Fundamentos de Desenvolvimento Web

Aula 1 – Como a internet funciona?

Objetivos

Contextualizar o surgimento da internet.

Apresentar os principais conceitos envolvidos no funcionamento da internet.

Conhecer a finalidade e o funcionamento da internet.

Como tudo começou

Antes de falarmos sobre *web* ou sobre desenvolvimento para *web*, precisamos falar de internet. E, embora muitos utilizem os dois termos (*web* e internet) de forma indiscriminada, eles não são sinônimos. Quando falamos de internet, estamos nos referindo a uma grande rede de dispositivos computadorizados de alcance mundial, podemos entendê-la como uma grande infraestrutura em rede. A *web* (uma derivação abreviada para a expressão *World Wide Web*) é apenas uma das funcionalidades da internet – no caso específico, navegar através do **hipertexto**.

A-Z

hipertexto
O termo hipertexto (*hipertext*) foi criado por Theodore Nelson, na década de 60, para denominar a forma de escrita/leitura não linear. Pode ser entendido como uma espécie de texto onde alguns trechos se intercalam com referências a outros textos.

Antes da internet se tornar o que conhecemos hoje, houve um grande percurso na evolução dos computadores e das tecnologias de telecomunicações. Assim como muitas das descobertas da humanidade, a internet também teve forte motivação militar. Durante o período pós-guerra (anos 60 do século XX), especialmente na Guerra Fria (EUA × Rússia), havia um grande temor em relação a possíveis ataques nucleares. Pesquisas buscavam desenvolver uma cadeia de comunicações onde não existisse um ponto central que, ao ser destruído, colocaria em colapso todo o sistema de comunicações (COMER, 2007). Em meados de 1962, os Estados Unidos criaram a Cadeia de Comunicação Distribuída (CCD), que era composta por vários computadores interligados por várias linhas telefônicas diferentes. A partir de tal estrutura, objetivava-se dividir o volume de dados a ser trafegado entre os computadores em pequenos "pacotes", despachando-os por meio das diferentes linhas telefônicas até um computador de destino. Observe que, neste modelo, na eventual falha de um dos pacotes por meio de um dos caminhos, o sistema poderia utilizar um

caminho equivalente, ou seja, não há um ponto único de falha. Uma eventual interrupção em alguma linha de transmissão não interrompe completamente o sistema (COMER, 2007).

Na Figura 1.1, pode-se observar o funcionamento da internet através de uma estrutura de interconexão física. Suponha que um computador da casa A queira enviar uma mensagem para o computador destino F. O caminho natural entre A e F está bloqueado (indisponível). No entanto, a casa A pode enviar sua mensagem para casa D que, por sua vez, envia para a casa E que, finalmente, entregaria para o destino F. Outra opção também seria realizar o caminho A – B – C – E – F, e ainda haveria outras opções.

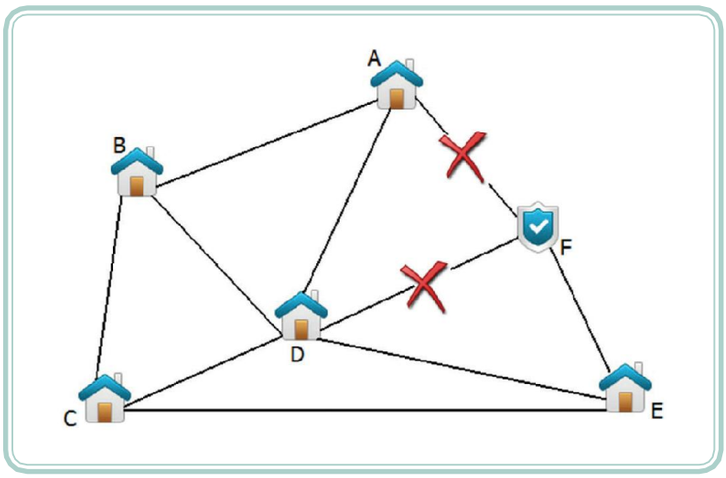


Figura 1.1: Exemplo de comunicação distribuída
Fonte: Autores

Em 1966, o Departamento de Defesa dos EUA, por meio da ARPANET (*Advanced Research Projects Agency* – Agência de Projetos e Pesquisas Avançadas), instalou, em 17 locais diferentes, computadores conectados às linhas telefônicas que, a partir de 1969, tornaram-se uma rede de computadores apenas para uso militar. A partir de então, as pesquisas desenvolvidas até aquele momento, que deram origem a cadeia de comunicação distribuída, passaram a ser chamadas de ARPANET (MANZANO; TOLEDO, 2008).

Nos anos seguintes, algumas agências do governo e universidades subordinadas ao Departamento de Defesa dos EUA começaram a fazer uso restrito da ARPANET com fins de pesquisa. Naquele período, algumas universidades e empresas de grande porte, inspiradas nas ideias da ARPANET, começaram a criar suas próprias soluções para interligar suas redes de computadores.

A internet, como a conhecemos hoje, é fruto de constantes otimizações e de novas tecnologias que se incorporaram às ideias iniciais da ARPANET. Merece destaque, nesse cenário, o desenvolvimento do **protocolo** TCP/IP (*Transmission Control Protocol/Internet Protocol*) adotado pela ARPANET em 1982 e que, posteriormente, foi liberado para utilização civil e, até hoje, tem se mostrado uma das melhores alternativas para comunicação entre computadores. Com a adoção de um protocolo único e padronizado, tornou-se viável conectar computadores de diferentes fabricantes em redes com diferentes meios de distribuição, potencializando ainda mais a utilização da internet.

A-Z

protocolo
Um protocolo de rede é um conjunto de regras que definem a forma como dois sistemas se comunicam. É uma espécie de língua falada entre os dispositivos. Se ambos "falam" o mesmo protocolo então a comunicação pode ser estabelecida.

Princípios de funcionamento

Como vimos anteriormente, a internet é uma rede de computadores de acesso público e ilimitado (sem um "dono") que utiliza a infraestrutura de telecomunicações. Embora não exista um dono, existem consórcios internacionais, como o W3C (*World Wide Web Consortium*), com a tarefa de agregar empresas filiadas na tentativa de, em conjunto, desenvolver padrões para a internet.

O W3C tem como missão conduzir a *World Wide Web* para que atinja todo seu potencial, desenvolvendo protocolos e diretrizes que garantam seu crescimento de longo prazo. Para saber mais sobre a W3C, acesse: ou o escritório no Brasil disponível em:

O acesso à internet se dá, normalmente, por meio de um ISP (*Internet Service Provider* ou Provedor de Serviço de Internet) e utiliza-se de, pelo menos, três componentes (CPE, Rede de Acesso e POP) ilustrados na Figura 1.2.

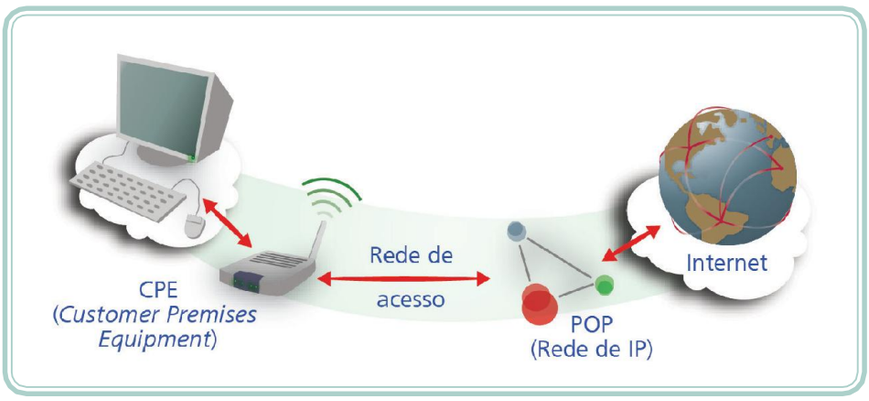


Figura 1.2: Acesso à internet
Fonte: CTISM, adaptado dos autores

A-Z

CPE
A sigla CPE – cuja tradução seria algo como "equipamento dentro das instalações do cliente" é um termo técnico genérico utilizado por fornecedores de serviços de comunicação. Sua definição está atrelada ao contexto em que é utilizada. Por exemplo, para uma empresa de telefonia, o CPE pode ser o aparelho de telefone (no caso dos serviços de voz) ou o *modem* (para serviços de dados). Já no caso de uma operadora de telefonia móvel, o CPE é o telefone celular. Qualquer equipamento que seja necessário para um cliente receber o serviço de comunicação é um CPE (roteadores, *cable modem*, receptor de ondas de rádio, etc.).

- **CPE** (*Customer Premises Equipment*) é o equipamento que conecta o dispositivo à rede de acesso (exemplo: modem).

- Rede de acesso é o tipo de infraestrutura que liga o dispositivo ao provedor de internet (exemplos: cabos de cobre, fibra ótica, Wi-Fi).

- POP (*Point of Presence*) é o ponto de presença do provedor onde estão os equipamentos que atribuem ao dispositivo um endereço IP, dando-lhe acesso à internet.

Como pôde ser observado, para que um computador possa se conectar à internet e se comunicar com outros computadores, faz-se necessário que este receba um número de identificação – esse número é conhecido como endereço IP (*Internet Protocol*). Para entender a importância de um endereço IP, podemos fazer uma analogia com o sistema de telefonia: para que duas pessoas conversem entre si, ambas precisam de um número telefônico (origem e destino). O mesmo se aplica aos computadores conectados à internet: cada um, no momento em que se conecta a um provedor de acesso, recebe um número (número IP) a partir do qual pode realizar "chamada" (conexões) com outros computadores que também estejam conectados à internet.

No Brasil, a entidade responsável pela coordenação da atribuição de endereços de internet é o Comitê Gestor da Internet no Brasil – cgi.br. Suas outras atribuições são estabelecer diretrizes estratégicas relacionadas ao uso e desenvolvimento da internet no Brasil e coletar, organizar e disseminar informações sobre os serviços de internet, incluindo indicadores e estatísticas.

Cabe ressaltar que qualquer dispositivo conectado à internet, independente de ser um computador, um celular/*smartphone*, uma impressora ou mesmo uma geladeira, deverá possuir um endereço IP. Normalmente, os provedores de serviços de internet adquirem/locam, de agências reguladoras, faixas de endereços IP que são atribuídas a seus usuários quando conectados à internet. Um usuário doméstico de internet, cada vez que se conecta, pode receber um endereço IP diferente. No entanto, usuários coorporativos (como empresas) podem adquirir endereços IP fixos (permanecendo os mesmos a cada conexão).

Um endereço de internet, IP na versão 4, é um número escrito em quatro partes (octetos), cada uma variando de 0 a 255 – por exemplo: 200.132.39.115. É importante ressaltar que um endereço IP não identifica, necessariamente, um equipamento individual, mas sim uma conexão. Podemos encontrar equipamentos (*gateways*) conectados a várias redes que possuem mais de um endereço IP (um para cada conexão).

Então, para que um recurso seja acessado na internet, precisamos conhecer um número IP que nos leve até ele. Para facilitar a memorização dos endereços, foi implementado um sistema de nomes de domínio – DNS (*Domain Name System*) – através do qual é possível traduzir um endereço, como **www.mec.gov.br**, em um endereço que nos remeta até a rede desejada – nesse caso, a do MEC (Ministério da Educação). O DNS é um sistema hierárquico que passou a ser utilizado em 1984, fundamentado em uma base de dados distribuída hierarquicamente na qual os equipamentos realizam consultas para descobrir o endereço IP dos computadores que precisam se conectar (COMER, 2007).

O endereço de um recurso, na internet, é, normalmente, dividido por seções separadas por "." (ponto). As seções mais à direita, comumente, identificam um nome de um domínio dentro do sistema de DNS. A Figura 1.3 ilustra a divisão do endereço **www.mec.gov.br**: o termo "br" remete a pesquisa para a base de nomes de domínio hospedados no "Brasil"; o termo "gov" indica que endereço é categorizado como "governamental"; o termo "mec" remete a conexão para a rede do Ministério da Educação, no qual existe um computador de nome "www" que responderá pela requisição. A partir de pesquisas nas bases DNS, o endereço **www.mec.gov.br** será convertido em um endereço IP.

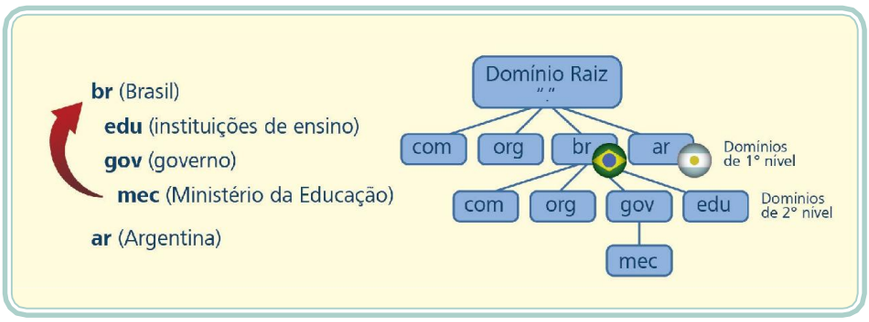


Figura 1.3: Sistema de nomes de domínio
Fonte: CTISM, adaptado dos autores

Agora que entendemos como o endereço de um recurso na internet é convertido em um endereço IP, podemos compreender, também, o significado do termo URL – *Uniform Resource Locator* ou localizador-padrão de recursos. URL é o termo correto que devemos utilizar quando nos referimos a um endereço na internet. Ela nos remete a um destino único (uma página, um vídeo, um documento, um serviço, etc.). A partir de uma URL, temos todas as informações necessárias para encontrar e para acessar uma informação na internet (protocolo utilizado, endereço do dispositivo, caminho para o recurso e o recurso propriamente dito). A Figura 1.4 ilustra as partes de uma URL hipotética:

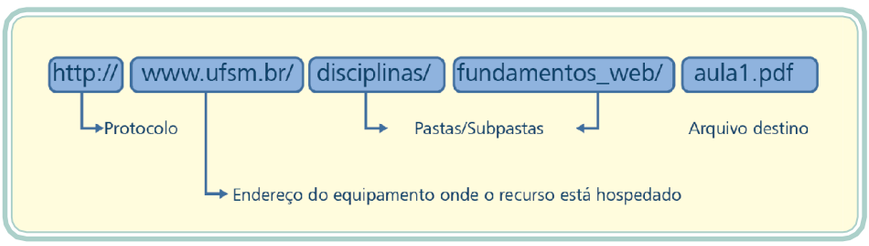


Figura 1.4: URL
Fonte: CTISM, adaptado dos autores

Com o aumento exponencial de dispositivos conectados a internet, o número de combinações possíveis de um endereço IP em sua versão 4 está praticamente esgotado. Tal situação motivou o desenvolvimento de uma "nova geração" de endereços IP, conhecida como IPv6, cuja representação se dá utilizando-se de oito grupos com 4 dígitos hexadecimais (exemplo: 2001:0db8:85a3:08d3:1319:8a2e:0370:7344), aumentando, substantivamente, a possibilidade de combinações e, consequentemente, a quantidade de dispositivos que podem estar conectados à internet.

Para saber mais sobre a lista de nomes de domínio de primeiro nível (DPNS), (*site* responsável pelo registro de nomes de domínio para materiais hospedados no Brasil).

O que se pode fazer com a internet?

Muitos estudiosos consideram a internet uma das maiores revoluções pela qual a humanidade passou e vem passando em um curto espaço de tempo. Entre tantas opções que temos ao utilizar a internet, podemos classificar alguns serviços clássicos que, de uma forma ou de outra, serviram de base ou fundamento para o surgimento de outras formas de aplicação.

Em geral, as aplicações que funcionam sobre a internet obedecem a um mecanismo conhecido como cliente/servidor (*client/server*), por meio do qual duas aplicações conversam entre si através de um protocolo predefinido. O modelo de aplicação cliente/servidor prevê que, em um dispositivo, exista uma aplicação (denominada de aplicação servidora ou *server application*) responsável por aceitar requisições de aplicações clientes. As requisições enviadas pelas aplicações clientes são como solicitações ou pedidos que demandam alguma ação por parte das aplicações servidoras.

Geralmente, as aplicações clientes preocupam-se mais com requisitos de apresentação das informações e coleta de dados de entrada, enquanto as aplicações servidoras preocupam-se com o desempenho, a disponibilidade e a segurança. Nesse modelo, um cliente não compartilha de seus recursos, apenas solicita o conteúdo do servidor. As seções são iniciadas pelos clientes. Os servidores, por sua vez, esperam as solicitações de entrada.

A seguir, abordaremos alguns aspectos de serviços clássicos que podemos encontrar na internet e que, de certa forma, influenciaram outros tantos surgidos posteriormente, estendendo, assim, suas funcionalidades.

1.3.1 Correio eletrônico

O correio eletrônico – também conhecido como *e-mail* – é um serviço através do qual podemos explorar a comunicação de forma *off-line*, ou seja, sem que ambos os interessados estejam conectados. Podemos fazer uma analogia ao correio tradicional, no qual as correspondências enviadas por um remetente a um destinatário somente serão lidas se este último se dirigir até sua agência de correio para retirar o material remetido (considerando, nesse exemplo, a inexistência do carteiro).

Por meio do serviço de correio eletrônico, uma aplicação (cliente de *e-mail*) oferece ao utilizador alguns campos para preenchimento (destinatários, assunto, texto da mensagem, etc.) que irão compor uma mensagem. Após a submissão da mensagem a uma aplicação servidora (servidor de *e-mail*), esta se encar-

rega de encaminhá-la às caixas de correio de cada um dos destinatários (que ficam armazenadas em aplicações servidoras). O usuário que deseja verificar se existem novas mensagens, realiza uma requisição por meio de seu cliente de *e-mail*, o qual consulta, no respectivo servidor, a existência ou não de mensagens. Se existirem, as mesmas são apresentadas ao destinatário na forma como foram concebidas.

Um endereço de *e-mail* é composto, basicamente, por duas partes que são separadas pelo sinal de "@" (que em inglês é lido como *at* – em). A Figura 1.5 ilustra um endereço de *e-mail*, evidenciando o nome do usuário (nome da caixa postal eletrônica) e o endereço da aplicação servidora responsável pelo recebimento e pelo envio de mensagens eletrônicas.

