículo do ASIT. A finalidade deste material é servir de apoio ao curso como fonte de informações teóricas e exercícios.

Os textos colecionados nesta apostila fazem parte de outras apostilas e livros que tratam de assuntos relacionados ao tema do curso, mas nem todo o material que está presente aqui será abordado no curso. Alguns assuntos são pré-requisitos para este curso mas foram incluídos nesta apostila para que os alunos que queiram revisar os pré-requisitos antes do início do curso possam fazê-lo, e como referência. Portanto, é o *programa do curso*, e não o índice de tópicos deste livro, a referência absoluta quanto ao programa a ser cumprido. O programa detalhado poderá ser encontrado no endereço http://www.ibpinet.net/helder/asit/.

O que você já deve saber

Um dos pré-requisitos do ASIT e que é requerido para este curso é o conhecimento de HTML. Você deverá saber criar uma página usando HTML e saber identificar, no código-fonte, as principais estruturas de uma página como títulos, células de tabelas, etc.

Não é necessário que você só desenvolva em HTML, mas, se você só cria páginas usando ferramentas gráficas, é possível que quase não veja código HTML na sua frente. Não há tempo para se ensinar HTML neste curso, portanto, se você não conhece algo mais que o básico de HTML, como listas e principalmente tabelas, procure aprender um pouco lendo o livro 2 (HTML) e fazendo os exercícios. Para avaliar seus conhecimentos, tente fazer os testes no final do livro 2. As respostas serão divulgadas em sala de aula.

Descrição do conteúdo

Esta apostila consiste de quatro livros. O primeiro, chamado de Web Design, apresenta uma definição do que seja essa disciplina, discute aspectos importantes do desenvolvimento de Web sites, define termos essenciais que serão usados ao longo do curso e contém informações técnicas sobre o funcionamento dos browsers e servidores Web. Como complemento ao material do primeiro livro, há uma apresentação (slides) sobre navegação, acessibilidade e projeto de um site.

O segundo livro trata exclusivamente da tecnologia HTML. Ele não será usado neste curso. Apresenta e mostra como usar os principais descritores HTML e seus atributos mais importantes, destacando o uso de tabelas, vínculos e imagens com diversos exemplos e ilustrações.

O terceiro livro apresenta aspectos avançados do HTML como o tratamento de imagens, o uso de tabelas para organizar o layout de páginas, imagens para espaçamento, janelas com múltiplos documentos (*frames*), imagens mapeadas, áudio, vídeo, plug-ins e applets Java. É muito assunto para conhecer em detalhes. Alguns tópicos (como applets), serão vistos em outros cursos. Apenas o layout com tabelas e *frames* são abordados neste curso.

O quarto e último livro introduz a linguagem CSS, usada para criar sites controlados por folhas de estilo. Com essa tecnologia é possível ter controle quase absoluto sobre a aparência de um site, com vantagens que facilitam a manutenção, tornam o site mais leve e acessível e permite o suporte a tecnologias interativas como o DHTML (Dynamic HTML).

Mídia eletrônica e atualizações

Todos os exemplos, exercícios resolvidos e soluções de alguns exercícios propostos estão ou distribuídos em um disquete que acompanha este livro ou disponíveis na Internet. Para esta edição, atualizações estarão disponíveis em http://www.ibpinet.net/helder/asit/ assim como correções e possíveis exemplos adicionais.

Críticas e sugestões

Os livros contidos nesta edição estão sempre sendo revisados, atualizados e ampliados periodicamente e cada vez que são utilizados em cursos. Cuidados foram tomados para garantir a apresentação dos assuntos de forma clara, didática e precisa, mas eventualmente podem escapar erros, imprecisões e trechos de pouca clareza. Sugestões, críticas e correções são sempre bem vindas e podem ser endereçadas por e-mail a `hlsr@uol.com.br` ou `helder@ibpinet.net`. Sua opinião é muito importante e contribuirá para que futuras edições possam ser cada vez melhores.

Helder L. S. da Rocha
São Paulo, 17 de fevereiro de 2000.

1 Web design

1.1. O que é Web design?

Web design é, em inglês, o nome da arte praticada pelos seres artrópodes da ordem *Aracneae*, ou aracnídeos, mais popularmente conhecidos como as *aranhas*.

Mas isto era há 10 anos atrás. Hoje, Web design é a arte de tecer uma outra teia, muito mais desorganizada e caótica, e que não é governada por nenhuma aranha. Web design é a concepção e projeto da interface interativa do serviço Web, formado por "páginas".

Alguns anos depois das aranhas, a administração da Web cabia à programadores que precisavam codificar toda a informação usando uma linguagem: o HTML, e depois armazená-las em uma área especial de uma máquina Unix, onde rodava um servidor Web – programa que permitia o acesso remoto às informações das páginas. Hoje, a arte de criar páginas Web continua a exigir mais e mais conhecimentos de programação ... visual! Saber HTML hoje é menos importante que ter noções de design.

O design de páginas para a Web hoje se assemelha mais à editoração eletrônica que à programação. Os caminhos do Web design, porém, são bem diferentes daqueles seguidos na criação para mídia impressa. O Web designer deve conhecer não só as possibilidades do meio onde publicará a sua informação, como suas limitações. A arte do Web design consiste em aproveitar ao máximo os recursos oferecidos pela Web, garantindo a melhor apresentação, navegabilidade e interatividade de um Web site. A idéia é atrair os visitantes e estimulá-los a voltar outras vezes. Às vezes é necessário sacrificar a qualidade da apresentação ou deixar de usar algum recurso útil devido a lentidão da rede ou incompatibilidades de browsers. É importante que o Web designer aprenda a traçar uma linha de equilíbrio entre os impedimentos tecnológicos e as possibilidades criativas desta nova mídia, para tirar o maior proveito do seu potencial.

O bom Web design começa fora do computador. Planejar a estrutura da teia e sua identidade com base nos objetivos à que se destina (comercio, informação, promoção, intranet, captura de insetos) antes de iniciar a implementação, preserva o conceito por trás do design da interface e diminui a limitação da tecnologia disponível.

Como já mencionamos acima, hoje é possível desenvolver todo um site sem escrever uma só linha de HTML. Obviamente, omitimos alguns detalhes nesta afirmação (senão, por que estaríamos ensinando HTML avançado?). O estado atual das ferramentas de *Web publishing* limita muito o que se pode fazer sem programar. HTML é um padrão em constante desenvolvimento e as ferramentas levam no mínimo seis meses para incorporar as extensões e novos padrões aos seus produtos. Para criar páginas com os mais modernos recursos interativos, com uma apresentação sofisticada, acessível e fácil de manter quase sempre ainda é preciso escrever alguma coisa em HTML. É a única forma de não se tornar escravo das ferramentas.

1.1.1. A internet e a world wide web

Para entender como criar e publicar páginas para a Web, precisamos primeiro saber um pouco como funciona a Teia, e a Internet, que está por trás de tudo. Nos parágrafos seguintes relembramos um pouco a sua história.

A Internet é uma rede antiga. Tem mais de 30 anos de idade. Uma das pessoas que teve um papel decisivo na sua criação foi Fidel Castro. Os Estados Unidos tinham passado por uma possibilidade real de ataque nuclear depois da instalação de mísseis russos em Cuba e a interligação das bases militares em rede foi uma estratégia militar para proteger a comunicação em caso de ataque nuclear. Ela ligava máquinas diferentes entre si através de linhas redundantes de maneira que, mesmo que uma ou várias bases fossem reduzidas a pó, as outras estações ainda conseguiriam se comunicar entre si. Então, com medo que Fidel e seus charutos nucleares paralisassem o sistema de defesa do país foi criada a rede ARPANET (*Advanced Research Projects Agency Network*). A rede interligou vários computadores em algumas universidades e centros de pesquisa envolvidos com projetos militares. Na época, computador ainda era coisa rara. Computador em rede, mais ainda.

Os primeiros usuários (cientistas) usavam a rede para trocar mensagens de correio eletrônico e se conectar remotamente em computadores distantes. O serviço tornou-se tão útil, que as universidades envolvidas começaram a conectar seus departamentos, mesmo os que não tinham a ver com o projeto. Na década de 80, foi a vez da NSFNET, a rede nacional de pesquisa dos Estados Unidos, que conectou os cinco grandes centros regionais de supercomputação e passou também a fazer parte da ARPANET. A NSFNET se tornou espinha dorsal das duas redes, depois chamada de Internet.

A NSFNET levou consigo uma legião de pequenas redes que a ela estavam conectadas. E assim a Internet foi crescendo e crescendo, sem nenhuma organização central. Qualquer centro que achasse seus serviços convenientes podia se amarrar ao nó mais próximo simplesmente pagando as despesas de uma linha dedicada de dados. Em pouco tempo, a Internet já interligava os maiores centros de pesquisa do mundo.

A expansão foi trazendo novos serviços e outras redes de pesquisa, como a Usenet, a Bitnet, EARN e redes de BBSs. O crescimento já era impressionante, mas a popularidade ainda era limitada. Grande parte do tráfego era acadêmico. Operar um sistema de correio eletrônico em geral significava saber operar um jurássico terminal IBM ou máquina Unix, e ter alguma familiaridade com suas interfaces hostis. Para transferir um arquivo era necessário saber usar um programa especial (FTP), mandar comandos para ele e entender os hieroglifos que ele retornava.

O maior problema era a desorganização generalizada da Internet. A Internet era um tremendo caos. Você sabia que poderia transferir um arquivo de qualquer lugar do mundo. O problema era saber se ele existia e onde estava!

A Internet começou a crescer de forma explosiva em 1992, com o surgimento da *World Wide Web*, um projeto do laboratório CERN em Genebra (eles não estudavam aranhas, mas

partículas subatômicas). A Web conseguiu finalmente organizar um pouco as informações da Internet através do hipertexto. No ano seguinte, chegou o *Mosaic* para Windows – programa desenvolvido por um grupo de estagiários do NCSA (*National Center for Supercomputing Applications*) da Universidade de Illinois a partir de uma versão anterior para o sistema Unix.

O Mosaic, oferecendo pela primeira vez uma interface gráfica multimídia para a Web, trouxe a grande massa de usuários de PC e Macintosh para dentro da rede. Os provedores de acesso e informação comerciais se multiplicaram, oferecendo às pessoas comuns o mesmo acesso que antes só tinham as grandes organizações e o meio acadêmico.

Os criadores do Mosaic pouco depois de deixar a universidade criaram uma empresa: a Netscape, que foi provavelmente a empresa que teve a maior influência nos rumos seguidos pela Web na sua evolução até os dias de hoje.

1.1.2. A web como um meio de comunicação

Apesar de ter surgido como um serviço de uma rede de computadores, a Web é hoje muito mais que isto e para explorá-la, nem computador é necessário mais. Tecnologias recentes como o *Network Computer* (NC) - que é um simples terminal para a Web, browsers que vêm embutidos em telefones celulares, e a rede *WebTV* mostram que a *World Wide Web* está destinada a preencher todos os espaços da mídia de difusão, não se limitando àqueles que possuem um computador.

A Web, dessa forma, possui um potencial inigualável na história das telecomunicações. É capaz de servir de interface à todos os serviços da Internet e ainda aos tradicionais serviços de voz (telefone), televisão, rádio e mídias impressas. Pode integrar tudo e interagir com tudo. Ainda estamos dando os primeiros passos neste novo terreno, e cada dia nos traz mais certeza de que trata-se de um caminho sem volta.

Diferente dos meios tradicionais de comunicação de massa, a *World Wide Web* é uma mídia democrática. O usuário não precisa possuir uma estação difusora, uma concessão, uma gráfica ou qualquer coisa do tipo para poder publicar sua informação e influenciar sua audiência. Todos podem receber as informações de todos. Qualquer um pode prover informação. O poder da informação está nas mãos de todos os que puderem ter um espaço na Teia, e não mais apenas com as aranhas ou com aqueles que possuem os meios de difusão tradicionais.

1.2. A World Wide Web e a Internet

A *World Wide Web* é o nome do mais popular dos *serviços* da Internet. Por esse motivo, é freqüentemente confundida com a própria Internet. Mas Web e Internet não são a mesma coisa, e precisamos conhecer bem a diferença entre as duas antes que possamos começar a desenvolver páginas e aplicações para a Web.

Internet é o nome dado ao conjunto de computadores, provedores de acesso, satélites, cabos e serviços que formam uma *rede mundial* baseada em uma coleção de *protocolos de comunicação* conhecidos como TCP/IP.

1.2.1. Protocolos

É através de protocolos de comunicação que um computador pode se comunicar com outro através de uma linha telefônica ou placa de rede sem que o usuário precise se preocupar em saber qual o meio físico que está sendo utilizado. O sistema Windows possui protocolos que permitem facilmente interligar computadores rodando Windows entre si. Os mesmos protocolos podem não servir para fazer com que uma máquina Windows se comunique com uma máquina Unix ou Macintosh, pois essas máquinas possuem arquiteturas diferentes. TCP/IP é uma suite de protocolos padrão que foi adotado como "língua oficial" da Internet. Para fazer parte da Internet, um computador precisa saber se comunicar em TCP/IP. Todas as operações de rede são traduzidas para TCP/IP antes que possam funcionar na Internet.

1.2.2. Endereços Internet

Um dos protocolos mais importantes da suite TCP/IP é o protocolo de rede IP - *Internet Protocol*. Ele define a *forma de endereçamento* que permite a localização de um computador na Internet, através de um conjunto de dígitos chamado de *endereço IP*. Qualquer máquina acessível através da Internet tem um endereço IP *exclusivo*. Esse endereço pode ser *temporário* ou *permanente*. Quando você se conecta a um provedor via linha telefônica, ele atribui um número IP temporário à sua máquina que permitirá que ela faça parte da Internet enquanto durar a sua sessão no provedor. Só assim é possível receber informações em um browser ou enviar e-mail. Computadores que hospedam páginas Web e que oferecem outros serviços pela Internet precisam de um endereço IP *fixo*, para que você possa localizá-los a qualquer hora. Por exemplo, 200.231.191.10 é o endereço IP da máquina onde está localizado o servidor Web do IBPINET em São Paulo. Você pode localizá-lo digitando http://200.231.191.10/ no campo de endereços do seu navegador.

1.2.3. Serviço de Nomes (DNS – Domain Name Service)

Embora cada computador seja identificado de forma exclusiva através de um endereço IP, não é dessa forma que costumamos localizá-los na Internet. Um dos serviços fundamentais ao funcionamento da Internet é o *serviço de nomes de domínio*. Esse serviço é oferecido por várias máquinas espalhadas pela Internet e que guardam tabelas que associam o *nome* de uma máquina ou de uma rede a um endereço IP. Quando você digita o nome de uma máquina no seu browser (por exemplo, *www.ibpinet.net*), o browser primeiro tenta localizá-la consultando uma outra máquina (cujo endereço IP o browser já conhece) que oferece o serviço de nomes. Essa máquina consulta outros serviços de nomes espalhados pela Internet e em pouco tempo

devolve o endereço IP correspondente ao nome solicitado (*www.ibpinet.net* devolverá 200.231.191.10).

Se o sistema de nomes falhar, o browser não conseguirá o número IP que precisa e assim não localizará a máquina correspondente (mesmo que ela não esteja fora do ar).

1.2.4. Portas e serviços da Internet

A Internet existe há mais de três décadas. Na maior parte desse período ela era restrita aos meios acadêmicos e militares e oferecia poucos serviços. Os principais *serviços* utilizados na rede eram a *transferência de arquivos* entre computadores (usando aplicações que se comunicavam através do protocolo FTP - *File Transfer Protocol*), o *correio eletrônico* e a *emulação de terminal*, que permitia o acesso a computadores remotos. Esses serviços eram oferecidos em algumas máquinas onde rodavam programas servidores, permanentemente no ar aguardando a conexão de um cliente em uma de suas *portas de comunicação*.

Uma mesma máquina pode oferecer vários serviços, desde que em portas diferentes. Imagine que o endereço IP de uma máquina seja como o endereço de um prédio de escritórios. Localizando o prédio, você procura por um determinado serviço que é prestado por uma empresa. Pode haver várias empresas no prédio. Cada uma tem uma sala identificada por um número. O número da sala é análogo à porta de serviços de uma máquina. Para facilitar a vida dos clientes, várias portas, identificadas por um número, foram padronizadas, ou seja, em computadores diferentes, você geralmente encontra os mesmos serviços localizados em portas com os mesmos números.

Para ter acesso a um serviço é preciso ter um cliente apropriado que saiba conversar na língua (protocolo) do servidor (programa que oferece o serviço na porta buscada pelo cliente). Como as portas são padronizadas, um cliente muitas vezes só precisa saber o nome ou endereço IP da máquina que tem determinado serviço, pois o número da porta ele supõe que seja o número padrão.

Se você usa o *Internet Explorer* para ter acesso ao site do IBPINET ou o *Outlook Express* para ler seu e-mail no IBPINET, é possível que você esteja se conectando à mesma máquina. O *Outlook Express* se conectará à porta 110 para verificar suas mensagens. Na hora de enviar, utilizará os serviços da porta 25. O *Internet Explorer* buscará a *home page* do IBPINET na porta 80, mas se você decidir fazer compras e utilizar o servidor seguro do IBPINET, se o browser utilizará a porta 443.

1.2.5. A Web entre os serviços da Internet

O serviço de emulação de terminal remoto - *Telnet*, requer que o usuário conheça o sistema remoto, tenha permissão de acesso e saiba utilizá-lo. O acesso é orientado a caracter e pode ser feito em MS-DOS. No início da popularização da Internet, vários serviços eram oferecidos apenas via *Telnet*, que funcionava como uma espécie de cliente universal. O usuário, ligado à Internet, poderia pesquisar o banco de dados da Nasa, bater papo com usuários

remotos e pesquisar repositórios de informações pelo mundo afora. A World Wide Web surgiu inicialmente como mais um desses serviços, que poderia ser utilizado através de um cliente Telnet apontando para o endereço info.cern.ch. O serviço ainda existe, por razões históricas. É possível acessá-lo via linha de comando no DOS ou Unix usando:

```
telnet info.cern.ch
```

É um acesso orientado a caracter e na época concorria com outros serviços de informações mais populares como o *WAIS*, o *Gopher* e o *Archie*, que também tentavam organizar as informações da Internet.

O acesso direto ao servidor, usando um cliente conectado à sua porta de comunicação é sempre mais rápido e eficiente que o acesso via Telnet. Os primeiros clientes Web eram orientados a caracter (não exibiam fontes nem imagens) e rodavam apenas em ambientes Unix. Com o surgimento do *X-Mosaic*, o primeiro browser gráfico rodando em ambiente *X-Window* (Unix), a Web começou a se tornar popular (pelo menos entre os acadêmicos). O crescimento explosivo da Internet aconteceu quando finalmente o Mosaic passou a ser acessível ao usuário comum não-acadêmico através do *Win-Mosaic* e posteriormente do *Mac-Mosaic*. O Mosaic, desenvolvido por Marc Andreesen quando estagiava no NCSA (*National Center for Supercomputing Applications*) abriu caminho para vários outros clientes Web, hoje chamados de browsers, e que passaram a ser o primeiro (e às vezes único) contato de muitos usuários com a Internet. O próprio Andreesen, ao deixar o NCSA, fundou a *Netscape Communications*, que dominou a Web por vários anos.

Os browsers de hoje não são mais apenas clientes Web. Eles lêem páginas locais, enviam e-mail, permitem que o usuário leia grupos de notícias, e-mail, execute aplicações locais, acesse aplicações remotas e diversos outros serviços da Internet. O browser moderno é um cliente universal para toda a Internet, embora esse acesso ocorra através da World Wide Web.

1.3. A plataforma Web

A World Wide Web é um serviço TCP/IP baseado no *protocolo de nível de aplicação* HTTP (*HyperText Transfer Protocol*) – *Protocolo de Transferência de Hipertexto*. A plataforma Web é o meio virtual formado pelos servidores HTTP (servidores Web que mantém sites), clientes HTTP (browsers) e protocolo HTTP (a língua comum entre o cliente e o servidor).

1.3.1. Hipertexto

Hipertexto é uma forma não linear de publicação de informações onde palavras que aparecem no decorrer do texto podem levar a outras seções de um documento, outros documentos ou até outros sistemas de informação, fugindo da estrutura linear original de um texto simples. O hipertexto baseia-se em ligações entre dois pontos chamados de *âncoras*. As ligações entre as âncoras são chamadas de *vínculos* (*links*). Vínculos de hipertexto são

implementados em textos publicados na Web usando uma linguagem declarativa chamada *HTML - HyperText Markup Language*.

1.3.2. HTML

HTML é usada para *marcar* um arquivo de *texto simples* (texto simples é texto sem formatação alguma, visualizável em *qualquer* editor de textos). Se um arquivo de texto simples receber uma extensão de nome de arquivo ".html" ou ".htm", um navegador como o Internet Explorer irá tentar interpretá-lo como HTML. Dentro do texto, pode-se definir descritores (ou comandos HTML) entre os símbolos "<" e ">":

```
<h1>Arquivo de texto</h1>
<p>Este é o <i>primeiro</i> parágrafo.</p>
```

Os descritores só serão visíveis quando o arquivo for visualizado em um editor de textos (como o Bloco de Notas do Windows). Ao ser visualizado em um programa capaz de entender HTML, apenas o texto aparece, com uma aparência determinada pelos descritores:

Arquivo de texto

Este é o *primeiro* parágrafo.

O texto com marcadores é chamado *código-fonte HTML*. O código-fonte é usado para produzir a página visualizada o browser que é chamada de *página HTML* ou *página Web*.

O browser, por ser capaz de exibir diversos tipos de informação, depende totalmente da *extensão do arquivo* para saber o que fazer com ele. Se a extensão ".htm" ou ".html" não estiver presente ou se o arquivo tiver a extensão ".txt", o browser exibirá o código-fonte.

Além da formatação da página, o HTML é responsável também pela inclusão de imagens e definição dos links que permitem a navegação em hipertexto.

1.3.3. Servidor HTTP

O serviço HTTP funciona de forma semelhante ao serviço *FTP - File Transfer Protocol* (protocolo de comunicação usado na Web para operações de transferência de arquivos). Ambos oferecem aos seus clientes um *sistema de arquivos virtual* onde podem localizar *recursos* (arquivos, programas, etc.) e transferi-los de um computador para outro. O sistema virtual pode ter uma *hierarquia* própria e totalmente diferente do *sistema de arquivos real do computador,* ao qual está vinculado. Geralmente um servidor tem acesso a uma área restrita da máquina e só permite a visualização dos arquivos lá contidos. O sistema de arquivos virtual usa uma notação diferente daquela usada pelo sistema real. Por exemplo, considere o seguinte sistema de diretórios no Windows:

```
C:\
```

```
C:\Windows
C:\Documentos
C:\Documentos\Web\
C:\Documentos\Web\Imagens
C:\Documentos\Web\Videos
```

Suponha que um servidor HTTP foi instalado nessa máquina. Na instalação, ele é configurado para administrar um sistema de diretórios a partir de um certo diretório. Suponha que esse diretório é C:\Documentos\Web\. Para o servidor, isto é seu diretório raiz. No sistema de diretórios virtual, o diretório raiz de um servidor é chamado de / (barra). O sistema de arquivos virtual (a parte que um browser poderá ter acesso) é:

```
/                (C:\Documentos\Web\)
/Imagens         (C:\Documentos\Web\Imagens)
/Videos          (C:\Documentos\Web\Videos)
```

Um browser jamais terá acesso ao diretório Windows, por exemplo. A principal função de um servidor Web é, portanto, *administrar* um sistema de arquivos e diretórios virtual e *atender à requisições dos clientes* HTTP (os browsers), que, na maior parte das vezes, enviam comandos HTTP pedindo que o servidor devolva um ou mais arquivos localizados nesses diretórios. Os pedidos são feitos através de uma sintaxe especial chamada de URI.

1.3.4. URIs (URLs [1])

Todas as comunicações na plataforma Web utilizam uma sintaxe de endereçamento chamada *URI - Uniform Resource Identifier* - para localizar os recursos que são transferidos. O serviço HTTP depende da URI que é usada para localizar *qualquer coisa* na Internet. Contém duas informações essenciais: 1) COMO transferir o objeto (o protocolo); 2) ONDE encontrá-lo (o endereço da máquina e o caminho virtual). URIs tipicamente são constituídas de três partes:

- mecanismo (protocolo) usado para ter acesso aos recursos (geralmente HTTP)
- nome da máquina (precedido de `//`) onde o serviço remoto é oferecido (e a porta, se o serviço não estiver em uma porta padrão) ou outro nome através do qual o serviço possa ser localizado (sem `//`).
- nome do recurso (arquivo, programa) na forma de um caminho (no sistema de arquivos virtual do servidor) onde se possa encontrá-lo dentro da máquina.

  Sintaxe típica:

```
protocolo://maquina:porta/caminho/recurso
```

[1] URIs também são frequentemente chamadas de URLs (Uniform Resource Locators). A URL é um tipo particular de URI mas, para a nossa discussão, essa distinção é irrelevante. A documentação HTML (especificação) sempre refere-se à essa sintaxe como URI.

As URIs mais comuns são os endereços da Web, que utilizam o mecanismo HTTP para realizar a transferência de dados:

```
http://www.maquina.com.br/caminho/para/minha/página/texto.html
```

Veja algumas outras URLs:

- `ftp://usuario:senha@maquina.com/pub/arquivo.doc`
  Acesso a servidor FTP que exige usuário e senha para fazer download de arquivo.doc
- `nntp://news.com.br/comp.lang.java`
  Acesso a servidor de newsgroups para ler o grupo comp.lang.java
- `news:comp.lang.java`
  Acesso ao grupo comp.lang.java através de servidor default (definido localmente)
- `http://www.ibpinet.net/`
  Acesso à página default disponível no diretório raiz do servidor Web de www.ibpinet.net
- `http://www.algumlugar.com:8081/textos/`
  Acesso à página default disponível no diretório textos do servidor Web que roda na porta 8081 da máquina www.algumlugar.net
- `http://www.busca.com/progbusca.exe?opcoes=abc&pesquisa=dracula`
  Passagem de parâmetros de pesquisa para programa de busca progbusca.exe que terá sua execução iniciada pelo servidor HTTP que roda na porta 80 (default) de www.busca.com.
- `http://www.ibpinet.net/helder/dante/pt/inferno/notas_4.html#cesar`
  Acesso à uma seção da página HTML notas_4.html identificada como "cesar", localizada no subdiretório virtual /helder/dante/pt/inferno/ do servidor Web de www.ibpinet.net.
- `mailto:helder@ibpinet.net`
  Acesso à janela de envio de e-mail do cliente de correio eletrônico local.

1.3.5. Browser

O browser é um programa que serve de interface universal a todos os serviços que podem ser oferecidos via Web. É para a plataforma Web o que o sistema operacional (Windows, Linux, Mac) é para o computador. A principal função de um browser é *ler* e exibir o conteúdo de uma página Web. A maior parte dos browsers também é capaz de exibir vários outros tipos de informação como diversos formatos de imagens, vídeos, executar sons e rodar programas.

Um browser geralmente é usado como *cliente HTTP* – aplicação de rede que envia requisições a um servidor HTTP e recebe os dados (uma página HTML, uma imagem, um programa) para exibição, execução ou *download*. Browsers também podem ser usados *off-line* como aplicação local do sistema operacional para navegar em sistemas de hipertexto

construídos com arquivos HTML (sem precisar de servidor HTTP). Nesse caso, não se comportam como clientes HTTP (já que não estão realizando operações em rede) mas apenas como *visualizadores de mídia interativa* capazes de visualizar HTML, imagens, sons, programas, etc.

Como os browsers precisam interpretar vários tipos de código (código de imagens GIF, JPEG, código de programas Java e Flash, códigos de texto HTML ou texto simples) é preciso que ele saiba identificar os dados que recebe do servidor. Isto não é a mesma coisa que identificar um arquivo carregado do disco local, onde ele pode identificar o tipo através da extensão. Quando os dados chegam através da rede, a extensão não significa nada. O servidor precisa informar ao browser o que ele está enviando. Na Web, isto é feito através de uma sintaxe padrão para definir tipos chamada *MIME - Multipart Internet Mail Extensions*.

1.3.6. Tipos MIME

MIME é uma sintaxe universal para identificar tipos de dados originalmente utilizada para permitir o envio de arquivos anexados via e-mail. O servidor Web possui, internamente, tabelas que relacionam os tipos de dados (na sintaxe MIME) com a extensão dos arquivos por ele gerenciados. Quando ele envia um conjunto de bytes para o browser, envia antes um *cabeçalho* (semelhante ao cabeçalho de e-mail) informando o *número de bytes* enviados e o *tipo MIME* dos dados para que o browser saiba o que fazer com a informação. A sintaxe MIME tem a seguinte forma:

```
tipo/subtipo
```

O *tipo* classifica um conjunto de bytes como imagens, textos, vídeos, programas (aplicações), etc. O *subtipo* informa características particulares de cada tipo. Não basta saber que o arquivo é uma imagem, é preciso saber qual o *formato*, pois o códigos usados para produzir imagens de mesma aparência gráfica podem diferir bastante entre si. Tanto no servidor como no browser há tabelas que relacionam extensões de arquivo a tipos MIME:

```
image/jpeg              .jpe, .jpg, .jpeg
image/png               .png
image/gif               .gif
text/html               .html, .htm, .jsp, .asp, .shtml
text/plain              .txt
x-application/java      .class
```

1.4. Tecnologias de Apresentação

As tecnologias utilizadas na plataforma Web podem ser classificadas de acordo com sua finalidade em *tecnologias de apresentação* e *tecnologias interativas*. As tecnologias de apresentação são aquelas que se destinam unicamente à formatação e estruturação das páginas Web. Podem ser usadas também para construir a interface de aplicações Web no browser. Os principais padrões

em uso atualmente são HTML, CSS, XML e XSL. As tecnologias interativas são as que permitem o desenvolvimento de aplicações e páginas com alto nível de interatividade com o usuário. Em geral consistem da combinação de uma linguagem de programação com uma arquitetura ou modelo que possibilita a sua integração com uma página HTML ou servidor HTTP. Podem, portanto, executar do lado do servidor (como CGI, ASP, ADO, Servlets, ISAPI, JSP, PhP, Cold Fusion e LiveWire) ou do lado do cliente (como JavaScript, DHTML, Java Applets, ActiveX e VBScript).

1.4.1. HTML

HTML - *HyperText Markup Language* é a linguagem universal da Web. É através dela que a informação disponível nas páginas da WWW pode ser acessada por máquinas de arquiteturas e sistemas operacionais diferentes. Não é uma linguagem de programação com a qual se possa construir algoritmos, mas uma linguagem declarativa que serve para organizar informações em um arquivo de textos que será visualizado em um browser. Define uma coleção de elementos para marcação (definição de estrutura) de texto. Se você, no passado, já usou um editor de textos como *WordStar* ou *Carta Certa*, deverá se "sentir em casa" com HTML. Coo foi mencionado anteriormente, um *arquivo HTML* é um arquivo de *texto simples* recheado de marcadores que se destacam do texto pelos caracteres especiais "<" e ">".

Existem várias linguagens para formatação de textos. Qualquer texto que aparece formatado (com fontes, cores, tamanhos) em um computador tem uma linguagem de formatação por trás. A maioria são linguagens proprietárias que só funcionam em softwares específicos (textos em formato *Word*, por exemplo). Existem alguns formatos, porém, que se tornaram padrões, servindo basicamente para realizar conversões entre os formatos proprietários (SGML e RTF, PostScript, TeX, PDF). HTML é um formato *público* (não pertence a um fabricante específico), e é *leve* (não produz arquivos enormes como o *Word*) sendo por essas e outras razões adequado à difusão de informações que serão visualizadas em máquinas diferentes.

Com HTML é possível publicar documentos estruturados *on-line*, recuperar informações através de vínculos de hipertexto, projetar uma interface interativa com formulários para acesso a serviços remotos como buscas e comércio eletrônico, e incluir imagens, vídeos, sons, animações e outras aplicações interativas dentro de documentos visíveis no browser.

1.4.2. Histórico do HTML

HTML foi desenvolvida originalmente por Tim Berners-Lee no CERN - Laboratório Europeu de Física de Partículas. Sua popularidade cresceu junto com a popularização da Web, através do *NCSA Mosaic*. Devido ao surgimento de vários browsers que utilizavam HTML para navegar no sistema de informações proporcionado pela Web, grupos de trabalho foram formados com a intenção de padronizar especificações para o HTML.

HTML 2.0, concluída em 1995, foi a primeira versão recomendada pelo IETF - *Internet Engineering Task Force* e se tornou um padrão da Internet. HTML 2.0 era uma linguagem simples que dizia como um browser deveria estruturar uma página, mas não como os títulos, parágrafos e listas deveriam aparecer graficamente. Durante o desenvolvimento do HTML 3.0, a Web estava em franca expansão e os esforços de padronização não puderam acompanhar as tendências do mercado, que exigiam maiores recursos de apresentação gráfica ao HTML 2.0. O HTML 3.0 acabou não sendo aprovado e anos depois, aprovou-se uma recomendação chamada HTML 3.2 em 1997, que introduzia recursos de apresentação gráfica no HTML.

Infelizmente a maioria dos recursos gráficos do HTML 3.2 foram incorporações de extensões proprietárias da *Netscape* e *Microsoft*, criadas sem levar em conta a filosofia do HTML de garantir a compatibilidade da linguagem em plataformas diferentes. Isto acabou atrasando o desenvolvimento de ferramentas de desenvolvimento eficientes, pois era impossível validar HTML para plataformas que não suportavam certos recursos gráficos mais sofisticados.

HTML foi desenvolvida originalmente para que *qualquer dispositivo* pudesse ter acesso à informação da Web. Isto inclui PCs com monitores gráficos de diversas resoluções, terminais orientados a caracter, telefones celulares, dispositivos geradores de voz, etc. HTML 3.2 tinha elementos que prejudicavam essa meta. Finalmente, depois de muita discussão, as empresas entraram em um acordo e desenvolveram o HTML 4.0, que estende o HTML com mais recursos visando um acesso mais universal à informação da Web, como recursos de acessibilidade à pessoas com deficiências, suporte a convenções internacionais (outras línguas, outros alfabetos), separação da estrutura, conteúdo e apresentação, recursos interativos do lado do cliente e otimização em tabelas e formulários.

1.4.3. HTML 4 e CSS

Vários elementos do HTML 3.2 foram considerados *deprecados* (candidatos a se tornarem obsoletos) pelo HTML 4. São todos elementos que permitem definir cores, fontes, alinhamento, imagens de fundo e outras características da apresentação da página que dependem da plataforma onde a informação é visualizada. HTML nunca realizou bem o trabalho de formatação gráfica de uma página. Foi criada inicialmente para apenas dar estrutura a um conteúdo. Nunca previu formas de posicionar imagens e texto de forma absoluta em uma página e as soluções desenvolvidas pelos Web designers, por não serem soluções previstas na especificação, têm causado problemas de acesso em vários sites e impedido o acesso de dispositivos mais limitados como a *WebTV* e *WebPhone* de terem acesso total à Web. A solução do HTML 4.0 foi separar a *estrutura* da *apresentação*, deixando que a linguagem HTML voltasse às suas origens (nos tempos do HTML 2) para definir apenas a função do texto marcado (o *que é* título, parágrafo, etc.) Uma outra linguagem foi criada para se preocupar com a aparência (*como* o título e parágrafo serão exibidos na tela). A principal linguagem usada para esse fim é CSS - *Cascading Style Sheets*, que permite a criação de *folhas de estilo* aplicáveis a várias páginas de um site. Se um dispositivo limitado não consegue exibir os estilos definidos no CSS, ele pelo

menos consegue entender a estrutura do texto e imagens de forma que mesmo usuários com menos recursos podem ter acesso à informação.

Escrever HTML não é difícil, apenas toma tempo, por isso usar um editor apropriado pode tornar o processo de criação de páginas mais produtivo. Um arquivo de texto simples com descritores HTML, quando carregado em um browser, tem os descritores interpretados e as suas informações formatadas na tela de acordo com a estrutura prevista pelos marcadores e uma folha de estilos, geralmente definida pelo próprio browser. A folha de estilos pode ser definida pelo programador usando uma linguagem como CSS e vinculada à página para mudar sua aparência.

1.4.4. XML e XSL

XML - *eXtensible Markup Language* e XSL - *eXtensible Style Language* são as novas criações do W3C - *World Wide Web Consortium* (consórcio das empresas que desenvolvem os padrões para a WWW). Não pretendem substituir o HTML mas, em vez disso, oferecer meios de estender e ampliar as possibilidades da Web. XML é uma especificação ou *meta-linguagem* que define uma sintaxe que pode ser usada para criar novas linguagens semelhantes a HTML. A própria linguagem HTML pode ser vista como um tipo especial de XML. Com XML você pode criar sua própria linguagem de marcação "MinhaML", definir seus próprios marcadores e esquemas para aplicações específicas, por exemplo, poderia conter algo como:

```
<compra id="xyz"><data>26/12/1999</data>...</compra> ...
```

Depois você pode usar CSS ou XSL para definir a aparência dos seus marcadores em um browser XML. Para que um browser XML seja capaz de compreender a linguagem que você criou, é preciso definir um *dicionário* e uma *gramática* (usando as regras da especificação XML) para ela. A gramática para a análise da sua linguagem deve ficar armazenada em um documento chamado DTD - *Document Type Definition*. Carregando o DTD, o browser XML aprenderá a nova linguagem e será capaz de formatar a informação que você estruturou com seus marcadores, e apresentar as informações na tela de acordo com as regras de estilo definidas no CSS ou em um XSL criado por você.

As tecnologias XML e XSL não serão abordadas neste curso mas você pode obter maiores informações sobre as duas tecnologias, além de ter acesso a guias de referência e tutoriais através do site do World Wide Web Consortium (W3C) em http://www.w3.org/XML/.

1.5. Ferramentas de desenvolvimento HTML

Podemos criar uma página Web simplesmente usando um editor de textos (como o bloco de notas do Windows) e um browser para visualização. Embora esta seja uma boa forma de aprender, é pouco produtiva para desenvolver sites complexos que usam tabelas, frames e design sofisticado. Alguns editores são bastante visuais e mostram todo o processo de criação

do site, fazem busca e substituição em todo o site, utilizam *templates*, permitem edição direta do código e fazem a previsão sem a necessidade de um browser. É o caso do *Microsoft FrontPage*, do *Macromedia DreamWeaver*, do *Corel WebMaster*, e do *Adobe GoLive*. Outros, simplesmente servem para economizar batidas de teclas e evitar erros de sintaxe na hora de escrever os descritores HTML. Editores de código HTML como o *Hot Dog*, *Hot Metal*, *HomeSite*, *BareBones*, etc. realizam esta função. Nas seções a seguir, discutiremos um pouco de cada um.

1.5.1. Editores gráficos (WYSIWYG)

Os editores mais fáceis de usar, que dispensam totalmente o uso de HTML são os editores WYSIWYG. Esse nome esquisito é uma sigla bastante usada nos primeiros tempos da editoração eletrônica para caracterizar programas que representavam na tela do computador uma página da forma como seria impressa. Naquele tempo, era comum a existência de processadores de texto ou formatadores de texto que usavam comandos para descrever como o texto iria aparecer na hora da impressão. Para se ter uma idéia do resultado final, era necessário ou imprimir ou rodar um programa a parte que fizesse um *preview* da página criada. Com o advento da editoração eletrônica, surgiram programas como o *Aldus PageMaker* no *Macintosh* e *Ventura Publisher* no PC que mostravam na tela a própria página, e não um monte de códigos de controle misturados ao texto. WYSIWYG quer dizer *What You See Is What You Get*, ou, "o que você vê é o que você obtém", sugerindo que o que o autor visualiza na tela é uma representação bastante fiel do resultado final que obterá na impressora.

WYSIWYG na Web é menos fiel que na impressão. Para trabalhos impressos, tem-se uma representação da página em uma determinada impressora, com determinadas fontes, cores, etc. Na Web é impossível saber se a pessoa que vai ver a sua página tem as mesmas fontes, cores, versões de browser que aquela onde você fez o seu teste "WYSIWYG". Portanto, os editores gráficos de páginas Web são WYSISWIG ou What You See is *Sometimes* What You Get.

A multiplicidade de browsers e plataformas que existem na Internet exige do Web designer mais que saber fazer páginas que ficam boas em uma única plataforma e browser. É necessário testar o site em sistemas diferentes, levando em conta o público-alvo e muitas vezes é necessário recorrer à codificação HTML para resolver algum problema não previsto pelos editores.

1.5.2. Editores de texto

Pode-se usar qualquer editor de texto que tenha a capacidade de salvar um arquivo de texto "puro", em formato ISO-Latin-1 ou ASCII para criar páginas HTML. Se o seu editor só conseguir salvar ASCII 7-bit, será mais difícil o uso de acentos (você terá que usar códigos especiais). Alguns exemplos de editores que podem ser usados são o *Bloco de Notas* do Windows, o *WordPad* (desde que se salve a página como "texto"), o *EditPad*, o *WinEdit* e outros editores shareware/freeware populares disponíveis na Internet.

1.5.3. Editores de HTML

Editores de HTML são como os editores de texto mas possuem uma série de atalhos para agilizar a entrada de código HTML. Os atalhos também evitam que se cometam erros de sintaxe. Os editores HTML mais sofisticados possuem um sistema de *preview* embutido, permitem a definição de gabaritos a serem aplicados a um conjunto de páginas, suportam macros, busca e substituição em diversos arquivos e filtros diversos.

A maioria dos editores HTML estão disponíveis na Internet pelo sistema de *freeware* ou *shareware*. O site das duas vacas, ou TUCOWS (http://www.tucows.com) possui uma lista crescente destes programas. Você também encontra editores em http://www.shareware.com. Entre os comerciais, um dos mais populares é o *Allaire HomeSite*.

1.5.4. Browsers e servidores

Qualquer web designer que queira desenvolver páginas seriamente hoje em dia deve ter as versões dos browsers mais populares entre o público. Não basta testar somente na última versão. Geralmente a grande massa de usuários que navega na Internet não possui a última versão do seu browser.

Os browsers são a janela do usuário para a Web. Dependendo da versão ou do tipo de browser que o seu visitante está usando, e qual o seu computador, a sua página pode aparecer para ele como uma experiência agradável que o fará voltar muitas e muitas vezes; ou como um pesadelo horrível, que ele jamais esquecerá. É trabalho do Web designer resolver ao máximo esses problemas de compatibilidade de forma a garantir que pelo menos a informação essencial seja acessível (isto inclui a identidade visual e a navegação do site). É essencial testar as páginas em browsers e plataformas diferentes antes de publicá-las.

Os browsers mais populares hoje continuam sendo o *Netscape Communicator* e *Microsoft Internet Explorer*. As plataformas de navegação mais populares ainda são PC e Mac. Mas o que é usado por todo o mundo pode não ser o mais importante no seu caso: se seu *público-alvo* é formado em grande parte por usuários de máquinas Sun é essencial testar suas páginas nessas plataformas também.

Há ainda uma outra questão. Mesmo que você saiba que 99% do seu público usa browsers Netscape ou Microsoft, é importante saber as *versões* que eles usam. A grande maioria dos usuários *não possui* as últimas versões dos seus browsers (os provedores continuam distribuindo versões antigas). Há muitos recursos novos nas novas versões do *Communicator* e do *Internet Explorer* que não são suportados nas versões anteriores. Se você usá-los sem prestar atenção ao efeito que causam em versões antigas, uma grande parte do público que visitar suas páginas poderão nunca mais voltar.

1.5.5. Por que aprender HTML?

Se é possível fazer todo um site, com *scripts* interativos, imagens, animações, Java, etc. sem escrever uma linha sequer de HTML, por que perder tempo aprendendo mais hieroglifos que não vão servir para nada?

A resposta à pergunta do parágrafo anterior depende do que você pretende fazer e do controle que você pretende ter sobre a sua obra. Se o seu objetivo é fazer páginas práticas e rápidas, como uma *home page* pessoal ou para um sistema Intranet simples, pode realizar todo o trabalho sem sequer tomar conhecimento do HTML, usando as ferramentas que existem por aí. Mas se você deseja fazer coisas mais sofisticadas, como ter mais controle sobre o *design* da sua página, criar páginas que interajam com outros programas, como servidores de bancos de dados, e usar recursos novos como *Dynamic HTML*, *JavaScript*, canais e folhas de estilo, você provavelmente não vai escapar de ter que editar sua página pelo menos uma ou duas vezes para fazer alguns ajustes, ou para analisar o código para poder criar rotinas JavaScript, DHTML, etc.

A notícia boa é que HTML é fácil. Não é linguagem de programação, não tem recursão, ponteiros ou nada do tipo. É somente marcação de texto. É trivial para aqueles que usam computadores desde o tempo do MS-DOS e WordStar – programas bem mais complexos que o HTML.

2 HTML

2.1. Fundamentos do HTML

Esta seção apresenta a estrutura e sintaxe da linguagem HTML. *HyperText MarkUp Language* é uma linguagem declarativa que tem a finalidade de dar estrutura a um arquivo de texto, identificando a função de suas seções (título, parágrafo, etc.) e vincular as páginas entre si através de hipertexto.

Não é possível programar em HTML pois a linguagem não possui recursos através dos quais se possa construir procedimentos.

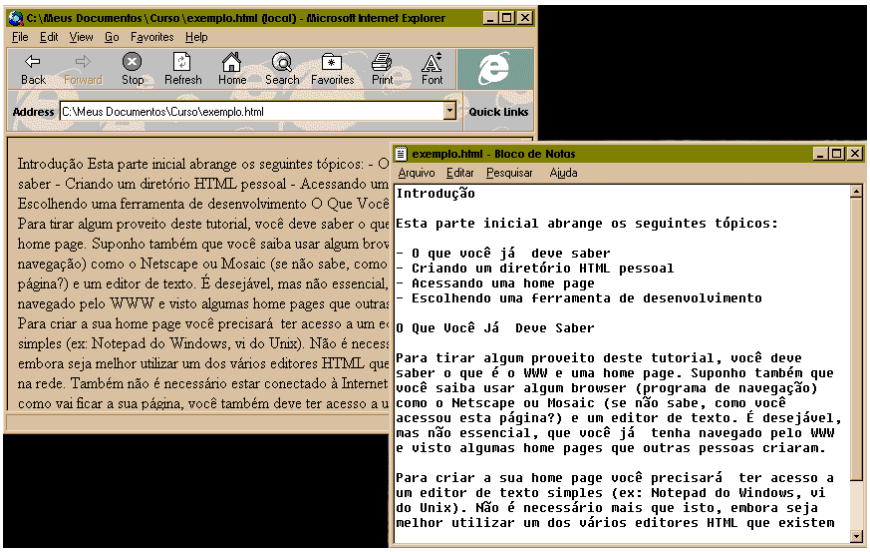
2.1.1. Elementos e descritores (tags)

O arquivo usado para construir uma página Web é texto simples. Mesmo que a página visualizada no browser mostre imagens, applets, plug-ins (como Flash) e outros recursos, por trás de tudo existe uma página de texto e vários outros arquivos separados. É texto, mas não é *só* texto. É texto *marcado* com HTML. HTML define *toda* a estrutura de uma página para que um browser possa formatá-la e produzir a apresentação desejada. HTML também *importa* as imagens, programas, sons e vídeos que uma página exibe em seu interior. Mas como HTML é texto simples, pode ser editada em qualquer editor de texto (como por exemplo, o *Bloco de Notas* do *Windows*).

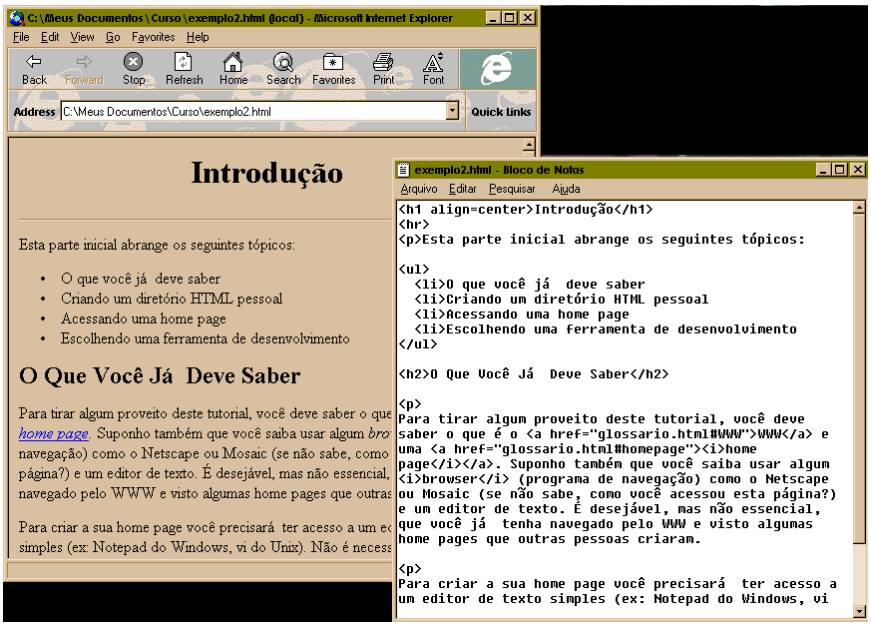
Ao ler um documento HTML, um browser tenta interpretar todas as seqüências de caracteres que ficam entre os símbolos "<" e ">". O browser entende que *qualquer coisa* que estiver entre esses caracteres é um *descritor HTML* (*tag*) e não deve ser mostrado na tela. Se o descritor for desconhecido ele simplesmente o *ignora* (e não mostra o conteúdo na tela) mas se realmente for um *elemento HTML* (definido na especificação suportada pelo browser), ele usará as informações contidas entre os símbolos estruturar a página, formatando-a de acordo com alguma regra de estilo previamente definida.

Toda a formatação de um arquivo HTML é feita *exclusivamente* através dos descritores. O browser sabe que o arquivo contém código HTML através de sua extensão (ou tipo MIME enviado pelo servidor).

Se um arquivo de texto com extensão HTML mas sem descritores for carregado, no browser, toda a sua informação aparecerá em um único parágrafo, sem destaques, sem formatação alguma. Veja as figuras abaixo. Elas ilustram um mesmo texto de um documento HTML digitado no *Bloco de Notas* do Windows e lidos através do *Internet Explorer*. Na primeira figura, não há descritores HTML e, apesar dos espaços, tabulações e novas-linhas existente no arquivo de texto visualizado no editor de textos, o browser ignora toda a formatação. Já na segunda figura descritores HTML foram incluídos e o browser foi capaz de formatar a página de uma maneira estruturada.



Arquivo de texto (com extensão .html) sem marcação HTML ao ser visualizado no browser e no editor Bloco de Notas (Windows).



Arquivo de texto (com extensão .html) contendo marcação HTML ao ser visualizado no browser e no editor Bloco de Notas (Windows).

A maioria dos *elementos HTML* possui um *descritor inicial* e um *descritor final*, cercando o texto que é marcado por eles. A sintaxe básica é

<Elemento> `Texto marcado por Elemento` **</Elemento>**

O texto marcado passa a ter uma *função*, determinada pelo Elemento. Nem sempre isto significa que o texto marcado terá sua aparência alterada quando a página HTML aparecer no browser. Há browsers orientados a caractere que não são capazes de exibir tamanhos de fonte, por exemplo. Outros podem não ter uma forma associada a um elemento. Neste caso, o autor da página poderá definir a forma através de uma folha de estilos usando CSS, caso o browser a suporte. Os elementos também podem influenciar o comportamento de mecanismos de busca, que atribuem maior importância ou filtram dados de acordo com a sua função determinada pelos elementos HTML. O texto marcado e o elemento HTML também assumem o papel de

objeto abstrato utilizável por linguagens embutidas na página como CSS e JavaScript. Essas linguagens interagem com o HTML através de um modelo de objetos que relaciona as estruturas HTML com comportamentos interativos (em JavaScript e DHTML) e forma gráfica (CSS).

Veja alguns exemplos de texto rotulado com descritores HTML:

```
<b> Texto em negrito </b>
<h2> Subtítulo </h2>
```

Observe que o descritor final é praticamente idêntico ao descritor inicial. A única diferença é que este último é precedido por uma barra ("/"). Não faz diferença utilizar letras maiúsculas ou minúsculas para os descritores HTML.

Os blocos de texto marcados pelo HTML podem conter outros blocos. Sempre que isto ocorrer, o bloco mais interno deverá ser fechado antes do descritor de fechamento do bloco externo. Veja um exemplo:

```
<h1> Texto <b>muito <i>importante</i></b> para todos</h1>
```

Este outro exemplo está incorreto pois o bloco <b> fecha sem que o bloco <i> tenha fechado.

```
<h1> Texto <b>muito <i>importante</b></i> para todos</h1>
```

Nem todo elemento HTML pode conter outros descritores e mesmo os que podem têm regras que definem quais os elementos permitidos.

Existem elementos que não precisam ter descritores de fechamento. O parágrafo, por exemplo, marcado por <P> e pode omitir o descritor final. Isto é permitido porque não pode haver um parágrafo dentro de outro, portanto, se o browser encontra um pouco depois de outro ele automaticamente coloca um antes.

```
<p>Primeiro parágrafo. <p>Início de outro parágrafo.</p>
```

Finalmente, há também elementos que não podem conter texto ou quaisquer descritores HTML. Eles precisam ser vazios. O descritor final nesses casos sequer é permitido (a não ser que apareça logo após o descritor inicial).

```
<p>Esta é uma frase<br>que ocupa duas linhas.
```

No exemplo acima
 (quebra de linha) marca a posição onde a linha deve ser quebrada (não contém ou marca texto algum).

2.1.2. Atributos

Alguns elementos HTML podem ter um ou mais *atributos*, opcionais ou não, que modificam alguma propriedade do texto marcado ou acrescentam alguma informação necessária. Geralmente os atributos são pares do tipo:

```
propriedade="valor"
```

Quando presentes, os atributos aparecem apenas no descritor inicial separados por espaços, logo após o nome do elemento:

```
<h1 align="center">Título centralizado</h1>
```

Um descritor pode ter vários atributos. A *ordem* em que aparecem *não faz diferença* desde que apareçam no *descritor inicial* e estejam separados dos outros atributos e do nome do elemento por *espaços*. Se um atributo for escrito incorretamente ou se o browser não o reconhecer, ele será ignorado. Diversos atributos são comuns a todos os elementos HTML. Outros só fazem sentido em certos elementos.

O valor do atributo não precisa estar entre aspas se contiver apenas letras (`A-Z, a-z`), números, ponto (`.`), dois pontos (`:`), hífen (`-`) e sublinhado (`_`). Se o valor atribuído à propriedade tiver espaços, é preciso colocá-lo entre aspas ou apóstrofes. Não pode haver espaço entre a propriedade e o "=" nem entre o "=" e o valor:

```
propriedade="Valor com espaços"
propriedade = "Sintaxe incorreta"
```

O segundo exemplo acima será interpretado incorretamente pelo browser como tendo três atributos e não um (não pode haver espaços!).

Atributos geralmente são opcionais. Em alguns casos são obrigatórios, como os atributos `HREF` e `SRC` em vínculos de hipertexto e imagens, respectivamente. Ambos informam uma URI onde se encontra um recurso na Web.

```
<a href="http://www.ibpinet.net/"
   title="Vai logo! Clica!"> Clique aqui </a>

<img src="antas.gif"
     height=120 width=400
     alt="Um casal de antas">
```

2.1.3. Caracteres de escape

Os caracteres "<" e ">", por definirem o início e final dos descritores, não podem ser impressos na tela do browser. Quando é necessário produzi-los, deve-se utilizar uma *seqüência de escape*. Esta seqüência é iniciada por um "&" seguido de uma abreviação e um ponto-e-vírgula, que indica o final da seqüência. Como o "&" também é caractere especial, há também uma seqüência para produzi-lo. As principais seqüências de escape, necessárias para produzir "<", ">", "&" e aspas (quando necessário) são:

Caractere	*Seqüência de Escape*
<	`<`
>	`>`
&	`&`
"	`&quot;`

Por exemplo, para produzir as seguintes linhas no browser:

```
144 < 25 + x < 36 + y
Fulano, Sicrano & Cia.
```

é preciso digitar no editor de textos (código HTML):

```
144 &lt; 25 + x &lt;  36 + y
Fulano, Sicrano &amp; Cia.
```

As seqüências de escape também são usadas para produzir caracteres que não são encontrados no teclado, como letras acentuadas, símbolos de *copyright*, etc. Por exemplo, para produzir, no browser, as palavras:

```
Plantação
maßgebend
håndbfger
enciclopædia
©
sueño
```

pode-se usar, no editor de textos:

```
Planta&ccedil;&atilde;o <BR>
ma&szlig;gebend <BR>
h&aring;ndb&oslash;ger <BR>
encilop&aelig;dia <BR>
&copy; <BR>
sue&ntilde;o
```

É possível usar também códigos numéricos, porém eles são dependentes do alfabeto utilizado. Consulte a documentação HTML para uma lista completa de todos os códigos.

Os editores mais sofisticados, como por exemplo o *HomeSite*, facilitam a entrada de caracteres especiais e geram automaticamente as seqüências de escape. Se você está usando um editor comum, como o *Bloco de Notas*, deve ter o cuidado de digitar corretamente as seqüências. Alguns browsers simplesmente ignoram seqüências que não conhecem, outros engolem linhas e palavras. Em qualquer um dos casos há perda de informação. Nunca é necessário digitar seqüências de escape se:

- você tem os caracteres no seu teclado
- eles fazem parte do conjunto de caracteres ISO-Latin-1 (alfabeto *default* do HTML)
- alfabeto default da página não foi alterado através de um descritor <META> ou de um cabeçalho HTTP.

Todos os browsers há mais de 4 anos entendem acentos e cedilha sem que seja necessário digitar os códigos.

É indiferente usar letras maiúsculas ou minúsculas para o nome do descritor ou seus atributos. Tanto faz usar <BODY>, <body>, <Body> ou <bOdY>; <a HREF="a.html"> ou

<A href="a.html">. Deve-se tomar cuidado, porém, com o *conteúdo* dos atributos (o valor que às vezes precisam vir entre aspas). Em alguns casos, principalmente quando se lida com URIs ou quando se usa JavaScript, o formato maiúsculo ou minúsculo *faz* diferença e deve ser respeitado.

2.1.4. Comentários

Elementos desconhecidos pelo HTML são ignorados quando colocados entre sinais de < e >. Se você deseja fazer isto intencionalmente para acrescentar comentários no código (que não aparecerão na página), deve usar os comentários HTML, identificados entre <!--e -->. Qualquer texto ou código entre esses símbolos será ignorado na formatação da página pelo browser.

2.1.5. Estrutura do documento

A maioria dos elementos HTML segue uma estrutura hierárquica. Há uma estrutura básica (mínima) para uma página HTML, construída de acordo com a especificação padrão, que deve ser respeitada para garantir uma compatibilidade com o maior número de browsers possível. Esta estrutura está mostrada na figura abaixo:

```
<!DOCTYPE HTML Public "-//IETF//DTD HTML 4.0//EN" -->
<HTML>
   <HEAD>
     <TITLE> Descrição do documento </TITLE>
        Aqui ficam blocos de controle, folha de estilo, funções de
        scripts, informações de indexação, atributos que afetam todo
        o documento, etc.
   </HEAD>
   <BODY>
        Toda a informação visível da página vem aqui.
   </BODY>
</HTML>
```

A primeira linha: <!DOCTYPE HTML Public "-//IETF//DTD HTML 4.0//EN" --> é um *descritor SGML* que informa ao browser que ele deve interpretar o documento de acordo com a definição do HTML versão 4.0, se possível. HTML é uma linguagem derivada e completamente especificada a partir da linguagem SGML – *Standard Generalized Markup Language*. Você não precisa saber SGML para usar HTML, basta recortar e colar a linha acima que serve para orientar o browser e permitir a validação do código de sua página.

O elemento <HTML>...</HTML> marca o início e o final do *documento HTML*. Deve conter duas sub-estruturas distintas: o *cabeçalho*, delimitado por <HEAD> e </HEAD>, e o *corpo do documento*, entre os descritores <BODY> e </BODY>.

2.1.6. Elementos do HEAD

O bloco do cabeçalho, marcado por <HEAD> e </HEAD> pode conter informações *sobre* o conteúdo do documento utilizada para fins de indexação e organização. Não contém informação que será exibida na página.

TITLE

<TITLE> é o único elemento obrigatório do bloco do cabeçalho. Deve conter o título do documento que aparece fora da página, na barra de título do browser. É o que aparece também nos *hotlists* e *bookmarks*. O título deve conter informações que descrevam o documento.

```
<TITLE>HTML e CSS: Introdução</TITLE>
```

META

<META> é usado para incluir meta-informação como palavras-chave, descrições, etc. que podem ser usadas por mecanismos de busca, softwares de pesquisa e catalogação. A informação adicional deve vir nos atributos NAME (descreve o tipo de meta-informação, por exemplo Keywords) e CONTENT (descreve o conteúdo da meta-informação, por exemplo, uma lista de palavras-chave separadas por vírgula). <META> também pode ser usado para adicionar ou redefinir *cabeçalhos HTTP*. Isto é feito através do atributo HTTP-EQUIV. Neste caso, o CONTENT deve conter o conteúdo do cabeçalho.

```
<META HTTP-EQUIV="Set-Cookie" CONTENT="pag=12">

<META NAME="Keywords" CONTENT="html, css, folhas de estilo, estilo">

<META NAME="Description" CONTENT="Esta página explica os fundamentos
básicos de HTML e folhas de estilo usando a linguagem CSS.">

<META HTTP-EQUIV="Refresh" CONTENT="10;url=pag13.html">

<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
```

LINK

<LINK> é usado para vincular uma página a outro recurso. Na maioria dos casos, o vínculo é simbólico e não tem outra finalidade a não ser facilitar a indexação e organização do site por ferramentas de validação e busca. Vínculos a folhas de estilo incorporam o arquivo que é interpretado pelo browser antes de formatar a página..

```
<LINK REL="StyleSheet" HREF="../estilos/default.css">
```

BASE

<BASE> altera os vínculos de origem e destino da janela, ou seja, a parte absoluta da URL das imagens e vínculos, especificados com URLs relativas, e a janela onde o resultado dos vínculos será mostrada. Normalmente a URL base de origem é o local onde a página se encontra no momento em que carrega a imagem ou o usuário clica no vínculo (protocolo atual, máquina atual e diretório atual ou "./"). Com <BASE>, o autor pode alterar a URL base para que as imagens e links sejam buscados em outros lugares. Normalmente a janela atual (_self) é a responsável por receber o resultado dos vínculos. O autor também pode, com <BASE>, fazer com que links abram em outra janela, em novas janelas (_blank) ou em outras partes de estrutura de *frames* (_top e _parent).

```
<BASE HREF="." TARGET="_self"> <!-BASE default -->

<BASE HREF=" http://www.a.com/dados/">
<!-Os links relativos serão procurados em http://www.a.com/dados/-->

<BASE TARGET="lateral2"> <!- links abrirão na janela lateral2 -->
```

2.1.7. Elementos de BODY

O bloco marcado por <BODY> e </BODY> contém a parte do documento onde será colocada a informação que efetivamente será mostrada e formatada na tela pelo browser. Todos os elementos das seções seguintes tratam da formatação do texto (ou imagens) da página e devem, portanto, estar presentes no bloco <BODY>.

Elementos HTML podem ser de vários tipos e têm regras específicas sobre o que podem conter e onde podem ser usados. Quanto à estrutura que assumem na página, podem ser classificados da seguinte forma:

- Elementos de *bloco* (contém texto ou agrupam outros elementos de bloco)
- Elementos *inline* (em linha – usados dentro do texto)
- Elementos de *lista* (usados para construir listas)
- Elementos de *tabela* (usados para construir tabelas)
- Elementos de *formulário* (usados para construir formulários)
- Elementos para embutir *objetos* (usados para incluir imagens, applets, vídeos, etc.)

Entre os elementos *inline* estão os vínculos (*links*) de hipertexto, que permitem marcar texto que servirá de ligação a outra página ou recurso na Internet.

O objetivo das seções seguintes é apresentar uma introdução básica aos principais elementos HTML. Não é uma introdução exaustiva. Consulte as referências e documentação oficial para uma abordagem completa de cada elemento HTML.

2.2. Parágrafos e blocos de texto

Blocos são elementos que podem ocorrer dentro de <BODY>. Não é permitido existir texto diretamente abaixo de <BODY>. O texto deverá estar dentro de um bloco, tabela, lista ou formulário. Alguns blocos HTML só permitem que haja outros blocos dentro deles (é o caso de <BODY>. Outros só admitem texto e elementos *inline* (*em linha* – elementos que não são blocos e podem ser usados dentro de parágrafos para formatar texto, marcar posições ou incluir imagens). Alguns admitem blocos e texto (<TD>, <LI>, etc.).

2.2.1. Parágrafos <P>

O descritor <P> abre um parágrafo em HTML. O descritor </P> o fecha. Tudo o que está entre <P> e </P> é tratado como um bloco. Se esse nosso parágrafo tiver atributos, eles afetarão todo o bloco.

Se seu arquivo HTML contiver:

```
<P>Um parágrafo</P> <P>Outro parágrafo</P>
```

um browser como o Netscape ou o Internet Explorer mostrará na tela algo parecido com:

Um parágrafo

Outro parágrafo

<P> pode ter atributos. Um deles é o atributo ALIGN. Se ele não for usado, cada parágrafo alinha pela esquerda. Ele pode ser usado para fazer o parágrafo alinhar pelo centro ou pela direita, por exemplo:

```
<P ALIGN=center>Um parágrafo</P> <P ALIGN=right>Outro parágrafo</P>
```

vai resultar em:

Um parágrafo

Outro parágrafo

O atributo ALIGN, como qualquer outro atributo que altera o posicionamento ou aparência através do HTML, foi *deprecado* na especificação HTML 4 em benefício das folhas de estilo. Nem todos os browsers suportam, porém, os recursos de folha de estilo que substituem o ALIGN. Mas as folhas de estilo, quando presentes, sempre têm precedência. Se uma folha de estilos ditar um alinhamento, aquele proposto pelo atributo ALIGN é ignorado.

O elemento </P> de um parágrafo é opcional. Como não pode haver um parágrafo dentro de outro ou qualquer *elemento de bloco* dentro de um parágrafo, o browser automaticamente fecha o parágrafo quando encontra um <P>, <H1>, <DIV>, <BLOCKQUOTE>, <PRE> ou outro elemento de bloco. Blocos em HTML nunca podem ocorrer dentro do texto de um parágrafo.

2.2.2. Cabeçalhos <H1> a <H6>

Um outro bloco importante na organização do texto é o cabeçalho. Há seis níveis diferentes. O mais importante é definido com <H1> e pode ser usado para destacar, por exemplo, o título de um documento. Não confunda esse título com <TITLE>, elemento de <HEAD>, que define um título para a página mas não aparece na mesma.

```
<h1>La Cosa Nostra Pizzaria i Ristorante</h1>
```

Observe no seu browser que o texto aparece grande e em negrito (por causa da folha de estilos do browser).

Assim como <P>, <H1> também suporta o atributo ALIGN, portanto, pode-se usá-lo para colocar o título alinhado pelo centro:

```
<h1 align=center>La Cosa Nostra Pizzaria i Ristorante</h1>
```

Além do H1, há mais 5 níveis de cabeçalho que podem ser usados para organizar os parágrafos de uma página em seções. Veja na figura abaixo o código e o resultado no browser de textos marcados com H1, H2, H3, H4, H5 e H6.

2.2.3. Controle de fluxo

*Quebras de linha
*

Os parágrafos sempre têm uma linha em branco separando-os dos outros blocos (por causa da folha de estilos do browser). Pode-se quebrar uma linha e ainda assim se manter no mesmo parágrafo usando o marcador <BR>.
 é um marcador vazio. Não pode ter descritor de fechamento.
 quebra a linha na posição onde for colocado:

```
<p align=center>"Tradição, qualidade e<br>
segurança são nossas<br> prioridades"</p>
```

Quebras de linha só podem ser usadas dentro de parágrafos, cabeçalhos e outros elementos de bloco que contém texto.