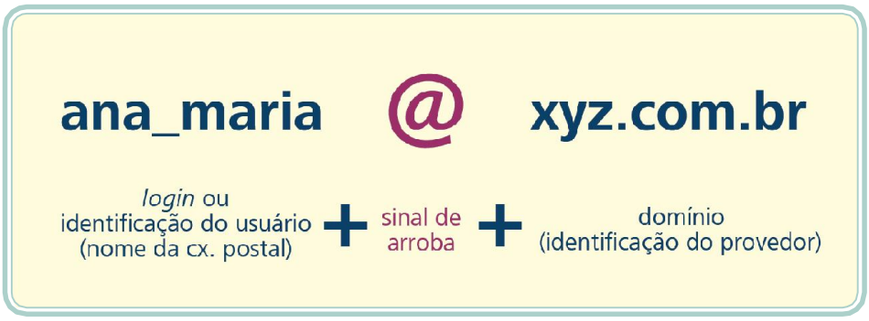


Figura 1.5: Anatomia de um endereço de *e-mail*
Fonte: Autores

1.3.2 Transferência de arquivos

O serviço de transferência de arquivos é baseado em um protocolo específico denominado de FTP (*File Transfer Protocol*). Por meio desse serviço, uma aplicação cliente pode realizar duas operações básicas:

- ***Download*** – quando um arquivo originalmente localizado no equipamento servidor é copiado para o equipamento do cliente. O termo "baixar um arquivo" está diretamente associado à operação de *download*.

- ***Upload*** – quando um arquivo que está localizado no equipamento do usuário é submetido para a aplicação servidora de forma que uma cópia do mesmo seja realizada. Nesse caso, a expressão equivalente seria de "subir um arquivo", embora não seja muito utilizada.

Um cliente de FTP é uma aplicação que oferece uma interface, normalmente, composta por duas visualizações: na primeira, é apresentada a estrutura de pastas do computador do usuário e, na segunda, a estrutura de pastas do

computador servidor. Tal aplicação oferece a possibilidade de navegar pela estrutura de pastas até encontrar o local exato em que o *upload* será efetivado ou onde o *download* será descarregado.

É por meio de um cliente de FTP que, normalmente, as páginas desenvolvidas são enviadas para o computador responsável por sua publicação. A Figura 1.6 ilustra a janela de um cliente de FTP. Observe que, do lado esquerdo, são exibidos os arquivos locais (do equipamento do usuário) e, do lado direito, são exibidos os arquivos remotos (no servidor de arquivos).

Embora existam serviços de FTP anônimo, no qual não é necessária a identificação do usuário, normalmente, por questões de segurança, os servidores exigem a informação de um nome de usuário (*login*) e de uma senha.

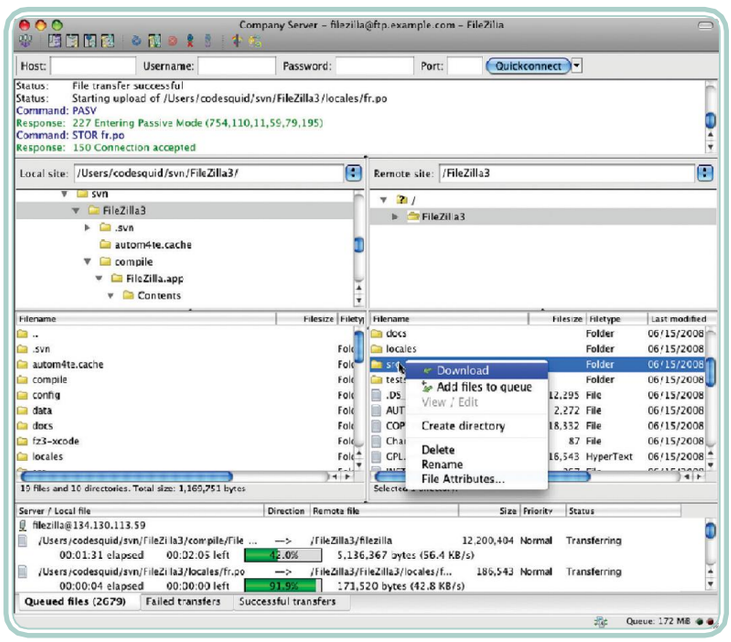


Figura 1.6: Exemplo de aplicativo para transferência de arquivos
Fonte: http://filezilla-project.org/images/screenshots/fz3_osx_main-small.png

Conversa em tempo real

Conhecida e popularizada como *chat*, a conversa em tempo real, diferentemente do serviço de correio eletrônico, exige que ambas as partes interessadas estejam conectadas (*on-line*) ao mesmo tempo. Existem diferentes formas de apresentação do serviço de *chat*. Algumas utilizam protocolos e aplicativos específicos. Nesses casos, é necessário que o usuário esteja cadastrado junto ao serviço de mensagens instantâneas e, à medida em que se conecta a ele, outros usuários, previamente autorizados, podem enviar e receber mensagens.

Outro formato bastante difundido são as salas de bate-papo, nas quais o usuário se conecta a um servidor específico (que geralmente não exige cadastro prévio, apenas uma identificação de usuário), escolhe uma "sala" (geralmente atrelada a um tema) e pode se comunicar com todos os usuários presentes na mesma. Nesse formato, também é possível que os usuários se comuniquem diretamente sem que a mensagem seja exibida a todos os participantes da conversa.

Independente do formato, a conversa em tempo real utiliza-se do modelo cliente/servidor. Uma aplicação cliente responsabiliza-se por apresentar as mensagens recebidas da aplicação servidora e direcionadas ao usuário utilizador, além de coletar e de submeter as mensagens do usuário para a aplicação servidora de forma que esta se encarregue de disponibilizá-las aos destinatários.

Acesso remoto

Acessar remotamente um recurso significa ter controle total sobre tal dispositivo, como se estivesse sentado diante dele, porém a distância. Essa é uma aplicação muito comum desde os primórdios da internet. Através de um acesso remoto, um usuário pode controlar, por exemplo, seu computador a quilômetros de distância. O equipamento que o usuário está operando conecta-se com o recurso remoto que por sua vez envia as ações que estão sendo executadas para a tela do equipamento do usuário.

Existem diferentes aplicativos para a realização de acessos remotos. Em alguns deles, somente é possível acessar terminais em modo caractere (modo texto, não gráfico), em outros, é possível receber toda a tela do computador remoto e interagir, inclusive, utilizando dispositivos apontadores como o *mouse*, por exemplo. Esse tipo de aplicação/serviço é muito utilizado por equipes de suporte ao usuário, que conseguem ter acesso aos computadores dos usuários sem a necessidade de um deslocamento físico (nesse caso, pressupondo que o problema a ser resolvido não está relacionado à conectividade do equipamento).

Navegação no hipertexto

A navegação entre páginas é, especialmente, o conteúdo que mais nos interessa. Ao final deste material, objetiva-se que você esteja apto a construir páginas, formatando e disponibilizando conteúdos. Certamente, essa atividade, possibilitada pela internet, pode ser considerada a mais importante ou a de maior impacto entre a sociedade. Muito do que se faz hoje, na internet, está fortemente apoiado nos fundamentos da navegação pela *web* (ou grande teia, como também é conhecida).

O princípio de navegação pelo hipertexto foi pensado, inicialmente, por Tim Berners-Lee, um britânico que, em meados dos anos 90, trabalhava no núcleo de computação do CERN (Organização Europeia de Pesquisa Nuclear). Tim Berners-Lee buscava uma forma de organizar, eletronicamente, os textos e as pesquisas dos cientistas do CERN (e também de outras partes do mundo) de forma que os documentos produzidos pudessem ser interligados e compartilhados.

Partindo-se desse anseio, Tim Berners-Lee desenvolveu um *software* próprio e um protocolo para recuperar hipertextos que foi denominado de HTTP (*Hypertext Transfer Protocol*). O formato do texto criado para ser transportado pelo protocolo foi chamado de HTML (*Hypertext Markup Language*) e consiste de uma linguagem de marcação pela qual é possível, por meio de comandos (*tags*), incluir ligações entre textos – inclusive entre materiais publicados em diferentes locais. Boa parte do que estudaremos neste material está diretamente relacionada às ideias de Tim Berners-Lee.

Além de um protocolo (conjunto de regras para que dois dispositivos "conversem") e de uma linguagem de marcação (para permitir que os usuários se expressem), era necessário um *software* que, utilizando-se do protocolo desenvolvido, conseguisse obter os documentos escritos em HTML, interpretá-los e exibi-los. Por meio deste *software*, esperava-se que o usuário "navegasse" pelo hipertexto, ou seja, ao encontrar no texto uma ligação com outro material, com um simples clique, o usuário seria direcionado para uma nova página/conteúdo.

Diante de tal necessidade, Tim Berners-Lee criou um protótipo daquele que viria a ser um dos *softwares* mais indispensáveis para quem deseja utilizar a internet: o navegador (ou *browser*). O primeiro navegador foi, inicialmente, nomeado de *World Wide Web*, mas, posteriormente, para evitar confusão com a expressão "*World Wide Web*", foi renomeado para Nexus. A Figura 1.7 demonstra a interface do primeiro navegador (SILVA, 2007).

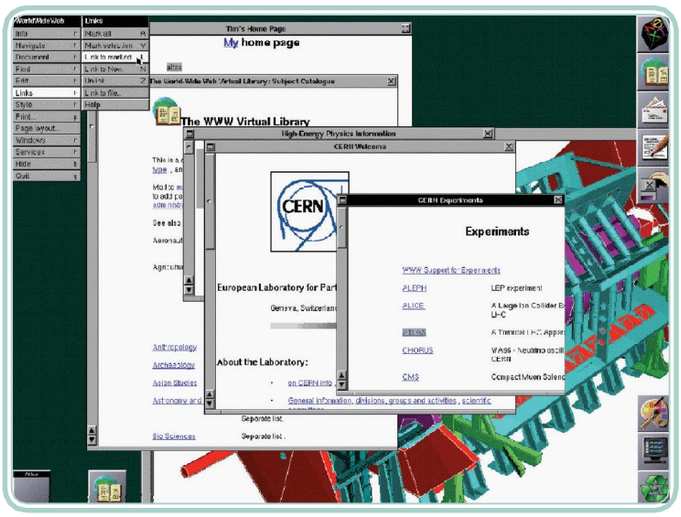


Figura 1.7: Protótipo *World Wide Web* um navegador primitivo
Fonte: Silva, 2007

Assim como os principais serviços que estudamos anteriormente, a navegação pelo hipertexto obedece aos princípios dos sistemas cliente-servidor. Um documento HTML, ao qual podemos chamar de página, está hospedado em um servidor *web* (*web server*) que, nesse caso, desempenha o papel de aplicação servidora. Um servidor *web* recebe requisições de documentos por meio de URLs e as entrega por meio do protocolo HTTP. Pelo lado cliente, temos o navegador ou *browser* cuja função principal é submeter uma requisição no formato de uma URL e, após receber o resultado, na forma de um documento HTML, interpretá-lo e apresentá-lo ao utilizador.

Observe que, quando o navegador recebe a resposta do servidor *web*, o documento HTML retornado não é apresentado tal como foi recebido. O documento passa por um processo de interpretação e o resultado de tal etapa é o que é mostrado ao usuário. Atualmente, não é comum que uma mesma página seja exibida de forma diferente em navegadores concorrentes. Por isso, é muito importante que você, que está começando a desenvolver páginas e aplicações para a internet, preocupe-se sempre em seguir padrões estabelecidos, minimizando esse tipo de problema.

Além de disponibilizar conteúdo através de páginas interligadas, os conceitos difundidos pelas ideias de Tim Berners-Lee evoluíram. Hoje, em muitas situações, o conteúdo a ser apresentado é gerado dinamicamente. Isso é possível através do desenvolvimento de aplicações *web*, ou seja, programas de computador executados no servidor *web* e produtores de conteúdo que é enviado para interpretação e apresentação pelo navegador. Esse tipo de

aplicação, normalmente, faz uso de servidores de banco de dados e de outros recursos comuns em sistemas de informação. A Figura 1.8 ilustra os principais elementos presentes em um cenário de navegação pela internet.

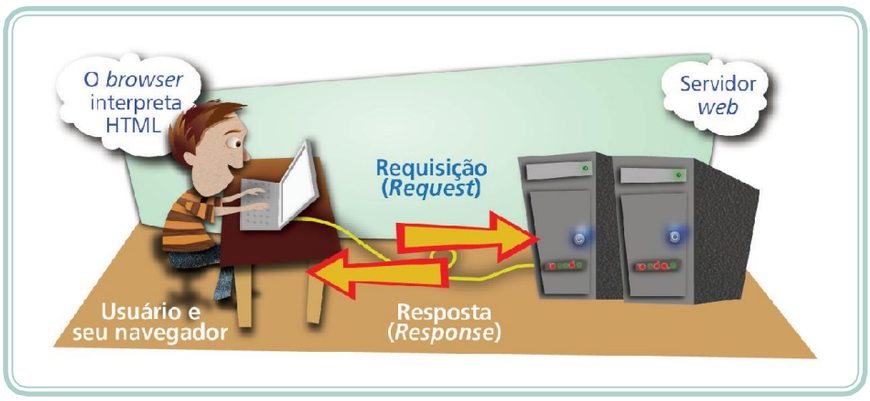


Figura 1.8: Elementos de navegação pelo hipertexto
Fonte: CTISM, adaptado dos autores

Resumo

Nesta aula, estudamos um pouco da história e das motivações que levaram ao surgimento da internet. Também, abordamos algumas tecnologias que balizam seu funcionamento, entre elas os conceitos de protocolo, número IP, sistema de nomes de domínio, etc. De forma sucinta, tratamos de algumas coisas que podemos fazer com a internet, como transferir documentos entre dispositivos separados fisicamente, conversar com outras pessoas – tanto de forma *on-line* como *off-line* –, acessar recursos remotos e navegar pela grande teia de informações.

Atividades de aprendizagem

1. Qual a principal motivação dos cientistas para desenvolver a internet?

2. O que você entende por URL?

3. Imagine a seguinte situação: você precisa enviar para Belo Horizonte (MG) uma cópia de um documento (certidão de nascimento, por exemplo) e o prazo que você tem é muito curto para enviar a cópia impressa pelo correio. Uma alternativa seria enviar o material por FAX, mas o único aparelho ao qual você tinha acesso parou de funcionar. Descreva com suas palavras uma alternativa para o envio do documento, utilizando recursos computacionais.

4. Muito do que se desenvolve para a internet é baseado no modelo cliente/servidor. Explique com suas palavras como esse tipo de aplicação funciona.

5. Teste seus conhecimentos sobre internet (não somente aqueles abordados durante a aula), respondendo V (para verdadeiro) ou F (para falso) no *quiz* a seguir:

( ) Um mecanismo de busca é responsável por distribuir endereços na internet.

( ) Navegadores *web*, como o Internet Explorer e o Mozilla Firefox, podem ser usados para acessar servidores, através dos quais se pode fazer a leitura e o envio de *e-mails*, conhecidos como servidores de *webmail*.

( ) *E-mails* podem ser lidos e enviados não somente por meio de computadores, mas também a partir de telefones celulares e PDAs (computadores de mão) com acesso à internet.

( ) A seguinte sequência de caracteres possui uma estrutura típica de URL de páginas da *web* brasileira: **www.com.empresa.bra**.

( ) A internet é uma ferramenta de utilização privada, sendo que o serviço principal de comunicação de dados está nos EUA.

( ) São exemplos de ferramentas que permitem que duas pessoas conversem em tempo real pela internet: *chat*, MSN e Skype.

( ) *Browser* (ou navegador) é o *software* utilizado para iniciar a conexão do computador com a rede que dá acesso à internet.

( ) Um *e-mail* deve ser direcionado para uma única pessoa.

( ) Todo endereço começa com a palavra www que significa (*World Wide Web* – Rede de Alcance Mundial).

( ) Para acessar um recurso na internet (seja ele uma página ou um arquivo), precisamos do seu endereço: uma URL.

( ) Todas as pessoas que recebem um mesmo *e-mail* sempre conseguem saber quais outras pessoas também o receberam.

( ) O conteúdo dos materiais disponíveis na *web* pode ser interligado por *links* (ou *hiperlinks*).

( ) Fazer um *download* significa enviar um arquivo do seu computador para outro através da internet.

Aula 2 – *Design* para *web*

Objetivos

Oportunizar a reflexão acerca dos conceitos inerentes à área de *design* – em especial o *design* para *web*.

Estruturar informações explorando a anatomia de página *web*.

Diferenciar os elementos presentes em uma página *web*.

O que é *design*?

O termo *design*, como utilizado na língua portuguesa, é um estrangeirismo apropriado do substantivo inglês "*design*" (significando propósito, objetivo, intenção) e do verbo inglês "*design*" (significando projetar/esquematizar) (NIELSEN; LORANGER, 2007). O *design* é uma importante ferramenta de agregação de valor. Podemos aplicar suas técnicas em diferentes áreas como a construção civil, o automobilismo, as mídias impressas, as embalagens, os produtos, etc.

Como área, o *design* é multidisciplinar e trata, em outras palavras, de escolher a melhor forma de apresentar uma ideia. Dependendo da mídia ou do alvo em que se deseja trabalhar, certas técnicas de *design* serão mais apropriadas do que outras. A diagramação de um cartaz convidando a população para uma festa, certamente, será diferente de uma chamada na televisão ou mesmo de um ***site*** de divulgação. As mídias oferecem possibilidades diferentes que o profissional deve estar apto para explorar.

Web design

O advento da *web* abriu espaço para uma nova forma de planejamento de transmissão de ideias/objetivos em materiais publicados em ambientes *on-line*: *web design*. Tais técnicas consistem da estruturação adequada de informações, utilizando recursos apropriados para veiculação em **páginas *web***, de maneira que o usuário possa alcançar seu objetivo de forma direta e agradável.

A-Z

site
Um *site* ou *website* é um conjunto formado por uma ou mais páginas *web* vinculadas.

páginas *web*
Uma página *web* se constitui num documento, normalmente, codificado, através de uma linguagem de marcação que, em conjunto com outros elementos (figuras, por exemplo), pode ser acessado através da internet por meio de uma URL.

O *design* para *web* não é como o *design* de impressão. A *web* é um meio único que, por sua natureza, força os profissionais a desistir do controlar coisas que eles, tradicionalmente, eram responsáveis por controlar (NIEDERST, 2002). Elementos, como cores, fontes e disposição, podem ser determinados pelo usuário (ou por seu *software* navegador) e não há garantias de que todos irão visualizar uma página da mesma forma como foi projetada e desenvolvida.

Projetar para o desconhecido consiste na principal atividade de um profissional que trabalha com *web design*. Como profissional, é importante que você tenha um bom entendimento sobre o ambiente *web* e consiga planejar-se para o desconhecido. Em seu livro "Aprenda *web design*", Niederst (2002) nos alerta sobre alguns itens, que podem ser desconhecidos, para os quais devemos estar preparados.