Linhas horizontais <HR>

Linhas horizontais decorativas podem ser produzidas pelo <HR>. <HR> é como <BR>. Não tem marcador final de fechamento. Acontece no lugar onde o colocarmos e ainda introduz uma quebra de linha antes e depois.:

```
<h1 align=center>La Cosa Nostra Pizzaria i Ristorante</h1>
<hr>
<h2 align=center>Una impresa de la famiglia Corleone</h2>
```

A linha atravessa toda a largura do browser. Podemos usar o atributo WIDTH para torná-la menor. WIDTH aceita um número absoluto em pixels ou uma porcentagem:

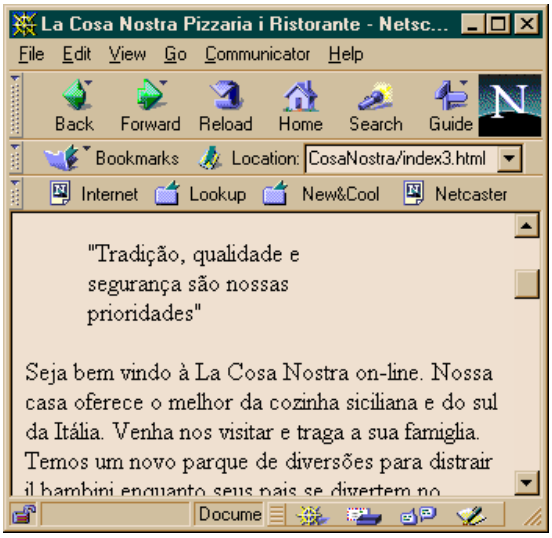
```
<hr width=50%>
```

Há vários outros atributos. Consulte a documentação (ou digite a tecla F1 quando estiver com o cursor dentro do elemento <HR> no *HomeSite*) e teste os possíveis resultados no seu browser.

Linhas horizontais são elementos de bloco, ou seja, eles quebram a linha antes e depois. Você não pode ter uma linha horizontal dentro de um parágrafo ou cabeçalho.

2.2.4. Citação de bloco <BLOCKQUOTE>

Com a função de dar o destaque de um bloco de citação em um texto, foi criado o descritor <BLOCKQUOTE>. Alguns browsers o formatavam em itálico, outros em fonte menor, outros endentavam. Hoje, todos os browsers formatam um bloco BLOCKQUOTE endentado e com uma linha em branco antes e depois. Sua função acabou sendo a de um bloco de parágrafos endentados:



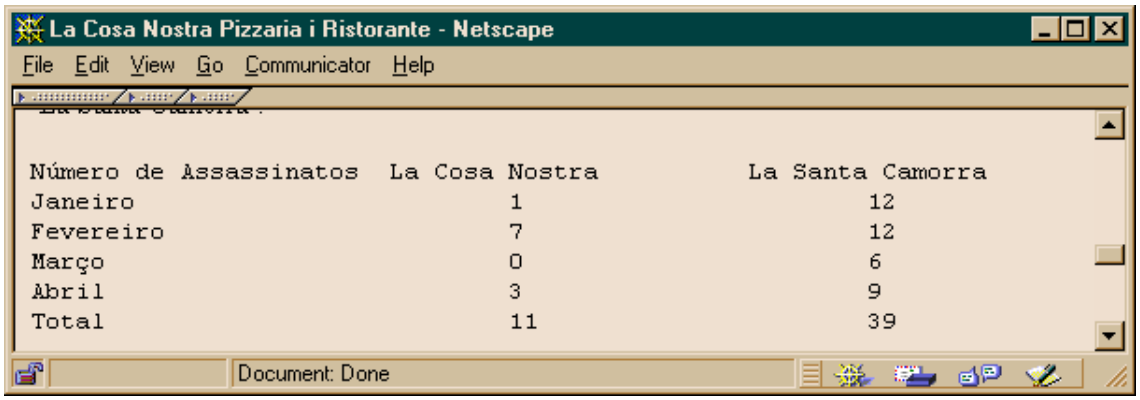

```
<blockquote>
<p>"Tradição, qualidade e<br>
segurança são nossas<br>
prioridades"</p>
</blockquote>
```

Mas <BLOCKQUOTE> não deve ser usado pensando só na aparência. Qualquer folha de estilos pode endentar outro parágrafo ou mudar a aparência de <BLOCKQUOTE>.

2.2.5. Texto pré-formatado <PRE>

Vimos que o HTML ignora espaços, tabulações, novas-linhas e coisas parecidas. Se isto for necessário, podemos resolver o problema usando o descritor <pre>. Esse descritor preserva as características do texto previamente formatado e leva em considerações as novas-linhas, espaços e tabulações:

```
<pre>
<p>Número de Assassinatos      La Cosa Nostra    La Santa Camorra
Janeiro                   1                12
Fevereiro                 7                12
Março                     0                6
Abril                     3                9
Total                     11               39</p>
</pre>
```



Se a largura da tabela passar da largura da página, é só ajustar diminuindo um pouco o número de tabulações ou trocando por espaços que o ajuste fica perfeito.

Há formas melhores de criar tabelas em HTML, como veremos adiante.

2.2.6. Bloco genérico de seção <DIV>

Podemos decidir criar divisões na página pelos mais diversos motivos. Até por motivos de estrutura lógica (mesmo que o resultado não apareça visualmente, pode ser usado por um mecanismo de busca ou indexação). Com folhas de estilo, podemos atribuir um estilo totalmente diferente a uma seção da página, como mudar fundo, fontes, cores, etc.

O descritor <DIV> é usado para definir uma nova seção (ou divisão). Este descritor não faz nada se não tiver atributos. Um dos atributos é ALIGN, que tem a mesma função que o ALIGN de <P> e <H1> só que afeta um bloco inteiro, com todos os descritores que houver dentro. A precedência ocorre de dentro para fora. Se houver um <P ALIGN=center> dentro de um <DIV ALIGN=right>, o alinhamento de <P> prevalece.

2.2.7. Outros blocos

A finalidade desta introdução ao HTML não é explorar todos os elementos disponíveis para se construir páginas, portanto, deixamos alguns outros blocos importantes de fora (como <ADDRESS>, que é usado para marcar endereços) e descritores deprecados como <CENTER> e <NOBR>. Também não exploramos os vários atributos desses elementos. Consulte a especificação ou uma referência HTML para conhecer mais detalhes.

2.2.8. Resumo

parágrafos e controle de fluxo

`<H`*n*`> Texto </H`*n*`>` (onde *n* = 1, 2, 3, 4 ,5 ou 6)

Marca um cabeçalho de nível *n*. Pode ainda receber um atributo `align="posição"`, onde posição pode ser `left`, `center` ou `right`.

`<P> Parágrafo </P>`

Marca um parágrafo. O descritor de fechamento pode, em geral, ser omitido. Também pode receber um atributo `align="posição"` como ocorre com `<H`*n*`>`.

`<BR>`

Marca a posição onde ocorre uma quebra-de-linha em um parágrafo ou cabeçalho.

blocos

`<BLOCKQUOTE> Bloco de Texto </BLOCKQUOTE>`

Endenta um bloco de texto.

`<PRE> Bloco de Texto </PRE>`

Preserva a formatação original (espaços, tabulações, novas-linhas) de um bloco de texto.

`<DIV align="`*posição*`"> Bloco de Texto </DIV>`

Alinha um bloco de acordo com a *posição*, que pode ser `left` (esquerda), `center` (centro) ou `right` (direita).

`<DIV> Bloco de Texto </DIV>`

<DIV align=center> faz o mesmo que `<CENTER>`.

`<ADDRESS> Texto </ADDRESS>`

Marca um texto como endereço (a formatação varia de browser para browser).

separadores

`<HR` *atributos*`>`

`WIDTH=`*m*`% ou WIDTH=`*n*

Determina a largura horizontal da linha, em porcentagem da largura da página ou em pixels.

`SIZE=`*n*

Determina a altura da linha em pixels.

`ALIGN="left | center | right"`

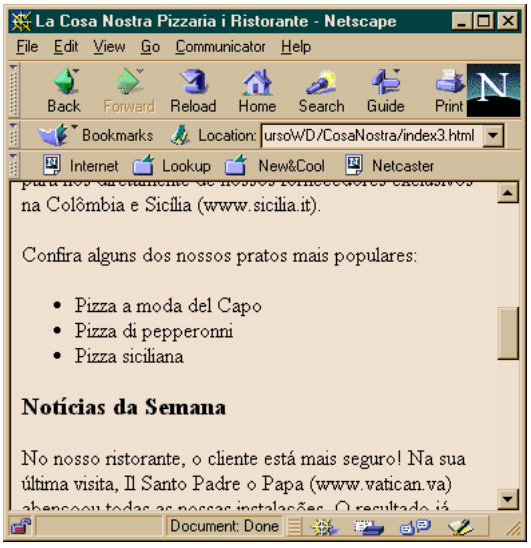
Alinha um separador pela esquerda, pelo centro ou pela direita.

2.3. Listas

HTML define várias formas de apresentar listas de informação em um documento. Toda lista é um elemento de bloco que possui um descritor inicial e final e só pode conter "itens de lista". Os itens de lista também são blocos que podem conter texto, parágrafos ou ainda outras listas. HTML define três tipos de listas: as listas ordenadas, marcadas pelos descritores <OL> e </OL>; as listas não-ordenadas marcadas por <UL> e </UL> e as listas de definições, marcadas por <DL> e </DL>.

2.3.1. Listas não-ordenadas <UL>, <LI>

Nas listas UL, cada item da lista é contido dentro de <LI> e </LI>. O browser geralmente formata os itens com marcadores (bolinhas pretas). Veja a imagem ao lado e o código abaixo:




```
<ul>
  <li>Pizza a moda del Capo</li>
  <li>Pizza di pepperonni</li>
  <li>Pizza siciliana</li>
</ul>
```

Você também pode criar listas de nível hierárquico inferior. O browser formatará a sub-lista de uma forma diferente (geralmente endentada e com um marcador diferente). Por exemplo, para fazer a seguinte lista:

- item 1
- item 2
  - item 2.1
  - item 2.2
- item 3

pode-se utilizar o seguinte código:

```
<ul>
  <li>item 1</li>
  <li>item 2</li>
  <ul>
    <li>item 2.1</li>
    <li>item 2.2</li>
  </ul>
  <li>item 3</li>
</ul>
```

2.3.2. Listas ordenadas <OL>, <LI>

As listas ordenadas são marcadas pelos descritores <OL> e </OL>. Da mesma forma que nas listas não-ordenadas, cada item é contido dentro de <LI>. Poderíamos numerar as pizzas trocando os UL por OL:

```
<ol>
<li>Pizza a moda del Capo</li>
<li>Pizza di pepperonni</li>
<li>Pizza siciliana</li>
</ol>
```

1. Pizza a moda del Capo
2. Pizza di pepperonni
3. Pizza siciliana

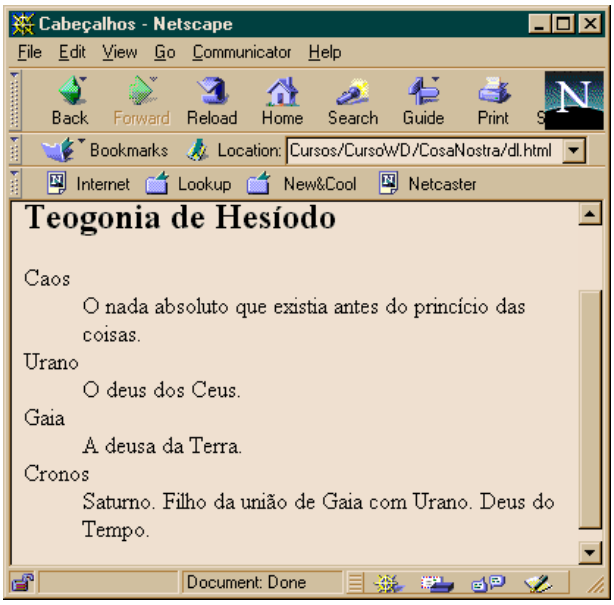
As listas e seus itens podem ter atributos que mudam suas características como tipo de marcador, início da contagem (para listas ordenadas), tipo de numeral, etc. A maior parte dessas alterações pode ser realizada em folhas de estilo.

2.3.3. Listas de definições <DL>, <DT>, <DD>

Finalmente, existem as listas de definições. Esse tipo de lista é usado quando há vários tópicos curtos acompanhados de uma definição mais longa (em um glossário, por exemplo). Consiste de três descritores: <DL> e </DL> que delimitam toda a lista; <DT></DT> que marca o termo; e <DD></DD> marca a definição.

A imagem ao lado foi produzida com os descritores abaixo:

```
<dl>

<dt>Caos</dt>
  <dd>O nada absoluto que existia antes do princípio das coisas.</dd>
<dt>Urano</dt>
  <dd>O deus dos Céus.</dd>
<dt>Gaia</dt>
  <dd>A deusa da Terra.</dd>
<dt>Cronos</dt>
  <dd>Saturno. Filho da união de Gaia com Urano. Deus do Tempo.</dd>

</dl>
```



2.3.4. Resumo

`<UL> ... </UL>`
: Lista não-ordenada

`<OL> ... </OL>`
: Lista ordenada

`<LI> ... </LI>`
: Elemento de lista. O descritor final pode ser omitido. Tanto **<LI>** como **<UL>** e **<OL>** podem receber atributos que modificam a apresentação da lista

`<DL> ... </DL>`
: Lista de definições

`<DT> ... </DT>`
: Termo. O descritor final pode ser omitido.

`<DD> ... </DD>`
: Definição do termo de lista. O descritor final pode ser omitido.

2.4. Imagens

Já vimos que uma página HTML é uma mera página de texto. Mesmo com imagens, ela continuará a ser uma mera página de texto, não vai aumentar de tamanho pois as imagens sempre ficarão separadas. As imagens são carregadas na hora em que o browser lê a página, através de um vínculo (link) para a sua localização através de uma URL. Ela pode estar no mesmo diretório que a página ou em outra parte do mundo. Desde que se informe o seu endereço corretamente e que a rede não esteja sem comunicação, o browser localizará a imagem e a colocará na página no local onde estiver o descritor <IMG> que a solicitou.

2.4.1. Objeto Imagem <IMG>

<IMG> é como <BR> e <HR> que não marcam texto, mas uma posição, por isso não têm descritores de fechamento. No caso de <IMG>, o atributo SRC é obrigatório pois é ele quem informa a localização da imagem através de sua URL. A URL pode ser tanto um endereço absoluto (http://alguma.coisa/) como um endereço relativo à URL da própria página. Por exemplo, se a URL página está em http://blablabla.com/dirdir/pagina.html e a URL da imagem que a página carrega é http://blablabla.com/dirdir/imagem.gif, pode-se simplesmente chamar a imagem pelo endereço relativo "imagem.gif", usando <IMG SRC="imagem.gif"> na posição, relativa ao texto, onde ela deva aparecer.

URIs relativas

O browser, como está remoto, precisa utilizar uma *URI absoluta* que contenha o protocolo, o nome da máquina onde roda o servidor e a porta de serviços (se não for a porta padrão). A URI usada internamente no servidor (o sistema de arquivos virtual) contém apenas o cami-

nho absoluto dentro do mesmo (/, /Imagens ou /Videos). As páginas HTML disponíveis via servidor Web estão armazenadas em determinada máquina e diretório e podem usar, para comunicação entre si, *URIs relativas* à sua localização.

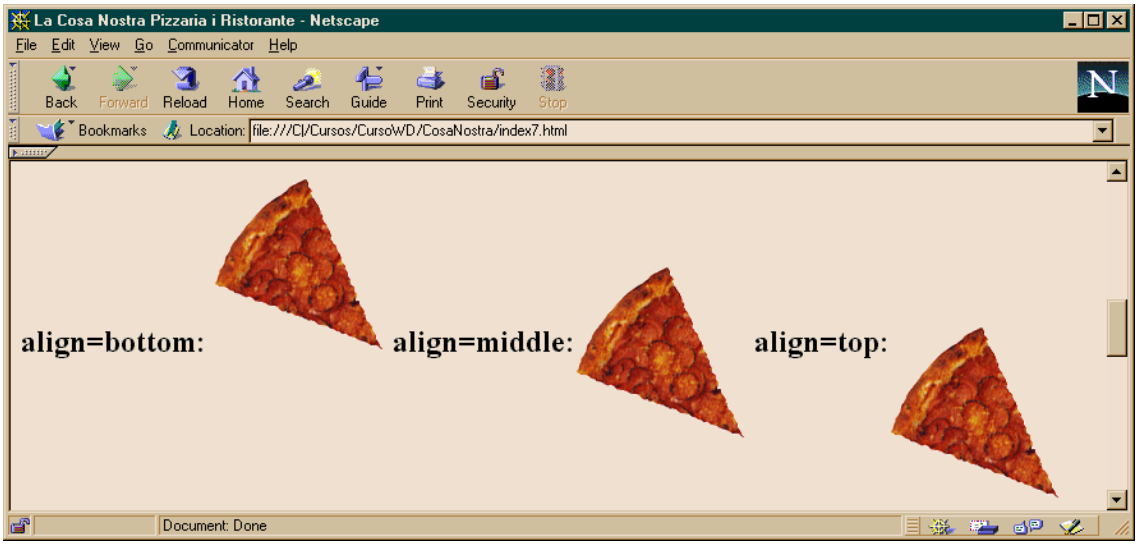
As URIs relativas são úteis em HTML. Um vínculo em uma página pode se referir a um arquivo que ocupa o mesmo diretório que ela simplesmente usando o nome do arquivo como URI (e omitindo http://maquina:porta/caminho/). Para ter acesso a um diretório anterior, usa-se a notação Unix: ../arquivo.txt refere-se a um arquivo chamado arquivo.txt localizado no diretório anterior..

Resumindo: servidores e browsers só localizam recursos usando URIs. Não faz sentido escrever C:\qualquercoisa\x.txt no browser a não ser para utilizá-lo como visualizador de páginas HTML armazenadas localmente. Também não faz sentido usar caminhos desse tipo em arquivos HTML (como veremos). Eles podem até funcionar localmente, sem servidor, mas não mais funcionarão quando forem publicados em um servidor Web. O ideal é usar URIs relativas que funcionam em qualquer situação.

2.4.2. Alinhamento de imagens

A imagem pode ser alinhada de várias formas em relação ao texto usando HTML (ou folhas de estilo). Veja abaixo alguns exemplos.

```
<img src="pizza_calabresa.gif" align=middle> e outros...
```

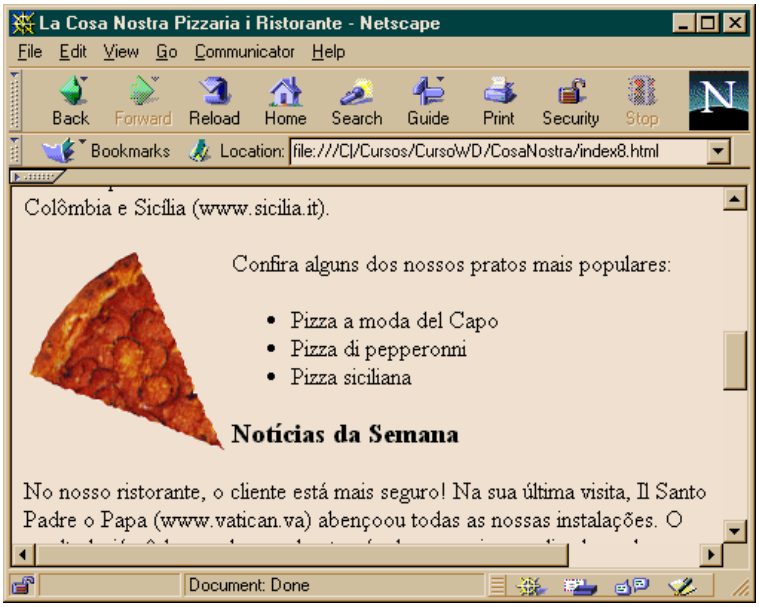


Use ALIGN=left ou ALIGN=right para fazer com que o texto corra à esquerda ou à direita da imagem:

```
<img src="pizza_calabresa.gif" align=left>
```

2.4.3. Dimensionamento

Se a imagem ficar muito grande, podemos reduzi-la um pouco usando os atributos HEIGHT e WIDTH. Através deles podemos mudar quaisquer dimensões da imagem ao ser exibida, em pixels. Mas cuidado! A qualidade da imagem e seu tamanho continuam os mesmos. Carregar uma imagem gigantesca para depois reduzi-la é desperdício assim como há perda de qualidade ao se ampliar uma imagem pequena.



Height e width não servem só para redimensionar imagens. Usando esses atributos em todas as suas imagens com as suas dimensões verdadeiras torna o carregamento da página mais rápido. Por que isto? Porque o browser não vai precisar calcular as dimensões da imagem para saber onde colocar o texto. Se não houver height e width ele só pode mostrar o texto depois de saber o tamanho das imagens. Para descobrir o tamanho da imagem, use um editor gráfico como o Lview ou PhotoShop. Que tal uma pizza esticada?

```
<img src="pizza_calabresa.gif" align=left height=100 width=20>
```

Se você não gostou da pizza esticada, use as dimensões corretas em pixels que são height=137 e width=134.

Ainda há alguns atributos que permitem que se empurre o texto para longe da figura. Na verdade, criam uma margem invisível ao redor dela. HSPACE e VSPACE, respectivamente acrescentam margens horizontais e verticais, em pixels, em volta da figura. Todos os atributos de alinhamento podem ser substituídos por folhas de estilo.

```
<img src="pizza_calabresa.gif" align=left hspace=10>
```

2.4.4. Resumo

`<IMG SRC="URL da imagem"` *atributos*>

Inclui uma imagem na página. **SRC** localiza a imagem através de uma URL. Principais atributos

`ALT="comentário"`

Inclui um comentário que será visto por quem não conseguir visualizar a imagem.

`BORDER="n"`

Cria uma borda em torno da imagem. BORDER="0" remove a borda presente ao redor das imagens ativas (que são links).

`ALIGN="bottom | middle | top | left | right"`

Alinha uma figura em relação ao texto. As três primeiras opções alteram a posição vertical da figura. As duas últimas fixam a mesma em um dos lados da página, fazendo o texto fluir ao redor.

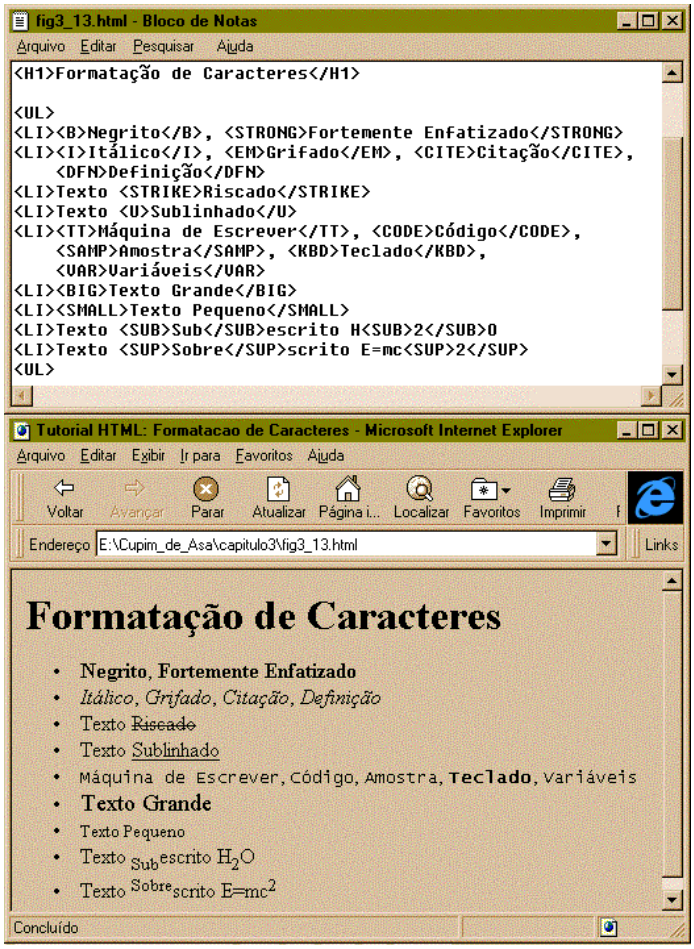
`HSPACE="n"; VSPACE="n"`

Colocam espaço horizontal e/ou vertical ao redor de uma imagem.

2.5. Formatação de caracteres (inline elements)

Há vários descritores que tem a função de atribuir um papel a trechos de texto. Na prática, este "papel" se reflete na forma de um destaque diferente. Somente com folhas de estilo o "papel" é aproveitado totalmente.

Os descritores de formatação de caracteres são divididos em duas partes: os lógicos (aqueles que tem uma função) e os físicos (aqueles que não tem função estrutural, só aparência). A única diferença entre eles está no nome: CITE (lógico) representa funcionalmente uma citação e aparece na tela em itálico. I (físico) representa texto em itálico (funcionalmente não é nada), e se apresenta na tela da mesma forma que CITE. Se você não está preocupado com estrutura tanto faz usar um ou outro. A figura abaixo ilustra vários deles.



2.5.1. Elementos

Os elementos abaixo significam mais pela sua função que pela sua aparência. São elementos lógicos.

Sintaxe: <Elemento> Texto </Elemento>, onde Elemento pode ser:

EM texto grifado

STRONG	texto fortemente grifado
DFN	definição
CODE	código
SAMP	amostra
KBD	teclado
VAR	variável
CITE	citação
SAMP	amostra
ABBR	abreviação
ACRONYM	sigla
SPAN	container genérico
Q	texto de citação (entre aspas)

Estes outros elementos têm um nome que enfatiza a sua aparência (são os elementos físicos). Folhas de estilo, porém, podem alterar tudo.

TT	texto de espaçamento fixo
I	texto em itálico
B	texto em negrito
S	texto riscado (mesmo que STRIKE)
STRIKE	texto riscado
U	texto sublinhado
BIG	texto em fonte maior
SMALL	texto em fonte menor
SUB	texto sub-escrito
SUP	texto sobre-escrito

2.6. Tabelas

Até a versão 2.0, HTML só fornecia meios de estruturar informação que fosse formatada seqüencialmente, ou seja, o fluxo do texto e imagens era único e acontecia da esquerda para a direita e de cima para baixo. Não havia meios de dispor, por exemplo, dois parágrafos lado-a-lado a não ser que todo o texto fosse formatado usando espaços e tabulações e posteriormente incluído em um bloco <PRE>. A introdução das tabelas, proposta no HTML 3.0 (mas adotada pelos browsers mais populares como uma extensão do HTML 2.0), tornou possível a classificação da informação com marcadores HTML que seria formatada no browser em linhas e colunas.

Os marcadores eram inicialmente destinados à organização de dados em forma de tabelas, mas em pouco tempo passaram a ser usados pelos Web designers para obter maior controle sobre a apresentação do conteúdo das suas páginas. Imagens com partes dinâmicas, páginas

de mais de uma coluna e controle da largura da página são algumas das soluções possíveis em HTML usando tabelas.

Por ter tantas utilidades, as tabelas são um tópico complexo em HTML, mas as possibilidades que oferecem compensam o esforço de conhecer bem cada um dos seus atributos.

2.6.1. Principais elementos estruturais <TABLE>, <TR>, <TD> ou <TH>

Uma tabela precisa ter no mínimo três elementos HTML. O elemento <TABLE> representa a tabela e contém uma ou mais linhas, representadas pelo elemento <TR>. O elemento <TR> é o único, dentro de um bloco <TABLE>...</TABLE> que contém a informação da tabela. Essa informação precisa estar dentro de elementos <TD> ou <TH>, que são os únicos permitidos dentro de um <TR>. <TR>...</TR> representa uma linha da tabela (*Table Row*). Cada linha pode ter uma ou mais células de dados em blocos <TD>...</TD> (*Table Data*) ou <TH>...</TH> (*Table Header*). O número de colunas da tabela é, portanto, derivado do número de células contidas no bloco <TR>...</TR> que tiver mais blocos <TH>...</TH> ou <TD>...</TD> .

Os exemplos abaixo ilustram tabelas simples:

```
<p><table border>
<tr> <td>1,1</td> <td>1,2</td> <td>1,3</td> </tr>
<tr> <td>2,1</td> <td>2,2</td> <td>2,3</td> </tr>
<tr> <td>3,1</td> <td>3,2</td> <td>3,3</td> </tr>
</table>
```

que resulta na seguinte formatação:

1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3

```
<TABLE BORDER=1>
<TR><TD>Mercúrio</TD><TD>Venus</TD><TD>Terra</TD></TR>
<TR><TD>Marte</TD><TD>Júpiter</TD><TD>Saturno</TD></TR>
<TR><TD>Urano</TD><TD>Netuno</TD><TD>Plutão</TD></TR>
</TABLE>
```

Mercurio	Venus	Terra
Marte	Jupiter	Saturno
Urano	Netuno	Plutão

BORDER é um atributo de <TABLE> que faz com que a tabela seja desenhada com uma borda *default* (1 pixel). Se o nome BORDER não estiver presente, a borda não aparecerá,

embora o espaço de 1 pixel em volta da tabela permaneça. Veja o exemplo abaixo (tabela simples sem bordas):

```
<TABLE>
 <TR><TD>Maranhão</TD><TD>Piauí</TD><TD>Ceará</TD></TR>
 <TR><TD>Rio Grande do Norte</TD><TD>Paraíba</TD>
     <TD>Pernambuco</TD></TR>
 <TR><TD>Alagoas</TD><TD>Sergipe</TD><TD>Bahia</TD></TR>
</TABLE>
```

Maranhão	Piauí	Ceará
Rio Grande do Norte	Paraíba	Pernambuco
Alagoas	Sergipe	Bahia

Os elementos <TD> e <TH> de <TABLE> podem conter qualquer texto ou elementos HTML utilizados dentro de <BODY>, como imagens:

- Altímetro
- Taquímetro
- Indicador de velocidade do vento
- Bússola
- Nível de combustível e óleo
- Termômetros

```
<table>
 <tr><td><img src="ball.gif"></td> <td>Altímetro    </td></tr>
 <tr><td><img src="ball.gif"></td> <td>Taquímetro   </td></tr>
 <tr><td><img src="ball.gif"></td> <td>Indicador ...</td></tr>
 <tr><td><img src="ball.gif"></td> <td>Bússola      </td></tr>
 <tr><td><img src="ball.gif"></td> <td>Nível de ... </td></tr>
 <tr><td><img src="ball.gif"></td> <td>Termômetros  </td></tr>
</table>
```

2.6.2. Bordas e espaçamento

O elemento <TABLE> contém vários outros atributos que alteram a aparência de toda a tabela. BORDER=valor permite especificar a largura da borda da tabela através de um valor absoluto em pixels. A tabela abaixo possui uma borda de 10 pixels:

Estate	Autunno
Primavera	Inverno

```
<TABLE BORDER=10>
<TR><TD><a href="estate.html">Estate</a></TD>
```

```
    <TD><a href="autunno.html">Autunno</a></TD></TR>
<TR><TD><a href="primavera.html">Primavera</a></TD>
    <TD><a href="inverno.html">Inverno</a></TD></TR>
</TABLE>
```

`CELLSPACING=valor` controla a quantidade de espaço entre as células de uma tabela. Normalmente esse espaço é de 1 pixel. `CELLPADDING=valor` controla a quantidade de espaço entre o texto da célula e suas bordas (margens internas das células). O valor *default* é de 2 pixels, portanto, se as bordas forem zero, haverá 3 pixels entre os elementos da tabela a não ser que esse espaço seja alterado com `CELLSPACING` e `CELLPADDING`. Veja os exemplos abaixo:

Aranha	Lacrau	Escorpião
Barata	Formiga	Mosca

```
<TABLE BORDER CELLPADDING=10 CELLSPACING=0>
  <TR><TD>Aranha</TD><TD>Lacrau</TD><TD>Escorpião</TD></TR>
  <TR><TD>Barata</TD><TD>Formiga</TD><TD>Mosca</TD></TR>
</TABLE>
```

Aranha	Lacrau	Escorpião
Barata	Formiga	Mosca

```
<TABLE BORDER CELLPADDING=0 CELLSPACING=10>
   (...) </TABLE>

<TABLE BORDER CELLPADDING=10 CELLSPACING=10>
  (...) </TABLE>
```

Aranha	Lacrau	Escorpião
Barata	Formiga	Mosca

Se os atributos BORDER, CELLPADDING e CELLSPACING forem definidos com o valor "0", não haverá espaço algum entre os elementos da tabela a não ser que o espaço seja adicionado de forma explícita (linhas em branco,   tabulações ou espaços dentro do bloco <TD>).

2.6.3. Largura e altura

O atributo WIDTH=numero ou percentagem permite especificar a largura da tabela, através de uma medida absoluta em pixels ou através de uma percentagem da largura visível da página. Sem WIDTH, a tabela ocupará apenas o espaço horizontal necessário para conter to-

dos os seus elementos, as bordas e margens. Se uma tabela for definida com WIDTH=100%, ela ocupará toda a largura visível da página e mudará de tamanho caso a janela seja redimensionada (e a área visível seja alterada). Se for definida com WIDTH=500, terá uma largura fixa e que não mudará mesmo com o redimensionamento da janela.

No uso de tabelas como ferramenta para o layout das páginas, WIDTH é usado para determinar a largura da página que está contida em uma tabela de única célula. Páginas criadas para a resolução de 640x480 são tipicamente definidas dentro de tabelas com largura de 600 pixels. Para 800x600, é comum usar 750 pixels:

```
<body>
<table width=750>
<tr><td> (... a página inteira ...) </td></tr>
</table>
</body>
```

No exemplo abaixo há duas tabelas. A primeira não possui o atributo WIDTH. A segunda possui o atributo WIDTH=50%.

16	25	26
34	40	43

16	25	26
34	40	43

```
<TABLE BORDER>
<TR><TD>16</TD> <TD>25</TD> <TD>26</TD></TR>
<TR><TD>34</TD> <TD>40</TD> <TD>43</TD></TR>
</TABLE>

<TABLE BORDER WIDTH="50%">
<TR><TD>16</TD> <TD>25</TD> <TD>26</TD></TR>
<TR><TD>34</TD> <TD>40</TD> <TD>43</TD></TR>
</TABLE>
```

De forma análoga ao atributo WIDTH, HEIGHT=numero ou percentagem permite especificar a altura da tabela. O suporte a esse recurso, porém, não é consistente em browsers que não suportam totalmente o HTML 4.

2.6.4. Atributos de posicionamento, cor e imagens de fundo

A tabela, se tiver uma largura menor que o espaço horizontal visível, será alinhada pela margem esquerda da página a não ser que possua o atributo ALIGN com o valor right (à direi-

ta) ou center. HSPACE e VSPACE podem ser usados para acrescentar espaço vertical e horizontal como margens externas para toda a tabela, da mesma forma como são usados por imagens.

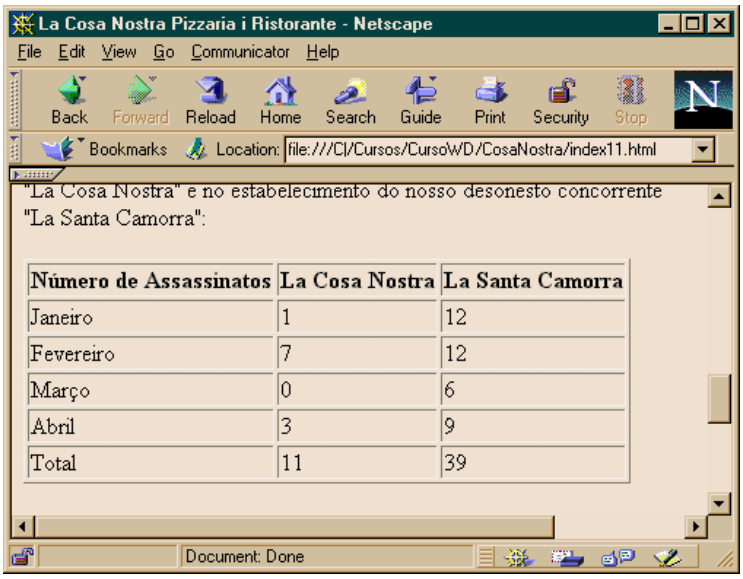
O atributo BACKGROUND, se presente, deve receber a URL de uma imagem que irá se repetir no fundo da tabela. BGCOLOR permite que se defina uma cor. É importante observar que a cor default da tabela (TABLE) das células (TD) e linhas (TR) é transparente, ou seja, se houver imagens ou cores de fundo na página, elas serão visíveis através da tabela. Isto é verdade apenas se a tabela ou células não tiverem atributos BACKGROUND e BGCOLOR. A cor da tabela poderá não ser visível se cores opacas tiverem sido atribuídas a blocos TD e TR.

Veja agora a tabela que antes construímos com <PRE>:

```
<table border>
<tr><th>Número de Assassinatos</th><th>La Cosa Nostra</th><th>La Santa Ca-
morra</th></tr>
<tr><td>Janeiro    </td><td>        1 </td><td>  12</td></tr>
<tr><td>Fevereiro </td><td>        7 </td><td>  12</td></tr>
<tr><td>Março      </td><td>        0 </td><td>  6 </td></tr>
<tr><td>Abril      </td><td>        3 </td><td>  9 </td></tr>
<tr><td>Total      </td><td>       11</td><td>  39</td></tr>
</table>
```

A tabela fica melhor alinhada pelo centro. Use <DIV> para fazer isto.

```
<div align="center">
<table border>
(...)
</table>
</div>
```



Número de Assassinatos	La Cosa Nostra	La Santa Camorra
Janeiro	1	12
Fevereiro	7	12
Março	0	6
Abril	3	9
Total	11	39

Observe que, por *default*, os dados de <TH> são alinhados pelo centro, enquanto que os dados de <TD> são alinhados com a margem esquerda. Tudo isto e muito mais podem ser alterados usando atributos como ALIGN, VALIGN (alinhamento vertical) além, é claro, folhas de estilo.

Tabelas tendem a ficar muito complexas, principalmente quando usadas umas dentro das outras, o que é bastante comum no HTML. É importante entender como funcionam as tabelas mas o ideal é criá-las através de programas gráficos como o *DreamWeaver* ou *FrontPage*. Às vezes, porém, é necessário fazer pequenas alterações por causa de diferenças entre browsers e assim, saber identificar as células de uma tabela no código HTML é essencial.

2.6.5. Alinhamento e dimensionamento de células

Assim como as tabelas, células e linhas individuais de tabelas podem ter seu conteúdo alinhado em relação à tabela sem usar folhas de estilo através dos atributos ALIGN e VALIGN aplicados a <TD> e <TR>. ALIGN pode assumir os valores left, right ou center. VALIGN pode ser top, bottom ou center. Com os dois é possível colocar um objeto alinhado em qualquer posição da tabela.

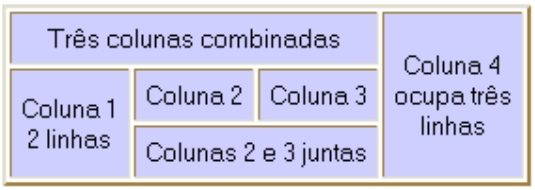
Qualquer <TD> também pode ter atributos HEIGHT e WIDTH para controlar a altura e largura da célula, respectivamente. Esse valor pode ser absoluto (em pixels) ou relativo à altura/largura totais da tabela.

Mas HEIGHT e WIDTH devem ser usados com cautela. Nem sempre é possível determinar essas dimensões. Blocos <NOBR> e <PRE> podem fazer a tabela crescer além de sua largura máxima estabelecida. A altura não pode ser limitada pois o texto contido na tabela sempre pode fazer com que ela tenha que crescer. Imagens tem todo o poder para destruir completamente quaisquer limitações de altura e largura definidas nas tabelas.

2.6.6. Combinação de células

Células podem ser mescladas em uma só atravessando colunas e linhas com os atributos ROWSPAN e COLSPAN aplicáveis a <TD> ou <TH>. Com esses recursos é possível construir tabelas

O seguinte exemplo utiliza COLSPAN e ROWSPAN para construir uma tabela contendo várias células mescladas:




```
<table border=2 cellpadding="5">
<tr>
 <td colspan="3">Três colunas combinadas</td>
 <td rowspan="3">Coluna 4<br>ocupa três<br>linhas</td>
</tr>
<tr>
 <td rowspan="2">Coluna 1<br>2 linhas</td>
 <td>Coluna 2</td>
 <td>Coluna 3</td>
</tr>
<tr>
 <td colspan="2">Colunas 2 e 3 juntas</td>
</tr>
</table>
```

2.6.7. Resumo

`<TABLE atributos> ... </TABLE>`

Define uma tabela. O atributo `border` torna visíveis as bordas das células.

`<CAPTION> Legenda </CAPTION>`

Legenda da tabela

`<TR> ... </TR>`

Linha de uma tabela. O descritor final pode ser omitido. Só pode incluir descritores **<TH>** e **<TD>**.

`<TD> ... </TD>`

Célula comum. O descritor final pode ser omitido.

`<TH> ... </TH>`

Célula de cabeçalho. O descritor final pode ser omitido.

2.7. Âncoras e Vínculos

Uma âncora é qualquer um dos lados de um vínculo de hipertexto. Em outras palavras, tanto a origem quanto o destino de um link são âncoras. Você pode criar uma âncora destino em uma página HTML usando o elemento <A> com atributo NAME (para dar um nome à âncora):

```
<A NAME="cap1"></A> <H1>Capítulo 1 </H1>
```

O descritor pode ficar vazio, já que só marca uma posição, mas o marcador final é necessário.

Para criar um vínculo à âncora, utiliza-se o mesmo elemento <A>, em outro lugar da página ou em outra página, marcando um texto ou imagem que servirá de ponto de partida, indicando o destino através do atributo HREF:

```
<A HREF="#cap1">Ir até o capítulo 1</A>
```

O trecho acima vincula a frase "Ir ao capítulo 1" com a posição onde está a âncora "cap1", desde que ela esteja definida no mesmo arquivo.

Para especificar um destino externo, é preciso usar uma URI (relativa ou absoluta). A URI pode apontar para qualquer coisa (não precisa ser uma página HTML).

```
<A HREF="http://www.abc.com/dados/index.html">Ir até o arquivo In-
dex.html</A>
```

Se o destino for uma determinada seção da página, pode-se complementar a URI com o fragmento correspondente ao nome da âncora definida no arquivo:

```
<A HREF="http://www.abc.com/dados/index.html#cap1">Ir até o capítulo
1 no arquivo Index.html</A>
```

2.7.1. Resumo

`<A HREF="URL do recurso vinculado"> ... </A>`

Faz com que o texto marcado seja um link para a URL especificada.

`<A HREF="#nome"> ... </A>`

Faz com que o texto marcado seja um link para a âncora local nome.

`<A NAME="nome"></A>`

Marca um lugar (âncora destino) em uma página.

2.8. Elementos de estilo *(deprecados no HTML 4.0)*

Não precisamos ficar satisfeitos com o fundo cinza, letras pretas e links azuis. Podemos mudar todas as cores. Veremos futuramente como fazer tudo isto usando folhas de estilo, mas o HTML também oferece meios de realizar alterações de estilo. A especificação HTML recomenda fortemente que esses elementos não sejam usados, mas que se prefira usar folhas de estilo, mas, como o suporte a folhas de estilo não chega aos browsers versão 3.0, e você ainda pode ter clientes com tais browsers que não vão ficar satisfeitos com páginas monocromáticas, é importante conhecer esses elementos. À medida em que os browsers versão 3.0 forem entrando em extinção, deve-se gradualmente abandonar toda a formatação de cores e fontes através do HTML, pois prejudica bastante a manutenção de um site, além de torná-lo maior e mais sujeito a erros.

2.8.1. Atributos de uma página <BODY atributos>

Os atributos de uma página (fundo, cores do texto e dos vínculos, papel de parede) podem ser mudados através do descritor <BODY>. Diversos atributos permitem definir uma imagem de papel de parede, uma cor de fundo, cor para todo o texto e para os diversos estados de um vínculo. As cores são informadas usando ou um código RGB em hexadecimal ou um nome padronizado.

Para mudar a cor de fundo, utilizamos o atributo BGCOLOR.

```
<body bgcolor="white" text="green" link="red" vlink="black">
```

Usamos nomes de cores. Se quiséssemos usar uma cor diferente das 16 cores daquela tabela básica (veja apêndice), precisaríamos usar códigos hexadecimais. Os códigos são um conjunto de 6 algarismos hexadecimais precedidos de um "#". Pode-se, através deles, obter mais de 200 cores diferentes.

O papel de parede é uma imagem que é repetida várias vezes no fundo. Pode-se usar qualquer imagem mas ela não deve atrapalhar a leitura do texto. Deve ser informada a URL completa ou relativa:

```
<body background="fundo.jpg"
 bgcolor="white" text="green" link="red" vlink="black">
```

2.8.2. Cores e fontes <FONT>

Cores de texto individuais, fontes e tamanhos podem ser alterados com um descritor chamado <FONT>, que pode receber três atributos:

- SIZE, que informa o tamanho (variando de 1 a 7 ou de -3 a +3);
- FACE, que informa o nome exato de uma fonte ou uma lista delas separadas por vírgula; e
- COLOR, que informa uma cor em código hexadecimal ou pelo nome, da mesma forma como é feito em BODY.

<FONT> é totalmente dispensável forem utilizadas folhas de estilo, mas, como alguns browsers ainda não suportam folhas de estilo, pode-se usar esse elemento para garantir uma aparência melhor às páginas Web.

2.8.3. Resumo

`<BODY` *atributos*`>` ... *todo o documento* ... `</BODY>`

`BGCOLOR="#rrggbb"` ou `BGCOLOR="nome_da_cor"`

Muda a cor do fundo da área de visualização. `#rrggbb` é o código RGB das 16,8 milhões de cores possíveis. `nome_da_cor` pode ser uma das 16 cores universalmente conhecidas ou uma das 465 suportadas pelo Netscape.

`TEXT="#rrggbb"` ou `TEXT="nome_da_cor"`

Mudam a cor de todo o texto que aparece no documento, com exceção dos links.

`LINK="#rrggbb"` ou `LINK="nome_da_cor"`

Muda a cor dos links ainda não selecionados.

`VLINK="#rrggbb"` ou `VLINK="nome_da_cor"`

Muda a cor dos links já visitados.

`BACKGROUND="URL_de_imagem"`

Inclui uma imagem cobrindo todo o fundo da tela.

2.9. Exercícios

2.9.1. Testes sobre Web e HTML

1. Marque as opções verdadeiras. Uma página HTML ...
   a) ... geralmente é um arquivo de texto com extensão `.htm` ou `.html`
   a) ... poderá exibir imagens, texto formatado, vínculos de hipertexto e cores ao ser lida por uma aplicação como o *Internet Explorer*.
   b) ... exibirá seu código-fonte, consistindo de símbolos especiais entre "<" e ">" quando for lida por um editor de textos comum, e não mostrará imagens.
   c) ... não pode ser criada através de um editor de textos qualquer. É preciso usar uma ferramenta como o *DreamWeaver* ou *HomeSite*.
   d) ... pode ter vínculos interligando-a com outras páginas e com imagens.
   e) ... se tiver vínculos deve expressá-los usando a notação de URIs, absolutas ou relativas, para que funcionem quando a página for publicada em um servidor Web.

2. O que acontece quando um browser carrega um arquivo de texto simples, com extensão `.html` ou `.htm`, mas sem formatação HTML alguma?
   a) A página não é carregada
   f) A página é carregada mas toda a formatação do texto original (parágrafos, títulos, quebras de linha, etc.) é perdida na visualização
   g) A página é interpretada como texto simples e exibida em fonte de largura fixa (Courier, por exemplo), preservando a formatação original
   h) O browser causa a abertura de uma aplicação para a leitura de textos (o *Word*, por exemplo).
   i) O browser exibe uma mensagem de erro

3. Considere o seguinte sistema de diretórios no *Windows*, em uma máquina acessível via Internet chamada `www.tribos.com.br`:

   `C:\`
   `C:\Windows`
   `C:\Apache`
   `C:\Apache`**`\htdocs\`**
   `C:\Apache`**`\htdocs\ongs\`**

Suponha que um servidor HTTP foi instalado nessa máquina e configurado para administrar um sistema de diretórios a partir do diretório `C:\Apache\htdocs\` (diretório raiz do servidor). Qual das URLs abaixo permite a visualização do arquivo `index.html`, armazenado em `C:\Apache\htdocs\ongs\` através de um browser localizado em uma máquina remota?

a) `http://www.tribos.com.br/Apache/htdocs/ongs/index.html`
j) `http://www.tribos.com.br/C:/Apache/htdocs/ongs/index.html`
k) `http://www.tribos.com.br/ongs/index.html`
l) `http://www.tribos.com.br/C:/Apache/htdocs/index.html`
m) `http://www.tribos.com.br/index.html`
n) `http://www.tribos.com.br/`
b) `C:\Apache\htdocs\ongs\index.html`

4. No mesmo servidor Web da questão anterior há uma imagem "`logotipo.gif`" armazenada no diretório `C:\Apache\htdocs\`. O arquivo `index.html` localizado em `C:\Apache\htdocs\ongs\` refere-se a essa imagem através de um descritor `<IMG>`. Quais, entre as sintaxes abaixo para o descritor `<IMG>` de `index.html`, causarão a exibição da imagem dentro da página, quando visualizada por um browser em uma máquina remota? Marque no mínimo uma.
   a) `<IMG SRC="logotipo.gif">`
   b) `<IMG SRC="C:\Apache\htdocs\logotipo.gif">`
   c) `<IMG SRC="../logotipo.gif">`
   d) `<IMG SRC="/logotipo.gif">`
   e) `<IMG SRC="http://www.tribos.com.br/logotipo.gif">`
   f) `<IMG SRC="http://www.tribos.com.br/Apache/htdocs/logotipo.gif">`

5. Qual dos caminhos abaixo não é uma URI (ou URL)?
   a) `ftp://usuario:senha@maquina.com/pub/arquivo.doc`
   b) `news:comp.lang.java`
   c) `http://www.algumlugar.com:8081/textos/`
   d) `c:\wd\paginas\html\`
   e) `/progbusca.exe?opcoes=abc&pesquisa=dracula`
   f) `mailto:helder@ibpinet.net`

6. Marque apenas as alternativas verdadeiras:
   a) Em um arquivo HTML, um bloco de texto cercado por um par de descritores HTML poderá não ter sua aparência alterada quando o arquivo for carregado em um browser.
   b) Atributos só podem ocorrer no descritor inicial de um elemento HTML
   c) Atributos podem ocorrer em qualquer ordem dentro de um descritor HTML.
   d) Descritores e atributos desconhecidos pelo browser são ignorados
   e) Os valores dos atributos devem ser definidos entre aspas, sempre.

7. O que acontecerá com o trecho de código HTML abaixo

```
<!-- Informação muito importante! -->
```

   quando a página no qual está contigo dor visualizada em um browser?
   a) Ele aparecerá em destaque na barra de título do browser

b) Ele aparecerá na página entre duas linhas horizontais
c) Ele será completamente ignorado pelo browser
d) Ele não aparecerá na página mas causará o aparecimento de uma linha em branco no lugar onde deveria aparecer.
e) O browser mostrará uma mensagem de erro.

8. Quais trechos de código HTML abaixo estão incorretos? Marque um ou mais.
   a) `<p>Este é um parágrafo <b>importante</b>.</p>`
   b) `<h1>Título do capítulo</h1 align="center">`
   c) `<td>Voltar para <a href="casa.html">Casa</a></td>`
   d) `<p><a href="../index.html"><b>Retornar</a></b></p>`
   e) `<p>Parágrafo um. <p>Parágrafo dois.`
   f) `<A HREF="arquivo.gif">Link para imagem</A>`
   g) `<IMG SRC = "imagem.jpg" ALT = "Figura 1">`
   h) `<p align=center>Texto alinhado pelo centro</p>`
   i) `<p>O elemento <table> serve para construir tabelas</p>`
   j) `<p>Copyright &copy 2000, José Severino da Silva</p>`

9. Marque apenas as afirmações falsas:
   a) Espaços extras, tabulações e quebras de linha que ocorrem no texto são geralmente ignorados em páginas HTML.
   b) Tudo o que aparece na página deve estar dentro de um bloco delimitado em HTML pelos descritores `<BODY>` e `</BODY>`.
   c) O elemento `<META>` serve para incluir informações sobre a página que podem ser úteis para facilitar a localização da página por mecanismos de busca.
   d) Se uma página tiver o descritor `<BASE TARGET="nova">`, clicar em um de seus vínculos (*links*) poderá causar a abertura de novas janelas.
   e) O título do documento, expresso entre `<TITLE>` e `</TITLE>` aparece no início da página HTML, antes do restante do texto ou imagens.

10. Para criar uma lista simples de itens numerados, qual conjunto de elementos HTML você utilizaria? Escolha uma alternativa.
    a) `<P> e <LI>`
    b) `<UL> e <LI>`
    c) `<OL> e <LI>`
    d) `<TD> e <LI>`
    e) `<UL>, <OL> e <LI>`

11. Quais trechos de código HTML abaixo sempre serão exibidos em itálico em qualquer browser? Marque uma ou mais opções.
    a) `<I>Texto grifado</I>`
    b) `<EM>Texto grifado</EM>`
    c) `<CITE>Texto grifado</CITE>`
    d) `<ADDRESS>Texto grifado</ADDRESS>`
    e) Nenhuma das alternativas acima garante que o texto aparecerá em itálico.

Para as três próximas questões a seguir, considere a seguinte estrutura de arquivos (em *itálico*) e diretórios (em **negrito**):

```
/____ index.html
|____ documentos
|               |____ mar.html
|               |____ ondas.jpg
|____ imagens
            |___ barco.gif
```

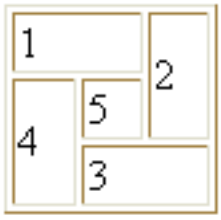
12. Qual das alternativas abaixo contém o código HTML para que o arquivo `index.html` mostre a imagem `barco.gif` e contenha um *link* para o arquivo `mar.html`? Marque uma alternativa.
    a) `<img src="barco.gif"> <a href="mar.html">Mar</a>`
    b)
    ```
    <img src="../imagens/barco.gif">
    <a href="../documentos/mar.html">Mar</a>
    ```
    c)
    ```
    <img src="imagens/barco.gif">
    <a href="documentos/mar.html"></a>Mar
    ```
    d)
    ```
    <img src="/imagens/barco.gif">
    <a href="documentos/mar.html">Mar</a>
    ```
    e) `<img src="../barco.gif"> <a href="../mar.html">Mar</a>`

13. Qual das alternativas abaixo contém o código HTML para que o arquivo `mar.html` mostre respectivamente as imagens `ondas.jpg` e `barco.gif`? Marque uma alternativa.
    a) `<img src="documentos/ondas.jpg"> e <img src="/imagens/barco.gif">`
    b) `<img src="ondas.jpg"> e <img src="../imagens/barco.gif">`
    c) `<img src="/documentos/ondas.jpg"> e <img src="../barco.gif">`
    d) `<img src="ondas.jpg"> e <img src="/barco.gif">`
    e) `<img src="ondas.jpg"> e <img src="barco.gif">`

14. Suponha que o arquivo `index.html` possua uma seção ancorada por um elemento `<a name="secao2"></a>`. Qual das alternativas abaixo contém o código HTML para que o arquivo `mar.html` contenha um link que aponte para essa âncora do arquivo `index.html`? Marque uma alternativa.
    a) `<a name="/index.html#secao2">Seção 2 do Índice</a>`
    b) `<a href="index.html#secao2">Seção 2 do Índice</a>`
    c) `<a href="../index.html">Seção 2 do Índice</a>`
    d) `<a href="#secao2">Seção 2 do Índice</a>`
    e) `<a href="/index.html#secao2">Seção 2 do Índice</a>`

15. Qual dos trechos de código HTML abaixo desenhará a seguinte tabela?

    a)
    ```
    <table border="1" width="80">
        <tr> <td colspan="2">1</td> </tr>
        <tr> <td rowspan="2">4</td>
             <td>5</td>
             <td rowspan="2">2</td> </tr>
        <tr> <td colspan="2">3</td> </tr>
    </table>
    ```



b)
```
<table border="1" width="80">
    <tr> <td colspan="2">1</td>
         <td rowspan="2">2</td>
         <td rowspan="2">4</td>
         <td>5</td>
         <td colspan="2">3</td> </tr>
</table>
```

c)
```
<table border="1" width="80">
    <tr> <td colspan="2">1</td>
         <td rowspan="2">2</td> </tr>
    <tr> <td rowspan="2">4</td>
         <td rowspan="1">5</td> </tr>
    <tr> <td colspan="2">3</td> </tr>
</table>
```

d)
```
<table border="1" width="80">
    <tr> <td rowspan="2">1</td>
         <td colspan="2">2</td> </tr>
    <tr> <td colspan="2">4</td>
         <td>5</td>             </tr>
    <tr> <td rowspan="2">3</td> </tr>
</table>
```

e) Nenhuma das alternativas anteriores

3 Além do HTML

3.1. Imagens

No meio eletrônico há basicamente dois tipos de imagens: as que são formadas por matriz de pixels (*bitmap*) e as *vetoriais*. As imagens do tipo bitmap (ou raster) são criadas a partir de uma matriz de pontos. A sua resolução é fixa, portanto, se uma imagem é criada em uma matriz de 100x100 pixels e depois ampliada para 1000x1000, cada ponto será dez vezes maior na versão ampliada. Imagens vetoriais são definidas através de equações e dados que permitem a sua reprodução. Uma imagem vetorial, quando ampliada, preserva a aparência da forma que representa. Os pontos na tela, que não fazem parte da informação armazenada com a imagem, não são ampliados como ocorre com as bitmaps, resultando em uma imagem mais nítida.

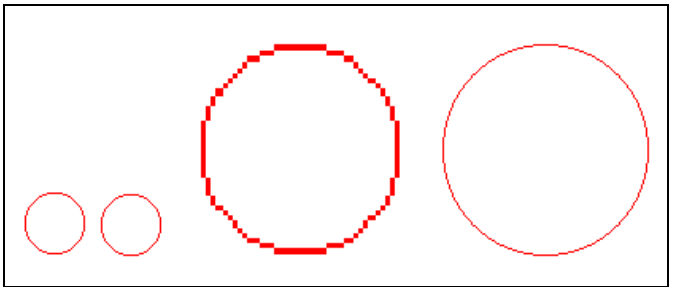


Figura 1 – O primeiro círculo é uma imagem bitmap. O segundo é uma imagem vetorial. O primeiro amplia o mapa de bits. O segundo amplia o raio da equação do círculo.

Existem softwares que trabalham exclusivamente com imagens vetoriais, chamados Line Art. Os mais populares são o *Corel Draw* (PC), *CorelXara*, *AutoCAD*, *Illustrator* (PC e Macintosh), *FreeHand* (Macintosh) e *Flash* (PC e Macintosh). O principal formato aberto utilizado é o *PostScript*. Imagens vetoriais são em geral *muito menores* que imagens bitmap, mas atualmente não são suportadas na World Wide Web a não ser através de extensões de browser como as extensões *Flash* e *Shockwave*..

O programa de criação e tratamento de imagens de bitmap mais popular é o *Adobe PhotoShop* (PC, Macintosh, Unix), que é praticamente o padrão de mercado. Seus concorrentes são *Corel PhotoPaint* (PC), o *Fractal Design Painter* (Macintosh), sharewares como o *LViewPro*, *PaintShopPro*, etc. Recentemente têm surgido pacotes criados especificamente para a Web como o *Adobe ImageReady*, o *Macromedia Fireworks* e o *Microsoft ImageComposer*. Esses pacotes são capazes de trabalhar com imagens vetoriais e gerar bitmaps otimizados para a Web.

Por serem muito grandes, *bitmaps sem compressão* raramente são usados na Web. Os formatos de compressão utilizados mais populares são *GIF* e *JPEG*. Na média, JPEG proporciona uma compressão maior que GIF. Mas em imagens pequenas, com poucas cores ou poucas variações, as imagens GIF tendem a ser menores.

3.1.1. Cores

A dimensão e o número de cores de uma imagem determina o seu tamanho. Imagens em preto-e-branco só precisam de um bit. Seus dois estados representam as cores desejadas: ligado

(branco) e desligado (preto). Com dois bits é possível representar quatro cores (como fazia o padrão de vídeo CGA) como preto (00), magenta (01), azul ciano (10) e branco (11). Cores adicionais podem ser produzidas através de dispersão, alternando pixels de cores diferentes e conseqüentemente, reduzindo a resolução. É possível construir cinza alternando bits pretos e brancos, mas a resolução será duas vezes menor que usando apenas bits cinza.

Quanto maior o tamanho de uma imagem em altura e largura, maior espaço será necessário para armazená-la na memória de vídeo e no disco. Uma imagem de um bit de profundidade com 100 pixels de largura por 100 pixels de altura ocupará um espaço de 100 x 100 = 10000 bits de informação. O armazenamento da imagem no computador poderá ocupar um espaço maior ainda, dependendo do formato utilizado.

Imagens coloridas tem mais bits por pixel. Se a imagem do exemplo anterior tivesse 4 cores (2 bits), o espaço necessário para armazená-la seria pelo menos duas vezes maior.

O número de cores que o seu sistema mostra na tela depende da quantidade de memória de vídeo disponível. Embora muitas máquinas mais sofisticadas têm capacidade de exibir imagens de 24 bits (16,777 milhões de cores), grande parte das máquinas populares só consegue exibir imagens de 8 bits, o que significa que seus usuários só podem ver 256 (2^8) cores na tela. Imagens de 24 bits são chamadas de *true color*, pois possuem 8 bits para representar cada cor, e conseguem exibir fotos de forma realística. Uma imagem *true color* com 100 pixels de largura por 100 pixels de altura ocupará um espaço de 100 x 100 x 24 = 240000 bits ou 29,4 kB, sem contar o *overhead* do formato de armazemanento (TIFF, BMP, etc.). A imagem terá esse tamanho mesmo que não use todas as cores.

Na Web, os tamanhos das imagens não aumentam desta forma quando se aumenta o número de cores, pois geralmente armazenadas em formatos de compressão como GIF e JPEG que reduz seu tamanho em um fator que varia 4 a 100 vezes.

3.1.2. Paletas de cores

Se uma imagem que foi criada com 16 milhões de cores (24 bits) for exibida em um ambiente gráfico de 8 bits, será necessário que o sistema realize conversões para que seja possível exibir a imagem. Sistemas de 256 cores são freqüentemente indexados (atribuem um número de 0 a 255 a cada cor que exibem). Esse conjunto de cores que são exibidas ao mesmo tempo é a paleta de cores. Ela pode ser escolhida de forma a permitir a melhor visualização possível. Por exemplo, se uma imagem é totalmente composta de tons de vermelho, pode-se usar uma paleta de 256 cores com predominância de tons avermelhados.

Mas as cores não servem apenas para a imagem. Elas têm que ser usadas também para colorir o ambiente gráfico do sistema operacional. Se uma imagem rouba todas as cores, o sistema operacional não será capaz de exibir as suas cores corretamente. Para evitar que uma aplicação roube todas as cores disponíveis, o número de cores que a aplicação pode usar poderá ser limitado.

Os browsers têm uma paleta própria chamada de *cubo de cores* para a exibição de imagens em monitores de 256 cores. O cubo consiste de 216 cores. As outras 40 são reservadas para o sistema. Os cantos do cubo são todas as oito possíveis combinações de 255 e 0:

- preto: 0,0,0;
- azul 0,0,255;
- verde 0,255,0;
- azul ciano 0,255,255;
- vermelho 255,0,0;
- magenta 255,0,255;
- amarelo 255,255,0 e
- branco 255,255,255

As cores internas do cubo são 4 cores intermediárias, uniformemente espaçadas, formando um total de 216. Quando uma imagem é carregada no browser, ela tem seus pixels adaptados à paleta do sistema operacional. Se a cor não existir, o sistema tentará criá-la por dispersão, alternando pixels. O browser automaticamente reduz as cores do browser (cores de fundo, textos, etc.) para cores da paleta.

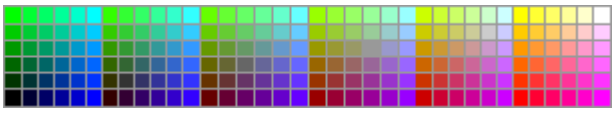


Figura 2 - Netscape Color Cube

Se você cria uma imagem em uma aplicação gráfica, sobre um fundo, digamos, azul, e esse azul não faz parte do cubo de cores, ele não terá uma cor correspondente no browser. Se você quiser que o fundo da imagem mescle de forma imperceptível com o fundo azul da página, você terá que escolher um azul que exista no cubo de cores, caso contrário, perceberá a diferença quando a imagem for carregada em um sistema de 256 cores. Utilizando-se as cores do "Color Cube", evita-se a dispersão incorreta das cores. A maior parte das aplicações gráficas possuem a paleta da Web.

3.1.3. Anti-aliasing

Anti-alising (ou anti-serrilha) é o processo que adiciona cores intermediárias para otimizar as áreas serrilhadas da imagem. O processo aumenta o número de cores da imagem, aumentando o tamanho da paleta (e conseqüentemente o tamanho da imagem). Se você abre o *PhotoShop* e faz um grande círculo de uma cor só, em um fundo uniforme, são grandes as chances de se ter mais de 200 cores! Além de aumentar o tamanho, *anti-aliasing* também reduz a capacidade de compressão da imagem.

Anti-aliasing é uma técnica necessária. Imagens com curvas ou linhas inclinadas sem *anti-aliasing* ficam com uma qualidade inaceitável. A solução é realizar o *anti-aliasing* com o menor número de cores possível.

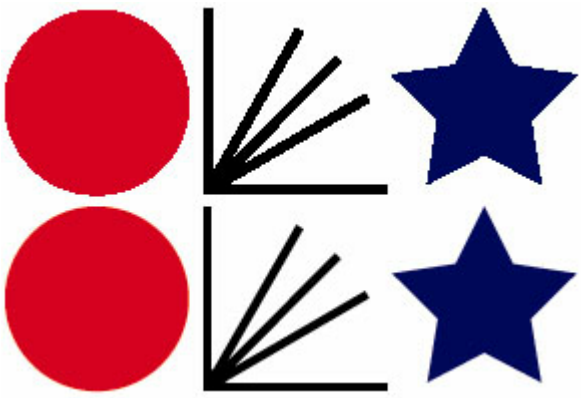


Figura 3 - Sem anti-serrilha (em cima) e com anti-serrilha (em baixo). Fonte: http://www.killersites.com

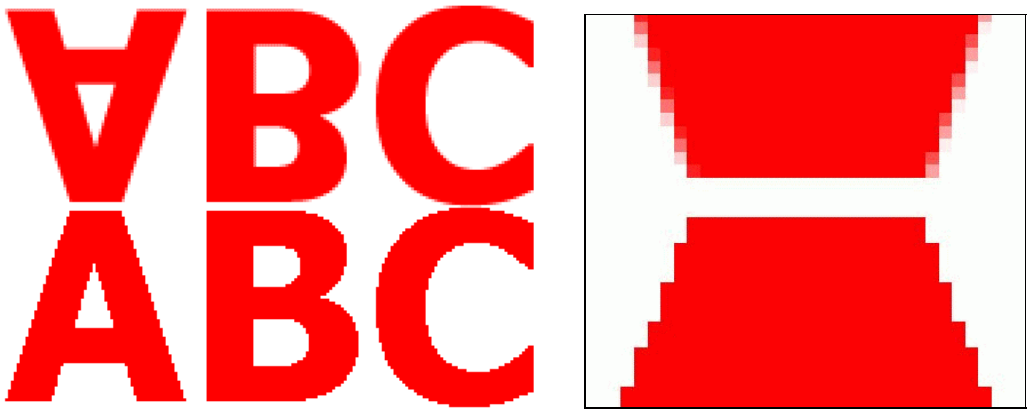


Figura 4 - Outro exemplo com anti-serrilha mostrando detalhe

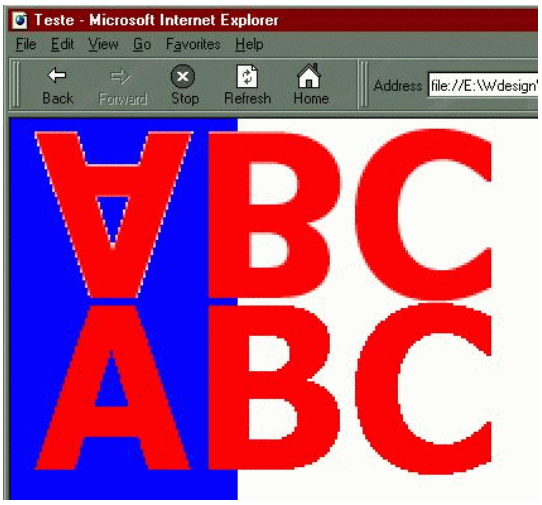


Figura 5 – Exemplo de "A" fantasma que aparece por trás do "A" que foi anti-serrilha.