- **Navegadores desconhecidos** – atualmente, existe uma grande variedade de *softwares*, denominados de *browsers* ou navegadores, cujo intuito é de nos fornecer acesso à *web*. Além disso, um mesmo navegador pode ter inúmeras versões. Evite instruções ou recursos que, sabidamente, são específicos de um ou de outro *browser*. Procure sempre utilizar linguagens ou notações padronizadas pela W3C – esta é uma garantia de que seu *site* irá se comportar da mesma maneira independente do *software* utilizado para exibí-lo.

- **Plataformas desconhecidas** – assim como temos uma vasta gama de navegadores, o mesmo também ocorre com as plataformas de *hardware* e de *software* que são utilizadas. Não temos como saber se os usuários de nossas páginas utilizam plataformas MS Windows, Linux ou MAC OS (ou ainda outra). Evite, dessa forma, recursos muito específicos que possam não estar disponíveis em plataformas concorrentes. É frustrante quando encontramos um *site* com o conteúdo que buscávamos e recebemos a informação de que o conteúdo é incompatível com nossa plataforma ou, então, que, para utilizar a página, é necessária a instalação de *softwares* adicionais. Normalmente, nessa situação, o usuário desiste da página.

- **Preferências de usuários desconhecidas** – em um ambiente *web*, não conhecemos nosso usuário, não sabemos que tipos de recursos ele tem disponíveis ou habilitados para uso. Dessa forma, é importante prever que, por exemplo, um usuário possa ter desabilitado as permissões sobre gravação de ***cookies***. Nesse caso, recursos muito específicos e que dependem de preferências pessoais dos usuários devem ser utilizados de forma opcional.

A-Z

cookies
Um *cookie* é um pequeno arquivo de texto que é trocado entre o navegador e o servidor de páginas durante a conexão. Por meio de um *cookie*, uma aplicação *web* poderia gravar informações sobre a navegação na máquina do usuário. São, normalmente, utilizados para relembrar opções de escolha do usuário e novas visitas feitas ao mesmo *site*.

- **Tamanho de janela desconhecido** – não sabemos se a tela de nosso usuário é grande ou pequena, certo? Com a popularização acelerada de dispositivos móveis, essa variável torna-se ainda mais crucial. Este é um dos aspectos mais incômodos do *design web*, mas não podemos perdê-lo de vista. Procure desenvolver a página de forma que esta ocupe todo o espaço disponível e, quando isso não for possível, avise seu usuário ("para melhor visualizar essa página recomendamos a seguinte **resolução**...").

- **Velocidade de conexão desconhecida** – uma página excessivamente pesada pode impedir que muitos de seus usuários a visitem ou, então, não suportem a demora em carregá-la e desistam. Nesse sentido, os maiores vilões são os elementos gráficos. Utilize, adequadamente, formatos de imagem de acordo com a quantidade de cores e mantenha-os com o menor tamanho possível.

A-Z

resolução
O termo resolução é utilizado para determinar o tamanho máximo que uma tela pode exibir informações. A resolução de uma tela é medida em pontos (*pixels*) e cada usuário, em função do tamanho de monitor disponível, pode utilizar diferentes configurações.

Planejamento e organização de informações

Design e projeto são termos intimamente ligados. Isto evidencia a necessidade de nos planejarmos antes de qualquer ação. Antes de começar a criar um *site*, é preciso planejar sua estrutura (página principal e páginas adjacentes), definindo de forma clara e coerente a sequência das informações que se deseja apresentar (MANZANO; TOLEDO, 2008).

Independente do tamanho ou do objetivo do que se pretende publicar ou desenvolver para *web*, faz-se necessário, inicialmente, desenvolver a ideia e a motivação em torno do que se quer apresentar. Não é necessário se apegar a ferramentas ou a tecnologias para criar um esboço. O mais importante é organizar objetivos e criar estratégias para alcançá-los. Uma forma interessante de alcançar tais estratégias é se questionar acerca do que se pretende fazer. Nessa etapa, são comuns questionamentos como: o que você espera realizar ou o que pretende oferecer com o *site*? A que público o *site* se destina? Quem será o responsável pela geração do conteúdo original e com que frequência tais informações serão atualizadas? Que tipo de aparência ou sensação você espera para o *site*? Observe que a lista de questionamentos pode ser ampliada, mas o mais importante é compreender exatamente onde se deseja chegar e de que forma (isto é o planejamento).

Uma vez que os objetivos estão claros, o passo seguinte é criar e organizar o conteúdo. Certamente, ao longo de sua trajetória profissional, você irá ouvir expressões como: "A pior parte é definir o conteúdo!" ou "Dê-me o conteúdo

e deixe o resto comigo!". De fato, o conteúdo é a parte mais importante de uma página *web*. É ele o fator determinante em relação à quão atrativo ou não será o material a ser publicado. Aparência agradável e bem acabada ajuda, mas um *site* bonito e sem conteúdo não faz com que o visitante retorne a ele.

A definição do conteúdo a ser apresentado é determinada por dois fatores muito importantes e diretamente relacionados: a criação (concepção) e o *design* das informações (LEMAY, 2002). Em relação à criação, é importante deixar claro quem será o responsável por tal concepção. É comum encontrarmos boas ideias, mas, quando vamos "olhar mais de perto", elas são vazias, não têm conteúdo. Isto vale também para quando você estiver desenvolvendo um *site* para algum cliente. É importante que ele lhe ofereça, explicitamente, o conteúdo que deseja apresentar de forma *on-line* e que se estabeleça uma política de atualização de tal conteúdo.

Estruturar informações para um material *on-line* também será uma atividade desafiadora. Esse processo, conhecido como *design* de informações, consiste em organizar e planejar a melhor maneira de apresentar o conteúdo produzido. Aqui, cabe lembrar que cada material tem suas especificidades, ou seja, a forma de abordar um conteúdo em um cartaz ou em um manual não é a mesma que em uma página para internet ou em um programa de televisão.

É comum que o demandante de uma página *web* lhe entregue um *folder* impresso com as informações que ele deseja colocar em seu *site*. Da mesma forma, na medida em que você solicita maiores informações, para extrair mais conteúdo, recebe um manual com 100 páginas. Observe que o trabalho nessa etapa é tentar organizar as informações de forma que seu conteúdo seja adaptado para a *web* e se torne atrativo em tal plataforma.

Quando tratamos de *design* de informações, temos que levar em conta também o público-alvo que irá visitar nossas páginas. Em geral, o público usuário da internet tem pressa e está à procura de informações instantâneas. Da mesma forma, na medida em que encontra o que procura, este mesmo público pode sentir a necessidade de se aprofundar. Precisamos tirar proveito dessas situações e dos recursos que temos em um ambiente *on-line*, adequando nossos conteúdos.

Materiais excessivamente extensos em ambientes *on-line* geram alguns inconvenientes, como o tempo que levam para ser carregados e formatados ou, ainda, por cansar o usuário que, conforme já mencionamos, está à procura

de informação instantânea. Por outro lado, conteúdos muito reduzidos não conseguem transmitir a informação de forma imediata e a necessidade de ficar "clicando", interminavelmente, para chegar até a informação desejada também cansa e irrita os visitantes. Para evitar ambas as situações, precisamos fazer uso do que de melhor a *web* nos oferece: ligações, vínculos ou *links* que nos interligam com conteúdos novos ou anteriores, permitindo a livre navegação por parte do usuário.

O resultado do *design* de informações é, em geral, um diagrama que revela a organização das interligações entre as páginas. Há diferentes formas de organização, cada uma adequada a uma determinada situação. As principais técnicas são: organização sequencial (rígida ou flexível) e organização em árvore (MANZANO; TOLEDO, 2008).

A apresentação de textos longos, comumente, utiliza a técnica de organização sequencial. A Figura 2.1 ilustra duas estruturas de *sites* empregando a organização sequencial. Na primeira parte da figura, observamos uma organização sequencial rígida, na qual as páginas possuem, basicamente, dois elos: o próximo conteúdo e o conteúdo anterior. Esse tipo de abordagem é útil quando a navegação obedece a um fluxo de informações bem definido (etapa1, etapa2, etapaN), como um formulário de pesquisa ou um processo de compra.

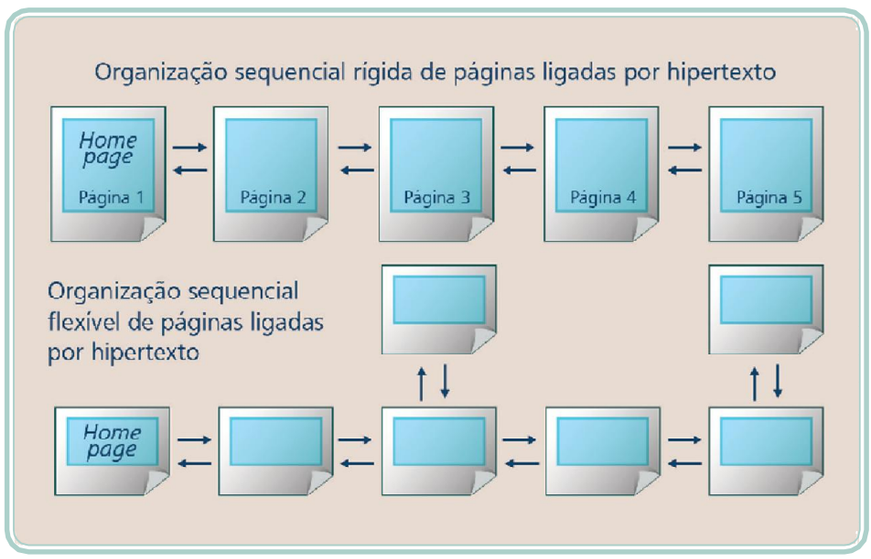


Figura 2.1: Organização sequencial de páginas
Fonte: CTISM, adaptado de Manzano e Toledo, 2008

Ainda, na Figura 2.1, podemos observar uma forma mais flexível de organização sequencial. Nesse caso, ligações (*links*) são empregadas para acrescentar informações complementares (em alguns casos elementos gráficos: figuras, diagramas, fotos, etc.). Tal técnica é, normalmente, utilizada na estruturação de textos longos ou de conteúdos específicos que são desenvolvidos em torno de um assunto principal.

Quando um *site* aborda diferentes assuntos ou se utiliza de tópicos individuais, aconselha-se o emprego da organização em árvore, conforme pode ser visualizado na Figura 2.2. Nesse caso, a característica mais marcante é a utilização de uma página raiz (*homepage*), contendo um índice ou menu com os temas de cada assunto abordado. A partir de então, os temas são relacionados às suas respectivas páginas por meio de ligações de hipertexto e cada página do *site* se relaciona com outras subpáginas.

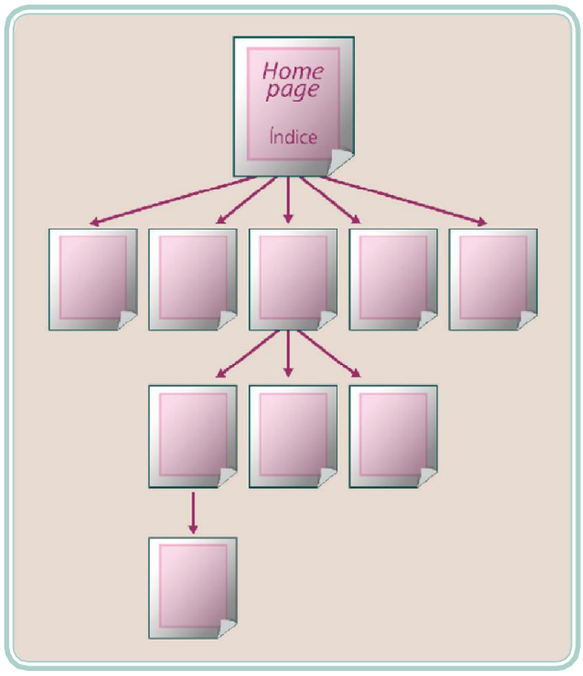


Figura 2.2: Organização em árvore
Fonte: CTISM, adaptado de Manzano e Toledo, 2008

A *web* é um espaço bastante democrático e as técnicas discutidas para estruturação do conteúdo não são, necessariamente, utilizadas de forma rígida e inflexível. Pelo contrário, o que se observa na prática são estruturas mistas, ou seja, organização dos vínculos entre as páginas de forma lógica dando ao usuário a liberdade de navegar e de se aprofundar de acordo com sua necessidade, permitindo, inclusive, começar do princípio a qualquer momento.

Layout e aparência

Layout (ou leiaute) é a forma pela qual os itens estão dispostos/diagramados, em outras palavras, refere-se ao *design* gráfico e a aparência visual. Durante o desenvolvimento de páginas para internet o projeto de *layout* é uma atividade crítica e é importante que esteja definido de forma consistente e prioritária, pois a partir dele o restante do trabalho será colocado em prática.

Na etapa de especificação do *layout*, devem ser definidos os itens que irão compor a identidade visual do *site*, como o esquema de cores, a **tipografia**, o estilo das imagens (fotos ou ilustrações, por exemplo) e a **ergonomia**. A matéria-prima para essa atividade é, normalmente, uma relação de elementos gráficos e alguns manuscritos que indicam o que deve ser utilizado. No entanto, você irá perceber que também não são raras as situações onde o profissional necessita "captar" essas informações e fazer a proposição de algo novo (NIEDERST, 2002).

Em geral, a definição de um *layout* segue alguns princípios básicos (CARRION, 2006): hierarquia das informações, foco/ênfase, equilíbrio, relacionamento dos elementos e unidade/integração. A hierarquia das informações determina qual a disposição da informação, baseando-se em sua importância em relação aos demais elementos visuais. De acordo com esse princípio, precisamos definir a informação mais importante para posicioná-la em um lugar estratégico, porque o usuário interage de imediato com aquilo que ele vê primeiro.

O princípio do foco/ênfase nos auxilia na disposição dos elementos mais prioritários, utilizando os espaços mais nobres do *layout*. Determinar o foco envolve identificar a ideia central, ou o ponto focal do material a ser publicado. Aspectos culturais, assim como decisões do projeto de uma página na *web*, influenciam diretamente esse princípio. Em geral, informações mais importantes devem estar localizadas no canto superior esquerdo, sendo o início ou a base da tela os melhores locais para mostrar informações sobre orientações que devem estar sempre visíveis.

O equilíbrio dos elementos e das informações de um *site* afeta, diretamente, a forma como o *layout* é compreendido. Mistura de assuntos sem conexão lógica ou mistura de elementos gráficos (sem deixar claro onde um começa e outro termina) são os principais problemas que podem afetar o equilíbrio. Um típico exemplo de falta de equilíbrio em um *layout* é a mistura de conteúdo com anúncios. Isso passa para o usuário uma sensação de estar sendo enganado ou persuadido. A falta de estrutura e de equilíbrio torna uma página na *web* mais difícil de ser entendida pelo usuário.

A-Z

tipografia
É a arte e o processo de criação na composição de um texto, física ou digitalmente. Seu objetivo é dar ordem estrutural e forma aos sinais gráficos utilizados.

ergonomia
Consiste no entendimento das interações entre seres humanos e outros elementos de um sistema. A ergonomia é a qualidade da adaptação de um dispositivo a seu operador e à tarefa que ele realiza. No projeto de uma interface *web*, a ergonomia objetiva facilitar e otimizar o trabalho do usuário junto ao computador.

De forma complementar, o princípio de relacionamento entre os elementos prega que estes precisam estar agrupados, categorizados, conduzindo a um fluxo de navegação lógico. A utilização de elementos visuais pode ajudar a comunicar uma relação/conexão específica de uma página com o *site* do qual ela faz parte. Da mesma forma, a utilização de esquemas de cores pode potencializar a relação entre diferentes elementos do *layout*.

Por fim e não menos importante, um *layout* deve se traduzir em uma unidade integrada de elementos. Esse princípio evidencia-se, especialmente, quando, em um grande projeto, há necessidade de manter subpáginas visualmente unificadas com o projeto de *layout* principal. Isso facilita a navegação uma vez que oferece um ambiente consistente, integrado e previsível.

Além desses princípios que norteiam a organização de um *layout*, precisamos observar algumas propriedades ou qualidades que potencializam a experiência positiva com o *site*, especialmente a usabilidade e a navegabilidade. Também, não podemos nos distanciar da legibilidade, pois de nada adianta ser bonito se não pode ser compreendido. E, por fim, destacamos a relevância da acessibilidade, permitindo um acesso universal e democrático às informações.

Usabilidade é um atributo de qualidade relacionado à facilidade de uso de algo. Refere-se à rapidez com que os usuários podem apreender e utilizar algo e o quanto lhes agradam utilizá-la. Se um recurso não pode ser utilizado ou não é utilizado por seus usuários, então, ele não precisa existir.

Navegabilidade é uma propriedade da organização das informações que nos remete à capacidade de navegação, ou seja, nos permite percorrer intuitivamente os caminhos virtuais que nos são propostos de forma que saibamos onde estamos, de onde viemos e quais são nossas possibilidades futuras.

A legibilidade é uma qualidade que determina a facilidade de leitura de alguma coisa; mede o quão legível esta se apresenta ao leitor.

Acessibilidade, no contexto da internet, é a propriedade que um *site* tem para permitir que portadores de necessidades especiais se utilizem dos recursos que oferece. Dependendo como os elementos gráficos são utilizados no *site*, eles inviabilizam qualquer tentativa de um deficiente visual, por exemplo, conseguir acessar as informações disponibilizadas pelo *site*.

Anatomia de um *site*

Em função dos princípios discutidos até então, estudiosos da área de *web design* têm experimentado diferentes formas de organização de *layouts*. Certamente, enquanto usuário da internet, você já deve ter tido boas e más experiências. Contudo, observamos que há uma espécie de padrão ou de roteiro com bastante aceitação e altamente difundido entre grandes e pequenos *sites*. Tal modelo é composto por alguns elementos de conteúdo que podem ser visualizados na Figura 2.3, na forma de um esquema numerado.

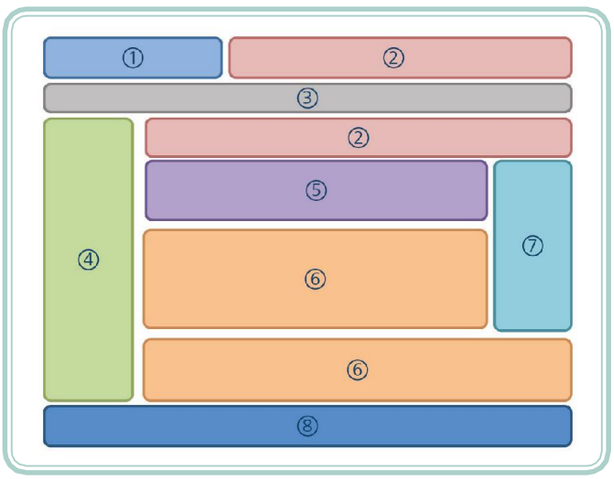


Figura 2.3: Anatomia de uma página *web*
Fonte: Autores

1. **Logotipo** – o logotipo ou logomarca do *site*, normalmente, ocupa o espaço mais nobre, onde, comumente, a visualização ocorre primeiro (canto superior esquerdo).