Anti-aliasing pode causar problemas em imagens com fundo transparente. Quando for necessário por uma imagem na página combinando com a imagem de fundo é necessário criar a imagem, no aplicativo gráfico, sobre uma *réplica* da imagem de fundo. Isto garantirá que a variação de cores será feita corretamente. Como apenas uma cor pode ser escolhida como transparente, as cores intermediárias permanecerão na sua cor original. Se no PhotoShop o desenho for feito sobre o fundo branco, na hora de escolher a transparência, as cores próximas ao desenho permanecerão brancas e não transparentes, aparecendo, portanto, sobre um fundo de outra cor (veja figura).

Pode não haver solução para esse problema se a imagem de fundo for irregular. O *registro* (casamento da imagem de frente com a de fundo) deverá ser perfeito, mas isto é impossível

com os browsers atuais. Nesses casos, é preciso avaliar o que é pior: ficar sem o *anti-aliasing*, ou correr o risco de ter "fantasmas" em volta de partes da imagem. Se ambos forem inaceitáveis, será preciso repensar o projeto.

3.1.4. Compressão GIF

A compressão reduz bastante o tamanho da imagem. O primeiro formato comprimido suportado na Web foi o formato GIF ou *Graphics Interchange Format* . Trata-se de um formato de compressão de bitmaps pertencente à Unisys (SPRY/Compuserve). Suporta imagens indexadas até 8 bits (256 cores). Usa como método de compressão o algoritmo Lempel-Ziv-Welch (LZW) que proporciona uma compressão média da ordem de 4:1 sem perdas. A imagem parece idêntica à original.

Use GIF para comprimir qualquer coisa que não seja fotográfica como desenhos vetoriais, fotos pequenas, etc. Imagens fotográficas comprimem melhor em JPEG.

O algoritmo GIF consiste em substituir seqüências horizontais de pixels da mesma cor com um número indicando o tamanho da seqüência. Linhas horizontais idênticas comprimem linha por linha. Qualquer regularidade horizontal será comprimida. Desta forma, imagens que têm semelhanças horizontais são beneficiadas: uma imagem bitmap de um círculo, comprimido com GIF, tem quase o mesmo tamanho que a imagem GIF de sua metade superior.

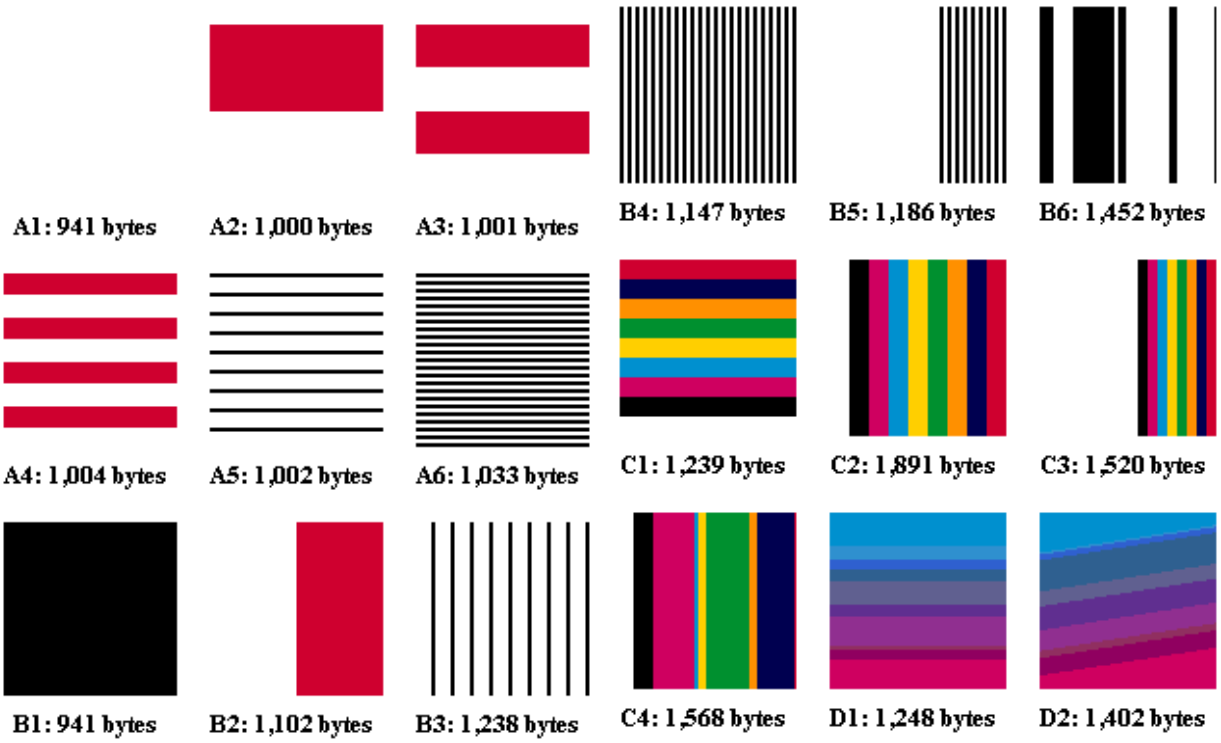


Figura 6 - Várias imagens gifs de mesma dimensão e conteúdo diferente (www.killersites.com)

Entrelaçamento e transparência

O formato GIF mais antigo em uso na Web é chamado de GIF87. O formato mais novo, GIF89, é mais eficiente e possui mais recursos como a capacidade de entrelaçamento, transparência e animação.

Normalmente, GIFs armazenam pixels de cima para baixo. Imagens entrelaçadas armazenam pixels fora de ordem linear. As imagens são transferidas totalmente em quatro passos. Na web, a exibição de imagens entrelaçadas costuma demorar mais que imagens comuns mas produz um efeito que aparenta uma velocidade maior.

Em GIF89, uma cor qualquer pode ser substituída pela cor transparente. Somente uma cor pode ser trocada.. Se houver uma variação pequena da cor escolhida como transparente, está continuará visível como antes. A transparência consiste em escolher o índice de uma cor e defini-la como transparente. A vantagem de imagens transparentes é que elas não são "retangulares", mas aparentam ter a forma do objeto que manteve as cores visíveis.

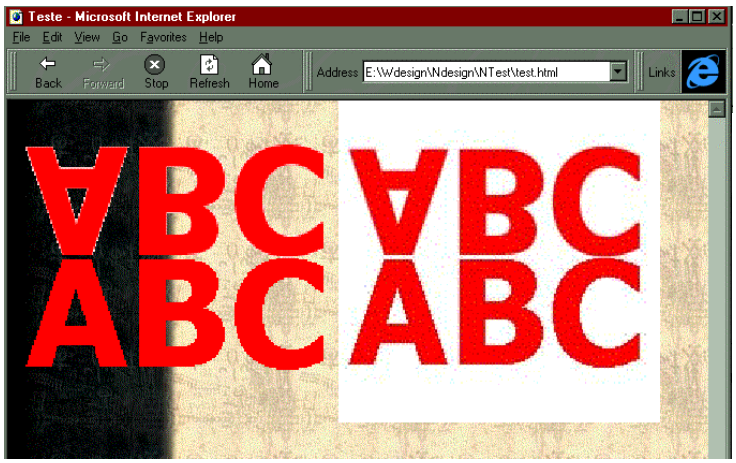


Figura 7 – Imagem transparente versus imagem não-transparente. O branco da primeira imagem foi trocado pelo transparente.

Animação

Várias imagens GIF podem ser armazenadas em um único arquivo para serem exibidas em seqüência. Com isto, é possível simular animação. GIFs animados são bastante versáteis e contém diversos efeitos diferentes: repetição eterna, tempos diferentes para cada imagem, etc.

Redução de cores e tamanho

Reduzir cores em GIF as torna menores. O processo deve ser realizado com cuidado para evitar uma redução excessiva que resulte em grande perda de qualidade. No *PhotoShop*, pode-se usar o *histograma de cores* e influenciar a redução de cores específicas.

Hoje já existem várias ferramentas que fazem automaticamente a redução de cores. Algumas ainda opinam sobre o melhor formato (GIF ou JPEG) para uma imagem.

3.1.5. JPEG

JPEG, ou *Joint Photographers Expert Group*, é a melhor forma de comprimir imagens fotográficas. Com JPEG, é possível obter uma compressão típica que varia de 10:1 a 100:1 dependendo da qualidade desejada.

Diferente de GIF, JPEG suporta o formato RGB e, portanto, pode armazenar informações de cor 24 bit (16,777 milhões de cores). JPEGs progressivas obtém o mesmo efeito que as GIFs entrelaçadas.

JPEG não garante a mesma compressão em imagens pequenas ou desenhos devido ao seu *overhead*. Veja os tamanhos de imagens de um pixel (1x1) de 4 cores em vários formatos:

JPEG:	949 bytes
BMP (sem compressão):	72 bytes
GIF:	41 bytes

Já uma imagem grande e detalhada (uma fotografia) comprime bastante bem em JPEG. Arquivo earthrise.jpg/gif/bmp (400x302 pixels; 256 cores) do CD do ASIT:

JPEG:	8,8 kB
GIF	32 kB
BMP	119 kB

Para reduzir o tamanho, JPEG separa as matizes de cor dos componentes de brilho e mantém uma boa cópia da versão em preto-e-branco da imagem. Joga fora diferenças de cor indistinguíveis. Divide a figura em zonas que são comprimidas individualmente.

A compressão é sempre realizada com perdas, mesmo quando na melhor qualidade. A imagem final, em geral, é menor que a equivalente GIF (desde que não seja muito pequena). Imagens mais nítidas são maiores que imagens mais borradas. A forma de reduzir o tamanho é borrar a imagem (cria mais cores intermediárias que são descartadas) diminuindo o seu fator de qualidade. JPEG não trabalha com cores indexadas (é inviável indexar 16 milhões de cores – o *overhead* seria imenso). Reduzir cores, portanto, não reduz o tamanho.

3.1.6. PNG

O PNG – *Portable Network Graphics* é um novo formato que foi desenvolvido pela W3C especialmente para a Web. É um formato que suporta compressão sem perdas, imagens de 24bits, efeitos progressivos (entrelaçamento) mais eficientes, transparência de vários níveis, canais alfa de 8 bits e uma série de vantagens que devem torna-la a substituição ideal para imagens GIF.

Vários programas já suportam PNG, mas não todos os seus recursos. Uma das vantagens não suportadas pelos browsers é a capacidade de exibir 256 níveis de transparência. Além dos browsers, os programas de desenvolvimento e tratamento de imagens como o *PhotoShop 4* também suportam e manipulam com PNG, que é o formato nativo do *Macromedia Fireworks*.

Veja mais informações sobre PNG em http://www.w3.org/PNG.

3.2. Layout com tabelas

Vimos na primeira parte deste curso os principais usos das tabelas HTML. Neste capítulo mostraremos exemplos de tabelas aplicadas à diagramação da página. Os recursos aqui apresentados são totalmente substituíveis por folhas de estilo, porém, a visualização da página poderá ser prejudicada seriamente se o browser não suportar CSS. Como as tabelas existem há mais tempo, é muito mais raro encontrar browsers que não suportam tabelas, embora os que não suportam posicionamento com folhas de estilo estão bastante próximos, nos browsers versão 3.0.

Para diagramar com tabelas é preciso saber como alinhar objetos dentro de tabelas, como combinar duas ou mais células em uma, remover os espaços entre células e entre a célula e o texto além de criar tabelas dentro de tabelas. As seções seguintes tratarão de alguns desses temas.

Por não ser uma linguagem apropriada à diagramação, HTML impõe diversas limitações quando à organização do texto e imagens em uma página. Como vimos na seção anterior, as tabelas oferecem algum controle. Um pouco mais controle é conseguido utilizando imagens invisíveis (pois a dimensão das imagens é certa enquanto a das tabelas não é). Mas há ainda várias outras limitações também referentes a colocação das imagens que só se resolvem com folhas de estilo. Em programas como o *PageMaker*, você coloca o texto e as imagens onde você quer. HTML sem folhas de estilo não oferece esse tipo de manipulação bidimensional. É necessário usar truques para simular este controle se você não puder depender de folhas de estilo.

3.2.1. GIF de um pixel

Este truque, proposto por David Siegel em seu livro *Criando Sites Arrasadores* consiste da utilização de uma imagem de um pixel (1x1) de cor transparente ou igual à cor de fundo da página. A imagem resultante é invisível e pode ser usada numa página HTML para "empurrar" textos, colunas e linhas de tabelas, imagens, etc.

A imagem pode ser esticada usando os atributos HEIGHT E WIDTH ou usar margens, através dos atributos HSPACE e VSPACE. Por exemplo:

```
<IMG SRC="pixel.gif" HEIGHT=11> ou
<IMG SRC="pixel.gif" VSPACE=5>      Espaço vertical de 11 pixels
```

```
<IMG SRC="pixel.gif" WIDTH=11> ou
<IMG SRC="pixel.gif" HSPACE=5>       Espaço horizontal de 11 pixels
```

A imagem alinha-se geralmente com a base do texto ou da outra imagem que está ao seu lado. Isto pode ser alterado através dos atributos ALIGN. É possível definir larguras mínimas precisas para tabelas criando uma linha <TR> que contenha blocos <TD> contendo, cada um, uma única imagem de 1 pixel de altura/largura pelo valor desejado de largura/altura.

3.2.2. Layout de página com tabelas

Tabelas sem bordas dentro de tabelas sem bordas podem ser usadas para realizar o layout de texto e imagens em uma página HTML. Com essa técnica pode-se construir um espelho de diagramação sofisticado e organizar texto e imagens de forma criativa, fugindo do estilo tradicional das páginas HTML.

Deve-se desligar as bordas da tabela e fazer o `cellspacing` e `cellpadding` iguais a zero. Use uma largura absoluta para a tabela e para as células para se ter maior controle. Havendo necessidade de espaço em branco, pode-se usar tanto `cellspacing` como `cellpadding`.

Imagens recortadas

Para colar imagens horizontalmente ou verticalmente a melhor opção, quando não se tem folhas de estilo, é usar tabelas. O controle é maior entre plataformas. A "fragmentação e colagem" das imagens é uma técnica que visa diminuir o tempo de transferência das imagens, se aproveitando do cache do browser, que não carrega imagens repetidas. A imagem é partida em pedaços de forma que as partes que mudam sejam imagens diferentes das partes que se mantém através das páginas.

Antes de dividir uma imagem pelas células de uma tabela, ela precisa ser recortada. Isto pode ser feito manualmente em um programa como o *Adobe PhotoShop* ou de forma automática no *Macromedia Fireworks*, por exemplo.

Ao acrescentar as imagens dentro das células, elas se acomodarão e ocuparão o espaço que for necessário. Se for preciso colocar duas imagens sobre outra ou imagens ao lado de uma imagem deve-se usar os atributos COLSPAN e ROWSPAN para rearrumar a tabela e permitir que as peças se encaixem. No final, deve-se desligar o espaço entre células, as margens e as bordas para que a imagem volte a ser uma só.

É preciso ter o cuidado de não deixar espaços nem quebras de linha dentro do bloco <TD> ... </TD> que contém a imagem . Se isto ocorrer podem acontecer situações como a página ao lado, onde a imagem não se recompõe totalmente.

3.3. Frames

Frames (quadros) são painéis que dividem a área de visualização do browser em subjanelas, cada uma capaz de exibir uma página diferente. Para mostrar duas páginas ao mesmo tempo dentro de uma janela do browser é preciso ter pelo menos três arquivos. Dois são os arquivos HTML das páginas que serão exibidas. O outro é uma página HTML que contém apenas as instruções para que o browser saiba se dividir em subjanelas.

Logo, para dividir uma janela em frames, é preciso criar essa página HTML que especifica as dimensões relativas ou absolutas das subjanelas em relação à janela que irá conter a página. Uma página de frames não é um *documento* HTML, pois não contém informação. Todo documento HTML, como sabemos, deve ter a forma:

```
<html>
  <head> ... </head>
  <body> ... </body>
</html>
```

O bloco <body> contém a informação da página. O bloco <head>, contém meta-informação, ou seja, informação sobre a página. Páginas que definem a estrutura de subjanelas (páginas de frames) têm uma estrutura diferente:

```
<html>
  <head> ... </head>
  <frameset atributos> ... </frameset>
</html>
```

e não podem conter blocos <body>[1].

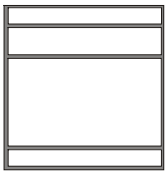
[1] Até podem conter blocos <BODY>, mas isto ou os transforma em páginas de informação, ou não causa efeito algum. Um bloco <BODY> antes do <FRAMESET> faz com que o browser ignore o <FRAMESET>. Um bloco <BODY>

3.3.1. Estrutura básica

O bloco `<frameset>` define a divisão da janela em *linhas* (usando o atributo `rows`) ou *colunas* (usando o atributo `cols`) ou simultaneamente em linhas e colunas (usando ambos os atributos). Os atributos especificam a largura ou altura de cada frame usando valores absoultos, em pixels, ou relativos, em percentagens da largura ou altura da janela principal. Por exemplo, um `<FRAMESET>` com a forma mostrada figura ao lado é definido pelo código

```
<FRAMESET COLS="25%,25%,50%"> ... </FRAMESET>
```

que divide a janela principal em três colunas, tendo as duas primeiras ¼ da largura total, e a última, metade da largura total. De forma semelhante pode-se dividir a janela em linhas. Neste outro exemplo (figura ao lado):



```
<FRAMESET ROWS="100,200,*,100"> ... </FRAMESET>
```

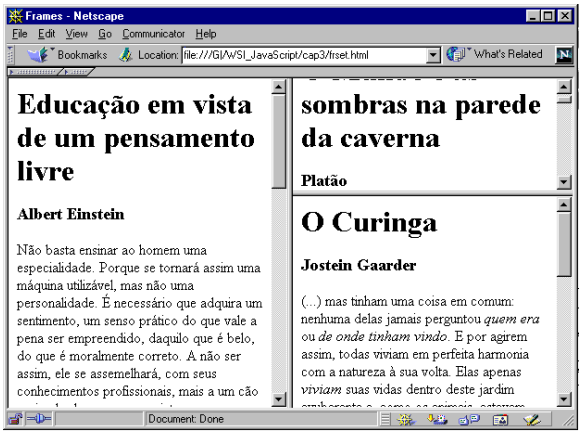
a janela foi dividida em quatro linhas, tendo a primeira e quarta 100 pixels cada de altura, a segunda 200 pixels e a terceira, o espaço restante. Combinando os dois argumentos obtém-se uma janela com 12 subjanelas.

Um bloco `<FRAMESET>...</FRAMESET>` só pode conter dois tipos de elementos:

- descritores `<FRAME>`, que definem a página HTML que ocupará uma janela. A página HTML poderá ser uma página de informação comum ou outra página de frames que dividirá a subjanela novamente em linhas e/ou colunas.
- sub-blocos `<FRAMESET> ... </FRAMESET>` que dividirão outra vez a subjanela (em linhas e/ou colunas) e poderão conter descritores `<FRAME>` e novos sub-blocos `<FRAMESET>`.

O número de sub-blocos para cada `<FRAMESET>` dependerá do número de linhas (ou colunas) definidas. Para dividir uma janela em linhas e colunas de forma irregular (com células que atravessam mais de uma linha ou coluna), pode-se proceder de duas formas:

- usar um *único* `<FRAMESET>`, contendo elementos `<FRAME>` que referem-se a páginas de frames (páginas que também definem um `<FRAMESET>`), ou
- usar vários `<FRAMESET>` em cascata na mesma página.



---

após o `<FRAMESET>` será ignorado por browsers que suportam frames, mas será lido por browsers antigos que não os suportam.

Usaremos as duas formas para montar a janela ao lado. Na primeira versão, utilizaremos *dois* arquivos de frames: `frset1.html` dividirá a janela principal em duas colunas, e `frset2.html` dividirá a segunda coluna em duas linhas. Na segunda versão, precisaremos de apenas *um* arquivo de frames (`frset.html`). As duas versões utilizarão três arquivos de informação: `um.html`, `dois.html` e `tres.html`. Visualmente, o resultado final é o mesmo, mas as duas formas podem ser manipuladas de forma diferente.

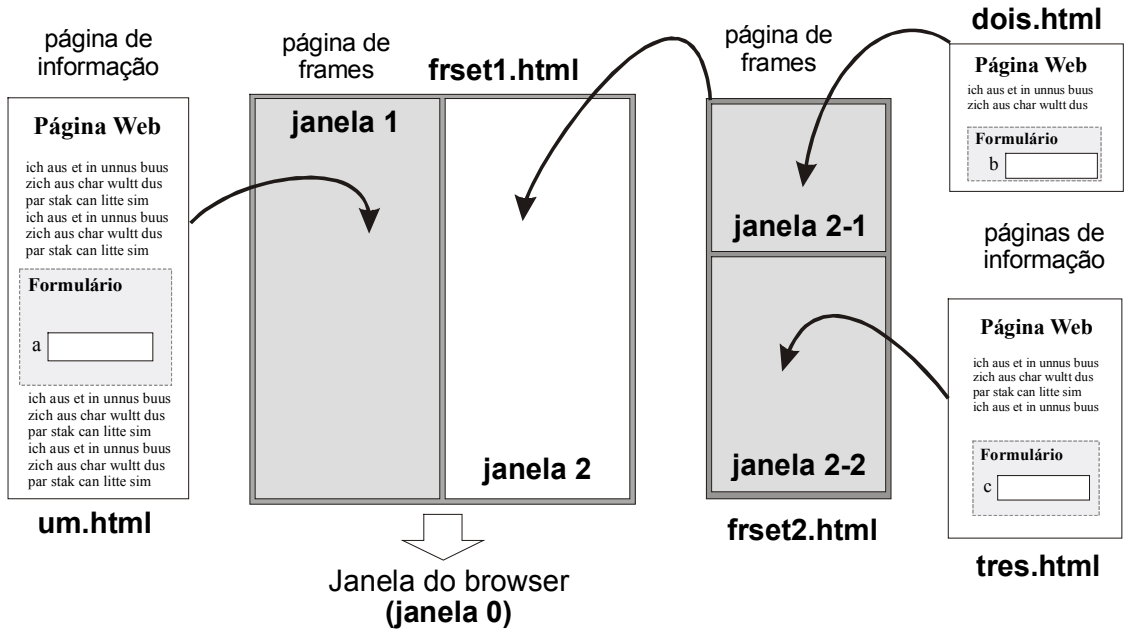
Na primeira versão temos *dois* arquivos. Os trechos em negrito indicam as ligações entre eles. O primeiro é `frset1.html`, que referencia uma página de informação:

```
<html>
<head> ... </head>

<frameset cols="50%,50%">
   <frame name="janela1" src="um.html">
   <frame name="janela2" src="frset2.html">
</frameset>
</html>
```

e chama `frset2.html`, com mais duas páginas de informação, listado abaixo:

```
<html>
<head> ... </head>
<frameset rows="35%,65%">
   <frame name="janela2_1" src="dois.html">
   <frame name="janela2_2" src="tres.html">
</frameset>
</html>
```

A figura abaixo mostra a organização das páginas de informação e das páginas de frames na janela do browser.



Observe que há *três níveis* de páginas. No nível mais alto está a página `frset1.html`, que ocupa toda a janela do browser. No segundo nível estão os arquivos `um.html` e `frset2.html`. E no terceiro nível, encontramos os arquivos `dois.html` e `tres.html`, que estão dentro de `frset2.html`.

Na segunda versão, temos apenas *um* arquivo de frames contendo referências para os três arquivos de informação. Em negrito está mostrado o segundo *frameset* (observe que ele substitui o elemento `<FRAME>` que chamava `frset2.html` no último exemplo.)

```
<html>
<head> ... </head>
<frameset cols="50%,50%">
   <frame name="janela1" src="um.html">
   <frameset rows="35%,65%">
      <frame name="janela2_1" src="dois.html">
      <frame name="janela2_2" src="tres.html">
   </frameset>
</frameset>
</html>
```

Esta segunda versão possui apenas dois níveis. No primeiro, a página de frames `frset.html`, no segundo, as páginas de informação. A aparência final é a mesma, nas duas versões, mas na `primeira` versão há uma janela a mais (`janela2`) que pode ser manipulada via JavaScript ou HTML.

3.3.2. Controle dos alvos dos vínculos

O comportamento usual dos browsers é o de abrir uma nova página sempre dentro da janela atual. É possível alterar esse comportamento através do atributo TARGET em vínculos de hipertexto <A HREF> e elementos <BASE>.

Se a `janela2` for utilizada como alvo de um link HTML:

```
<a href="pagina.html" TARGET="janela2"> link </A>
```

os frames `janela2_1` e `janela2_2`, que estão em um nível abaixo de `janela2` deixarão de existir e `pagina.html` ocupará toda a segunda coluna da janela do browser. Isto não poderá ser feito na segunda versão, pois ela só possui dois níveis.

Se o link `estiver` dentro da página `dois.html` ou `tres.html`, a sintaxe abaixo, usando o nome especial `_parent` causará um resultado equivalente:

```
<a href="pagina.html" TARGET="_parent"> link </A>
```

3.4. Multimídia

3.4.1. Imagens mapeadas

Para fazer com que uma imagem sirva como origem de um vínculo de hipertexto basta colocá-la dentro de um bloco <A HREF>. Mas se o que se deseja é que partes diferentes da imagem apontem para lugares diferentes, é preciso usar imagens mapeadas ou *imagemaps*. Imagens mapeadas são muito usadas em barras de navegação, mapas, diagramas, etc. Sempre que for necessário selecionar uma área irregular de uma imagem e fazer com que ela esteja relacionada a um destino (página, âncora local, etc.) pode-se usá-la.

Há basicamente dois tipos de imagens mapeadas: as que funcionam no servidor (no seu provedor) e as que funcionam do lado do cliente (no seu computador). O primeiro tipo necessita de um programa CGI e um arquivo especial que contém as coordenadas das áreas ativas armazenados no servidor. É uma solução obsoleta e mais complicada. O segundo realiza tudo sem acessar a rede e portanto é mais rápido e mais eficiente. Todos os browsers mais recentes (desde as versões 2.0) suportam imagens mapeadas do lado do cliente.

Para criar uma imagem mapeada é necessário ter uma imagem e calcular as coordenadas de suas áreas "ativas". A maior parte dos editores HTML possuem ferramentas para gerar coordenadas, mas se a sua não tiver, você pode sempre obter ferramentas gratuitas na rede como o *MapEdit para Windows* (http://www.boutell.com/mapedit/) ou o *MapMaker para Mac* (http://www.kickinit.com/mapmaker/).

Imagens mapeadas são imagens comuns. A diferença é que têm um atributo adicional informando o nome do bloco que define o mapa que será utilizado. O atributo é USEMAP:

```
<BODY>
<IMG SRC="banner.gif" USEMAP="#isoptera" ALT="Menu de Opções">
</BODY>
```

O atributo USEMAP acima possui um valor #isoptera que corresponde ao nome de um mapeamento. Incluímos uma estrutura de mapeamento na página através do bloco <MAP>. Todo o bloco <MAP> é gerado pelo programa que calcula as coordenadas. Para a imagem abaixo (banner.gif) o mapa, sem as coordenadas, é o seguinte:

```
<MAP NAME="isoptera">
 <AREA SHAPE="rect" COORDS="..." HREF="core.html" target=_top>
 <AREA SHAPE="rect" COORDS="..." HREF="livraria.html">
 <AREA SHAPE="rect" COORDS="..." HREF="cdshop.html">
 <AREA SHAPE="rect" COORDS="..." HREF="eventos.html">
 <AREA SHAPE="rect" COORDS="..." HREF="pulgas.html">
 <AREA SHAPE="rect" COORDS="..." HREF="sair.html" target=_top>
</MAP>
```

Para selecionar as áreas ativas da figura, você precisará descrevê-las, informando a *forma* (`POLY`, `RECTANGLE` ou `CIRCLE`) e suas coordenadas (lista de números separados por vírgulas) além da URL que devem seguir (atributo `HREF`). Esses atributos do descritor `<AREA>` definem a área utilizada.

Os polígonos (`SHAPE="poly"`) são definidos em termos das coordenadas de seus vértices. Cada coordenada (par de dois números) corresponde a um vértice (x, y) do mesmo. A origem (0,0) corresponde ao canto superior esquerdo da imagem `banner.gif`. Círculos (`SHAPE="circle"`) devem incluir três números separados por vírgulas como coordenadas. Os dois primeiros são a coordenada do centro do círculo e o último seu raio. Retângulos (`SHAPE="rectangle"`) são definidos em termos de quatro números: as coordenadas no canto superior esquerdo e do canto inferior direito.

O bloco `<MAP>` pode estar em qualquer lugar da página pois é identificado pelo nome e só se torna ativo depois que a página inteira for carregada.

3.4.2. Áudio, vídeo e plug-ins

Componentes Flash e Shockwave, imagens e vídeos podem ser incluídos em uma página usando o elemento <EMBED> ou o elemento <OBJECT>. Embora o último seja recomendado pela especificação do HTML 4.0, o suporte a <EMBED> é mais antigo e funciona tanto nos browsers Internet Explorer como Netscape.

Os atributos de <EMBED> variam conforme o componente instalado. Na maior parte dos casos, o componente ocupa uma parte da página como uma imagem, e pode ser alinhado e redimensionado usando os mesmos atributos disponíveis em <IMG>. A sintaxe mínima de <EMBED> é:

```
<EMBED SRC="URL do componente" HEIGHT=100 WIDTH=150>
</EMBED>
```

Audio

Há vários formatos de áudio suportados na Web. Os que possuem maior suporte são os formatos `.au` e `.mid` suportados pelo componente *LiveAudio* disponível no browser Netscape ou Internet Explorer. Mais recentemente tem crescido o suporte ao formato `.mp3`. O *LiveAudio* exibe (opcionalmente) um controle de tela que permite ao usuário manipular o som.

O elemento `<EMBED>` usado para incluir áudio possui vários atributos adicionais. A sintaxe típica é:

```
<EMBED SRC="musica.mid" CONTROLS=console HEIGHT=60
       WIDTH=150 AUTOSTART=false></EMBED>
```

O atributo `CONTROLS` pode receber os valores `console`, `smallconsole`, `playbutton`, `pausebutton`, `stopbutton` ou `volumelever` e alterar a aparência do *display*. O atributo `LOOP` pode ter os valores `true`, `false` ou um número indicando o número de vezes que o som deve tocar. `AUTOSTART` pode ser `true` ou `false` indicando se o som deve começar imediatamente ou esperar pela decisão do usuário. `VOLUME` varia de 0 a 100 e estabelece o volume do som.

No Internet Explorer (e apenas no Internet Explorer) som de fundo pode ser incluído usando o elemento <BGSOUND>:

```
<BGSOUND SRC="musica.mid" LOOP=3>
```

Vídeo

Assim como os recursos de áudio, vários formatos de vídeo como `.mpg`, `.mov`, `.avi`, `.rm`, etc. podem ser incluídos em páginas HTML usando o elemento `<EMBED>`. A sintaxe típica é:

```
<EMBED SRC="filme.mov" CONTROLS=console HEIGHT=260
       WIDTH=450 AUTOPLAY=false></EMBED>
```

Cada *plug-in* tem seus próprios atributos que podem ser acrescentados ao descritor `<EMBED>`. O formato `.mov` (*QuickTime*) suporta vários outros atributos, entre eles `CONTROLLER=true` ou `false`, para ligar ou desligar o controle do vídeo; `LOOP = true`, `false` ou `palindrome` para fazer o vídeo executar continuamente e de trás para frente.

No *Internet Explorer* (e somente no *Internet Explorer*) é possível incluir vídeos usando o atributo `DYNSRC` de `<IMG>`:

```
<IMG DYNSRC="filme.mov">
```

3.4.3. Flash

Flash é um formato de multimídia desenvolvido pela Macromedia que oferece a possibilidade de criar animações, interatividade e integração com áudio e vídeo. Os arquivos são muito menores que os utilizados nas tecnologias padrão da Web pois o formato das imagens e animações do Flash se baseia em vetores (vector) e não em bitmaps. Imagens vetoriais descrevem os desenhos, fontes e comportamentos em termos de equações, que ocupam bem menos espaço de armazenamento que as animações baseadas em bitmaps, que consistem da repetição de arquivos que contém o mapa completo de cores e pixels de cada tela.

Há diversas vantagens no formato oferecido pela tecnologia Flash. Os arquivos são menores, são escaláveis sem perda de qualidade, suportam som integrado, podem ser desenvolvidos sem a utilização de programação, contam com amplo suporte e ainda podem ser controlados através de parâmetros passados em HTML ou JavaScript.

As desvantagens se baseiam principalmente em se tratar de um formato proprietário. É preciso adquirir um software para produzir arquivos Flash. Para exibi-lo, é preciso fazer o download de um plug-in. Não há suporte em Linux ou Unix e nem para browsers não-gráficos, que perdem totalmente o conteúdo da página.

Para criar componentes Flash é preciso ter o pacote Macromedia Flash 4. Há uma versão de demonstração que vale por 30 dias no site http://www.flash.com/. O objetivo desta seção é apenas mostrar como se pode usar um componente Flash genérico que já foi criado previamente.

Configuração do servidor

Pode ser necessário configurar o servidor Web para que ele saiba servir arquivos Flash. Informe-se no seu provedor sobre a configuração. Se você for o webmaster responsável por um servidor, você deverá acrescentar a seguinte relação tipo MIME/extensão de arquivo para que o servidor reconheça o formato:

- *Tipo/subtipo:* application/x-shockwave-flash
- *Extensão*: .swf

Para acrescentar Flash em uma página HTML, use o elemento `<EMBED>`. A sintaxe mínima é:

```
<EMBED SRC="filme.swf" HEIGHT=260 WIDTH=450>
</EMBED>
```

3.5. Applets Java

3.5.1. O que são applets Java?

A Web é popular porque suas informações, na forma de texto e imagens, são visíveis independente da plataforma (tipo de máquina e sistema operacional) do usuário. Nem sempre é possível ver determinado vídeo dependente de plug-in, ou um certo site interativo dependente de Flash, pois às vezes o programa necessário à execução da aplicação depende de plataforma e pode ser até que não exista para a plataforma onde roda seu browser.

Applets são pequenas aplicações *independentes de plataforma* (por isso populares na Web), geralmente escritas em uma linguagem de programação chamada Java, que têm sua execução iniciada pelo browser. Diferentemente do que ocorre com as linguagens embutidas em blocos `<SCRIPT>` (JavaScript), o *código* Java escrito pelo programador da aplicação não é *interpretado* nem visualizado pelo browser, já que foi antes *compilado* para uma linguagem de máquina. Browsers que suportam Java possuem uma *plataforma virtual*, a "Java Virtual Machine", que simula uma máquina Java em sistemas operacionais diferentes como Windows, Macintosh, Unix, etc. Só a máquina virtual Java é capaz de interpretar a linguagem de máquina Java. Através dela, browsers são capazes de rodar programas sofisticados que podem desde oferecer uma interface de formulários mais segura interativa, até mostrar áudio, vídeo, 3D e sistemas de comunicação visual interativa em qualquer plataforma.