2. ***Banner* padrão, área de busca, anúncios externos** – este é, também, um espaço nobre que, frequentemente, é explorado com um *banner*. Nessa área, recomenda-se a utilização de uma ferramenta de busca ou de um mapa do *site* (para simplificar e agilizar a localização de informações).

3. **Menu administrativo** – o menu administrativo é uma opção importante, no entanto deve ser discreto. É nele que colocamos informações sobre a empresa, identificação (*login*) ou cadastro de usuários, contato, ajuda, etc.

4. **Menu de navegação** – categorização das informações disponibilizadas pelo *site*. É comumente encontrado de forma horizontal, como o menu

administrativo. É importante que as categorias ou opções do menu sejam curtas e claras, indicando ao usuário, rapidamente, a opção que ele deve escolher (excessivas opções também não são recomendadas).

5. **Área de destaque** – nesse espaço, recomenda-se enfocar algo de maior importância dentro do contexto do *site*. Um *site* de comércio eletrônico poderia divulgar uma promoção; um *site* de notícias poderia noticiar o fato mais marcante do dia; um *site* pessoal poderia remeter para a atividade mais recente ou última postagem.

6. **Conteúdo** – área de conteúdo é o lugar onde são exibidas as informações na medida em que navegamos pelo *site*.

7. **Anúncios** – a barra lateral, da direita, pode ser utilizada para divulgação de atividades afins ao *site*, como anúncios ou *links* para outras páginas.

8. **Rodapé** – o rodapé é, normalmente, utilizado para informações sobre o portal. Não é, necessariamente, usada por visitantes comuns, mas sim por aqueles com algum interesse específico (como anunciar, trabalhe conosco, política de privacidade, termos de uso, etc.).

A Figura 2.4 sobrepõe a imagem de um portal (**www.brasil.gov.br**), utilizando o esquema numerado da Figura 2.3 de forma que, através de um exemplo real, possamos visualizar os elementos de uma página *web* e sua anatomia.

Figura 2.4: Elementos de uma página *web*
Fonte: Autores

Cabe ressaltar que os elementos discutidos até então são aqueles comumente encontrados. De acordo com o propósito da página, ela poderá não utilizar todos os elementos ou ainda adaptar a posição dos mesmos, segundo o que seu projeto de *design* considerou mais importante. A Figura 2.5 ilustra um dos *sites* mais famosos e que, certamente, para muitos de nós serve como "porta de entrada" para a *web*. Observe como a simplicidade de detalhes, aliada ao propósito do *site* (busca de informações), oferece uma sensação de equilíbrio e de integração entre seus elementos.

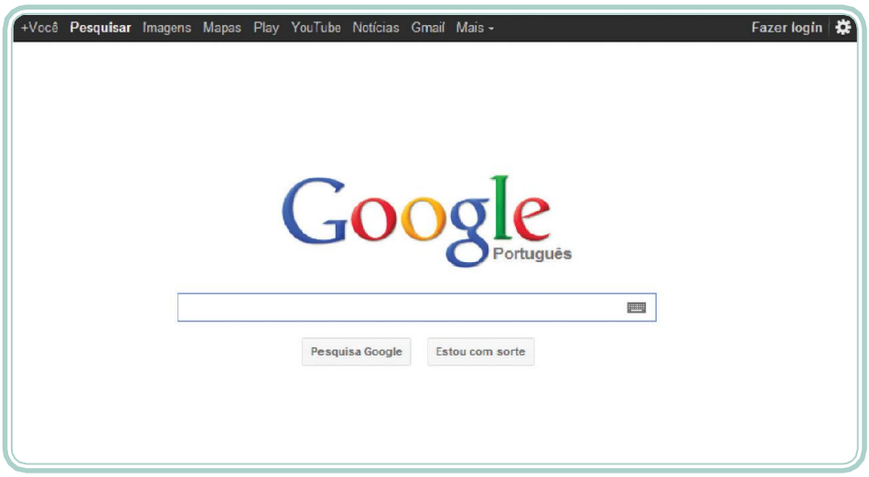


Figura 2.5: Página do mecanismo de busca Google
Fonte: www.google.com

Resumo

Nesta aula, discutimos alguns princípios e técnicas que nos orientam a organizar conteúdos para publicação em ambientes *on-line*, bem como noções de como organizá-los em uma proposta de *layout*. No decorrer da aula, conhecemos alguns conceitos chaves que precisam ser aprofundados, mas que nos dão uma noção do que significa desenvolver *sites* e aplicações para a internet. É importante salientar que a *web* é um espaço muito democrático sempre aberto a inovações, no entanto existem alguns princípios e técnicas largamente discutidos e que, quando seguidos, oferecem ao usuário uma melhor experiência.

Atividades de aprendizagem

1. Explique por que uma das principais atividades do profissional que trabalha com *web design* é projetar para o desconhecido?

2. Considere a necessidade de desenvolver um *site* para operacionalizar uma votação eletrônica. Nesse contexto, qual das técnicas de organização de informações seria mais indicada? Justifique sua resposta.

3. Que tipos de iniciativas podem habilitar um *site* a oferecer acessibilidade a seus usuários?

4. Acesse o endereço da Enciclopédia Livre Wikipedia (**www.wikipedia.org**), descreva de que forma as informações apresentadas estão estruturadas e quais os elementos que estão sendo utilizados.

5. Que tipo de qualidade/característica está faltando em um *site* cuja principal reclamação dos usuários é a dificuldade de utilização?

6. Faça um esboço para o *layout* de um *site* cujo objetivo é a divulgação de anúncios de veículos usados. Considere que diferentes revendas/garagens ou até mesmo pessoas físicas podem publicar seus anúncios. Os anúncios podem ser pagos (nesse caso ganhando maior destaque) ou gratuitos, ficando disponíveis no sistema de busca. Questione-se acerca do que precisa ser divulgado e de que forma.

Aula 3 – Introdução à linguagem de marcação

Objetivos

Possibilitar a compreensão da estrutura básica de uma página.

Entender o conceito de *tags*.

Utilizar *tags* para formatar blocos de texto.

Entender como as páginas são disponibilizadas na *web*.

HTML – a linguagem para escrever páginas para *web*

Antes de começarmos a dar vida aos *layouts* que aprendemos a planejar e a estruturar na aula anterior, precisamos nos instrumentalizar. O primeiro passo consiste em entender de que forma um documento ou *layout* deve ser codificado para que seja entendido por um *software* navegador. Parte dessa introdução já foi abordada em nossa primeira aula, no entanto, naquele momento, não foi possível aprofundarmos os conhecimentos.

Os documentos disponíveis na internet, independentemente da temática que abordem (notícias, entretenimento, ciência, comércio, etc.), são estruturados através de uma Linguagem de Marcação de Hipertexto conhecida como HTML (*Hypertext Markup Language*). Uma linguagem de marcação é um mecanismo para adicionar marcas com algum significado a um texto. Tais marcas são omitidas na versão do texto que é apresentada ao usuário (NIEDERST, 2002).

Certamente, com um exemplo ficará mais simples de entender e apresentar os conceitos subsequentes. Utilizando um editor de texto não formatado (como bloco de notas ou *gedit*), digite o bloco de texto ilustrado na Figura 3.1 que representa a estrutura básica de uma página HTML. Após digitar, salve o arquivo com o nome "exemplo.html".

Existem inúmeros *softwares* que permitem a construção de páginas HTML sem que seu utilizador conheça a fundo os detalhes da linguagem. No entanto, é recomendado que, neste momento, você utilize um editor de textos sem recursos para formatação para que conheça e se aproprie dos conceitos básicos da linguagem de marcação para poder explorá-los, futuramente, com maior profundidade, utilizando-se de um editor que irá lhe facilitar muito o trabalho.

```
<html>
  <head>
    <title>Minha Página</title>
    <meta name="author" content="Bruno e Leticia" />
  </head>

  <body bgcolor="yellow">
    Essa é minha primeira página
  </body>
</html>
```

Figura 3.1: Estrutura básica de uma página HTML
Fonte: Autores

O arquivo "exemplo.html", se aberto por um navegador (*browser*), apresentará o resultado conforme pode ser visualizado na Figura 3.2. Observe que a maior parte do texto digitado (comandos ou *tags*) é suprimida, de forma que apenas o conteúdo entre as marcas é visualizado.

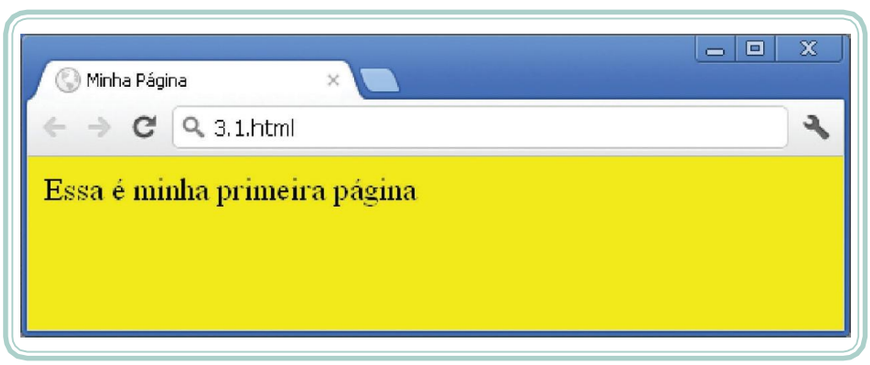


Figura 3.2: Visualização de uma página HTML em um navegador
Fonte: Autores

A partir desse primeiro exemplo, podemos dar continuidade às nossas discussões. A linguagem HTML é utilizada para trabalhar a estrutura de um *site*. É verdade que ela também permite que se atue sobre a apresentação desta, embora não seja essa sua especialidade. Observe que, no exemplo anterior, apenas dois itens são visíveis após o processamento do *browser*: o título, "Minha Página", e a informação "Essa é minha primeira página". O

restante do texto serviu apenas para informar de que forma o *browser* deve apresentar a informação.

As marcas utilizadas para dar significado ao texto são conhecidas como *tags*, cuja tradução literal seria algo como "etiqueta". A maior parte das *tags* é do tipo *container*, ou seja, há uma marca inicial que delimita o começo da instrução e outra de fechamento, encerrando o efeito da instrução sobre o conteúdo. A marca de fechamento tem o mesmo nome da marca inicial e se diferencia desta apenas por começar com uma barra (/). Algumas *tags*, no entanto, são independentes, ou seja, não há necessidade de uma marca para fechamento. As *tags* podem conter também atributos, como é caso da *tag* **<body>** que delimita o início do corpo da página e utiliza-se do atributo *bgcolor* para definir a cor de fundo da área de conteúdo. O valor de um atributo é sempre colocado entre aspas.

A especificação atual da linguagem HTML é a versão 5 e é padronizada pelo consórcio W3C. Contudo, ao longo dos nossos exemplos, procuraremos utilizar instruções compatíveis entre as diferentes versões. Não perca de vista, entretanto, os avanços e os recursos disponíveis a partir de novas versões. É importante que você esteja sempre atualizado e disponível para aprender.

Estrutura básica de uma página HTML

A Figura 3.1 ilustrou um típico exemplo de uma estrutura básica para uma página HTML. Observe que a *tag* **<html>** delimita o início e término do documento HTML (ela é a primeira *tag* a ser aberta e a última a ser fechada). Observe, também, que as *tags* podem se apresentar de forma hierárquica, ou seja, umas dentro das outras. Este é o caso das *tags* **<head>** e **<body>** que demarcam, respectivamente, a área de cabeçalho e de corpo do documento.

A seção de cabeçalho não produz informações visuais e é utilizada como uma seção de configuração, na qual podemos indicar comandos que devem ser lidos antes de se carregar o conteúdo que será apresentado. É nessa seção que indicamos, por exemplo, o título da página (*tag* **<title>**) e também **metadados**, como o autor da página (*author*), descrição (*description*), palavras chave (*keywords*), idioma (*language*), entre outros. A utilização da *tag* de cabeçalho **<head>** é opcional.

A-Z

metadados
São informações sobre os dados, ou seja, dados que descrevem de que forma os dados referenciados devem ser interpretados ou categorizados.

Após a seção de cabeçalho, inicia-se, obrigatoriamente, o corpo da página – a *tag* **<body>**. O corpo da página define a área que será apresentada

Para saber mais sobre outras opções da *tags* de metadados, acesse a referência oficial do *site* da W3C: http://www.w3.org/wiki/HTML/Elements/meta

como conteúdo, ou seja, a área visível da página. É a partir da *tag* **<body>** que o documento HTML será estruturado. Alguns atributos da *tag* **<body>** permitem, por exemplo, alterar a cor de fundo (*bgcolor*) ou, então, utilizar uma imagem de fundo (*background*).

Elementos que não são interpretados

Antes de dar sequência à estruturação de conteúdos a partir de documentos HTML, é importante que tenhamos claro que algumas de nossas ações podem vir a ser ignoradas pelo *browser*. Para entender melhor, vamos a mais um exemplo. Digite o bloco de texto ilustrado na Figura 3.3, salve-o como um arquivo html e abra-o a partir do navegador.

```
<html>
  <head>
    <title>Observe</title>
  </head>

  <body>

    Olá turma!

    Abaixo vou elencar algumas coisas que vocês precisam estudar

        Conceitos de redes de computadores
        Tratamento de imagens digitais
        Teoria das cores

    Por hoje é isso <abraco> um abração a todos </abraco>

    <!--
        Não se esqueçam de que isso pode cair na avaliação
    -->

  </body>
</html>
```

Figura 3.3: Código com exemplo de marcas/caracteres ignorados pelo navegador
Fonte: Autores

O resultado da interpretação do código da Figura 3.3 pode ser visualizado na Figura 3.4. Observe, por exemplo, que as quebras de linhas e os múltiplos espaços ou tabulações são ignoradas pelo navegador no momento em que interpreta o código. Da mesma forma *tags* não conhecidas, como a *tag* **<abraco>**, também são ignoradas pelo navegador. Por fim, quando se deseja realizar anotações no código da página sem que as mesmas sejam apresentadas, podemos utilizar de um conceito denominado de comentário. Um comentário é uma anotação feita sobre um código que é ignorado no momento em que é interpretado. Os comentários em HTML devem ser colocados entre os sinais de <!-- e --> (conforme pode ser visualizado na Figura 3.3, na frase “Não se esqueçam de que isso pode cair na avaliação”).

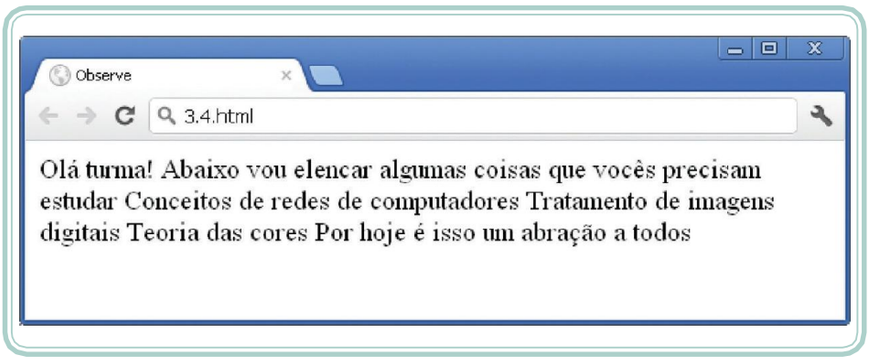


Figura 3.4: Visualização de uma página com elementos que são ignorados pelo navegador conforme Figura 3.3
Fonte: Autores

Formatação de parágrafos e blocos de texto

Agora que já entendemos de que forma uma linguagem de marcação funciona, podemos explorar alguns recursos ligados à organização de parágrafos e de blocos de texto. Lembremos que o objetivo da linguagem HTML é permitir a estruturação de conteúdo. Nas subseções seguintes, vamos apresentar e descrever algumas *tags* e, ao final da seção, as Figuras 3.6 e 3.7 ilustrarão o efeito do emprego destas.

3.4.1 Cabeçalho (<h1> </h1> ... <h6> </h6>)

As *tags* **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>** e **<h6>** são utilizadas para demarcar uma área do documento que indica um cabeçalho – *head* (um título ou subtítulo, por exemplo). Quanto menor for o valor, mais destaque receberá a apresentação do cabeçalho. Os cabeçalhos são exibidos em negrito e, ao final dos mesmos, é feita uma quebra de linha. A utilização de demarcam uma área do texto que merece realce e, normalmente, são utilizados para iniciá-la, informando, por exemplo, o título do mesmo. Opcionalmente, pode-se utilizar o parâmetro *align* para indicar o alinhamento do cabeçalho: *right* (a direita), *left* (a esquerda) ou *center* (centralizado).

Parágrafos ( )

A *tag* ** ** (*paragraph*) demarca um parágrafo textual. Dividir um texto em parágrafos é uma atividade presente em qualquer redação. Associado à *tag* ** **, existe um parâmetro denominado de *align* o qual informa o alinhamento do texto, podendo este ser centralizado (*center*), justificado (*justify*), alinhado à esquerda (*left*) ou alinhado à direita (*right*). A utilização da *tag* ** ** de forma vazia, como ** **, produz uma quebra de linha. Todavia, existe uma *tag* especial para esse propósito, conforme veremos a seguir.

Textos pré-formatados (<pre>)

Há situações nas quais temos a necessidade de manter a apresentação de um texto tal como ele foi digitado (ou seja, respeitando espaços e tabulações e quebras de linha). Uma típica situação em que isto acontece é quando precisamos representar um exemplo de uma **linguagem de programação**. Nesse caso, a opção indicada é a *tag* **<pre>** (*predefined*). Seu efeito visual é um recuo à esquerda e a apresentação do texto em uma fonte de tamanho fixo tal como a usada quando este foi digitado (respeitando quebras de linha, espaços e tabulações). Observe, na Figura 3.5, o desenho de uma vaca apenas com sinais gráficos. Veja que, com a *tag* **<pre>**, no momento em que o código é visualizado, ele mantém as posições do texto tal como foram predefinidas no documento HTML.

A-Z

linguagem de programação
É constituída de um conjunto de instruções que expressam uma tarefa a ser executada por um dispositivo. É um conjunto de palavras e de expressões estruturadas que nos permitem programar um dispositivo ou sistema computadorizado.

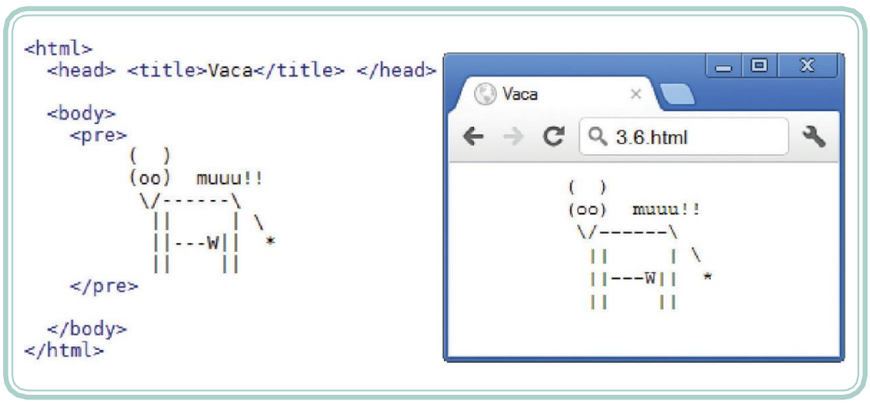


Figura 3.5: Exemplo de utilização da *tag* <pre>
Fonte: Lemay, 2002

Quebras de linha (<br />) e linhas horizontais (<hr />)

Conforme discutimos anteriormente, o navegador ignora as quebras de linha informadas ao longo do texto. Para produzir o efeito desejado, devemos utilizar a *tag* **
** (*break*) que é independente, ou seja, não precisa de outra *tag* para fechá-la, bastando, para isto, indicar, na própria instrução de abertura, o sinal de "/" para fechá-la automaticamente.

A *tag* **<hr />** (*head row*) tem o mesmo princípio da *tag* **
**, no entanto seu efeito é a produção de uma linha que divide a página horizontalmente. Seu intuito é produzir seções de divisão ao longo do conteúdo.

```
<html>
  <head>
    <title>Formatação de Parágrafos e Blocos de Texto</title>
  </head>

  <body>

    <h1> Cabeçalho de primeiro nível </h1>
    <hr>
    <h2> Cabeçalho de segundo nível </h2>

    <p align="justify"> Este é um parágrafo com alinhamento justificado
    observe que todo o seu conteúdo será exibido de forma alinhada tanto
    a direita quanto a esquerda, as outras opções de alinhamento que
    existem são: right, left e center </p>

    <p align="right"> Neste caso estamos visualizando um parágrafo
    alinhado a direita e neste ponto <br /> faremos uma quebra
    de linha </p>

    <h3> Aqui temos um cabeçalho de terceiro nível </h3>

    <p align="center"> Este é um texto centralizado</p>

  </body>
</html>
```

Figura 3.6: Código HTML com recursos de formatação de parágrafos e de blocos de texto
Fonte: Autores

Figura 3.7: Visualização de uma página com recursos de formatação de parágrafos e de blocos de texto
Fonte: Autores

Formatação de fonte

Embora prover a aparência de uma página não seja o objetivo principal da linguagem HTML, existem alguns recursos que nos permitem enfatizar expressões ao longo do texto, bem como aplicar-lhes algum formato. A principal *tag* que atua sobre os atributos da fonte utilizada ao longo do texto é a *tag* **<font>**. Entretanto, existem várias outras *tags* que têm influência sobre o

estilo ou opções de formato de um bloco de texto. Vejamos, a seguir, alguns recursos para formatação de textos e, ao final, vamos colocá-los em prática.

Fonte (<font>)

A *tag* **<font>** – cuja tradução é dispensável – atua sobre atributos do texto em si. Em síntese, são três os atributos que podemos alterar: o tamanho (*size*), o tipo (*face*) e a cor (*color*). O tamanho é definido a partir da utilização de um número inteiro e pode ser empregado de forma absoluta (informando diretamente o número) ou de forma relativa, indicando, através dos sinais positivo (+) ou negativo (-), o valor de incremento ou decremento em relação ao valor padrão do navegador (MARCONDES, 2005). O valor absoluto para o atributo *size* (que define o tamanho) pode variar entre 1 e 7 – quanto maior o número, maior será o tamanho de sua exibição.

Lembre-se que desenvolver para *web* é desenvolver para o desconhecido. Diferentes navegadores podem ter valores padrão distintos para o tamanho de fonte ou mesmo o usuário pode alterar suas preferências e definir outro valor.

A alteração do tipo de fonte (da tipografia ou do formato das letras) pode ser feita com o atributo *face*. Nesse caso, é importante ressaltar que nem todos os usuários, dispositivos ou plataformas compartilham das mesmas fontes. Dessa forma, procure utilizar tipos de fontes altamente difundidas (ex.: Arial, Times New Roman, Verdana). O atributo *face* permite especificar mais de um tipo de fonte. Assim, se o primeiro não existir, usa-se o segundo e assim por diante. Para indicar mais de uma fonte, os nomes destas devem ser separados por vírgulas.

Por fim, o atributo *color* permite alterar a cor de exibição de uma fonte. Sua utilização é bastante simples, contudo precisamos entender de que forma o HTML nomeia ou utiliza as cores. A representação das cores em HTML utiliza do modelo de cores RGB (*Red/Green/Blue*), cujo propósito é a reprodução de cores em dispositivos eletrônicos. A partir das cores básicas do modelo (vermelho, verde e azul), são feitas combinações com diferentes intensidades de cada uma das cores, conseguindo uma variedade de 16 milhões de cores.

hexadecimais
O sistema de numeração hexadecimal utiliza os 10 dígitos numéricos (0...9) e as letras A, B, C, D, E e F para representar informações. Com dois dígitos hexadecimais é possível realizar 256 combinações diferentes. A expressão #1A corresponde ao número inteiro 26 em base decimal.

As combinações mais básicas podem ser expressas através de nomes, no entanto a forma mais interessante de se conseguir uma cor pelo sistema RGB é misturar os tons. Cada tom pode variar de 0 a 255, o que pode ser representado por dois dígitos **hexadecimais**. A cor cujo código é #FF0000 representa, por exemplo, o vermelho puro, em função de que os dois primeiros dígitos

estão com o valor hexadecimal mais alto possível. Já a cor de código #000000 representa o preto (a ausência total de cor, em qualquer tonalidade). Observe, no Quadro 3.1, a relação dos nomes de cores reconhecidos pela W3C. Além destes, é possível variar os valores de cada dupla de dígitos hexadecimais e conseguir 16.777.216 combinações (2563).

Quadro 3.1: Nomes de cores reconhecidos pela W3C

black	#000000 0 0 0		lime	#00FF00 0 255 0	
silver	#C0C0C0 192 192 192		green	#008000 0 128 0	
gray	#808080 128 128 128		olive	#808000 128 1280	
white	#FFFFFF 255 255 255		yellow	#FFFF00 255 2550	
maroon	#800000 128 0 0		navy	#000080 0 0 128	
red	#FF0000 255 0 0		blue	#0000FF 0 0 255	
purple	#800080 128 0128		teal	#008080 0 128128	
fuchsia/ma-genta	#FF00FF 255 0255		aqua/cyan	#00FFFF 0 255255	

Fonte: Marcondes, 2005

Outros recursos de formatação

Além de alterar o tipo da fonte, podemos demarcar áreas de texto que precisam ser enfatizadas. Estilos físicos, como negrito, itálico e subscrito, podem ser conseguidos com *tags* específicas. O Quadro 3.2 demonstra os principais recursos que podemos utilizar para enfatizar blocos de conteúdo textual.

Quadro 3.2: Estilos físicos

Tag	Propósito	Exemplo	Resultado
(*bold*)	Negrito	escuro	**escuro**
(*italic*)	Itálico	deitado	*deitado*
<u> (*underline*)	Sublinhado	<u>com linha</u>	com linha
<s> (*strike*)	Riscado	<s>tachado</s>	~~tachado~~
<tt> (*teletype*)	Espaçamento fixo	<tt>fixo</tt>	fixo
<sub> (*subscript*)	Subscrito	H₂O	H_2O
<sup> (*superscript*)	Sobrescrito	km²	km^2

Fonte: Marcondes, 2005

Caracteres especiais

Imagine uma situação na qual necessitamos separar um texto por um conjunto de espaços em branco. É sabido que espaços em branco, em sequência, são ignorados pelo navegador. Então, como conseguir representá-los? Imagine agora uma situação em que precisamos representar o sinal de micro (µ) ou algum outro sinal não encontrado, normalmente, no teclado. Essas situações podem ser resolvidas com a utilização de caracteres especiais, também conhecidos como caracteres ISO.

O alfabeto de caracteres ISO (*International Organization for Standardization*) é obtido por um código especial formado pelos caracteres & ("e" comercial), # (tralha ou sustenido), a definição de um valor numérico de três dígitos, o caractere ";" (ponto-e-vírgula) para finalizar. É possível, ainda, utilizar uma espécie de abreviação ou entidade equivalente ao código ISO. Por exemplo, para representar o sinal de "menor que" podemos utilizar o código < ou então a entidade *&lt*; onde *lt* é a abreviação para *letter then* (MANZANO; TOLEDO, 2008).

A utilização dos códigos do alfabeto de caracteres ISO, por meio de entidades, garante que o conteúdo do documento HTML será exibido da forma como foi definido, independente do navegador ou da plataforma utilizada. O Quadro 3.3 apresenta os caracteres do alfabeto ISO e o nome da entidade correspondente.

Quadro 3.3: Alfabeto de caracteres ISO

Alfabeto ISO	Entidade	Símbolo	Alfabeto ISO	Entidade	Símbolo
&#034;	&quot;	"	&#206;	&Icirc;	Î
&#038;	&amp;	&	&#207;	&Iuml;	Ï
&#060;	&lt;	<	&#208;	&ETH;	Ð
&#062;	&gt;	>	&#209;	&Ntilde;	Ñ
			&#210;	&Ograve;	Ò
&#161;	&iexcl;	¡	&#211;	&Oacute;	Ó

Alfabeto ISO	Entidade	Símbolo	Alfabeto ISO	Entidade	Símbolo
&#162;	&cent;	¢	&#212;	&Ocirc;	Ô
&#163;	&pound;	£	&#213;	&Otilde;	Õ
&#164;	&curren;	¤	&#214;	&Ouml;	Ö
&#165;	&yen;	¥	&#215;	&times;	×
&#166;	&brvbar;	¦	&#216;	&Oslash;	Ø
&#167;	&sect;	§	&#217;	&Ugrave;	Ù
&#168;	&uml;	¨	&#218;	&Uacute;	Ú
&#169;	&copy;	©	&#219;	&Ucirc;	Û
&#170;	&ordf;	ª	&#220;	&Uuml;	Ü
&#171;	&laquo;	«	&#221;	&Yacute;	Ý
&#172;	&not;	¬	&#222;	&THORN;	Þ
&#173;	&shy;		&#223;	&szlig;	ß
&#174;	&reg;	®	&#224;	&agrave;	à
&#175;	&macr;	¯	&#225;	&aacute;	á
&#176;	&deg;	°	&#226;	&acirc;	â
&#177;	&plusmn;	±	&#227;	&atilde;	ã
&#178;	&sup2;	²	&#228;	&auml;	ä
&#179;	&sup3;	³	&#229;	&aring;	å
&#180;	&acute;	´	&#230;	&aelig;	æ
&#181;	&micro;	µ	&#231;	&ccedil;	ç
&#182;	&para;	¶	&#232;	&egrave;	è
&#183;	&middot;	•	&#233;	&eacute;	é
&#184;	&cedil;	¸	&#234;	&ecirc;	ê
&#185;	&sup1;	¹	&#235;	&euml;	ë
&#186;	&ordm;	º	&#236;	&igrave;	ì
&#187;	&raquo;	»	&#237;	&iacute;	í
&#188;	&frac14;	¼	&#238;	&icirc;	î
&#189;	&frac12;	½	&#239;	&iuml;	ï
&#190;	&frac34;	¾	&#240;	&eth;	ð
&#191;	&iquest;	¿	&#241;	&ntilde;	ñ
&#192;	&Agrave;	À	&#242;	&ograve;	ò
&#193;	&Aacute;	Á	&#243;	&oacute;	ó
&#194;	&Acirc;	Â	&#244;	&ocirc;	ô
&#195;	&Atilde;	Ã	&#245;	&otilde;	õ
&#196;	&Auml;	Ä	&#246;	&ouml;	ö
&#197;	&Aring;	Å	&#247;	&divide;	÷
&#198;	&AElig;	Æ	&#248;	&oslash;	ø
&#199;	&Ccedil;	Ç	&#249;	&ugrave;	ù
&#200;	&Egrave;	È	&#250;	&uacute;	ú
&#201;	&Eacute;	É	&#251;	&ucirc;	û
&#202;	&Ecirc;	Ê	&#252;	&uuml;	ü
&#203;	&Euml;	Ë	&#253;	&yacute;	ý
&#204;	&Igrave;	Ì	&#254;	&thorn;	þ
&#205;	&Iacute;	Í	&#255;	&yuml;	ÿ

Fonte: Manzano e Toledo, 2008

Para finalizar, vamos realizar um exemplo com opções para formatação de fonte. Digite o código ilustrado na Figura 3.8, o qual demonstra algumas opções que empregaremos para caracterizar e estruturar o conteúdo de nossas páginas. O arquivo, depois de salvo e executado, deverá apresentar o resultado que pode ser visualizado na Figura 3.9.

```
<html>
 <head> <title>Formatação de Fonte</title> </head>
 <body>
   <font face="Arial, Verdana" size="3" color="black">
      Texto em fonte arial, tamanho 4 e cor preta. </font> <br />

   <i> Texo em IT&#193;LICO </i> <br />
   <s> Texto riscado </s> <br />
   <tt> Isso &eacute; uma frase com espa&ccedil;amento fixo </tt> <br />

   <font size="5" face="Tahoma">
      CO<sub>2</sub> &eacute; a f&oacute;rmula do
      <b> g&aacute;s carb&ocirc;nico </b> <br /><br />
   </font>

   <b><u> CORES PRIM&#193;RIAS </u></b>  <br />

   <font size="7" face="Courier">
       256<sup>3</sup> &eacute; o n&ordm; de
       combina&ccedil;&otilde;es do sistema
       <font color="#FF0000">R</font>
       <font color="#00FF00">G</font>
       <font color="0000FF">B</font>
   </font>
  </body>
</html>
```

Figura 3.8: Código HTML com recursos de formatação de fonte
Fonte: Autores

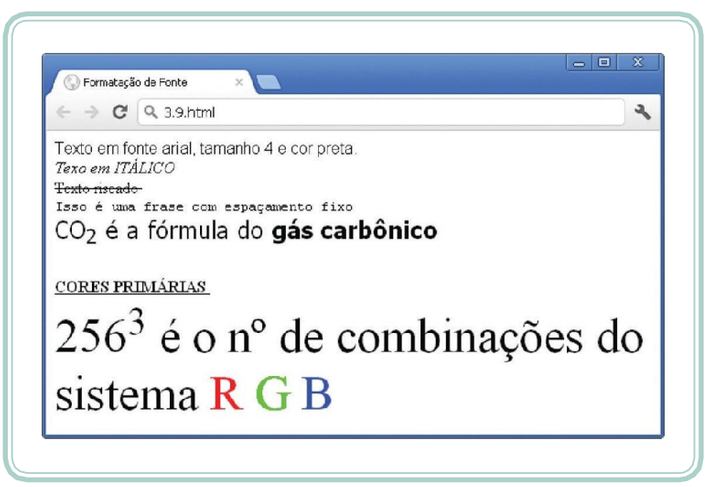


Figura 3.9: Visualização de uma página com recursos de formatação de textos
Fonte: Autores

Como uma página é publicada

Ao longo desta aula, você deve ter produzido, pelo menos, dois ou três arquivos com a estruturação de informações, utilizando-se da linguagem HTML.

Para que uma página seja disponibilizada na internet, será necessário que tenhamos acesso a um servidor *web*. Um servidor *web* é, comumente, um computador com um *software* permanentemente em execução. Esse *software* recebe requisições por meio de URLs e é responsável por identificar o recurso requisitado e entregá-lo ao usuário requisitante. Recursos são arquivos HTML, mas também podem ser imagens ou outros tipos de arquivos.

No momento em que temos acesso a um servidor *web*, vamos nos conectar a ele por meio de um serviço FTP, ou seja, um serviço de transferência de arquivos. A partir de nossa identificação (*login* e senha), o serviço de FTP nos disponibiliza um local (diretório) onde podemos realizar o *upload* de nossas páginas *web*. Podemos entender um servidor *web* como um *software* capaz de entregar os recursos solicitados a partir de uma determinada estrutura de diretórios.

Para que você tenha acesso a um servidor de páginas, precisa entrar em contato com seu provedor de serviços de internet ou com a sua escola ou, ainda, contratar o serviço de uma empresa especializada. Existem servidores de página gratuitos, todavia a grande maioria restringe o envio de documentos HTML, permitindo apenas a edição de documentos padronizados disponibilizados de forma gratuita.

Resumo

Certamente, essa foi, até agora, nossa aula mais atrativa, pois, a partir dela, começamos a colocar a "mão na massa" e dar vida as nossas ideias. Durante a aula, compreendemos a estrutura básica de um documento HTML e as opções para a estruturação de blocos de texto e para a formatação de expressões. Além disso, conversamos um pouco sobre o sistema de cores RGB e o alfabeto de caracteres ISO.

Atividades de aprendizagem

1. O que são metadados e de que forma podemos expressá-los por meio de HTML?

2. Qual é a diferença entre as *tags* **<body>** e **<head>** no contexto de um documento HTML?

3. Por que a expressão "Eu vou viajar de avião" é exibida pelo navegador como "Eu vou viajar de avião"?

4. Codifique um documento HTML para apresentar a seguinte piada tal como está formatada:

Os loucos e o cocô

Dois **loucos** andavam pelo pátio do hospício quando viram um tolete de ~~merda~~ cocô. Então, um deles diz:
— *O que é isso?*
O outro responde:
— *Eu acho que é ~~merda~~ cocô.*
Eles se olham e o primeiro diz:
— *Será? Vamos experimentar para ver se é.*
Os dois experimentaram e o segundo comenta:
— *É ~~merda~~ cocô mesmo,* ***ainda bem que nós não pisamos****.*

5. Escreva uma página HTML para demonstrar as 16 cores básicas nomeadas pela W3C (utilize um fundo diferente de branco para que todas elas possam ser lidas).

6. Escreva, utilizando HTML, as seguintes expressões:

a) área_do_círculo = $\pi \times raio^2$

b) resultado = $5 \div 2 \times 8^2$

c) CH_3CH_2OH

Aula 4 – Ligações de hipertexto e de imagens

Objetivos

Compreender como funcionam as ligações de hipertexto.

Diferenciar ligações internas, externas, absolutas e relativas.

Utilizar elementos externos no corpo da página.

Disponibilizar materiais para *download* a partir de uma página..