3.5.2. Quando usar applets

Applets podem ser usados para desenvolver aplicações que seriam impossíveis em HTML com ou sem linguagens embutidas (como JavaScript) e fugir das limitações do HTML, do protocolo HTTP e do próprio browser. Com um applet, é possível *estender* um browser fazendo-o suportar, por exemplo, novos protocolos de comunicação e segurança, novos formatos de mídia, etc. O preço dessa liberdade é sua fraca integração com o HTML da página. Aplicações Web baseadas em Java pouco ou nada aproveitam do HTML da página a não ser um espaço gráfico para sua exibição. O browser simplesmente reserva um espaço para rodar o programa, e o programa assume aquele espaço e faz o que bem quiser (limitado apenas por restrições de segurança). É possível aumentar essa integração com linguagens embutidas no HTML como JavaScript.

Applets são aplicações gráficas e precisam de uma página HTML para poderem executar. São exibidos na página de forma semelhante a imagens: carregam um arquivo externo, ocupam um espaço com determinada altura e largura, e podem ter seu alinhamento em relação ao texto definido pelos mesmos atributos presentes em `<IMG>`. A sintaxe do elemento HTML `<APPLET>` está mostrada abaixo. Tudo o que não estiver em negrito é opcional:

```
<APPLET
   CODE="nome_do_programa.class"
   HEIGHT="altura em pixels"
   WIDTH="largura em pixels"
   NAME="nome_do_objeto"
   CODEBASE="diretório base do arquivo de classe Java"
   ARCHIVE="arquivo .jar contendo o programa"
   ALT="texto descritivo"
   HSPACE="margens externas laterais em pixels"
   VSPACE="margens externas verticais em pixels"
   ALIGN="left" ou "right" ou "top" ou "middle" ou "bottom" ou
         "texttop" ou "absmiddle" ou "absbottom" ou "baseline"
   MAYSCRIPT>
      <PARAM NAME="nome" VALUE="valor">
      <PARAM NAME="nome" VALUE="valor">
         ...
      <PARAM NAME="nome" VALUE="valor">
</APPLET>
```

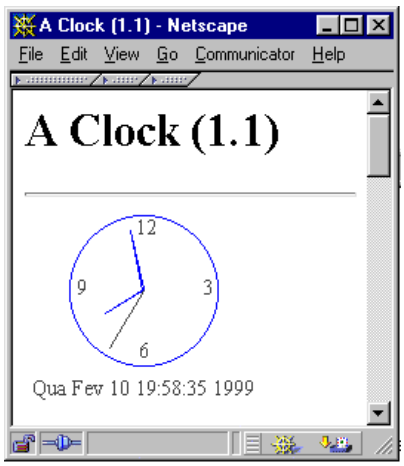
Diferentemente de <IMG>, o elemento <APPLET> é um bloco e possui um descritor de fechamento </APPLET>. Entre <APPLET> e </APPLET> pode haver nenhum ou vários elementos <PARAM>, que contém parâmetros necessários ou não (depende do applet) para o funcionamento da aplicação. Cada elemento <PARAM> contém um par de atributos obrigatórios. O valor do atributo NAME é definido pelo programador do applet. Através dele, o programa pode recuperar um valor que o autor da página (que não precisa saber Java) definiu no atributo VALUE.

Os atributos CODEBASE e ARCHIVE são usados opcionalmente para localizar applets em outros servidores e importar outros módulos do programa, respectivamente. O atributo MAYSCRIPT é necessário se o applet pretende ter acesso ao código JavaScript da página. Sem este atributo, qualquer tentativa do applet de acessar variáveis ou executar funções ou métodos JavaScript causará em uma exceção de segurança no applet.

No HTML 4.0, applets também podem ser incluídos em uma página usando o bloco genérico <OBJECT>, que também serve para importar imagens, *plug-ins* e construir *frames* internos.

Existem muitos applets úteis disponíveis gratuitamente na Web que podem ser usados por autores de páginas Web e programadores JavaScript sem que precisem saber Java. Os mais populares implementam *banners* para rolagem de texto, ícones inteligentes, gráficos, planilhas de dados e interfaces para bancos de dados. A maioria são configuráveis através de parâmetros que o autor da página define em elementos <PARAM>. No apêndice A há uma lista de sites onde se pode encontrar tais applets. Exemplos são o *Gamelan* (http://www.gamelan.com) e o site da Sun sobre Java (http://java.sun.com). A seção a seguir mostrará como incluir um applet obtido em um desses sites.

3.5.3. Como incluir um applet em uma página



O exemplo a seguir mostra como incluir o applet `Clock2` em uma página Web. Este applet é distribuído pela *Sun* gratuitamente em `http://java.sun.com` e juntamente com o ambiente de desenvolvimento Java. O applet pode ser incluído na página da forma *default*, sem especificar parâmetros, ou de forma personalizada, definindo parâmetros que alteram a cor de fundo, dos ponteiros e do mostrador.

Como é que o autor de uma página HTML saberá quais parâmetros usar e configurar? Vendo-o funcionar! O applet é distribuído com uma página HTML que mostra como usá-lo. Ele deve ser incluído na página HTML usando o nome do arquivo executável java, que neste caso é `Clock2.class` e deve ocupar uma área de no mínimo 170x150 pixels (tudo isto está na página):

```
<applet code="Clock2.class" height=150 width=170></applet>
```

Com o código acima, o relógio aparece na página como mostrado na figura, com ponteiros azuis, visor com letras pretas e fundo branco. O autor do applet permite, porém, que o autor da página altere esses parâmetros através de descritores `<PARAM>`. Os três parâmetros modificáveis são:

- `bgcolor` – cor de fundo (branco é *default*)
- `fgcolor1` – cor dos ponteiros e dial (azul é *default*)
- `fgcolor2` – cor dos números e ponteiro de seguntos (preto é *default*)

Todos os parâmetros devem receber como valor um número hexadecimal representando uma cor no formato RGB (mesmo formato usado em HTML): `ff0000` – vermelho, `ffffff` – branco, `0000ff` – azul, etc.

Portanto, para incluir o relógio acima em uma página com um fundo cinza claro, ponteiros marrons e letras douradas, o código seria:

```
<applet code="Clock2.class" width=170 height=150>
      <param name=bgcolor value="dddddd">
      <param name=fgcolor1 value="800000">
      <param name=fgcolor2 value="808000">
</applet>
```

Applets remotos

Caso o applet esteja em um diretório diferente daquele onde está a página, será necessário usar o atributo `CODEBASE`, para informar a URL base. Por exemplo, se o arquivo `.class` que usamos acima estiver em `http://www.abc.com/clocks/`, precisamos usar:

```
<applet codebase="http://www.abc.com/clocks/" code="Clock2.class" ... >
   ...
</applet>
```

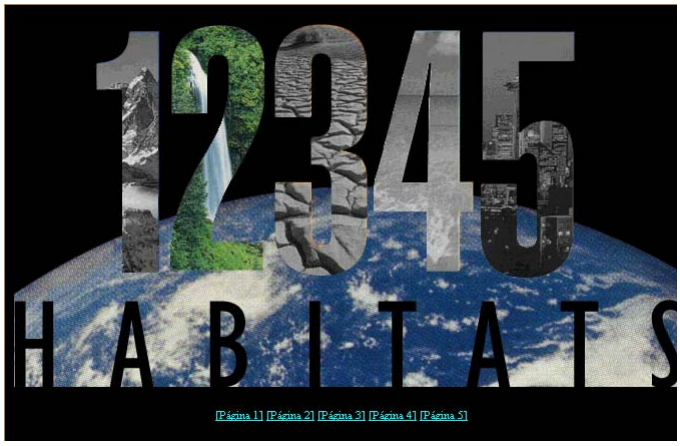
3.5.4. Conclusão

Applets oferecem um meio para escapar das limitações do browser e do HTML sem limitar o público-alvo da página de acordo com a plataforma de computação que possuem. É preciso, porém, lembrar que podem ocorrer problemas. Como applets são programas, podem conter ou provocar erros. Geralmente esses erros não são graves a ponto de comprometer a segurança do usuário, pois os browsers que suportam applets possuem um rigoroso mecanismo que restringe o que applets podem fazer. Eles não podem, por exemplo, escrever no seu disco ou imprimir. Mas os erros podem levar o applet a não funcionar, ou pior, a consumir os recursos de memória do browser e do sistema, forçando o usuário a abandonar sua execução (e possivelmente a do browser). Applets escritos para as versões Java 2 podem não funcionar até em browsers recentes, como o *Internet Explorer 4*. Embora funcionem em plataformas diferentes, applets podem rodar até 30 vezes mais rápido em certas plataformas. Isto depende da qualidade da máquina virtual suportada pelo browser.

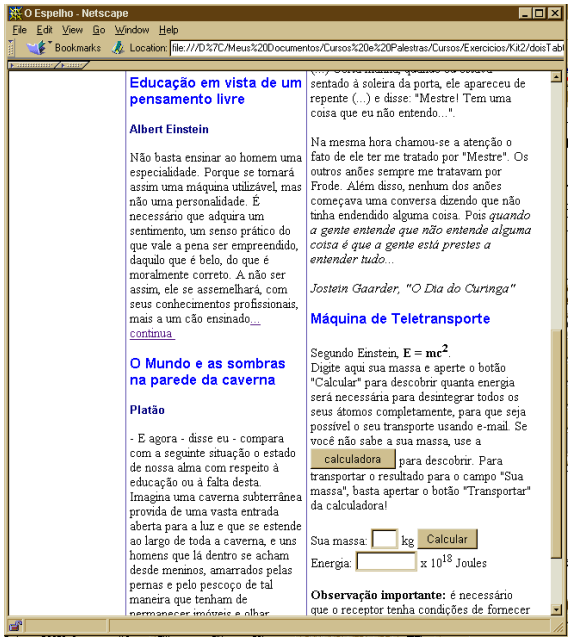
Levando em conta todas essas questões, você deve usar applets quando for necessário incluir recursos que o browser e o HTML sozinhos não seriam capazes de oferecer. É preciso, porém, levar em conta o risco de inacessibilidade caso o applet não funcione por um motivo ou outro. Se possível, deve haver uma forma alternativa de obter a informação que ele guarda ou a funcionalidade que ele implementa. Se o seu público-alvo permitir, você talvez decida que é melhor usar outra tecnologia mais simples, visível em menos plataformas, mas que atenda seus objetivos.

3.6. Exercícios

1. Monte o quebra cabeças abaixo com as imagens `im-01.jpg` a `im-05.jpg` do `kit1` (disponíveis no site do curso), juntamente com as imagens, `esquerda.jpg`, `direita.jpg` e `baixo.jpg` para obter o desenho abaixo usando tabelas HTML.



2. Organize o texto (do `kit2`) da forma mostrada nas figuras abaixo



4 Folhas de Estilo

4.1. Introdução

4.1.1. O que são folhas de estilo?

Uma folha de estilos é um conjunto de regras que informa a um programa, responsável pela formatação de um documento, como organizar a página, como posicionar e expor o texto e, dependendo de onde é aplicada, como organizar uma coleção de documentos.

A maior parte dos programas de editoração eletrônica e processadores de texto modernos trabalham com folhas de estilos. O processo consiste em definir um rótulo (nome do estilo) para um determinado parágrafo e em seguida alterar os seus atributos. Todo parágrafo que for rotulado com aquele estilo passará a exibir as características definidas anteriormente. Qualquer alteração nos atributos de um estilo afetará todos os parágrafos que estiverem rotulados com ele.

Esta descrição, que se aplica a estilos em processadores de texto e programas de editoração eletrônica, também vale para a Web. Na Web, os "parágrafos" são blocos marcados por descritores HTML como <H1>, <P>, etc. Para fazer com que todos os blocos de textos marcados com <H1> em um documento sejam exibidos em tamanho de 48 pontos, basta definir a regra:

```
H1 {font-size: 48pt}
```

dentro de uma "folha de estilos" aplicada ao documento.

A folha de estilos pode ser um arquivo de textos simples (alfabeto ISO-Latin1) com a extensão `.css`. Para vinculá-lo a uma página HTML, esta deve ter dentro do seu bloco <HEAD> ... </HEAD> o seguinte descritor:

```
<LINK REL="stylesheet" HREF="url_do_arquivo.css">
```

O restante deste artigo tratará dos fundamentos da tecnologia de folhas de estilos aplicáveis ao HTML, chamada de *Cascading Style Sheets* (folhas de estilo em cascata), mostrando como estabelecer as regras de estilo para um bloco de texto, uma página ou todo um site. Seções específicas abordarão cores, imagens, tipologia e posicionamento. Este texto não é completo. Omitimos propriedades e recursos não suportados nos browsers e nos limitamos àqueles recursos que constam da especificação CSS1 (não incluímos recursos proprietários nem a maior parte das novidades do CSS2 que não funcionam nos browsers disponíveis no mercado). Para uma abordagem mais completa, consulte a documentação oficial do W3C: `http://www.w3.org/Style/`.

4.1.2. Para que servem as folhas de estilo

Separar apresentação da estrutura

Com isto é possível voltar a suportar browsers antigos que antes estavam condenados por não conseguirem ler a informação sem perdas. Com a informação toda armazenada no HTML (estrutura), a apresentação (estilo) seria uma camada a mais, alterando a disposição do texto, cores, etc. mas sem afetar a estrutura essencial da informação. Isto permite que uma página tenha vários estilos e use *scripts* (programas embutidos) para decidir qual carregar (em função do browser e da plataforma). Isto é muito menos trabalho que fazer uma página para cada browser e plataforma, pois a atualização é feita apenas no HTML. Também, com isso, é possível ter uma folha de estilos especial somente para impressão, onde haveria informações de quebra de páginas, etc. (este recurso não faz parte da versão 1 do CSS).

Controle absoluto da aparência da página.

É algo que não se tem com o HTML. Pode-se usar tabelas, GIFs invisíveis de um pixel e mais uma dúzia de "macetes" mas não se consegue fazer o texto fluir suavemente em volta de uma imagem irregular, por exemplo. Além do mais, quanto mais sofisticada a técnica, mais difícil é de codificar e mais "sujo" fica o código, o que o torna mais sujeito a erros. Com CSS, pode-se colocar uma imagem em qualquer lugar da página (até fora dela), usando técnicas de posicionamento absoluto ou relativo. Pode-se escolher a posição exata da imagem de *background* e fazê-la combinar com algo no *foreground*. As dimensões e posições são exatas e dadas em unidades conhecidas no mundo da tipografia como pixels, pontos, *paicas*, milímetros.

Páginas mais leves

Com folhas de estilo é possível criar muitas páginas com um layout sofisticado que antes só era possível usando imagens, tecnologias como *Flash* ou applets Java. Estas páginas eram sempre mais pesadas, pois precisavam carregar imagens, componentes, programas. Com CSS é possível definir texto de qualquer tamanho, posicioná-lo por cima de outros objetos, ao lado ou por cima de texto e conseguir efeitos sofisticados a um custo (banda de rede) baixo. É possível ainda importar fontes (que o usuário talvez não tenha).

Manutenção de um grande site

Uma folha de estilos serve para toda uma coleção de páginas, podendo ser usada para dar um estilo consistente a todo o site. Sendo aplicada em separado da informação e estrutura, não precisará ser atualizada todas as vezes em que a informação for mudada. A página pode ser atualizada em um editor HTML ou gerador de HTML simples, sem recursos de cor ou alinhamento, e ser formatada na hora em que for carregada pelo browser. É possível também fazer o contrário: mudar o estilo sem alterar a informação, como ter uma página que sempre carrega com um estilo diferente.

O uso das folhas de estilo depende da boa estrutura do HTML. A linguagem CSS (é uma linguagem declarativa) trabalha com os elementos tratando-os como "objetos". Cada parágrafo <P>, cada <H1>, cada <IMG> é um objeto. Objetos podem ser agrupados de várias formas. A cada objeto ou grupo de objetos podem ser atribuídas propriedades de estilo definidas em regras. Por exemplo, considere a seguinte regra: "todo objeto P da classe 'editorial' será azul, terá tamanho de 12 pontos, espaçamento duplo, alinhamento pela direita e usará a família de fontes Arial, ou, se esta não existir, Helvetica, ou então a fonte sem-serifa *default* do sistema". Um arquivo CSS com apenas a regra acima conteria o texto:

```
P.editorial {color: 0000ff;
             font-size: 12pt;
             line-height: 24pt;
             text-align: right;
             font-family: arial, helvetica, sans-serif}
```

Se a folha de estilos acima for aplicada a uma página que possua parágrafos <P> rotulados com o nome "editorial" (atributo 'CLASS=editorial'), eles serão formatados de acordo com as propriedades especificadas se o browser suportar CSS. Se o browser não suportar CSS, a estrutura será mantida (embora a aparência não seja a ideal) e o usuário conseguirá ter acesso à informação estruturada, mesmo em um meio de visualização mais limitado.

Quando se usa CSS, são poucas as modificações necessárias no HTML. Não são necessários novos descritores ou extensões. No exemplo acima, teremos que incluir apenas um atributo a mais (o atributo CLASS, do HTML 4) classificando os parágrafos que fazem parte do nosso 'editorial' (parágrafos que tem uma função diferente dos demais).

A grande vantagem das folhas de estilo é a preservação da estrutura do HTML e um controle muito melhor do autor sobre a sua aparência na tela do usuário final. Uma página deverá aparecer da melhor forma possível tanto num *PowerPC* sofisticado como naquele IBM PCXT 4.77MHz rodando o *Lynx* for DOS. O primeiro utilizará todos os recursos gráficos determinados pelas folhas de estilo. O segundo as ignorará, mas preservará a estrutura e a informação

4.2. Regras Básicas

4.2.1. Regras, declarações e seletores

A estrutura dos estilos é bastante simples. Consiste de uma *lista de regras*. Cada regra possui um bloco, entre chaves ({ e }), de uma ou mais declarações aplicáveis a um ou mais *seletores*. Um seletor é algo no qual pode-se aplicar um estilo. Pode ser um descritor HTML, uma hierarquia de descritores ou um atributo que identifique um grupo de descritores. Uma *folha de estilos* consiste de uma ou mais linhas de regras, da forma:

```
seletores { declarações }
```

As regras podem estar dentro de um arquivo de texto (ISO Latin1 ou ASCII 8-bit) com extensão ".css" ou *embutidas* em um arquivo HTML (as várias maneiras de aplicar estilos a um arquivo HTML serão vistas na seção seguinte).

Um exemplo de folha de estilos com apenas uma regra foi mostrada na seção anterior:

```
H1 {font-size: 48pt}
```

Nesta regra, `H1` é o seletor e `{font-size: 48pt}` é o bloco da declaração, que estabelece um tamanho de fonte (prop. `font-size`) para todos os objetos (parágrafos) marcados com `<H1>`.

As declarações são feitas usando a sintaxe:

```
propriedade: valor
```

Observe que se usa dois-pontos (`:`) e não igual (`=`) para aplicar um valor a uma propriedade. Pode haver mais de uma declaração de estilo para um seletor. Isto pode ser feito em outro bloco. Cada linha acrescenta ou sobrepõe declarações feitas em linhas anteriores:

```
H1 {  font-size: 24pt }
H1 {  color: blue }
H1 {  font-size: 18pt }
```

No trecho acima, o texto marcado com `<H1>` será azul e terá tamanho de `18pt` porque a regra `H1 { font-size: 18pt }` ocorreu depois da regra `H1 { font-size: 24pt }`.

4.2.2. Múltiplas declarações e seletores

Várias declarações de estilo podem ser aplicadas de uma vez a um seletor. As declarações, então, precisam ser separadas por ponto-e-vírgula (`;`) :

```
H1 {font-size: 18pt; color: blue; font-family: Caslon, serif }
BODY { background : navy; color : white }
```

Os espaços em branco (espaços, novas-linhas e tabulações) são ignorados pelo browser na hora de interpretar uma folha de estilos e podem ser usados para melhorar a sua legibilidade. As duas linhas acima ficariam mais legíveis da forma:

```
BODY {background : navy;
      color : white }

H1   {color: blue;
      font-size: 18pt;
      font-family: Caslon, serif }
```

Uma declaração só termina com uma fecha-chaves (`}`) ou com um ponto-e-vírgula (`;`). Dependendo da propriedade, esta pode ter vários valores separados por vírgulas ou valores compostos com as palavras separadas por espaços:

```
P  {  font: 12pt "Times New Roman" bold }
H2 {  font-family: Arial, Helvetica, Sans-serif }
```

As aspas devem ser usadas quando uma palavra que tem espaços precisar ser usada. No exemplo acima, o nome "Times New Roman" deveria ser tratado como o nome de uma fonte distinta, e não como três valores, o que ocorreria se as aspas não estivessem presentes.

Assim como um seletor pode ter várias propriedades definidas para ele, um mesmo conjunto de propriedades pode ser aplicada a um grupo de seletores, separando-os com vírgulas:

```
H1, H2, H3 { color: blue;
             font-size: 18pt;
             font-family: Arial, Helvetica, Sans-serif }
```

As declarações de estilo podem ser aplicadas em quase qualquer descritor HTML - no mundo perfeito! Na prática, muitos browsers ainda têm problemas de compatibilidade, e não implementam a especificação à risca, como veremos adiante.

Ao utilizar folhas de estilo, deve-se respeitar os elementos HTML que possuem descritores finais freqüentemente ignorados, como </P>, </LI>, etc. A falta do </P> pode causar o "vazamento" das declarações de estilo para fora do parágrafo em alguns browsers.

4.2.3. Comentários e instruções

Além das regras, um arquivo CSS pode ter ainda comentários e instruções (precedidas de @). No CSS1 apenas uma instrução é definida: `@import`. Ela é usada para que uma folha de estilos possa importar estilos de outro arquivo CSS através de uma URL. Os estilos importados sempre têm menos precedência que os locais (ou seja, os locais podem sobrepor os importados). A sintaxe da instrução `@import` é:

```
@import url(url_da_folha_de_estilos)
```

Não deve haver outras estruturas (a não ser comentários) na linha onde há uma instrução. Exemplos do uso de `@import`:

```
@import url(../basico.css)
@import url(http://longe.com/estilos/basico.css)
```

Pode-se inserir trechos que serão ignorados pelo browser ao interpretar folhas de estilo usando blocos de comentário. Comentários em folhas de estilos são iguais a comentários em linguagens como C ou Java: entre `/*` e `*/`:

```
/* este texto é ignorado até que seja encontrado
um asterisco seguido por uma barra */
```

4.2.4. Valores

Os valores que são aplicados às propriedades têm uma sintaxe que depende da propriedade que os usa. Propriedades que envolvem tamanho (tamanho de fontes, espaçamento, etc.) geralmente recebem valores que consistem de um número e uma unidade ou porcentagem. O

sinal de porcentagem ou unidade deve estar junto ao número correspondente sem espaços. Ou seja, deve-se escrever `font-size: 24pt` e não `font-size: 24 pt`.

Cores e arquivos externos podem requerer uma função para serem definidos. São duas as funções (ou procedimentos) do CSS1: `rgb()`, que constrói uma cor, e `url()`, que retorna um vínculo para uma imagem ou arquivo CSS (usada em instruções `@import`).

Há quatro maneiras diferentes de especificar cores em CSS: usando o nome do sistema (red, yellow, blue, black, lightGray), usando seu código RGB hexadecimal (ff0000, ffff00, 0000ff, 34adfc, 80a7a7) ou usando a função `rgb()`. A função `rgb()` requer três argumentos que representam a intensidade dos componentes vermelho (R), verde (G) e azul (B) de uma cor em forma de luz (não opaca). A intensidade pode ser expressa em valores inteiros de 0 (mínimo) a 255 (máximo) ou em valores fracionários de 0% a 100%. As instruções abaixo definem a mesma cor para um parágrafo:

```
P {color: red}
P {color: ff0000}
P {color: rgb(100%, 0%, 0%)}
P {color: rgb(255, 0, 0)}
```

Não deve haver espaço entre o "b" de `rgb` e o abre-parênteses.

A função URL pode ser usada em propriedades que requerem arquivos (no caso, imagens) como valores. Ela recebe um argumento apenas com a URL (relativa ou absoluta) da imagem:

```
P {background-image: url(../imagens/tijolos.gif)}
P {background-image: url(http://longe.com/imagens/pedras.png)}
```

4.2.5. Herança

Os estilos "herdam" propriedades de várias maneiras. Uma das formas é através da própria hierarquia do HTML. Se você declara propriedades para BODY, todos os descritores serão afetados a não ser que tenham as suas propriedades redefinidas dentro de um novo bloco de declarações CSS. Se um <I> está dentro de um <P> e todos os <P> são declarados como tendo a cor vermelha, o <I> também será vermelho a menos que haja um bloco, posterior àquela declaração, redefinindo as propriedades de <I>, por exemplo:

```
P {font: 12pt "Times New Roman" bold;
   color: red }

I {color: black }
```

faria com que o texto "seletor", no texto a seguir permanecesse preto:

```
<P>Um <I>seletor</I> é algo no qual pode-se aplicar um estilo.</P>
```

Se você definir atributos para os descritores <BODY> ou <HTML>, toda a página será afetada. No exemplo a seguir, uma cor de texto definida para BODY será usada para colorir todo o texto do documento, a não ser que sejam sobrepostos por uma regra subseqüente:

```
BODY {color: navy }
H1, H2 {color: yellow }
```

Os blocos acima farão com que todo o texto seja azul marinho, exceto aquele marcado com H1 ou H2, que será amarelo.

Os browsers comerciais têm problemas principalmente com a aplicação de estilos em BODY, portanto, freqüentemente é preciso mexer nas declarações de estilo, acrescentando propriedades redundantes para adaptá-los à realidade. No site do W3C (`http://www.w3.org`) há links para documentos que analisam essas diferenças entre browsers. O site `http://www.w3.org/Styçe/CSS/Test/` é uma plataforma de testes que pode ser usada para verificar se um browser suporta ou não determinada propriedade.

4.2.6. Descritores HTML especiais

Dois descritores HTML têm importância fundamental em CSS. Eles são descritores estruturais puros que não definem apresentação específica na folha de estilos nativa do browser. Com CSS é possível definir propriedades de apresentação para esses descritores. Eles são <DIV> e <SPAN>.

<SPAN> é um descritor que deve ser usado dentro de blocos de texto apenas. É chamado de descritor em-linha (*inline*), já que não quebra a linha antes ou depois. Ele se assemelha a descritores como <B>, <I>, <SMALL>, <A HREF> e <SUP> que servem para formatar texto dentro de parágrafos, células de tabela, etc.

<DIV> é um descritor que define um bloco ou seção da página. Pode ser usado para dividir a página em seções (e subseções no *Internet Explorer*) e permitir que sejam aplicados estilos específicos a essas seções. Descritores de bloco são <P>, <H1>, <TABLE>, etc. <DIV> define um bloco sem função ou aparência definida. A função e aparência será determinada em CSS.

4.2.7. Como incluir estilos em uma página

Há três formas de aplicar uma folha de estilos a uma página Web. Estas formas irão determinar a abrangência dos estilos: se afetarão um trecho limitado de uma página, se a toda a página, ou se irão afetar todo um site.

A primeira forma, mais abrangente, é a vinculação a um arquivo CSS. Isto é feito ligando a página HTML a um arquivo de folha de estilo, usando do descritor <LINK>). Este método permite que múltiplas páginas ou um site inteiro usem a mesma folha de estilos.

Para fazer um vínculo à uma folha de estilos externa, deve-se criar um arquivo de texto contendo um conjunto de regras de estilo em CSS, salvá-lo com uma extensão ".css" e chamá-

lo a partir de todos os documentos HTML onde o estilo será aplicado. Não é preciso compilar ou qualquer coisa do tipo. Depois que as definições estiverem prontas, o estilo estará pronto para ser usado. Pode ser importado através do descritor LINK:

```
<HEAD>
(...)
<LINK REL=STYLESHEET
   HREF="http://internet-name/mystyles.css"
   TYPE="text/css">
</HEAD>
```

O elemento <LINK> não tem descritor de fechamento e deve ser usado dentro do bloco <HEAD>.

Uma segunda forma, permite que estilos sejam aplicados localmente, em uma única página, embutindo uma folha de estilos diretamente na página HTML dentro de um bloco formado pelos descritores <STYLE> e </STYLE>. Este método permite que você altere as propriedades de estilo de uma única página.

A linguagem que é embutida em um bloco <STYLE> é a mesma usada nos arquivos CSS. <STYLE> ... </STYLE> deve ser usado em <HEAD>. Um atributo *type* informa o tipo de arquivo utilizado:

```
<style type="text/css">
 P  {  font: 12pt "Times New Roman" bold;
            color: red }
 I  {  color: black }
</style>
```

As propriedades declaradas no bloco <STYLE> sobrepõem qualquer propriedades anteriores do elemento (inclusive as de uma folha de estilos importada, se houver). É comum colocar todo o código entre comentários HTML (<!--e -->) para proteger browsers antigos da exibição indesejada do código:

```
<style type="text/css">
<!--
 P  {  font: 12pt "Times New Roman" bold;
       color: red }
 I {   color: black }
-->
</style>
```

Finalmente, há uma forma de aplicar estilos diretamente a um descritor individual. Para fazer isto deve-se usar o atributo STYLE em quase qualquer descritor. Este método não corresponde exatamente a uma "folha" de estilos, pois os estilos aplicados não são reaproveitáveis.

Permite alterar a aparência de um único descritor, de um conjunto deles ou de um bloco de informações da página. Por exemplo:

```
<P STYLE="color: green; font-size: 12pt">Este texto</P>
```

Esta propriedade é mais interessante quando aplicada em um descritor que é usado para agrupar vários outros, como <DIV>, que divide a página em seções ou <SPAN>, usado em situações bem específicas. Usar estilos desta forma é pouco diferente de usar atributos locais. Os benefícios de poder mudar a fonte, cores e outras características em todo o documento ficam mais difíceis.

Pode-se usar um, dois ou os três métodos ao mesmo tempo. As características definidas pelos mais específicos sobrepõem as definidas pelos mais genéricos. Por exemplo, suponha que o arquivo `estilos.css` contenha apenas as seguintes regras:

```
H1 { color: green;
     font-size: 24pt}
P { color: blue}
```

Suponha que ele seja carregado na página a seguir que possui um bloco <STYLE> com uma nova definição de P e H1.

```
<HEAD>
<link rel=STYLESHEET
     href="estilos.css"
     type="text/css">
```

```
<style type="text/css"> <!--
      P  {font: 12pt "Times New Roman" bold;
          color: red }
      H1 {color: black }
-->   </style>
</HEAD>
```

Mais adiante, existe um parágrafo e um título da forma:

```
<h1 style="color: navy">Auto da Compadecida</h1>
<p style="color: black">Ariano Suassuna (Recife, 1955)</p>
```

Qual estilo irá predominar? Pela regra de que o mais específico sobrepõe o mais geral, teremos uma página onde os <h1> têm tamanho 24pt (do estilo importado), cor preta (valor sobreposto pelo estilo da página) e os <p> são vermelhos (sobreposto pelo estilo da página). Nas duas linhas acima (e nelas apenas), o <h1> será azul marinho (sobrepondo o estilo da página) e o <p> será preto.

4.2.8. Classes e IDs

Às vezes um parágrafo tem uma aparência diferente dos outros parágrafos em uma certa parte do texto. Para mudar o estilo dele, pode-se incluir as declarações em um atributo STYLE. Mas, se tal procedimento torna difícil a localização e a gerência dos estilos, pode-se usar um recurso para marcá-lo de forma que seja considerado diferente. Isto pode ser feito atribuindo-lhe uma identificação única. Em HTML 3.2, pode-se usar o atributo ID:

```
<P ID=w779>Texto especial</P>
```

Para alterar as características deste parágrafo agora, pode-se usar o seu ID como seletor, da forma:

```
#w779 {color: cyan }
```

Se isto estiver em um arquivo CSS, todas as páginas que o usam e que tiverem um elemento com o ID #w779 serão afetadas. Se houver mais de um com o mesmo ID apenas o primeiro será afetado.

Melhor que usar ID é agrupar características semelhantes em classes. Uma classe é uma variação de um determinado objeto. Por exemplo, um texto teatral pode ter três parágrafos com apresentação diferente, representando as falas de três personagens. Se quiséssemos que cada um tivesse uma cor diferente, poderíamos declarar cada um como sendo de uma classe distinta:

```
<p class=padre>Eu retiro o que disse, João</p>
<p class=grilo>Retirando ou não retirando, o fato é que o cachorro
                              enterrou-se em latim</p>
<p class=bispo>Um cachorro? Enterrado em latim? </p>
```

```
<p class=padre>Enterrado latindo, Senhor Bispo, Au, au, au, não sa-
be? </p>
```

Para dar a cada parágrafo de um mesmo personagem (mesma classe) os mesmos atributos, usa-se:

```
P.grilo { color: maroon }
P.padre { color: black }
P.bispo { color: navy }
```

Desta maneira, todos os textos que deverão ser lidos pelo personagem "Bispo" estarão em azul marinho.

Uma classe também pode conter descritores diferentes. Se todos os textos citados por um certo autor tivessem que estar em outra cor ou fonte, poderíamos criar uma classe sem citar o descritor:

```
.verde { color: green }
```

Todos os descritores que tiverem o atributo CLASS=verde serão afetados, por exemplo: <P class=verde>, <h3 class=verde>, <table class=verde>, etc.

4.2.9. Links (pseudo-classes e pseudo-elementos)

Para seletores especiais que mudam de estado, como o texto marcado com <A>, é possível atribuir propriedades diferentes para cada estado:

```
A:link {color: red}
A:active {color: 660011}
A:visited {color: black; text-decoration: none}
A:hover {color: blue; text-decoration: underline}
```

muda as características dos links não-visitados, ativos e visitados. Assim como qualquer seletor, os links podem ser combinados com outros descritores:

```
P, A:link, H2 {color: red}
```

4.2.10. Seletores de contexto

Você também pode definir seletores que só serão aplicados se estiverem no contexto de um outro seletor, por exemplo:

```
P.verde EM {color: 000040}
```

indica que o EM só terá sua cor alterada se ocorrer dentro de um bloco P da classe "verde".