4.1 *Hiperlinks*

Hiperlinks, *links* ou, simplesmente, ligações são uma das principais motivações de grande parte do que, hoje, conhecemos como *web*. A *web* em si (ou teia) é, na verdade, formada por infinitas ligações entre documentos em diferentes computadores ou recursos computacionais. A utilização de *links* em HTML é bastante simples e se utiliza de uma única *tag*, a qual possui um nome bastante singelo: **<a>** (*anchor*).

Os *links* podem ser internos – quando referenciam conteúdos do próprio documento – ou externos – quando referenciam recursos do mesmo *site* ou recursos de outros *sites*. Um *link* interno pressupõe a existência de uma marca (âncora) para o local onde o usuário deverá ser submetido no momento em que clicá-lo.

Vamos imaginar uma página na qual disponibilizamos uma receita. Antes de detalhá-la, desejamos incluir um índice, indicando, por exemplo, seções da página como ingredientes, modo de preparo, rendimento e tempo de preparo. Dessa forma, o usuário pode acessar, diretamente, o tempo de preparo sem, necessariamente, ter que rolar toda a página.

A partir desse cenário, precisamos entender dois atributos da *tag* **<a>**. O primeiro deles (o atributo *name*) é utilizado para nomear uma âncora, ou seja, demarcar o local para o qual a ligação irá submeter o usuário. Uma *tag* **<a>**, com o atributo *name*, não tem nenhum efeito visual. No local onde desejamos incluir o *link*, ou seja, no índice da receita, vamos utilizar a mesma *tag* **<a>**,

porém com o atributo *href*, indicando o nome da âncora à qual devemos submeter, precedido do sinal de # (tralha ou sustenido), que indica que o *link* é interno. Vamos demonstrar essa situação por meio de um exemplo. A Figura

ilustra um código HTML de um documento que estrutura as informações de uma receita de pão de queijo nos moldes do que discutimos anteriormente.

```
<html>
  <head> <title>Ancoras (links internos)</title> </head>

  <body>
    <h1>Receita de Pão de Queijo Mineiro</h1>
    <hr />

    <p align="center"> [
              <a href="#ing">Ingredientes</a> |
               <a href="#tempo">Tempo de Preparo</a> |
               <a href="#rend">Rendimento</a> |
               <a href="#preparo">Modo de Preparo</a> ]
    </p>

    <a name="ing"><h3>Ingredientes</h3></a>
        4 copos(americanos)de polvilho doce (500g) <br />
        Sal a gosto (ex. 1 colher de sopa) <br />
        2 copos de (americano)de leite (300ml) <br />
        1 copo (americano) de óleo (150 ml) <br />
        2 ovos grandes ou 3 pequenos <br />
        4 copos (americano) de queijo minas meia cura ralado <br />
        óleo para untar <br />

    <a name="tempo"><h3>Tempo de Preparo</h3></a>
        O tempo de preparo médio é de 30 minutos

    <a name="rend"><h3>Rendimento</h3></a>
        Essa receita rende 5 porções

    <a name="preparo"><h3>Modo de Preparo</h3></a>
        Colocar o polvilho em uma tigela grande<br />
        à parte, aquecer o forno, o leite e o óleo<br />
        Quando ferver escaldar o polvilho com essa mistura, mexer muito
        bem para desfazer pelotinhas<br />
        Deixe esfriar<br />
        Acrescentar os ovos um a um, alternando com o queijo e sovando
        bem após cada adição<br />
        Untar as mãos com óleo, se necessário<br />
        Enrolar bolinhos de 2 (cm) de diâmetro e colocá-los em uma
        assadeira untada<br />
        Levar ao forno médio (180°), pré-quecido<br />
        Assar até ficarem douradinhos<br />
    <hr />
    Para acessar o artigo original clique
    <a href="http://goo.gl/Ap59x">aqui</a>

  </body>
</html>
```

Figura 4.1: Código HTML com utilização de *hiperlinks*
Fonte: Autores

Observe que, logo no início do documento, a *tag* **<a>** é utilizada com o atributo *href* seguido do sinal de # e de um nome (ex.: ing, tempo, rend, preparo). Esses nomes estão presentes na sequência do documento quando a *tag* **<a>** é utilizada com o atributo *name*. Observe, também, que a *tag* **<a>** marca toda a palavra que será transformada em *link* quando o documento HTML for interpretado pelo navegador. Um *link*, normalmente, é apresentado em outra cor (comumente azul) e sublinhado. Quando o cursor do *mouse* passa sobre ele, a figura é substituída por uma "mãozinha" com o indicador levan-

tado. Imaginemos que o usuário clique sobre o *link* produzido com a palavra ingredientes (**Ingredientes**). Nesse caso, a página será posicionada no exato local onde a *tag* **<a name="ing">** estiver declarada.

Agora, note, ao final do documento, a palavra "aqui". Ela está marcada com a *tag* **<a>**, entretanto seu atributo *href* não está indicando uma seção do *site*, mas sim uma URL externa (**http://goo.gl/Ap59x**). Temos, então, um *link* externo que encaminhará a navegação para uma área fora da página atual, carregando outro documento HTML – nesse caso, localizado, inclusive, em outro servidor de páginas.

Imaginemos, agora, que você tenha feito todos os exemplos até aqui demonstrados. Neste caso você deve ter uma relação de arquivos (um para cada exemplo – exemplo1.html, exemplo2.html, exemplo3.html), que tal se construíssemos uma página para interligar estes arquivos? Assim como fizemos no exemplo anterior (Figura 4.1), onde o último *link* vinculava nossa página com uma página externa, podemos também vincular uma página a outras páginas do mesmo *site*. Observe o código da Figura 4.2.

```
<html>
 <head> <title>Ancoras (links exernos)</title> </head>
 <body>
  <h1>Exemplos de links externos</h1>

  <a href="exercicio1.html" target="_blank">Exercício 1</a> <br />
  <a href="exercicio2.html" title="Fontes">Exercício 2</a> <br />
  <a href="http://www.google.com.br">Site do Google</a>  <br />
  <a href="ftp://ftp.registro.br/pub/doc/introducao-dns-dnssec.pdf">DNS</a>

 </body>
</html>
```

Figura 4.2: Código HTML com utilização de *hiperlinks* externos
Fonte: Autores

Na Figura 4.2, os dois primeiros *links* são referências externas para arquivos do mesmo *site* que estão no mesmo diretório do arquivo HTML, o qual está sendo processado. Os dois últimos *links* (*site* do Google e DNS) referem se a informações externas ao *site*, sendo que, no último caso, como a URL identifica um alvo que não é um arquivo HTML, o *browser* poderá se oferecer para fazer um *download* do recurso (introducao-dns-dnssec.pdf).

Ainda sobre *links*, existem duas outras propriedades que merecem ser apresentadas. A primeira delas é a propriedade *title*: utilizada para atribuir uma espécie de descrição sobre o *link*. A descrição de um *link* é apresentada na forma de uma dica (*hint*) quando passamos o *mouse* sobre o mesmo. Na Figura 4.2, podemos observar que o segundo *link* utiliza a propriedade *title* com o valor "Fonte".

A-Z

frame
É um recurso que pode ser utilizado em páginas HTML, criando um ambiente onde mais de uma página pode ser visualizada ao mesmo tempo. Quando uma página utiliza *frames*, a propriedade *target* de um *link* pode indicar em qual das divisões da página o *link* será apresentado.

A outra propriedade que merece destaque é a propriedade *target*: determina a janela específica – ou **frame** – onde o *link* será aberto. Por meio da propriedade *target*, é possível que a página atual interaja com outras janelas. Existem alguns valores reservados para essa propriedade, como "**_blank**", indicando que o *link* deve ser exibido em uma nova janela, e "**_self**", para que o *link* seja exibido na mesma janela (MARCONDES, 2005). Na Figura 4.2, o primeiro *link* utiliza o valor "**_blank**" para o atributo *target*, indicando que o arquivo exercicio1.html será aberto em uma nova janela.

Imagens

Não é só com textos que se faz um documento HTML. Imagens são excelentes recursos para incrementar o *layout* e o visual de uma página ou para exemplificar e demonstrar algo (já dizia o velho provérbio "uma imagem vale mais do que mil palavras"). Antes de falarmos como vincular uma imagem a um documento HTML, precisamos conhecer um pouco sobre os tipos de imagens.

Os *browsers* possuem a capacidade de interpretar imagens do tipo "mapa de *bits*". Nesse tipo de imagem, o arquivo que a armazena é organizado como uma matriz (ou tabela) de pontos. Cada ponto armazena uma codificação específica de cor. Quanto mais colorida for uma imagem, maior será seu tamanho. Os formatos de imagens exibidos na *web* utilizam algum recurso de compactação, ou seja, uma técnica para reduzir o seu tamanho. Os mais comuns serão descritos a seguir. A Figura 4.3 ilustra um comparativo de qualidade.

- **GIF (*Graphics Interchange Format*)** – o formato GIF é, comumente, utilizado para representar ilustrações e sua compactação deve-se ao fato de armazenar, no máximo, 258 cores. O tipo GIF pode utilizar fundo transparente e permite armazenar, em um mesmo arquivo, uma sequência de imagens que, quando exibidas pelo navegador, produzem o efeito de animação (GIF animado).

- **JPG ou JPEG (*Joint Pictures Expert Group*)** – é o tipo de imagem mais comum na internet e, normalmente, é o padrão de salvamento da maioria das máquinas fotográficas digitais e dos *smartphones*. Uma imagem JPG tem seu tamanho reduzido em função de não armazenar os pontos (*pixels*) com cores iguais ou muito parecidas. Quanto maior a taxa de compactação, pior será a qualidade da imagem e menor será seu tamanho.

- **PNG (*Portable Network Graphics*)** – PNG é um formato mais recente de imagem. Oferece o que há de melhor entre os dois formatos anteriores: representação de milhões de cores, transparência e animação. Seu desenvolvimento foi motivado em função de alguns recursos patenteados do formato GIF. Atualmente, é um formato livre apoiado pela W3C.

Figura 4.3: Comparativo dos formatos de imagem GIF, JPEG e PNG
Fonte: http://www.infowester.com/imagens.php

Um detalhe importante na utilização de imagens em documentos HTML é que as mesmas não são "incorporadas" ao arquivo HTML, são apenas referenciadas por estes. Em função disso, no momento em que nosso *site* vai crescendo e aumentando o número de arquivos HTML que passam a utilizar imagens, devemos considerar que os arquivos dessas imagens precisam estar disponíveis, ou seja, precisam ser enviados conjuntamente com os arquivos HTML para o servidor de páginas. No momento em que o *browser* detecta a necessidade de exibição de uma imagem, ele solicita ao servidor de página para que a mesma seja descarregada (*download*) no computador do usuário (por este motivo as imagens devem ter o menor tamanho possível).

Para vincular uma figura em um documento HTML, utilizamos a *tag* **<img>**. Essa é uma *tag* independente, ou seja, que não precisa de outra *tag* para fechá-la. O parâmetro mais importante da *tag* **<img>** é o *src* que especifica o caminho (URL) da imagem a ser exibida. Da mesma forma que a *tag* **<a>**, a *tag* **<img>** pode referenciar um arquivo local ou um recurso disponível em outro servidor. Observe, na Figura 4.4, que três imagens são vinculadas. No primeiro caso, a imagem "tux.gif" é referenciada a partir de um servidor *web* externo (**http://ubuntu-br.org/**), enquanto que, no segundo e no terceiro exemplo, são utilizados arquivos locais (existentes no mesmo diretório do arquivo HTML que está sendo processado pelo *browser*).

```
<html>
  <head> <title>Imagens</title> </head>
  <body>
    <h1>Exemplos de imagens</h1>

     <img src="http://ubuntu-br.org/themes/ubuntu-br/tux.gif" /> <br />
     <img src="exemplo.jpg" alt="Exemplo de imagem JPG"/> <br />
     <img src="foto.png" width="50%" height="50%" border="2"/> <br />

  </body>
</html>
```

Figura 4.4: Código HTML para vinculação de imagens
Fonte: Autores

Além do atributo *src* que especifica a origem (*source*) da imagem, alguns outros atributos merecem destaque. O atributo *alt* (*alternative text*) especifica um texto que poderá ser exibido caso algum problema impeça a apresentação da imagem ou que poderá ser utilizado por leitores de páginas que auxiliam deficientes visuais a navegar pela internet (trata-se de um típico exemplo de acessibilidade). Além disso, o texto do atributo *alt* pode ser visualizado quando posicionamos o *mouse* sobre a imagem.

O tamanho da imagem pode ser redimensionado no momento de sua exibição. Essa opção é habilitada com a utilização dos atributos *width* (largura) e *height* (altura), cujos valores podem ser valores inteiros absolutos (expressos em *pixels*) ou então valores percentuais (considerando o tamanho original da imagem). Opcionalmente, podemos adicionar uma borda em torno da imagem, empregando o atributo *border*, cujos valores são números inteiros que especificam a largura da borda em *pixels*.

Caminhos relativos e absolutos

Ao longo dos conteúdos abordados nesta aula, ficou evidente o conceito de *site*, ou seja, um conjunto de páginas e de outros recursos (figuras, por exemplo) interligados entre si. Você deve estar questionando se todos esses arquivos (documentos HTML e imagens) precisam estar em um mesmo diretório ou se podemos organizá-los utilizando subdiretórios. A resposta para esse questionamento é: podemos e devemos, sim, dividir e organizar os arquivos de um *site* em subdiretórios. As figuras podem, por exemplo, ser agrupadas em uma pasta de imagens; as páginas de um mesmo assunto podem ser separadas por diretórios que as caracterizem.

Vamos imaginar que necessitamos desenvolver uma página para uma disciplina. O professor que nos contratou disse possuir alguns materiais que gostaria de disponibilizar, como exercícios, *slides* das aulas, definições de

trabalhos e provas comentadas. A partir dessa contextualização, poderíamos imaginar uma organização de diretórios e de pastas conforme a Figura 4.5.

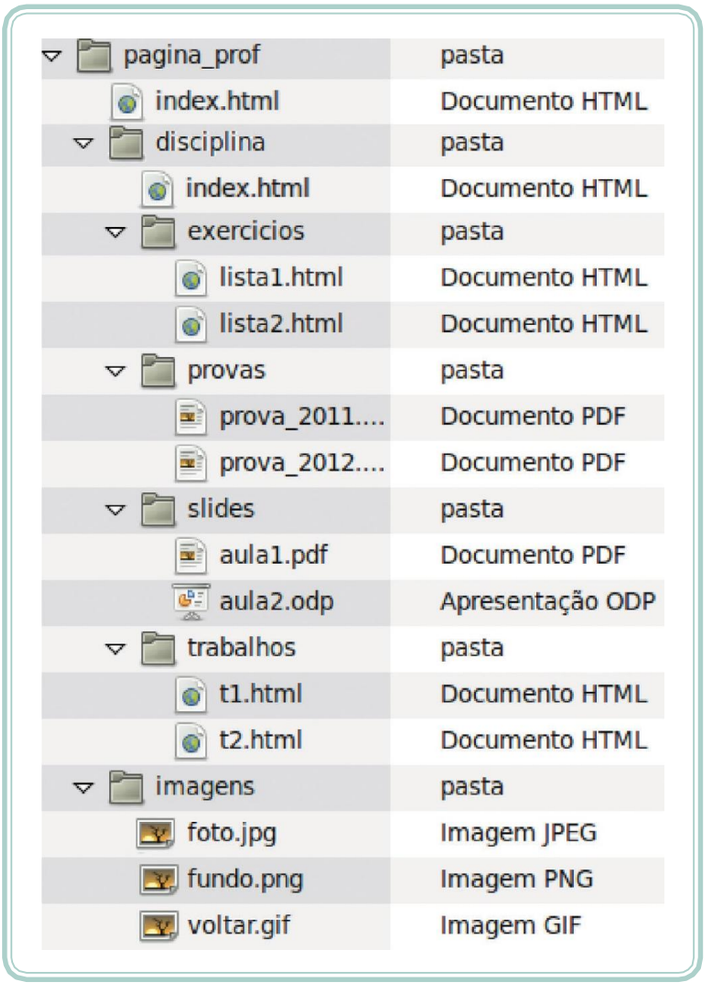


Figura 4.5: Exemplo de organização de diretórios e de arquivos em um *site*
Fonte: Autores

Através da Figura 4.5, podemos observar que há uma pasta principal denominada de "pagina_prof" dentro da qual existe um único arquivo ("index.html") e duas subpastas ("disciplina" e "imagens"). A pasta disciplina, por sua vez, é subdividida em outras quatro pastas (exercícios, provas, *slides* e trabalhos) e contém um único arquivo com o nome de "index.html". Supondo que todo esse material esteja disponível no seguinte endereço "**http://saber.edu.br/pagina_prof**", temos, a partir desse contexto, basicamente duas formas de vincular e de referenciar os arquivos entre si.

A primeira delas é conhecida como caminho ou referência absoluta e consiste em indicar, explicitamente, em cada uma das ligações, o caminho completo

(URL) do recurso que se quer relacionar. Por exemplo, se no arquivo "index.html" da pasta "disciplina" quisermos referenciar a lista de exercícios "lista1.html" da pasta "exercícios", isso poderia ser feito com a *tag* **<a>** da seguinte forma:

```
<a href= "http://saber.edu.br/pagina_prof/disciplina/
exercicios/lista1.html">lista 1</a>
```

O mesmo processo se aplica se quisermos, no arquivo "t1.html" da pasta "trabalhos", utilizar a figura "fundo.png" do diretório "imagens". Nesse caso, utilizaríamos a *tag* **<img>**:

```
<img src= "http://saber.edu.br/pagina_prof/imagens/fundo.png"  />
```

O problema da utilização de referências absolutas acontece quando, por exemplo, resolvemos mudar de domínio (**http://educacao.edu.br**) ou alterar a localização da pasta principal dentro do servidor de páginas. Nesses casos, todos os nossos arquivos HTML precisam ser revisados e, dependendo do tamanho do *site*, poderia levar muito tempo e demandar bastante esforço.

Em função de tal problemática, existe uma possibilidade de se referenciar recursos de forma relativa, ou seja, a partir de sua localização atual. Um caminho relativo pode utilizar dois sinais coringas que servem para representar o diretório atual (sinal de ponto ".") e o diretório anterior (sinal de ponto-ponto ".."). Nesse caso, podemos fazer a vinculação de figuras ou o relacionamento de *links* utilizando, como ponto de partida, o caminho atual do recurso em que o *browser* está interpretado.