Os seletores de contexto podem ser bem longos:

```
.bispo P UL UL LI A.classX:link {font-style: italic }
```

fará com que apenas os links não visitados da classe "classX" que estejam dentro de itens de lista de segundo nível situados dentro de um parágrafo dentro de um bloco qualquer da classe "bispo" sejam mostradas em itálico.

4.2.11. Cascata de folhas de estilo

Existem seis diferentes folhas de estilo que podem ser definidas. Além das três formas que mostramos em seção anterior deste capítulo, há ainda, segundo a especificação, mais três folhas de estilos que podem afetar uma página: 1) uma folha de estilos que é importada por outra folha de estilos (isto é diferente daquela que é vinculada ao HTML, dentro de um <link>), 2) uma folha de estilos definida pelo usuário (ou leitor da página) e 3) a folha de estilos *default* do browser (que é usada quando outra folha não define os estilos).

Todas estas folhas de estilo diferentes podem provocar uma grande confusão se não houver uma regra clara de como devem ser consideradas. Ainda há um sétimo fator que é a formatação introduzida pelo HTML, como nos descritores <font> e atributos align=center. Listando todas as folhas de estilos que podem afetar um texto, temos:

1. *Folha de estilos default do browser:* todos os browsers possuem regras comuns para formatar um texto. A especificação HTML não impõe um formatação padrão. *Netscape Navigator* por exemplo usa um fundo cinza como padrão e links azuis sublinhados. Já o *Internet Explorer* prefere um fundo branco.
2. *Folha de estilos definida pelo leitor:* a especificação define a possibilidade do leitor estabelecer uma folha de estilos própria. Isto é parcialmente conseguido quando o browser permite que se escolha diferentes cores para fundo, texto e links.
3. *Folha de estilos vinculada ao HTML:* é a folha de estilos que é importada pelo arquivo HTML através do descritor de ligação <link>
4. *Folha de estilos importada:* uma folha de estilos externa (arquivo CSS) pode ser importada de dentro de outra folha de estilos (um outro arquivo CSS ou bloco <style> no HTML) usando um comando especial @import:

   ```
   @import url (outroestilo.css)
   ```

5. *Folha de estilos embutida no HTML:* é a folha de estilos que aparece na página HTML entre os descritores <style> e </style>.
6. *Folha de estilos local:* é aquela que é aplicada localmente a um descritor usando o atributo style="lista de declarações".
7. *Estilo definido pelo HTML:* atributos e descritores podem provocar alterações na aparência do texto, por exemplo: <font size>, <big>, <body bgcolor>, <p align=center>, etc.

Um browser que siga a especificação CSS à risca obedece a seguinte ordem de precedência: 1. Local, 2. Embutida, 3. Vinculada, 4. Importada, 5. HTML, 6. Leitor, 7. Browser

Na prática, as coisas não são tão bonitas. No *Internet Explorer* para *Macintosh*, a ordem é respeitada. Na versão do *Internet Explorer 3.0* para *Windows*, os estilos vinculados ao HTML têm

mais importância que os embutidos (o mesmo ocorre com *Explorer 4* e *Navigator 4*). No *Netscape Navigator 4* e *Internet Explorer 4*, os estilos aplicados via HTML têm precedência máxima (no *Internet Explorer 3* a precedência funciona corretamente, mas não no 4). Com as diferenças existentes nos browsers de hoje, não vale a pena ainda se aprofundar neste assunto. A solução ainda é testar, testar, testar antes de colocar no ar.

4.3. Fontes

Fontes são estilos de apresentação consistentes aplicados a alfabetos. Uma fonte consiste de atributos que alteram a aparência de um símbolo, sem alterar o seu significado. Oferecem as informações necessárias para que uma letra ou símbolo possa ser representado graficamente.

Os atributos essenciais de uma fonte são:

- Seu tipo (ou família)
- Seu tamanho
- Seu estilo (regular, itálico, *outline*, etc.)
- Seu peso (normal, negrito, *light*, *black*)

Para representar qualquer texto, portanto, é preciso escolher uma fonte, ou seja: um tipo, um estilo, um peso e um tamanho. Letras maiúsculas e minúsculas não são consideradas fontes diferentes, pois têm um significado distinto.

Os quatro atributos acima podem ser definidos em CSS através das propriedades `font-family`, `font-size`, `font-style` e `font-weight`. Não é preciso definir todas pois sempre têm valores *default*. CSS oferece ainda `font-variant`, que permite considerar outras variações de uma fonte.

4.3.1. font-family

Uma família de fontes (tipo) é selecionada com a propriedade `font-family`. Esta propriedade aceita uma lista de valores separados por vírgulas representando nomes de fontes existentes ou não no sistema do usuário. No final da lista, pode ser incluída uma referência a uma família genérica, que será usada caso nenhum dos nomes coincida com o nome de uma fonte do sistema.

A sintaxe é:

```
font-family: fonte1, fonte2, fonte3, ..., fonte-genérica
```

Exemplos:

```
H1 { font-family: garamond }
H2 { font-family: arial, helvetica, sans-serif }
H3 { font-family: courier, "courier new", monospaced }
H4 { font-family: monospaced }
```

As fontes `sans-serif` e `monospaced` são nomes genéricos. Não se referem a uma fonte em particular mas a um grupo genérico. Os outros são `serif`, `cursive` e `fantasy`.

Nome genérico	Serif	Sans-Serif	Monospace
Default do Unix	Times	Helvetica	Courier
Default do Mac	Times	Helvetica	Courier
Default do Win	Times New Roman	Arial	Courier New

O browser usará a primeira fonte da lista se a encontrar. Se não encontrar, irá procurar a fonte seguinte.

Se o nome de uma fonte tiver mais de uma palavra, este deverá ser colocado entre aspas. As aspas podem ser apóstrofes simples (') ou aspas duplas ("). Os apóstrofes são necessários quando for preciso especificar estilos dentro de um atributo HTML:

```
<p style="font-family: 'times new roman', sans-serif">...</p>
```

4.3.2. font-size

O tamanho de uma fonte é alterado usando `font-size`. Pode ser especificado em valores absolutos ou relativos. Para especificar um valor absoluto, pode-se usar:

```
font-size: número(pt | px | cm | mm | pc | in)
font-size: xx-small|x-small|small|medium|large|x-large|xx-large
```

O tamanho também pode ser especificado relativo ao elemento no qual o atual objeto está contido.

```
font-size: tamanho_relativo (smaller, larger)
font-size: comprimento (em ou ex)
font-size: porcentagem%
```

Exemplos:

```
H1 { font-size: 24pt}
H1 { font-size: x-large}
H1 { font-size: smaller}
H1 { font-size: 1.5em}
H1 { font-size: 150%}
<H1 style="font-size: 1cm">
```

Os tamanhos de pontos devem ser especificados como valores inteiros (mesmo se usados cm ou in). Os browsers podem formatar os tamanhos de forma diferente e os mesmos podem ser alterados pelos usuários nos browsers atuais. As unidades válidas são: pt (pontos), px (pixels), pc (paicas), cm (centímetros), mm (milímetros) e in (polegadas).

Os tamanhos absolutos chamados pelo nome (xx-small, etc.) correspondem aos tamanhos de 1 a 7 do descritor <font size> e podem variar de acordo com a família de fontes usada (variam bastante entre plataformas também). Veja um exemplo comparativo e o resultado no Internet Explorer para Windows:

```
<style>
 p {font-family: serif}
 .t1 {font-size: xx-small}
 .t2 {font-size: x-small}
 .t3 {font-size: small}
 .t4 {font-size: medium}
 .t5 {font-size: large}
 .t6 {font-size: x-large}
 .t7 {font-size: xx-large}
</style>
(...)
<p><span class=t1>xx-small</span> |
<span class=t2>x-small</span> |
<span class=t3>small</span> |
<span class=t4>medium</span> |
<span class=t5>large</span> |
<span class=t6>x-large</span> |
<span class=t7>xx-large</span> <br>
<font size="1">size=1</font> |
<font size="2">size=2</font> |
<font size="3">size=3</font> |
<font size="4">size=4</font> |
<font size="5">size=5</font> |
<font size="6">size=6</font> |
<font size="7">size=7</font> |</p>
```

xx-small | x-small | small | **medium** | large | x-large | xx-large

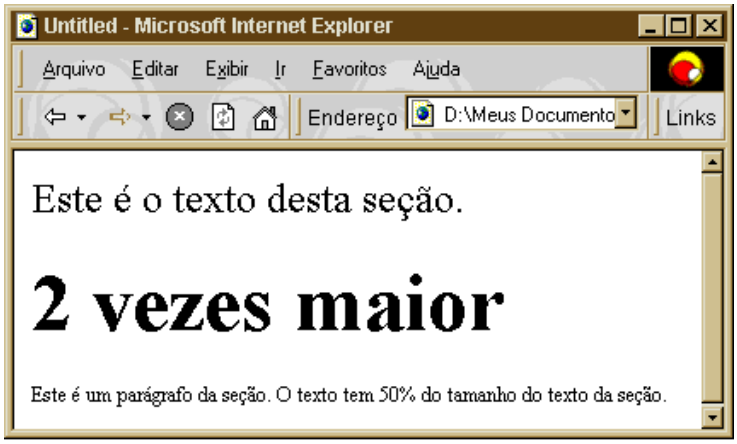
size=1 | size=2 | size=3 | **size=4** | size=5 | size=6 | size=7 |

Os tamanhos relativos funcionam como o <BIG> e <SMALL>, aumentando a fonte atual por 150%. A diferença é que podem passar além do limite xx-large (ou <font size=7>) ou xx-small (font size=1>).

Os comprimentos referem-se a unidades comuns em tipografia. Um "em" é uma distância horizontal equivalente ao tamanho de uma fonte (se uma fonte tem 20 pontos de tamanho, um em corresponde a uma distância de 20 pixels de largura). Um "ex" é a altura das letras de caixa-baixa (sem incluir as hastes ascendentes e descendentes). Tanto "em" como "ex" usam valores relativos ao elemento que contém o elemento atual. São úteis principalmente em espaçamento, sendo pouco usados em fontes.

As porcentagens são afetadas pela herança, por exemplo:

```
<style>
 .sec {font-size: 18pt};
 .sec H1 {font-size: 200%}
 .sec P  {font-size: 50%}
 </style>
</head>

<body>
<div class=sec>Este é o texto desta seção.
<h1>2 vezes maior</h1>
<p>Este é um parágrafo da seção. O texto tem 50% do tamanho do texto
da seção.</p>
</div>
```



As porcentagens de 50% e 200% são aplicadas na fonte atual, que é a fonte do bloco que contém os dois elementos (<DIV>), e que tem tamanho 18pt. O resultado é que o <H1> será exibido em tamanho 26pt e <P> em tamanho 9pt.

4.3.3. font-style e font-weight

O estilo de uma fonte é afetado através de duas diferentes propriedades: `font-weight`, que altera o peso da fonte, e `font-style`, que altera o estilo ou inclinação.

Sintaxe:

```
font-style: normal (ou italic, oblique)
```

Exemplos:

```
H1 { font-style: italic }
I  { font-style: normal }
<p style="font-style: oblique">...</p>
```

Sintaxe:

```
font-weight: normal | bold (normal=400 e bold = 700)
font-weight: bolder | lighter (valores relativos)
font-weight: 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
```

Exemplos:

```
H1 { font-weight: normal }
B { font-weight: 900 }
<b> ... <b style="font-weight: bolder">...</b> ... </b>
```

A palavra `oblique` deve fazer com que fontes chamadas de "oblique" (se existirem no sistema) sejam usadas, assim como ocorre com fontes "italic". A rigor, `italic` é uma fonte distinta da normal, e não, meramente uma versão inclinada da mesma. Os browsers, porém, não encontrando um equivalente "italic", "oblique", "kursiv" ou similar irão inclinar o texto, simulando um itálico.

Os valores numéricos para `font-weight` oferecem um controle excepcional sobre o peso da fonte na tela, embora isto esteja limitado aos níveis disponíveis nas fontes instaladas (para um mesmo nome de fonte). Na prática, dos 9 níveis disponíveis de peso, se consegue 4 ou 5 níveis diferentes de mais pesado ou mais leve. 700 é o "bold" típico e 400 é o "normal".

O exemplo a seguir ilustra o efeito com a fonte "Tahoma" (Windows):

```
<style type=text/css>
   P     {font-family: tahoma;
          font-size: 18pt;}
   .b100 {font-weight: 100}
   .b200 {font-weight: 200}
   .b300 {font-weight: 300}
   .b400 {font-weight: 400}
   .b500 {font-weight: 500}
   .b600 {font-weight: 600}
   .b700 {font-weight: 700}
   .b800 {font-weight: 800}
   .b900 {font-weight: 900}
   .nor {font-weight: normal}
   .bol {font-weight: bold}
</style>
(...)
<p><span class=b100>100</span>
<span class=b200>200</span>
<span class=b300>300</span>
<span class=b400>400</span>
<span class=b500>500</span>
<span class=nor>normal</span><br>
<span class=b600>600</span><br>
<span class=b700>700</span>
<span class=b800>800</span>
<span class=bol>bold</span><br>
<span class=b900>900</span></p>
```

Os valores `lighter` e `bolder` especificam pesos de fontes relativos a algum valor herdado. Eles avançam ou retrocedem na escala de 100 a 900 relativos aos pesos de fontes.

4.3.4. font-variant

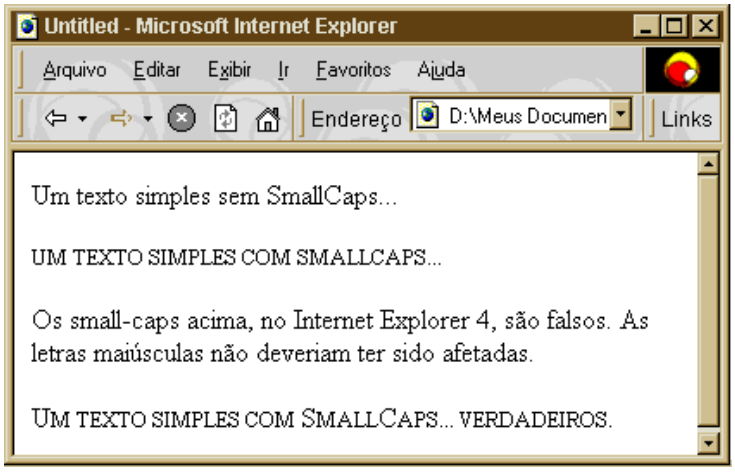
Atualmente a única opção disponível para esta propriedade é `small-caps`, que deve colocar o texto selecionado em maiúsculas, porém menores que as capitulares. Na prática, até as maiúsculas são reduzidas no *Internet Explorer*.

Sintaxe:

```
font-variant: small-caps
```

Exemplos:

```
<style>
 .sm {font-variant: small-caps}
 .tc {font-size: 120%}
 </style>
</head>

<body>
<p>Um texto simples sem SmallCaps...</p>
<p class=sm>Um texto simples com SmallCaps...</p>
<p>Os small-caps acima, no Internet Explorer 4, são falsos. As letras maiúsculas não deveriam ter sido afetadas.</p>
<p><span class=sm><span class=tc>U</span>m texto simples com <span class=tc>S</span>mall<span class=tc>C</span>aps... verdadeiros</span>.</p>
```



4.3.5. A propriedade font

Para especificar várias propriedades de um seletor de uma vez só, pode-se usar a propriedade `font` em vez de definir em separado `font-size`, `font-weight`, `font-family`, etc. Nesta sintaxe, a ordem dos fatores é importante, porém nem todos os elementos precisam estar presentes:

```
font: font-style font-variant font-weight font-size
      line-height font-family
```

Exemplos:

```
H1 {font: italic 700 24pt Tahoma, Arial, sans-serif }
```

4.4. Atributos de texto

As propriedades desta seção tratam de transformações e atributos aplicados a texto, não afetando a exibição das fontes. Os atributos tipográficos afetam a forma como o texto é apresentado na tela como o espaçamento entre linhas, entre palavras, entre letras, o alinhamento de parágrafos e a endentação.

A propriedade `text-transform` permite colocar letras em maiúsculas ou minúsculas e a propriedade `text-decoration` permite acrescentar ou tirar efeitos decorativos do texto como riscados e sublinhados.

4.4.1. text-transform

A propriedade `text-transform` realiza transformações no formato caixa-alta ou caixa-baixa do texto. *Sintaxe:*

```
text-transform: capitalize
text-transform: uppercase
text-transform: lowercase
text-transform: none (valor default)
```

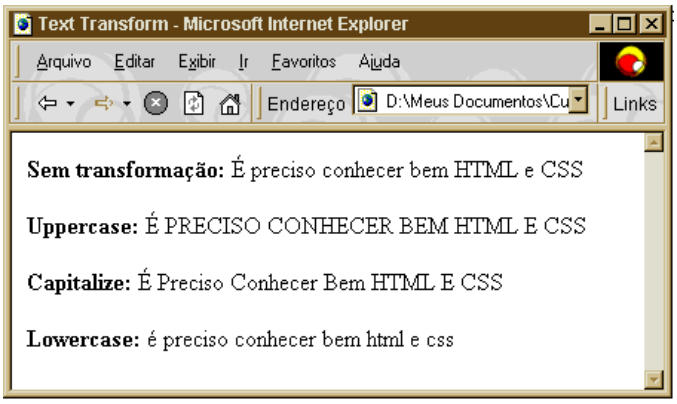
Exemplos:

```
H1 {text-transform: capitalize}
```

Capitalize coloca a primeira letra de cada palavra em maiúsculas. `Uppercase` coloca tudo em maiúsculas e `lowercase` coloca tudo em minúsculas. `None` remove qualquer transformação deixando o texto da forma como foi definido antes das transformações.

```
<style>
p {font-weight: bold}
span {font-weight: normal}
.non {text-transform: none}
.upp {text-transform: uppercase}
```

```
 .cap {text-transform: capitalize}
 .low {text-transform: lowercase}
 </style>
(...)
<p>Sem transformação:
<span class=non>É preciso conhecer bem HTML e CSS</span></p>
<p>Uppercase:
<span class=upp>É preciso conhecer bem HTML e CSS</span></p>
<p>Capitalize:
<span class=cap>É preciso conhecer bem HTML e CSS</span></p>
<p>Lowercase:
<span class=low>É preciso conhecer bem HTML e CSS</span></p>
```



4.4.2. *text-decoration*

A propriedade `text-decoration` permite colocar (ou tirar) sublinhados, linhas sobre e atravessando o texto, etc. *Sintaxe:*

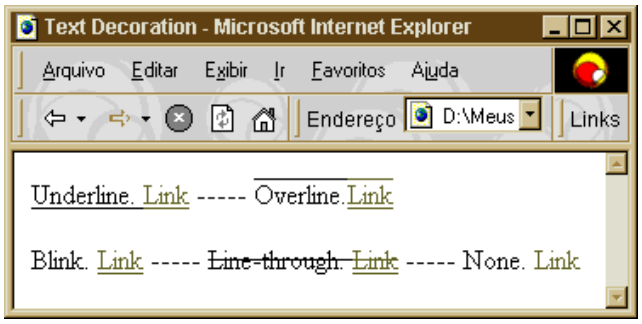
```
text-decoration: underline     (default em links)
text-decoration: overline
text-decoration: line-through
text-decoration: blink
text-decoration: none    (default)
```

Exemplos:

```
h1 {text-decoration: overline}
<a href="algumlugar.html"
   style="text-decoration: none">Link sem sublinhado</a>
```

```
<style>
 .und {text-decoration: underline}
 .ovr {text-decoration: overline}
 .blk {text-decoration: blink}
 .lin {text-decoration: line-through}
 .non, .non a {text-decoration: none}
</style>
```

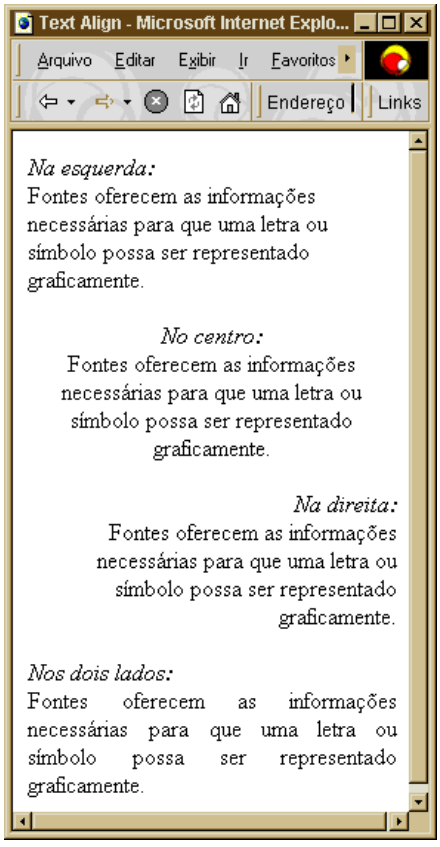
```
(...)
<p><span class=und>Underline.     <a href="#">Link</a></span> -----
   <span class=ovr>Overline.      <a href="#">Link</a></span></p>
<p><span class=blk>Blink.         <a href="#">Link</a></span> -----
   <span class=lin>Line-through. <a href="#">Link</a></span> -----
   <span class=non>None.         <a href="#">Link</a></span></p>
```



Vínculos (links) são normalmente sublinhados na maior parte dos browsers. O sublinhado pode ser removido com a propriedade text-decoration: none.

O browser Netscape 4 não suporta a propriedade overline. O Internet Explorer não suporta a propriedade blink.

4.4.3. text-align e vertical-align



CSS oferece propriedades que permitem controlar o alinhamento horizontal do texto, seu alinhamento vertical e endentação do texto na primeira linha. O alinhamento horizontal é o mesmo conseguido com o atributo `align` do HTML, só que o da folha de estilos tem precedência. A sintaxe é:

```
text-align: left | right | center |
            justify
```

O alinhamento só se aplica a elementos de bloco (<P>, <DIV>, H1, etc.) e elementos como applets e imagens. A característica é herdada para sub-blocos. O valor *default* é sempre `left`. Exemplo:

```
DIV { text-align: center }
```

Alinhamento vertical em HTML só pode ser aplicado a tabelas e imagens. Com CSS, esta propriedade é estendida a qualquer elemento de texto e imagens. A sintaxe é:

```
vertical-align: baseline | top | text-top | middle |
                bottom | text-bottom
vertical-align: sub | super
vertical-align: porcentagem %
```

O valor *default* é `baseline`. As porcentagens referem-se a altura da linha (`line-heght`) do próprio elemento. Usando porcentagens negativas consegue-se sobrepor elementos.

Na prática, apenas o *Internet Explorer 4* suporta `vertical-align` com os valores `sub` e `super` (coloca elementos em subscrito ou sobrescrito).

4.4.4. *text-indent*

A propriedade `text-indent` se aplica a elementos de bloco e realiza a endentação da primeira linha. A sua sintaxe é:

```
text-indent: comprimento
text-indent: porcentagem
```

A porcentagem ocorre em relação à largura do elemento que contém o seletor. Pode ser a largura total da página, a largura da célula de uma tabela, etc. Exemplos:

```
P { text-indent: 1 cm }
P {text-indent: 50% }
<P style="text-indent: 1in">
```

A endentação tratada aqui é apenas para uma linha de texto. Para endentar blocos inteiros, deve-se usar as margens (em seção mais a frente).

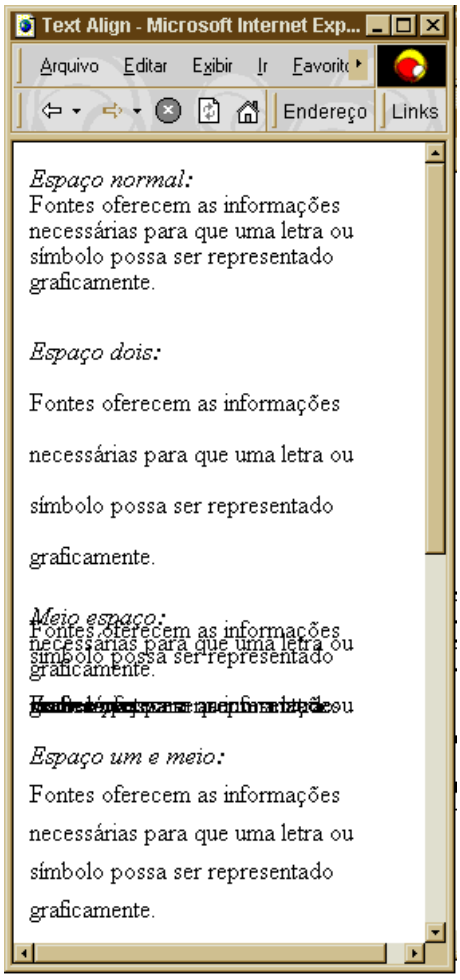
4.4.5. *line-height*

Este atributo é usado para controlar o espaço que existe antes e depois de uma linha de texto. Ela especifica a altura total de uma linha de texto. Se você tem um texto de 10 pontos e uma `line-height` de 20 pontos (`line-height`: 2), haverá 5 pontos antes e 5 pontos depois de cada linha de texto (espaço duplo). O espaço simples equivale geralmente a `line-height`: 120%. Uma `line-height` menor que o tamanho da fonte produzirá sobreposição de texto.

Embora ambos os browsers mais populares suportem este recurso, ele não ocorre da maneira correta. Os browsers não acrescentam a mesma quantidade de espaço antes e depois como esperado.

Um *bug* no *Internet Explorer 3* faz com que ele interprete valores absolutos (sem unidade) como valores em pixels. Por exemplo, 1.5 indica espaço 1 e meio ou 150%. No *Internet Explorer 3* as linhas ficam sobrepostas pois o browser interpreta a unidade como 1.5 pixels. Evite, portanto, usar valores absolutos (use porcentagens).

A sintaxe é:



```
line-height: número_absoluto
line-height: comprimento ou unidade
line-height: porcentagem
```

Exemplos:

```
H1 {line-height: 0 }     // sobreposição de linhas
H1 {line-height: 2 }     // espaço duplo
H1 {line-height: 0.3em }
H1 {line-height: 150% }  // espaço 1 e meio
```

Se você usar um valor percentual menor que 100%, um valor absoluto menor que 1 ou uma unidade menor que o tamanho da fonte, haverá sobreposição de linhas.

4.4.6. letter-spacing

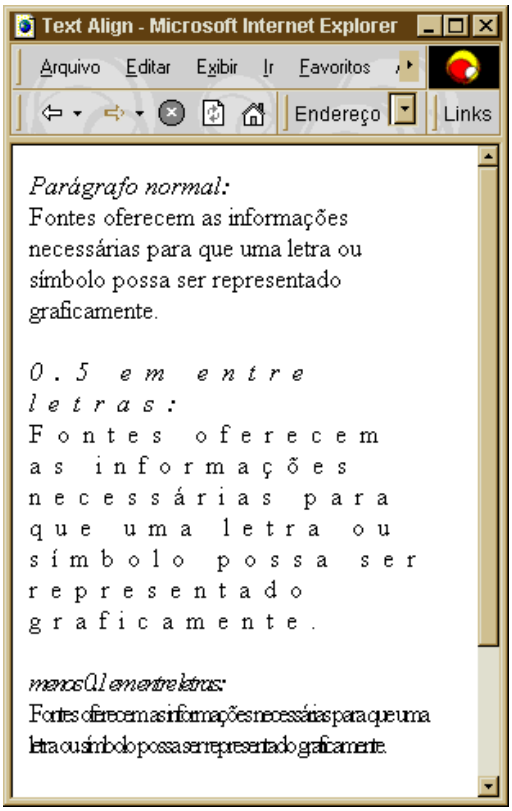
A propriedade `letter-spacing` altera o espaço entre as letras. A sua sintaxe é:

```
letter-spacing: normal
letter-spacing: comprimento
```

As unidades podem ser quaisquer uma das unidades válidas para tamanho de fontes (pt, px, pc, cm, in, mm, em e ex). Na tipografia, é mais comum usar "em" como medida de espaçamento por ser proporcional ao tamanho da fonte.

Pode-se usar também valores negativos para sobrepor caracteres, criar ligaduras (usadas em kerning) e caracteres especiais (por exemplo, sobrepondo / com \).

O suporte a `letter-spacing` nos principais browsers ainda é inconsistente. O *Netscape* (versão 4) não o suporta.



4.5. Cores

Com as propriedades de cores, podemos controlar as cores de várias partes da página, do texto, do fundo da página e de elementos HTML. Além disso, podemos aplicar imagens de fundo em qualquer elemento, não só no elemento BODY como já se faz em HTML.

As cores definidas em CSS, assim como em HTML, podem ser especificadas por um número em hexadecimal (representando um código RGB) ou por um nome. Além dessas duas formas, podem ainda ser especificadas por três números decimais, representando também o código RGB da cor.

Os códigos RGB informam a quantidade de luz vermelha, verde e azul que compõe a cor, respectivamente. Cada cor pode ter 16 níveis de intensidade: 0 a 256 (00 a FF, em hexadecimal). O total de combinações possíveis é de 16.777.216 cores.

A exibição correta das cores depende da capacidade do vídeo onde serão vistas. Poucos sistemas têm capacidade de mostrar mais que 65.536 cores simultâneas. A maioria só mostra 256.

A tabela abaixo relaciona em negrito os 16 nomes padrão, suportados por todos os browsers que exibem cores, e seus respectivos códigos RGB em hexadecimal e decimal.

Cor	Nome	Cód. Decimal	Cód. Hexa	Cor	Nome	Cód. Decimal	Cód. Hexa
	red	255, 0, 0	ff0000		maroon	128, 0, 0	800000
	lime	0, 255, 0	00ff00		green	0, 128, 0	008000
	blue	0, 0, 255	0000ff		navy	0, 0, 128	000080
	yellow	255, 255, 0	ffff00		olive	128, 128, 0	808000
	aqua	0, 255, 255	00ffff		teal	0, 128, 128	008080
	fuchsia	255, 0, 255	ff00ff		purple	128, 0, 128	800080
	white	255, 255, 255	ffffff		silver	192, 192, 192	c0c0c0
	black	0, 0, 0	000000		gray	0, 0, 0	808080

Há muito mais cores com nomes suportadas pelo *Netscape* e *Internet Explorer*. Estas listadas são as únicas que fazem parte da especificação oficial do HTML 4. São todas "seguras", ou seja, fazem parte da paleta básica de 216 cores.

4.5.1. color

Define a cor do texto. A propriedade `color` substitui totalmente o descritor <FONT COLOR> com vantagens. Pode ser aplicada localmente em um descritor (usando o atributo `style`) ou globalmente na página e no site, como qualquer outra propriedade de estilo.

A sintaxe da propriedade `color` é:

```
color: nome_de_cor
color: #número_hexadecimal
color: rgb(vermelho, verde, azul)
```

Exemplos:

```
H1 { color: green }
P { color: #fe0da4 }
EM { color: rgb (255, 127, 63) }
<EM STYLE="color: rgb (100%, 50%, 25%)">
```

Os nomes são qualquer nome válido de cor. Se o browser não encontrar o nome ao qual o estilo se refere, deve exibir a cor *default* (ou herdada). O símbolo "#" é opcional no código

hexadecimal. A intensidade da cor pode ser expressa em valores absolutos (0 a 255) ou percentagens (0.0-100.0%).

4.5.2. background-color

As cores de fundo de qualquer elemento podem ser alteradas através da propriedade `background-color`. A sintaxe é:

```
background-color: transparent        (valor default)
background-color: nome_de_cor
background-color: #número_hexadecimal
background-color: rgb(vermelho, verde, azul)
```

Exemplos:

```
H1 { background-color: green }
P { background-color: #fe0da4 }
EM { background-color: rgb (255, 127, 63) }
<EM STYLE="background-color: rgb (100%, 50%, 25%)">
```

O fundo transparente simplesmente deixa à mostra o fundo do objeto que o contém. O fundo, para texto, ocupa todo o espaço da fonte (inclusive espaço em branco acima e abaixo que fazem parte da fonte). A cor não é estendida quando o espaçamento entre linhas é aumentado em alguns browsers.

4.5.3. background-image

Com `background-image` é possível atribuir a qualquer elemento HTML uma imagem que será exibida no fundo, assim como as cores de fundo. A sintaxe é:

```
background-image: none          (valor default)
background-image: url(URL_da_imagem)
```

Exemplos:

```
H1 { background-image: url(http://www.xyz.com/abc.jpg) }
B {background-image: url(../monstro.gif) navy
<TD STYLE="background-image: url(dinheiro.gif)">...</TD>
```

As URLs são relativas à localização do arquivo que contém a folha de estilos (pode ser a própria página ou não). A cor de *backup* é usada para 'vazar' pelas partes transparentes da imagem ou prevenir contra o não carregamento do fundo (para permitir a leitura em fundo escuro pode-se usar preto como cor de *backup* de uma imagem escura).

4.5.4. background-repeat

CSS permite mais controle ainda sobre a imagem de fundo, facilitando a maneira como a mesma irá se repetir. A propriedade é `background-repeat`. *Sintaxe*:

```
background-repeat: repeat            (default)
background-repeat: repeat-x
background-repeat: repeat-y
background-repeat: no-repeat
```

Exemplos:

```
BODY {background-image: url(china.jpg);
      background-repeat: repeat-x }

TABLE{background-image: url(corinthians.gif)
      background-repeat: no-repeat }
```

O valor `repeat` é *default* e faz com que a imagem ocupe toda a tela. `repeat-x` faz com que a imagem seja repetida apenas horizontalmente e `repeat-y` faz com que ela seja repetida apenas verticalmente. `no-repeat` faz com que a imagem não seja repetida de forma alguma (aparecerá uma imagem apenas no canto superior esquerdo).

Para fazer a imagem aparecer em outros lugares, pode-se usar as propriedades de posicionamento do fundo da tela.

4.5.5. background-position e background-attachment

O posicionamento e a forma de exibição do papel de parede são controlados pelas propriedades `background-attachment` e `background-position`. A primeira define se o fundo irá ou não se mover com o texto ou ficar fixo na tela. A segunda permite o posicionamento do fundo em um local exato da tela. Infelizmente essas duas propriedades não têm suporte universal pelos browsers comerciais (apenas o *Internet Explorer* os suporta).