Para entender melhor, vamos utilizar os dois exemplos anteriores e adaptá-los para o conceito de caminhos relativos. No primeiro caso, a página em questão é o arquivo "index.html", da pasta "disciplina", que precisa se referir ao arquivo "lista1.html", da subpasta "exercicios". Veja que o arquivo alvo encontra-se hierarquicamente abaixo do documento HTML que estamos editando. Nesse caso, devemos utilizar o sinal de "." para fazer referência ao diretório atual e, a partir dele, referenciar o restante do caminho:

```
<a href= "./exercicios/lista1.html">lista 1</a>
```

No segundo caso, temos uma situação inversa, ou seja, o arquivo "t1.html" deseja incluir uma imagem que está hierarquicamente acima da pasta na qual "t1.html" está salvo. O sinal de ponto-ponto "..", nessa situação, deverá ser utilizado para que possamos nos referir aos diretórios anteriores. A *tag* **<img>**, nesse caso, seria assim codificada:

```
<img src= "../../imagens/fundo.png" />
```

Observe que foi necessário utilizar duas vezes o sinal de ponto-ponto (“..”), pois, a partir do diretório “trabalhos”, precisamos recuar dois diretórios para, só então, nos referirmos à pasta “imagens”.

Destacamos a importância de você observar, atentamente, os nomes dos recursos que deseja referenciar (documentos HTML, imagens, arquivos em geral), respeitando a forma exata como foram escritos (considerando letras maiúsculas e minúsculas). Não temos como saber de antemão o tipo de plataforma na qual o servidor de páginas será executado, visto que há plataformas que diferenciam nomes de arquivos escritos com maiúsculas/minúsculas. Prestar atenção a esse detalhe, desde o início, pode evitar surpresas desagradáveis na hora de publicar seu *site*.

Resumo

Nesta aula, discutimos, inicialmente, a razão de ser da *web*, ou seja, as ligações, os *hiperlinks*. Aprendemos a estruturar ligações dentro de uma mesma página e ligações para páginas externas. Também, iniciamos nosso trabalho com imagens e nos contextualizamos acerca dos tipos disponíveis e de suas características. Por fim, aprendemos duas formas de realizar vinculações: a forma absoluta, na qual a URL é colocada de forma completa, e a relativa, na qual partimos do local onde nosso documento HTML está armazenado e avançamos ou retrocedemos às pastas para relacionar outros recursos do mesmo *site*.

Atividades de aprendizagem

1. De que forma um *site* pode disponibilizar um arquivo para *download* (uma música em formato MP3, por exemplo)?

2. Qual seria o formato mais indicado de imagem para você publicar em um álbum de fotos da sua última viagem de férias?

3. Observando a estrutura de diretórios ilustrada na Figura 4.5 e os conceitos sobre caminhos relativos responda o que se pede:

a) A partir do arquivo “t2.html”, da pasta “trabalhos”, como você faria para incluir um *link* com o nome de “voltar” que retornasse até a página principal do professor (arquivo “índex.html”)?

b) Se você estivesse editando o arquivo “index.html”, da página principal do professor, como você faria para exibir a imagem “foto.jpg”, da pasta “imagens”?

4. Em que situação não se pode evitar a utilização de um caminho absoluto?

5. De que forma podemos indicar para o *browser* que determinado *link* deve ser aberto em uma nova janela?

6. Exemplifique de que forma o usuário poderia clicar sobre uma figura e ser redirecionado para outra página (como colocar um *link* em uma figura)?

Aula 5 – Tabulação de dados

Objetivos

Utilizar listas numeradas ou com marcadores.

Estruturar dados em tabelas.

Listas

A linguagem HTML nos oferece diferentes formas de estruturar textos por meio de tópicos (numerados ou com marcadores). A utilização desse tipo de recurso é bastante simples e consiste, inicialmente, em definir que tipo de lista se deseja utilizar. São três as opções disponíveis: listas com marcadores, listas numeradas e listas de definição. Vejamos, a seguir, alguns exemplos e as *tags* necessárias para representá-las.

Listas com marcadores

O tipo mais comum de lista são as listas com marcadores. No exemplo da quarta aula, quando apresentamos a receita de pão de queijo, poderíamos ter utilizado esse recurso para listar os ingredientes. Uma lista com marcadores é definida utilizando-se a *tag* **<ul>** (*undefinded list*), sendo que, entre as marcas de abertura e fechamento, utilizaremos a *tag* **<li>** (*list*) para indicar os itens da lista.

Opcionalmente, podemos utilizar, na *tag* **<ul>**, o atributo *type* de forma que possamos alterar o tipo de símbolo (marcador) utilizado. Alguns valores aceitos para o atributo *type* são: *circle* (círculo vazio), *disc* (círculo preenchido) e *square* (retângulo). A Figura 5.1 ilustra a utilização da *tag* **<ul>** e seu respectivo efeito.

```
<html>
  <head> <title>Listas com Marcadores</title> </head>
  <body>

     <h3>Ingredientes</h3>
     <ul>
        <li>Açúcar</li>
        <li>Farinha</li>
        <li>Leite</li>
     </ul>


     <h3>Tarefas</h3>
     <ul type="square">
        <li>Arrumar a cama</li>
        <li>Retirar a mesa do café</li>
        <li>Varrer o piso da cozinha</li>
     </ul>

  </body>
</html>
```

Figura 5.1: Documento HTML utilizando listas com marcadores
Fonte: Autores

Listas numeradas

A diferença de uma lista numerada para uma lista com marcadores é essencialmente a *tag* utilizada: **<ol>** (*ordinate list*). Inclusive, a *tag* para representar os itens da lista também se mantém a mesma (**<li>**). No caso de listas numeradas, o atributo *type* pode receber os seguintes valores: "1" (quando a numeração será feita com algarismos arábicos – 1, 2, 3), "I" ou "i" (para se utilizar de algarismos romanos maiúsculos ou minúsculos) e "A" ou "a" (para organizar a lista em ordem alfabética). Opcionalmente, o atributo *start* pode ser utilizado em conjunto com a *tag* **<li>** para indicar o valor inicial a partir do qual a lista deve partir. A Figura 5.2 ilustra a utilização de listas numeradas. Observe que tanto listas ordenadas quanto listas com marcadores podem ser utilizadas umas dentro das outras, dando a ideia de sublistas ou subitens.

```
<html>
  <head> <title>Listas Numeradas</title> </head>
  <body>

     <h3>Questionário</h3>
     <ol type="1">

        <li>Podemos combinar dois tipos de listas?</li>
        <ul>
           <li>Sim</li>
           <li>Não</li>
        </ul>
        <br />
        <li>A sigla HTML significa</li>
        <ol type="a">
           <li>Hypermedia Markup Language</li>
           <li>HyperText Markup Language</li>
           <li>Hypervisor Markup Language</li>
        </ul>

     </ul>

  </body>
</html>
```

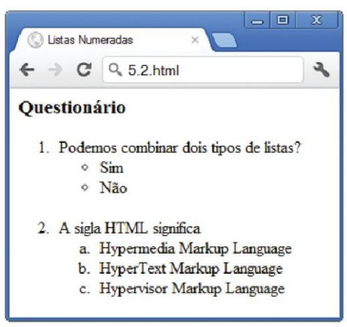


Figura 5.2: Documento HTML utilizando listas numeradas
Fonte: Autores

Listas de definição

Uma lista de definição é, normalmente, utilizada quando precisamos listar um conjunto de itens e defini-los. Esse tipo de lista também é conhecido como lista glossário e sua *tag* principal é **<dl>** (*definition list*). Além desta, outras duas *tags* são utilizadas em conjunto: **<dt>** (*definition term*) e **<dd>** (*definition detail*). Vamos usar uma lista de definição para determinar as 3 *tags*:

<dl>

A *tag* **<dl>** é utilizada para delimitar o início e o término da lista.

<dt>

A *tag* **<dt>** delimita o termo que será, na sequência, definido.

<dd>

A *tag* **<dd>** é utilizada como um item para a definição do termo.

Observe que a lista de definição é composta, basicamente, por duas partes: um termo a ser definido e o detalhamento para o termo. A Figura 5.3 ilustra o mesmo exemplo anterior, contudo utilizando-se da sintaxe HTML. Observe o efeito de cada uma das *tags* na visualização do documento, na própria Figura 5.3.

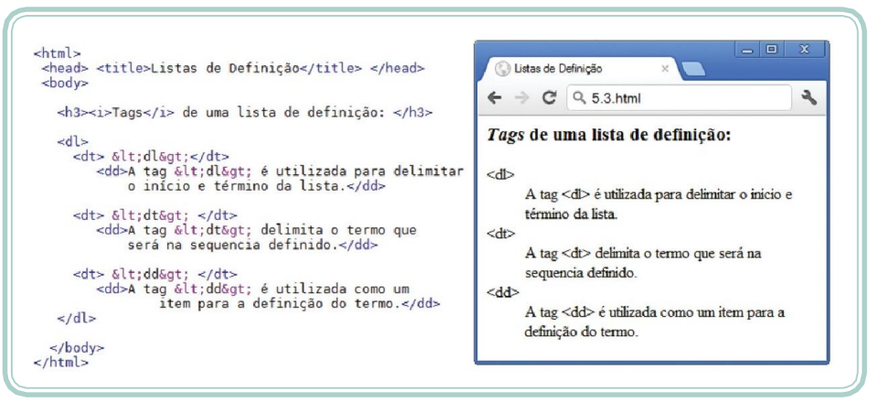


Figura 5.3: Documento HTML utilizando listas de definição
Fonte: Autores

Tabelas

Imaginamos que, neste ponto de nossas aulas, você esteja motivado e interessado em conhecer o que mais podemos fazer com a linguagem de marcação HTML. Um dos recursos disponíveis é a estruturação ou a tabulação de informações em formato de tabela. Uma tabela constitui-se de um quadro, que pode ter um título, dividido em linhas, sendo cada linha dividida em colunas. A

intersecção (ou encontro) de uma linha com uma coluna produz uma unidade de informação a qual, comumente, chamamos de célula.

Tabelas oferecem recursos interessantes para a linguagem de marcação HTML, especialmente no que diz respeito à organização de *layouts*. No entanto, temos que ter consciência e tomar o cuidado de não transformar tudo que temos em tabelas. É importante lembrar que HTML é uma linguagem para estruturação de conteúdo, não necessariamente de formatação ou "maquiagem". Para tal existem recursos apropriados que estudaremos nas aulas seguintes.

Tabelas devem ser usadas quando precisamos tabular dados na forma de uma grade. A *tag* inicial que demarca o início de uma tabela é a *tag* **<table>** e ela é do tipo *container*, ou seja, somente deverá ser fechada quando a estruturação da tabela estiver concluída. Opcionalmente, podemos utilizar a *tag* **<caption>** para definir um título ou descrição a uma tabela. O **<caption>** é exibido antes do quadro de forma centralizada.

O passo seguinte é definir as linhas contidas na tabela. Para tanto, a *tag* **<tr>** (*table row*) será utilizada. Essa também é uma *tag container*, cujo conteúdo será composto por outras *tags*. Cada conjunto de *tags* **<tr>** deverá ser dividido em tantas quantas forem as colunas. Para tal, emprega-se a *tag* **<td>** (*table data*). É entre as marcas de abertura e de fechamento da *tag* **<td>** que os dados serão informados. A Figura 5.4 ilustra a estrutura básica de uma tabela em código HTML para representar um quadro com três linhas e com quatro colunas.

```
<html>
 <head> <title>Tabelas</title> </head>
 <body>

   <table>
       <caption> Tabela 3 x 4 </caption>
       <tr>
           <td> 01 </td> <td> 02 </td> <td> 03 </td> <td> 04 </td>
       </tr>

       <tr>
           <td> 05 </td> <td> 06 </td> <td> 07 </td> <td> 08 </td>
       </tr>

       <tr>
           <td> 09 </td> <td> 10 </td> <td> 11 </td> <td> 12 </td>
       </tr>
   </table>

  </body>
</html>
```

Figura 5.4: Estrutura básica de uma tabela em HTML
Fonte: Autores

A Figura 5.5a representa a interpretação da tabela pelo navegador. Perceba que os dados são informados, exclusivamente, entre as *tags* **<td></td>**. Observe, também, que, por padrão, a tabela não apresenta linhas divisórias entre as células. Para apresentar as linhas divisórias, é necessário utilizar o parâmetro *border* (borda) junto à *tag* **<table>** da seguinte forma: **<table border="1">**. O efeito produzido com a utilização da borda pode ser visualizado na Figura 5.5b.

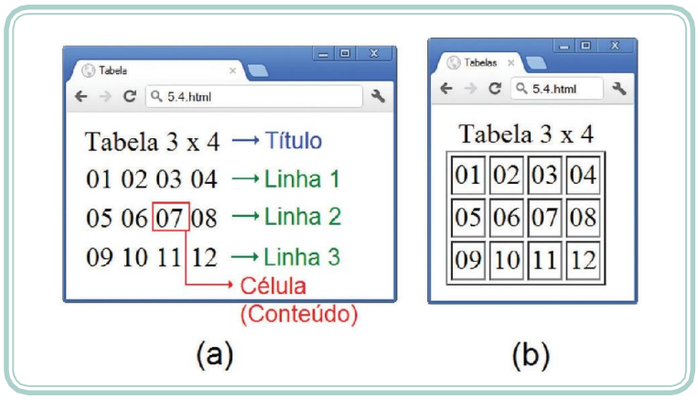


Figura 5.5: Visualização da tabela no navegador
Fonte: Autores

A linguagem HTML permite que tabelas sejam aninhadas, ou seja, dentro de uma célula de conteúdo (*tag* **<td> </td>**), ao invés de informar dados em si, uma nova tabela é declarada. Esse tipo de recurso nos permite explorar algumas situações interessantes. Observe, na Figura 5.6, o efeito visual da inclusão de 8 tabelas, uma dentro das outras.

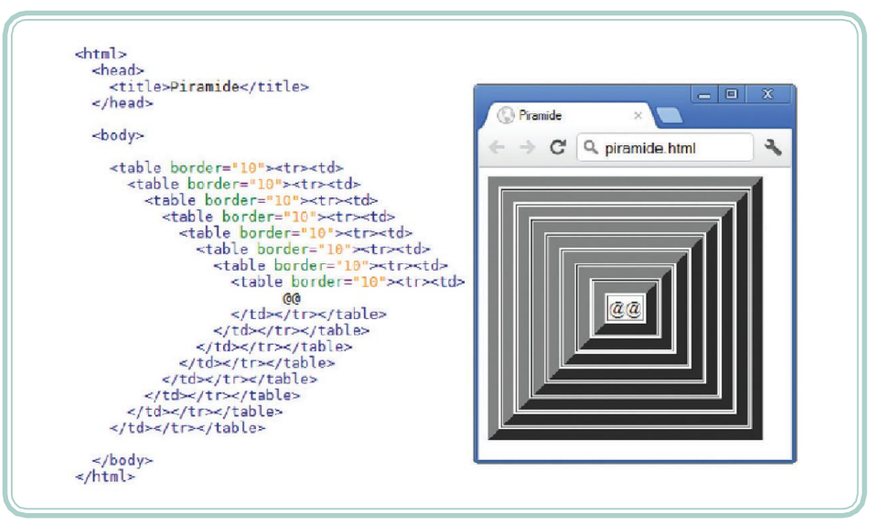


Figura 5.6: Tabelas aninhadas
Fonte: Marcondes, 2005

Vamos tentar imaginar alguns exemplos mais complexos de tabelas, envolvendo, por exemplo, células mescladas (ou unificadas). Para facilitar a compreensão, veja as Figuras 5.7a e 5.7b que ilustram, respectivamente, a fusão de células em uma linha (expressão "Região Sul") e em uma coluna (expressão "Grãos"). A partir dessas duas situações vamos verificar que tipo de atributos e em que *tags* os mesmos devem ser aplicados para que venhamos a conseguir tal efeito.

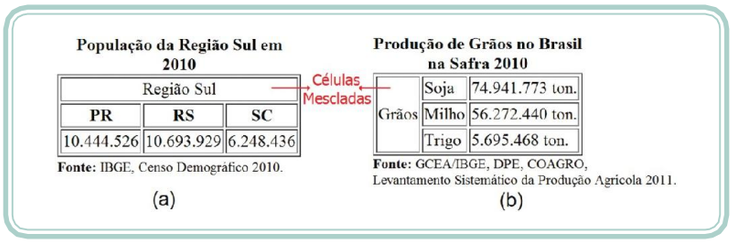


Figura 5.7: Tabelas com células mescladas
Fonte: Autores

Para mesclar células em uma mesma linha (situação representada na Figura 5.7a), será necessário utilizar o parâmetro *colspan* (indicando o número de colunas da linha que será mesclado). No caso da situação representada na Figura 5.7b, na qual temos células mescladas entre linhas, será necessário utilizar o parâmetro *rowspan*, indicando quantas linhas serão mescladas. O código representado na Figura 5.8 ilustra como a fusão de células pode ser feita com os parâmetros *colspan* e *rowspan*.

```
<table border="1">
   <caption> <b>População da Região Sul em 2010</b> </caption>
   <tr align="center"> <td colspan="3"> Região Sul </td>  </tr>
   <tr align="center"> <th> PR </th> <th> RS </th> <th> SC </th> </tr>
   <tr align="right">
       <td>10.444.526</td><td>10.693.929</td><td>6.248.436</td>
   </tr>
</table>
<font size="2"><b>Fonte:</b> IBGE, Censo Demográfico 2010.</font>
<br/>
<table border="1">
   <caption> <b>Produção de Grãos no Brasil na Safra 2010</b> </caption>

   <tr valign="center">
       <td rowspan="3">Grãos</td><td>Soja</td><td>74.941.773 ton.</td>
   </tr>
   <tr> <td> Milho </td> <td> 56.272.440 ton.</td> </tr>
   <tr> <td> Trigo </td> <td> 5.695.468 ton.</td> </tr>
</table>
<font size="2"><b>Fonte:</b> GCEA/IBGE, DPE, COAGRO, <br/>
Levantamento Sistemático da Produção Agrícola 2011.</font>
```

Figura 5.8: Código HTML para mesclar células
Fonte: Autores

Ainda em relação às tabelas, existe um tipo especial de *tag* cuja utilização é a mesma da *tag* **<td>**: trata-se da *tag* **<th>** (*table header*). Na Figura 5.8, podemos observar a utilização da *tag* **<th>** na segunda linha da tabela (no

local onde foram digitadas as siglas dos Estados). Essa *tag* objetiva indicar que aquela célula contém uma informação de cabeçalho para as demais células. Normalmente, é utilizada nas células da primeira linha quando estas contêm cabeçalhos (títulos) das colunas.

À medida que explorarmos alguns recursos mais avançados da linguagem HTML (como é o caso das tabelas), fica mais difícil imaginar que os profissionais da área *web* ficam, constantemente, codificando da forma como temos feito até então. Tabelas é um típico exemplo de elemento no qual um editor de código (um *software* que ofereça facilidades para edição e manipulação das instruções) é muito bem-vindo. Imagine-se organizando a tabela dos caracteres ISO ou mesmo as 16 cores básicas da palheta RGB. Um trabalho que parece simples pode ficar extremamente complexo se não utilizarmos algum tipo de ferramenta.