Sintaxe:

```
background-attachment: fixed
background-attachment: scroll
```

Exemplo:

```
BODY {background-image: url (china.jpg);
      background-attachment: fixed }
```

Sintaxe:

```
background-position: porcentagem_horiz% porcentagem_vert%
background-position: comprimento comprimento
background-position: posição_vertical posição_horizontal
```

Exemplos:

```
BODY {background-image: url(china.jpg);
      background-repeat: no-repeat;
      background-position: 50% 100% }
```

```
BODY {background-image: url(china.jpg);
      background-repeat: no-repeat;
      background-position: 25pt 2.5cm }

BODY {background-image: url(china.jpg);
      background-repeat: no-repeat;
      background-position: center top }

BODY {background-image: url(china.jpg);
      background-repeat: no-repeat;
      background-position: left bottom }
```

Os valores de porcentagem são relativos à posição do elemento sobre o qual se aplica o estilo. As posições são sempre dadas em pares, tendo os valores separados por espaços. O primeiro valor é sempre um valor horizontal e o segundo um valor vertical. O browser coloca o bloco afetado dentro de uma "caixa invisível" e a posiciona de acordo com as porcentagens. Um valor de 100% para o primeiro valor, empurra a margem direita (oposta) desta "caixa invisível" contra a margem direita do browser.

Os valores de comprimento, assim como os de porcentagem também são dados em pares. O primeiro é a distância da margem horizontal a partir do canto superior esquerdo do objeto; o segundo é a distância da margem superior. As unidades válidas são as mesmas usadas em fontes (cm, mm, in, pc, px, pt, em, ex) e podem ser misturadas nos dois valores do par.

Os valores de posição são palavras-chave usadas também aos pares. São equivalentes das porcentagens básicas de alinhamento. O primeiro par pode ter `left` (0%), `right` (100%) ou `center` (50%). O segundo par pode ser `top` (0%), `bottom` (100%) ou `center` (50%).

4.5.6. background

A propriedade `background` pode ser usada para definir várias características de fundo de uma única vez. Na sintaxe abaixo, a ordem dos fatores é importante. A sua sintaxe é:

```
background: background-color background-image background-repeat
            background-attachment background-position
```

Deve haver pelo menos um valor definido, mas qualquer número de valores pode ser atribuído de uma vez.

Exemplos:

```
BODY {background: url(../duke.gif) white no-repeat fixed 50% 25%}
```

4.6. Propriedades de classificação

Estas propriedades classificam os elementos em categorias que podem receber estilos. Categorias podem ser listas, blocos, trechos de blocos ou itens invisíveis.

4.6.1. display

Esta propriedade define *como* um elemento é mostrado. A propriedade `none` desliga o elemento e fecha o espaço que o objeto antes ocupava (torna o objeto invisível). `block` abre uma nova *caixa* onde o objeto é posicionado, relativo aos outros blocos, `list-item` é um bloco com um marcador de lista e `inline` define um elemento como parte de um bloco.

Sintaxe:

```
display: block | inline | list-item | none
```

Exemplo:

```
P {display: list-item}
IMG {display: none}      // desliga todas as imagens
```

4.6.2. white-space

Define como o espaço em branco do elemento é gerenciado (se as linhas devem ser quebradas para que apareçam na tela ou não (`nowrap`) ou se os espaços em branco, tabulações, etc. devem ser considerados (`pre`).

```
white-space: normal | pre | nowrap
```

4.6.3. list-style

Esta propriedade e as propriedades `list-style-type`, `list-style-image` e `list-style-position` definem atributos para objetos de lista, como tipo de marcador, imagem do marcador e posição. Esses elementos não são suportados no *Netscape*.

```
list-style-type: disc | circle | square | decimal | lower-roman
                 upper-roman | lower-alpha | upper-alpha | none

list-style-image: url(url_da_imagem)
list-style-position: inside | outside
```

Exemplo:

```
list-style-image: url(bullet.gif)
```

É possível definir as três propriedades através de um atalho usando `list-type`. A ordem dos fatores é importante neste caso.

```
list-style: list-style-type list-style-image list-style-position
```

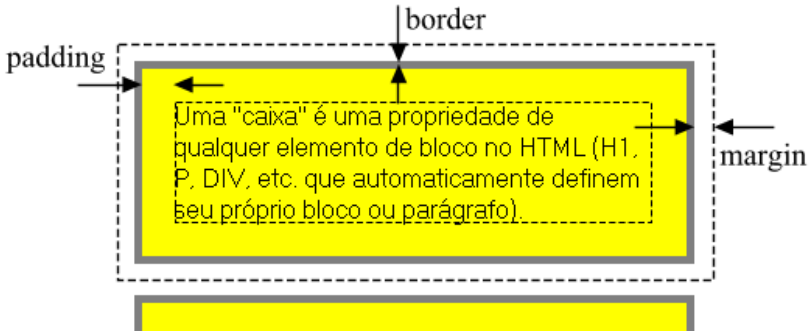
Exemplo:

```
list-style: url(bullet.gif)
list-style: square outside
```

4.7. Controle de blocos

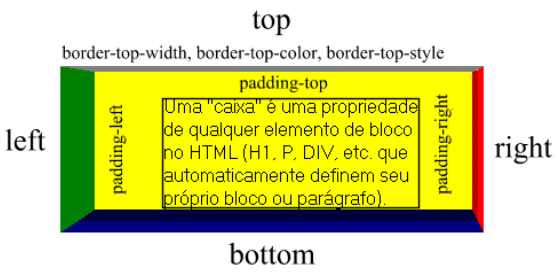
Uma "caixa" é uma propriedade de qualquer elemento de bloco no HTML (H1, P, DIV, etc. que automaticamente definem seu próprio *bloco* ou *parágrafo*). A caixa de um objeto consiste das partes seguintes:

- O elemento em si (texto, imagem)
- As margens internas do elemento (*padding*)
- A borda em torno das margens internas (*border*)
- A margem em torno da borda (*margin*)



Todo elemento de bloco tem essas propriedades. As propriedades CSS que veremos nesta seção mostrarão como alterá-las. A cor e tamanho da borda podem ser alterados assim como o fundo (como vimos na seção anterior). A margem externa é sempre transparente mas a margem interna herda a cor de fundo do objeto.

Também são alteráveis as margens internas e externas, larguras de borda, cor de borda e estilo de borda de cada um dos quatro lados de uma caixa individualmente, identificados pelos nomes *top*, *right*, *bottom* e *left*:



4.7.1. margin e padding

As margens externas são definidas usando a propriedade `margin` (que afeta todas as margens ao mesmo tempo) ou as propriedades `margin-top`, `margin-bottom`, `margin-right` e `margin-left` que permite alterar as margens individualmente.

Sintaxe:

```
margin-top: comprimento | porcentagem % | auto
margin-bottom: comprimento | porcentagem % | auto
margin-right: comprimento | porcentagem % | auto
margin-left: comprimento | porcentagem % | auto
```

Exemplo:

```
margin-top: 1cm; margin-left: 12pt;
```

A propriedade `margin` afeta vários aspectos das margens externas de uma vez só. A ordem dos fatores é importante. Podem ser incluídos todos quatro valores, apenas um (todas as margens iguais) ou dois (margens horizontais e verticais). *Sintaxe:*

```
margin: margin-top   margin-right   margin-bottom   margin-left
margin: margin-top% margin-right% margin-bottom% margin-left%
margin: espaço_vertical   espaço_horizontal
margin: margem_de_todos_os_lados
```

Exemplos:

```
margin: 5cm        // vale para as quatro margens
margin: 5cm 2cm    // 5cm margs verticais, 2cm margs horizontais
margin: 5cm 3cm 2cm 1cm  // sent. horário: top, right, bottom, left
                         // (em cima 5; à direita 3; em baixo 2;...
```

As margens internas (*padding*) são definidas usando a propriedade `padding` (que afeta todas as margens internas ao mesmo tempo) ou as propriedades `padding-top`, `padding-bottom`, `padding-right` e `padding-left`.

Sintaxe:

```
padding-top: comprimento | porcentagem %
padding-bottom: comprimento | porcentagem %
padding-right: comprimento | porcentagem %
padding-left: comprimento | porcentagem %
```

A propriedade `padding` afeta vários aspectos das margens internas de uma vez só. A ordem dos fatores é importante. Podem ser incluídos todos quatro valores ou apenas um. *Sintaxe*:

```
padding: padding-top padding-right padding-bottom padding-left
padding: padding-top% padding-right% padding-bottom% padding-left%
```

```
padding: espaço_vertical   espaço_horizontal
padding: margem_de_todos_os_lados
```

4.7.2. border-width

Pode se controlar vários aspectos das bordas como a sua espessura em cada um dos quatro lados, suas cores (também cada um dos quatro lados) e estilos (idem). Isto pode ser feito de diversas maneiras. Para que as bordas apareçam é preciso primeiro que o estilo (`border-style`) seja definido. Por exemplo:

```
border-style: solid
```

A espessura das bordas pode ser controlada através da propriedade `border-width`, afetando as espessuras de todos os lados da borda, ou individualmente através de `border-top-width`, `border-bottom-width`, `border-right-width` e `border-left-width`. *Sintaxe*:

```
border-top-width: comprimento | thin | medium | thick
border-bottom-width: comprimento | thin | medium | thick
border-right-width: comprimento | thin | medium | thick
border-left-width: comprimento | thin | medium | thick
```

Exemplos:

```
border-bottom-width: thick; border-right-width: 5.5px;
border-left-width: 0.2cm
```

As propriedades das bordas podem ser tratadas em grupo, com `border-width`. A ordem dos fatores é importante. Podem ser incluídos todos os quatro valores, dois (referindo-se às bordas horizontais e verticais) ou apenas um (afetando todas as bordas). *Sintaxe*:

```
border-width: border-top-width    border-right-width
              border-bottom-width border-left-width
```

Exemplos:

```
border-width: 5cm        // vale para as quatro bordas
border-width: 5cm 2cm    // 5cm verticais, 2cm horizontais
border-width: 5cm 3cm 2cm 1cm  // horário: top, right, bottom, left
```

4.7.3. border-color

A propriedade `border-color` é um atalho que permite que se altere a cor de uma ou de todas as quatro bordas ao redor de um elemento que também podem ser definidas individualmente através de `border-top-color`, `border-bottom-color`, `border-right-color` e `border-left-color`.

```
border-top-color: cor (nome ou código)
border-bottom-color: cor (nome ou código)
```

```
border-right-color: cor (nome ou código)
border-left-color: cor (nome ou código)
```

Exemplos:

```
border-bottom-color: rgb(231,45,112); border-right-color: 0fa97b;
border-left-color: navy
```

As propriedades das bordas podem ser tratadas em grupo, com `border-color`. A ordem dos fatores é importante. Podem ser incluídos todos os quatro valores, dois (referindo-se às bordas horizontais e verticais) ou apenas um (afetando todas as bordas). *Sintaxe*:

```
border-color: border-top-color      border-right-color
              border-bottom-color   border-left-color
```

Cada um dos `border-`*xxx*`-color` acima é uma cor (RGB, hexadecimal ou nome). Pode-se alterar todas as bordas de uma vez, apenas as duas verticais e horizontais ou as quatro individualmente.

Exemplos:

```
border-color: red              // red para as quatro bordas
border-color: rgb(255, 0, 0)   // red
border-color: rgb(100%, 0, 0)  // red
border-color: red 0000ff       // red verticais, 0000ff horizontais
border-color: red blue yellow cyan // 4 cores sentido horário
```

4.7.4. border-style

O estilo de cada uma das quatro bordas pode ser alterado com `border-style`. Também é possível defini-los individualmente usando `border-top-style`, `border-bottom-style`, `border-right-style` e `border-left-style`. São vários os estilos disponíveis (tracejado, pontilhado, várias versões de 3D, etc.).

```
border-top-style: none | dotted | dashed | solid | double |
                  groove | ridge | inset | outset
border-bottom-style: nome de estilo (um dos nomes acima)
border-right-style: nome de estilo
border-left-style: nome de estilo
```

Exemplos:

```
border-bottom-style: none border-right-style: solid;
border-left-style: inset
```

As propriedades das bordas podem ser tratadas em grupo, com `border-style`. A ordem dos fatores é importante. Podem ser incluídos todos os quatro valores, dois (referindo-se às bordas horizontais e verticais) ou apenas um (afetando todas as bordas). *Sintaxe*:

```
border-style:     border-top-style     border-right-style
                  border-bottom-style  border-left-style
```

Exemplos:

```
border-style: solid none inset outset;
border-style: solid
border-style: inset outset
```

Cada um dos `border-xxx-style` acima é um dos seguintes valores: `none`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, `outset`. No *Netscape* e *Internet Explorer*, funcionam os estilos `solid`, `inset` (efeito baixo-relevo) e `outset` (efeito alto-relevo). No *Netscape* `solid` é *default* mas no *Explorer*, o *default* é `none`, portanto, uma borda não aparece se a propriedade `border-style` não for definida antes.

4.7.5. border

As propriedades `border-top`, `border-bottom`, `border-left` e `border-right` agrupam as propriedades de cor, estilo e espessura para cada uma das quatro bordas.

```
border-top:    border-width  border-style  border-color
border-bottom: border-width  border-style  border-color
border-left:   border-width  border-style  border-color
border-right:  border-width  border-style  border-color
```

A propriedade `border` é uma forma reduzida de definir tudo isto de uma vez só para todas as bordas e de forma idêntica (não é possível especificar valores diferentes para as bordas neste caso). Todos os itens não precisam aparecer, mas a ordem dos fatores é importante:

```
border:        border-width  border-style  border-color
```

4.7.6. width e height

As propriedades `width` e `height` modificam a altura e largura de um bloco da mesma forma que atributos HTML `width` e `height` usados em imagens e applets no HTML. Com folhas de estilos podem ser usados para redimensionar a "caixa" de qualquer elemento de bloco. *Sintaxe*:

```
width: comprimento | auto
height: comprimento | auto
```

4.7.7. float

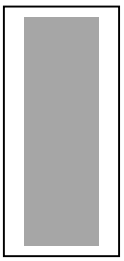
A propriedade `float` permite que um bloco qualquer seja posicionado à direita ou esquerda da janela do browser, fazendo com que o texto restante flua em sua volta. Permite que qualquer bloco se comporte como uma imagem que faz uso dos atributos `align=left` e `align=right` no HTML. *Sintaxe*:

```
float: left | right | none
```

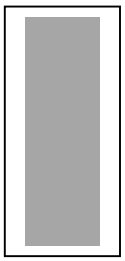
4.7.8. clear

E finalmente, para evitar que um bloco flua em torno de uma imagem ou bloco que utiliza a propriedade `float`, existe a propriedade `clear`, que deve ser atribuída a um bloco ou parágrafo que é afetado pela flutuação de um bloco. Faz a mesma coisa que o atributo `clear`, usado no `<BR>` em HTML, só que aqui ela é suportada em qualquer elemento (bloco ou não).

```
clear: none | left | right | both
```



Primeiro parágrafo flui ao lado da imagem. Outro parágrafo que não devia fluir.



Primeiro parágrafo flui ao lado da imagem.

Outro parágrafo que não devia fluir (com clear).

4.8. Posicionamento

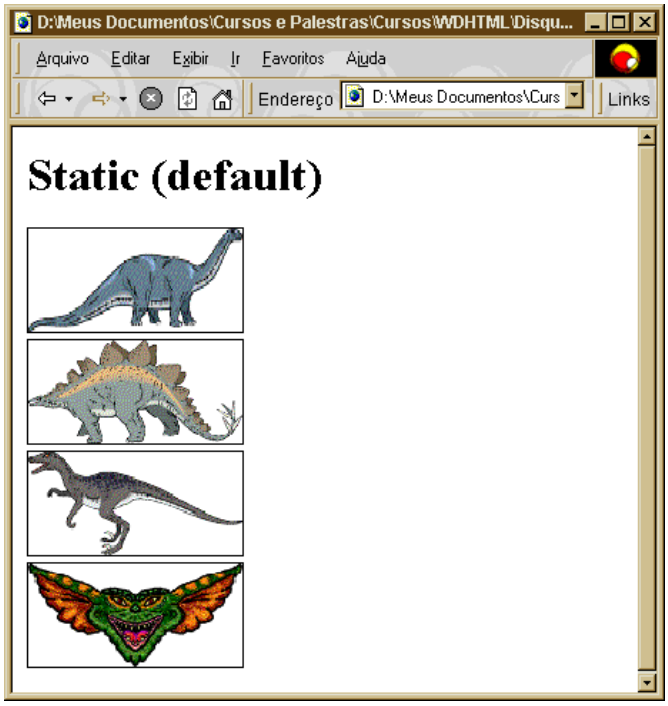
O posicionamento de objetos em HTML pode ser conseguido em termos absolutos ou relativos usando CSS 2. As propriedades permitem o posicionamento em três dimensões (horizontal, vertical e em camadas). Embora os recursos de posicionamento não façam parte do CSS1, tanto o *Netscape Navigator 4* como o *Internet Explorer 4* o suportam.

4.8.1. position, top e left

Esta propriedade precisa de mais duas propriedades que definem o posicionamento de um objeto. Os atributos localizam o objeto na tela de forma *absoluta* ou de forma *relativa*. A origem da posição absoluta será a posição do objeto no nível imediatamente superior. O posicionamento relativo se refere à posição anterior do objeto. *Sintaxe*:

```
position: absolute | relative
left: comprimento | porcentagem | auto
top: comprimento | porcentagem | auto
```

Exemplo: considere as quatro imagens a seguir:

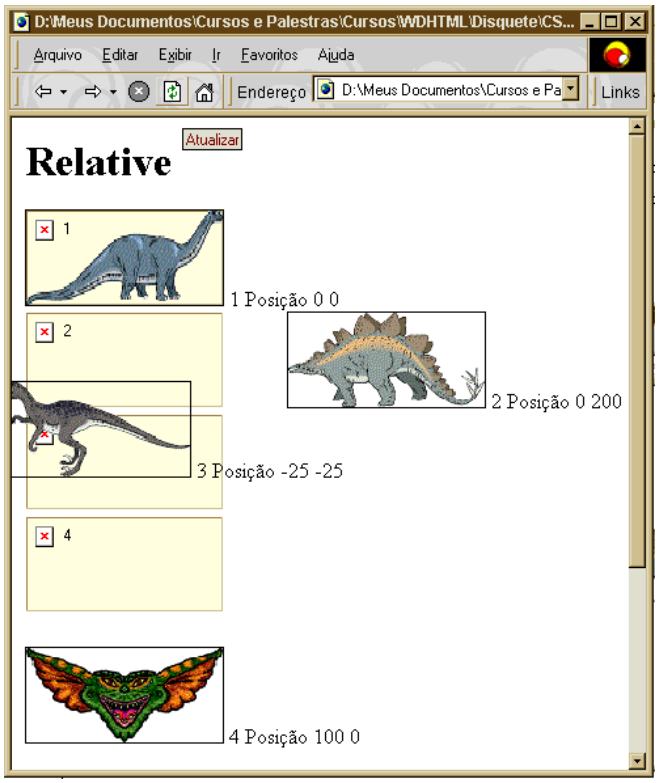



```
<div class="imagem1">
<img src="T1.gif" width=150 height=72 border=1> 1 Posição 0 0
</div>

<div class="imagem2">
<img src="T2.gif" width=150 height=72 border=1> 2 Posição 0 200
</div>

<div class="imagem3">
<img src="T3.gif" width=150 height=72 border=1> 3 Posição -25 -25
</div>

<div class="imagem4">
<img src="T4.gif" width=150 height=72 border=1> 4 Posição 100 0
</div>
```

Aplicando a seguinte folha de estilos para posicionar as quatro imagens relativamente à sua localização original obtemos a imagem ao lado. Os retângulos claros indicam a posição original da imagem:

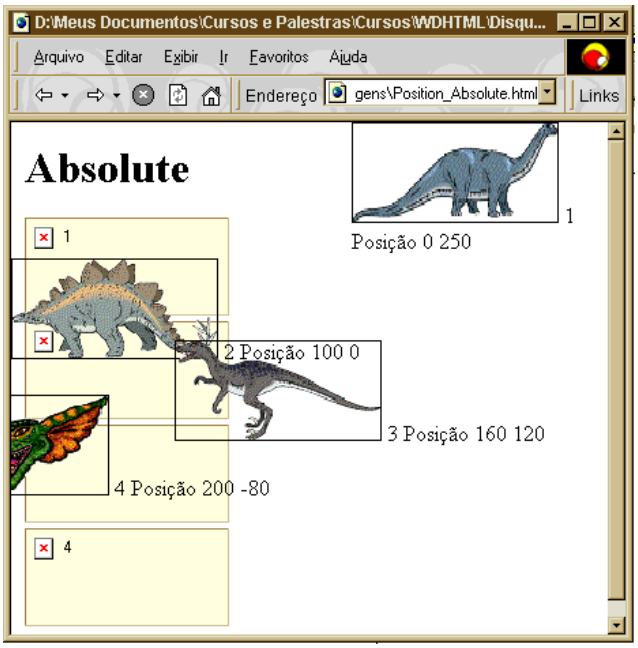
```
DIV.imagem1 {
   position: relative;
   top: 0px;
   left: 0px;
 }
```

```
DIV.imagem2 {
  position: relative;
  top: 0px;
  left: 200px;
}
DIV.imagem3 {
  position: relative;
  top: -25px;
  left: -25px;
}
DIV.imagem4 {
  position: relative;
  top: 100px;
  left: 0px;
}
```



Usando posicionamento absoluto, o bloco é movido em relação ao canto superior esquerdo do browser.

```
DIV.imagem1 {
   position: absolute;
   top: 0px;
   left: 250px;
 }
 DIV.imagem2 {
   position: absolute;
   top: 100px;
   left: 0px;
 }
 DIV.imagem3 {
   position: absolute;
   top: 160px;
   left: 120px;
 }
 DIV.imagem4 {
   position: absolute;
   top: 200px;
   left: -80px;
 }
```



4.8.2. *z-index*

A propriedade `z-index` permite ordenar os objetos em camadas. É um eixo de profundidade. Para usá-la, basta definir em cada objeto:

```
z-index: número
```

onde número é a camada de exibição. Quanto maior o número, mais alta a camada. o corresponde à camada do texto. Se um objeto tiver `z-index` menor que zero aparecerá atrás do texto. Se `z-index` for maior que zero, aparecerá na frente. Quando não é definido ou quando `z-index`: 0 o bloco ocupará a mesma camada que o texto.

4.8.3. *visibility*

Esta propriedade serve para tornar um bloco visível ou invisível. Difere de `display` porque não remove o espaço antes ocupado pela imagem:

```
visibility: hidden | visible
```

Exemplo:

```
IMG {visibility: hidden}       // torna invisíveis todas as imagens
```

4.9. Exercícios

4.9.1. Testes sobre Folhas de Estilo

1. Qual das seguintes regras de estilo está incorreta? Marque uma.
   a) `a:link {color: rgb(0%,40%,40%)}`
   b) `div.code pre {margin-bottom: 0px}`
   c) `body {font-size: 0.5cm, color: yellow, background: black}`
   d) `.botcor {font-size: 16pt; font-family: tahoma, sans-serif;}`
   e) Estão todas corretas.

2. Qual dos seguintes trechos de código é ilegal dentro de um arquivo `.css`? Marque uma.
   a) `span.value {color: maroon}`
   b) `/* <H1>Titulo</H1> */`
   c) `@import url(http://ww.estilos.org/estilo.css);`
   d) `<STYLE>`
   e) Nenhuma das alternativas é ilegal dentro de um arquivo CSS.

3. Qual das regras abaixo, de uma folha de estilos, declara que os parágrafos e células de dados de tabelas terão texto vermelho?
   a) `P TD {color: red}`
   b) `P: TD {color: ff0000}`
   c) `P, TD {color: rgb(100%, 0%, 0%)}`
   d) `P; TD {color: rgb(255, 0, 0)}`
   e) `P, TD {color=red}`

4. Qual das declarações abaixo, contida em uma página HTML, a vincula à folha de estilos `basico.css`, localizada no mesmo diretório que a página?
   a) `<LINK REL=StyleSheet HREF="basico.css">`
   b) `<LINK REL=StyleSheet SRC="basico.css">`
   c) `<A HREF="basico.css" TITLE="StyleSheet">Folha de estilos</A>`
   d) `<FRAME SRC="basico.css" REL="StyleSheet">`
   e) `<A HREF="basico.css">Folha de estilos</A>`

5. Considere o seguinte trecho de código HTML:

```
<div>
<p>Parágrafo</p>
</div>
```

   Quais declarações abaixo, em um bloco `<STYLE>` do arquivo que contém o trecho acima, farão com que o texto do parágrafo tenha tamanho 10pt em um browser que suporte folhas de estilo? Marque uma ou mais.

   a)
```
div {font-size: 20pt}
p {font-size: 50%}
```
   b) `div {font-size: 10pt}`

c) `p {font-size: 10pt}`
d) div `{font-size: 5pt}`
   `p {font-size: 100%}`
e) `p div {font-size: 10pt}`

6. Considere a seguinte folha de estilos, com uma única regra, vinculada a uma página HTML.

```
P {color: green}
```

Dentro dessa página, logo depois da instrução que vincula o estilo à página, há um bloco `<STYLE>`, com a seguinte regra:

```
P {color: red}
```

A página possui dez parágrafos. Um deles atribui um estilo local usando o atributo `STYLE`, da forma:

```
<P STYLE="color: blue">Parágrafo</P>
```

Supondo que a página seja visualizada em um browser que suporte folhas de estilo CSS, qual é a cor da maior parte dos parágrafos dessa página?

a) azul (`blue`)
b) vermelha (`red`)
c) verde (`green`)
d) preta (`black`)
e) indefinida

7. Identifique as alternativas que contém HTML ou CSS incorretos:
   a) `<span style="color: red; font-size: 24pt">Texto</span>`
   b) `<span color=red font-size="24pt">Texto</span>`
   c) `<div class="sec1">Tem <span class=item1>mais texto</span>.</div>`
   d) `<span>Itens e <div class="sec1">seções</div> especiais.</span>`
   e) `<div style="P {color: yellow}"><P>Texto amarelo</P></div>`

8. Considere o código HTML abaixo:

```
<div class=sec2>
  <p class=novo>Texto modificado</p>
</div>
```

Quais das regras de estilo abaixo fará com que o parágrafo seja exibido na cor azul, em um browser que suporte folhas de estilos CSS?

a) `P {color: blue}`
b) `.sec2 {color: blue}`
c) `P.novo {color: blue}`
d) `.sec2 .novo {color: blue}`
e) `P.sec2 {color: blue}`

9. Qual das regras abaixo retira o sublinhado apenas dos *links* visitados? Marque uma.
   a) `a: visited {text-decoration: none}`
   b) `a, visited {text-decoration: none}`

c) `a.visited {text-decoration: none}`
d) `a visited {text-decoration: none}`
e) Nenhuma das regras anteriores.

10. Marque apenas as alternativas verdadeiras
    a) Uma mesma folha de estilos pode ser usada por várias páginas.
    b) Uma mesma página pode usar várias folhas de estilo.
    c) Se um browser não suportar uma folha de estilos requerida pela página, poderá haver uma degradação na qualidade da apresentação mas nunca haverá perda de informação.
    d) É possível construir um site inteiro usando apenas CSS.
    e) A linguagem CSS usada para construir folhas de estilo é uma recomendação do W3C – consórcio de empresas que estabelece os padrões para a Web.

4.9.2. Exercícios com Folhas de Estilo

Os exercícios a seguir têm a finalidade de explorar as principais propriedades do CSS e permitir que se verifique o suporte a elas nos browsers populares. Eles são mais didáticos do que úteis. O objetivo é apenas praticar com folhas de estilos. Para realizá-los, use os arquivos disponíveis no CD do ASIT.

Conceitos básicos

1. Crie uma folha de estilos, chame-a de basico.css, e a carregue no arquivo StyleTest.html.

2. Nesta folha de estilos, usando o mínimo de declarações possível, declare:
    a) que todo H1 tenha fonte Tahoma, ou sans-serif, se Tahoma não existir
    b) que todo o texto do corpo (BODY) do arquivo tenha tamanho 10 pontos
    c) que todos os H1, H2 e H3 sejam vermelhos
    d) que os H1 tenham tamanho 24 pontos
    e) que os H2 tenham tamanho 18 pontos
    f) que os H3 tenham tamanho 14 pontos

3. Mude a cor do fundo da página para azul marinho (navy) e a cor *default* do texto para branco em uma única declaração.

4. Faça com que todo texto marcado em itálico apareça em azul ciano (cyan).

Formas de usar CSS

5. Carregue a folha de estilos basico.css em outros arquivos HTML e veja o que acontece. Faça com que uma dessas outras páginas tenha uma cor de fundo clara (amarela, por exemplo) e cor de texto escuro (diferente daquela definida por basico.css) sem que isto altere as outras paginas que usam o mesmo arquivo.

6. Faça com que o primeiro parágrafo do arquivo StyleTest.html tenha texto verde.

7. Faça com que a célula do meio da tabela de StyleTest.html tenha texto vermelho sobre fundo amarelo (a tabela 3x3 encontra-se no meio da página).

Classes, links, seletores de contexto

Para os exercícios abaixo, desligue a folha de estilos usada nos exercícios anteriores (mude o nome ou remova o elemento <LINK>) para que a página fique limpa outra vez. Use uma nova folha de estilos para aplicar as alterações a seguir.

8. Defina classes sec2, sec3, sec31 e sec32 para as seções (<DIV>) do documento StyleTest.html. As seções estão indicadas em comentários HTML (por exemplo: <!--Seção 2 -->). Aplique um fundo diferente (imagem ou cor) nessas seções para diferenciá-las das outras.

9. Tire os sublinhados de todos os links e substitua-os por um fundo cinza claro.

10. Faça com que o link ativo (active) fique em negrito; que o link normal tenha tamanho 10pt e que mostre fundo amarelo quando o mouse estiver sobre ele (hover); e que o link visitado não tenha mais cor de fundo mas recupere o sublinhado. Obs: Para fazer um link ainda não visitado, faça um link para qualquer recurso do sistema de arquivos; para ver o link ativo, clique no link e segure o mouse.

11. Faça com que:
    a) todos os itálicos dentro de negritos sejam colocados em maiúsculas (use text-transform: uppercase).
    b) todos os negritos dentro de itálicos sejam sublinhados
    c) todos os negritos que estejam dentro de um bloco itálico que está dentro de um bloco LI que está dentro de uma UL que está em outra UL, sejam colocados em fonte arial, em maiúsculas e em vermelho.

Fontes

Crie uma nova folha de estilos (fontes.css) para aplicar fontes. Vincule (LINK) ou importe-a (@import) em seus arquivos.

12. a) Aplique Verdana como fonte *default* em todo o site. Garanta que, se Verdana não existir, Arial será usada, e se esta não existir, será usada a *default* sans-serif. Para testar, mude os nomes das primeiras fontes para nomes desconhecidos do sistema. b) Teste a compatibilidade dos dois browsers em relação ao suporte de fontes com nomes longos (entre aspas) em folhas de estilo locais e remotas.

13. Faça com que os <B> de seus parágrafos sejam 20% maiores que o texto normal destes parágrafos.

Atributos de texto e classificação

Crie uma nova folha de estilos para esses exercícios.

14. a) Aplique um espaçamento de 1 cm entre palavras de um parágrafo seu texto (isto poderá não funcionar devido à falta de suporte dos browsers). b) Aplique um espaçamento de 1 cm entre as letras de outro parágrafo. Teste nos dois browsers.

15. Defina todos os títulos H2 como sendo caixa alta (uppercase).

16. Experimente com as propriedades text-decoration (use overline e outras propriedades em blocos <SPAN> criados para mostrar cada propriedade.

17. Elimine o espaçamento entre os parágrafos (<P>) usando {margin-top: 0pt}, endente a primeira linha e coloque todos os seus parágrafos, com exceção dos parágrafos da terceira seção, com alinhamento justificado. O alinhamento deve ser aplicado apenas nos parágrafos e não em listas ou tabelas.

Cores

18. Experimente com cores, aplicando cores em textos, backgrounds de diversos componentes da página, inclusive formulários (<INPUT> e <SELECT>). Use as três formas (url(r, g, b), rrggbb e nomes) e veja como ocorre o suporte dos browsers em folhas de estilo locais e externas. Dica: crie uma folha de estilos só para este exercício.

Fundos, Imagens e Repetições

19. Inclua a imagem rabbit.gif (ou outra qualquer do subdirectório 3_Imagens do CD do A-SIT) no fundo da página StyleTest.html (usando uma nova folha de estilos: background.css). Experimente com posicionamento, fazendo a imagem ficar fixa em vez de rolar na tela. Teste nos dois browsers. Experimente com repetições, fazendo a imagem repetir na vertical, na horizontal e não repetir. Veja o que acontece nos dois browsers.

20. Numa outra folha de estilos (para este exercício), posicione a imagem no centro da página, sem repetições e uma outra imagem no centro da tabela, também sem repetições.

21. Posicione (outra folha de estilos) o rabbit.gif (ou outra imagem) em uma posição a 4cm da margem esquerda e a 7cm da margem superior. Na seção 2 (sec2), posicione bart.gif, repetindo somente na vertical, afastado 11cm da margem esquerda e iniciando 1 cm abaixo da margem superior.

Posicionamento e Layout

22. Remova o espaço entre todos os parágrafos de StyleTest.html. Aplique um text-indent de 1cm em cada parágrafo.

23. Faça com que os blocos (parágrafos e cabeçalhos) da seção 3.1 e 3.2 (DIV) tenham 0,3 cm de margem esquerda e direita, e 0,5cm de margem superior e inferior, em relação às bordas da seção.

24. Faça com que as seções 3.2 e 3.1 tenham uma margem externa de 0,5 cm em relação à seção 3.

25. Aplique uma borda azul, sólida, de 3mm acima da seção 2 (<DIV>), uma outra, também de 3mm, abaixo, na cor amarelo. Dos lados, coloque bordas vermelhas, de 5mm.

26. Aplique uma borda verde, de 4mm à esquerda de todos os parágrafos da seção 2. Entre a borda e o texto deve haver um espaço de 5mm. Entre a borda e a margem da página, mais 5mm (sem levar em conta o offset).

27. Sem usar tabelas, aplique uma largura máxima de 500 pixels em toda a página.

28. Faça com que a seção 3.1 tenha largura máxima de 220 pixels e flutue para a direita, deixando o restante do texto fluir em volta dela.

Posicionamento absoluto

29. Monte o quebra-cabeças do exercício 1, do livro 3 (Além do HTML) sem usar tabelas (usando apenas posicionamento de folhas de estilo).

30. Diagrame a página do exercício 2, do livro 3, sem usar tabelas (usando apenas posicionamento de folhas de estilo).