Se você se interessou pela possibilidade de ter um ajudante na hora de fazer o código HTML mais complexo, faça uma pesquisa em *sites* que distribuem aplicativos como:
http://www.baixaki.com.br/
ou
http://www.superdownloads.com.br
Existem excelentes opções tanto pagas quanto gratuitas. Inclusive, mesmo as versões pagas oferecem a possibilidade de testá-las por um tempo determinado.

Por fim, existem, ainda, alguns atributos das *tags* utilizadas em tabelas que merecem destaque, pois podem possibilitar configurações interessantes. Vejamos os principais atributos:

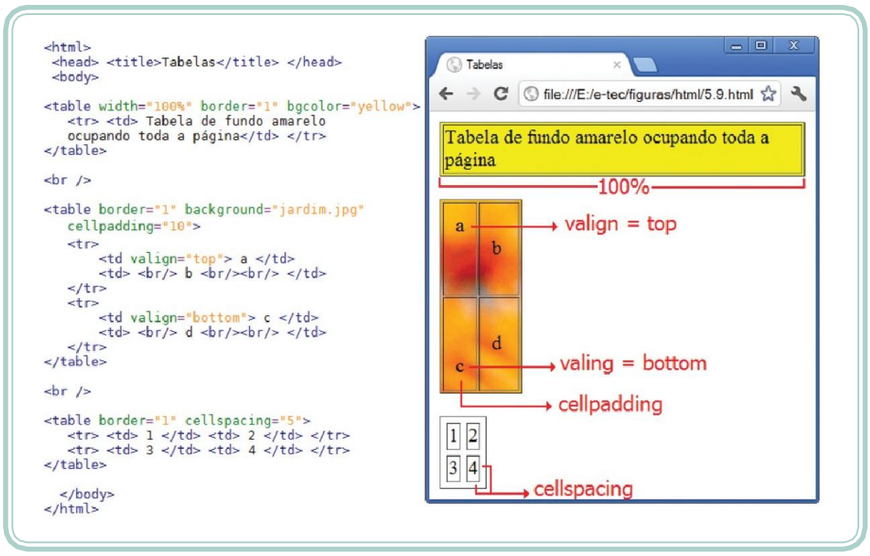
- ***width*** e ***height*** (largura e altura) – podemos especificar, de forma absoluta, o tamanho de uma tabela (*tag* **<table>**) ou de uma célula (*tags* **<td>** ou **<th>**) utilizando os atributos *width* e *height*. Nesse caso, o valor informado para o atributo é um número inteiro indicando o número de pontos (*pixels*). Também é possível informar os valores em forma de percentual. Nesse caso, indicando o tamanho percentual do elemento em relação à janela do *browser*.

- ***align*** e ***valign*** (alinhamento horizontal e vertical) – o alinhamento da tabela em relação à página ou do conteúdo de uma célula no sentido horizontal pode ser alterado em função do atributo *align* (que pode receber os valores *center*, *right* e *left*). Já o alinhamento vertical do conteúdo de uma célula pode ser alterado por meio do atributo *valign* (que recebe os seguintes valores *botton* – inferior, *top* – superior ou *middle* – centro).

- ***bgcolor*** e ***background*** (cor de fundo e imagem de fundo) – o preenchimento do fundo de uma tabela pode ter sua cor alterada com a utilização do atributo *bgcolor* e com a indicação de uma cor. Caso a opção seja por uma imagem, então, devemos utilizar o atributo *background*, indicando o caminho para o arquivo que será utilizado como fundo.

- ***cellpadding*** e ***cellspacing*** (espaçamento entre bordas/conteúdo e espaçamento entre células) – o atributo *cellpading* especifica um número inteiro que indica (em *pixels*) qual é a distância entre o conteúdo de uma célula e sua borda. O atributo *cellspacing* indica a quantidade de *pixels* que deve existir entre uma célula e outra.

A Figura 5.9 ilustra, por meio de exemplos, a utilização e o efeito dos atributos apresentados.




```
<html>
 <head> <title>Tabelas</title> </head>
 <body>

<table width="100%" border="1" bgcolor="yellow">
   <tr> <td> Tabela de fundo amarelo
   ocupando toda a página</td> </tr>
</table>

<br />

<table border="1" background="jardim.jpg"
   cellpadding="10">
   <tr>
       <td valign="top"> a </td>
       <td> <br/> b <br/><br/> </td>
   </tr>
   <tr>
       <td valign="bottom"> c </td>
       <td> <br/> d <br/><br/> </td>
   </tr>
</table>

<br />

<table border="1" cellspacing="5">
   <tr> <td> 1 </td> <td> 2 </td> </tr>
   <tr> <td> 3 </td> <td> 4 </td> </tr>
</table>

  </body>
</html>
```

Figura 5.9: Atributos adicionais para estruturação de tabelas
Fonte: Autores

Resumo

Durante esta aula, aprendemos duas formas para tabular informação: listas e tabelas. No caso das listas, conhecemos os tipos principais e suas peculiaridades. No caso das tabelas, exploramos de que forma as mesmas são organizadas e quais são seus componentes mínimos. A partir desta aula, ficou evidente a necessidade de explorarmos os recursos de algum editor de código, especialmente para estruturar dados em grande volume.

Atividades de aprendizagem

1. Quais são os três tipos de listas que a linguagem HTML nos oferece? Caracterize situações nas quais cada um dos tipos pode ser utilizado.

2. Reescreva o documento HTML com a receita do pão de queijo (Aula 4, Figura 4.1) utilizando uma lista com marcador para os ingredientes e uma lista numerada para as etapas do preparo.

3. De que forma você pode redigir um texto de forma que o mesmo fique dividido em duas colunas? Utilize um gerador de texto aleatório (ex.: **http://www.lipsum.com/feed/html**) para demonstrar como isso pode ser feito.

4. Codifique, em HTML, a estrutura das seguintes tabelas:

a) Lista de produtos (formato 1)

Código	Descrição
P001	*Webcam*
P002	*Mouse* ótico
P003	*Pen drive* 8 Gb

b) Lista de produtos (formato 2)

Produtos	
Código	**Descrição**
P001	*Webcam*
P002	*Mouse* ótico
P003	*Pen drive* 8 Gb

5. Organize uma tabela (sem utilização de bordas) para esquematizar uma galeria de imagens, conforme o modelo a seguir:

Aula 6 – Interação com formulários

Objetivos

Compreender o funcionamento dos formulários eletrônicos.

Diferenciar os métodos de envio de informações.

Identificar e utilizar os principais componentes dos formulários eletrônicos.

Introdução ao uso de formulários *web*

A partir do momento em que começamos a desenvolver aplicações para a internet (não apenas páginas de conteúdo estático), surge a necessidade de permitir, por meio de alguns elementos, a interação. A interação consiste em apresentar algo ao usuário, uma pergunta, por exemplo, e permitir que o mesmo se manifeste (fornecendo uma resposta, por exemplo).

O conteúdo que abordaremos, nesta aula, é de extrema importância para as atividades ligadas à programação para a internet; atividades essas que não são discutidas neste material. No entanto, pretendemos instrumentalizá-lo acerca de como funcionam as interações entre uma página e uma aplicação e de que forma podemos estruturar os elementos que possibilitam tal relação.

Um formulário *web*, como o próprio nome já sinaliza, é um mecanismo pelo qual o *browser* poderá coletar informações. No entanto, por que o *browser* coletaria tais informações? A resposta para essa pergunta está no servidor *web*, pois é lá que existe uma aplicação (também desenvolvida por um profissional da área *web*) aguardando a informação enviada pelo documento HTML para realizar algum processamento e produzir ou apresentar outro documento HTML.

Vamos a um exemplo. Você já deve ter utilizado o serviço de consulta de CEPs (códigos postais) dos Correios. Através desse serviço, um formulário semelhante ao que está ilustrado na Figura 6.1 é apresentado ao usuário. Ao digitar uma informação e clicar sobre o botão “Buscar”, os dados fornecidos serão enviados

para um servidor de página dos Correios que irá procurar a informação em sua base de dados e organizar uma página HTML, apresentando o resultado ao usuário. A Figura 6.2 demonstra a resposta da ação do formulário.

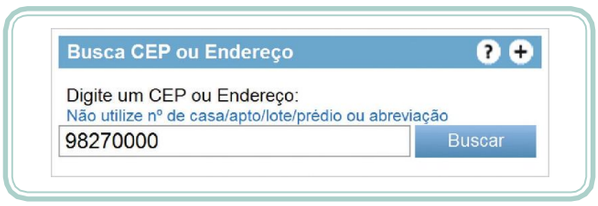


Figura 6.1: Serviço de consulta CEP do *site* dos Correios
Fonte: www.correios.com.br

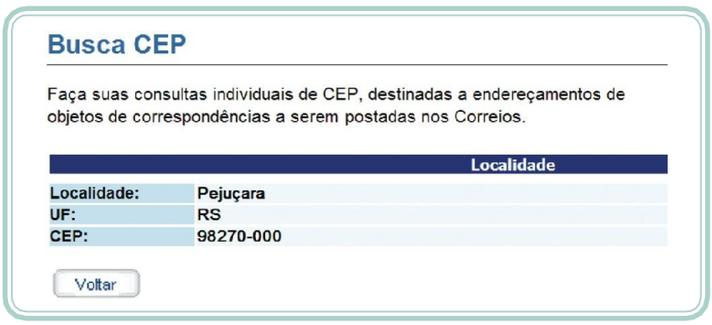


Figura 6.2: Resultado do serviço de busca de CEP do *site* dos Correios
Fonte: www.correios.com.br

Observe que a informação digitada na Figura 6.1 (o valor "98270000") foi submetida para o servidor de páginas quando o botão "Buscar" foi acionado. Após algum tempo de processamento, uma nova página (Figura 6.2) foi apresentada ao usuário. Atente que a informação utilizada no formulário serviu de base para a apresentação do resultado. Podemos imaginar que a página exibida, na Figura 6.2, foi gerada de forma dinâmica em função do que foi informado no formulário da página anterior.

As aplicações para a *web* funcionam, basicamente, dessa forma, ou seja, um formulário coleta informações que são submetidas para uma aplicação *web*. Esta, por sua vez, baseada nos dados recebidos do formulário, decide qual ação tomar. Durante esta aula, discutiremos de que forma podemos empregar a linguagem HTML utilizando formulários para estruturar documentos. Exploraremos quais elementos e recursos podem ser utilizados para coletar, da melhor forma possível, os dados desejados.

A *tag* **<form>** é responsável pela organização de um formulário. Ela é uma *tag* do tipo *container*, ou seja, irá agrupar, dentro de suas marcas de início

e de fim, outros elementos (nesse caso, os campos do formulário). Existem, basicamente, três atributos da *tag* **<form>** que merecem destaque. Vamos analisá-los a partir do exemplo ilustrado na Figura 6.3, cujo objetivo é construir uma página com um formulário para realizar pesquisas na base do Google. Experimente digitar o código da Figura 6.3 e testar sua funcionalidade.

```
<html>
 <head> <title>Formulários</title> </head>
 <body>

   <h2> O que você quer pesquisar? </h2>

   <form action="http://www.google.com/search" name="f" method="get">

      <input type="text" name="q" size="40" />

      <input type="submit" name="pesquisar" value="Pequisar" />

   </form>

  </body>
</html>
```

Figura 6.3: Documento HTML com exemplo de formulário
Fonte: Autores

Na *tag* **<form>**, três atributos são utilizados: *action* (ação), *name* (nome) e *method* (método). O primeiro deles (*action*) indica, para o formulário, qual é a aplicação *web* para onde os dados serão submetidos. Essa é a aplicação *web* da qual falamos anteriormente. No caso desse exemplo, podemos deduzir que os cientistas e os profissionais do Google desenvolveram uma aplicação com o nome de "*search*" a qual está hospedada no servidor disponível em "**http://www.google.com/**". Os dados digitados, nesse formulário, serão enviados para lá.

O segundo atributo (*name*) serve para nomear o formulário (que neste caso tem o nome de "f"). Por que um formulário precisa de um nome? Em um mesmo documento HTML, podemos ter vários formulários e, em determinadas situações, precisamos identificar elementos específicos de um ou de outro formulário. Uma das formas de fazer essa diferenciação é por meio do atributo *name*.

O terceiro atributo, ao qual precisamos dedicar atenção, é conhecido por *method* (que, nesse formulário, contém o valor "*get*"). O atributo *method* informa para o *browser* de que forma os dados coletados pelo formulário serão enviados para a aplicação definida no atributo *action*. Existem, basicamente, dois tipos de métodos de envio:

Experimente digitar diretamente no seu *browser* a seguinte URL http://www.google.com/search?q=gato.
Qual foi o resultado? Veja que a aplicação "*search*" do servidor *web* "http://www.google.com" foi executada com o parâmetro "q" (*query* – consulta) contendo o valor "gato".

- ***get*** – os dados coletados pelo formulário são enviados pela URL da requisição, ficando, inclusive, visíveis ao usuário. É o método mais simples, pois dispensa até mesmo a utilização de um formulário. No entanto, apresenta algumas desvantagens, como o fato de exibir as informações do formulário na URL (no caso de uma senha, isso não seria adequado). Outra desvantagem é que o método *get* não permite o envio de determinados tipos de dados, como arquivos, por exemplo.

- ***post*** – quando utilizamos o método *post*, os dados do formulário são empacotados junto com os dados do protocolo HTTP por meio de uma variável de ambiente que fica disponível para o acesso da aplicação *web*. Neste caso, os dados não ficam visíveis e, por meio do método *post*, é possível enviar arquivos. A desvantagem fica por conta de não poder ser acessado, diretamente, pela URL, necessitando, obrigatoriamente, ser chamado por meio de um formulário *web*.

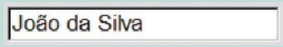
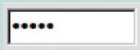
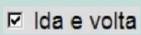
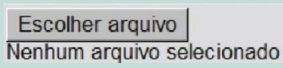
Tipos de controles

O formulário em si (*tag* **<form>**) é, basicamente, um *container* pelo qual configuramos a ação e o método de envio de dados. No entanto, sozinho, ele não é capaz de realizar nenhuma interação com a aplicação *web*. Para que tal interação seja possível, precisamos de alguns elementos que nos possibilitem coletar dados em diferentes formatos. Existem três tipos de *tags* com este propósito: **<input>** (entradas curtas diversas, como caixas de texto, botões de escolha, senhas, arquivos), **<select>** (para entradas predefinidas em listas de seleção, únicas ou múltiplas) e **<textarea>** (para entrada de textos longos). A seguir, detalharemos os principais controles de entrada em formulários, explorando os atributos e opções de visualização que os mesmos nos oferecem.

Input – campo de entrada

A *tag* **<input>** nos permite coletar entradas de texto curtas ou escolhas simples por meio de botões. Há, ainda, duas ações importantes que a *tag* **<input>** indica para o formulário: o envio (submissão) e a reinicialização (ação na qual os dados do formulário voltam ao seu estado padrão). O Quadro 6.1 demonstra alguns códigos HTML de elementos de formulário e o efeito visual que os mesmos produzem. Na sequência, discutiremos os atributos que podem ser configurados para a *tag* **<input>**.

Quadro 6.1: Tipos de componentes da *tag* <input>

Componente/código	Visualização
Caixa de texto `<input type="text" name="nome" value="João da Silva" size=8 maxlength=20 />`	
Caixa de texto com leitura protegida (campo senha) `<input type="password" name="senha" value="12345" size=6 maxlength=6 />`	
Botão de opção única `<input type="radio" name="sexo" value="M" />` `Masculino  ` `<input type="radio" name="sexo" value="F" />` `Feminino`	
Botão de opção múltipla `<input type="checkbox" name="ida_e_volta" value="S" checked/> Ida e volta`	
Entrada oculta (não possui visualização) `<input type="hidden" name="escondido" value="este valor estava escondido" />`	
Arquivo (abre uma janela para que o usuário escolha o arquivo desejado) `<input type="file" name="curriculo" />`	
Botão com ação a ser definida pelo programador `<input type="button" name="ok" value="ok" onClick="alert('ok');" />`	
Botão com ação de reiniciar os dados do formulário `<input type="reset" name="limpar" value="Limpar" />`	
Botão com ação de enviar (submeter) os dados para a ação do formulário `<input type="submit" name="enviar" value="Enviar" />`	
Imagem clicável para submeter os dados para a ação do formulário caixa de texto `<input type="image" src="ok.jpg" value="submit" />`	

Fonte: http://www.cafw.ufsm.br/~bruno/disciplinas/desenvolvimento_web/material/formularios.html

O Quadro 6.1 nos permite explorar a variedade de valores possíveis para o atributo *type* da *tag* **<input>** (*text*, *password*, *radio*, *checkbox*, *hidden*, *file*, *button*, *reset*, *submit* e *image*). Dois desses valores merecem destaque, pois executam ações específicas. Quando um elemento <**input**> contiver o valor *submit* para o atributo *type*, ele será responsável por ativar o envio dos dados do formulário. Funciona como um gatilho. Para que um formulário seja enviado, é necessário que pelo menos um de seus componentes seja do tipo *submit*. O outro valor que merece destaque é o *reset*, cuja ação é "reiniciar" o estado do formulário, atribuindo a todos os seus elementos o valor padrão existente quando este foi apresentado pela primeira vez.

A *tag* **<input>** nos permite realizar outras configurações junto aos componentes visuais por meio de alguns atributos. A seguir, detalhamos os principais (também, podem ser visualizados nos códigos do Quadro 6.1):

- ***name*** – define o nome do campo (nome do objeto para o formulário). É por meio do nome que um componente poderá ser referenciado por alguma linguagem de programação, por exemplo.

- ***value*** – define o valor do campo (quando especificado na definição do elemento, atribui-lhe um valor padrão).

- ***checked*** – em botões de escolha, indica se o mesmo está ou não selecionado.

- ***size*** – em objetos que coletam caracteres, esse atributo especifica o tamanho visível do campo, ou seja, a quantidade de caracteres a ser exibida.

- ***maxlength*** – em campos textuais, define a quantidade máxima de caracteres que o campo poderá coletar.

- ***src*** – se o *type* da *tag* **<input>** for *image*, então o atributo *src* indicará o caminho do arquivo de imagem que será utilizado.

Select – lista de seleção

A *tag* **<select>** nos permite coletar entradas a partir de dados previamente organizados os quais se apresentam ao usuário opções de escolha. Dependendo da configuração do controle, o usuário poderá realizar escolhas simples (indicando um valor de uma lista) ou escolhas múltiplas (nesse caso, escolhendo mais de um valor). É uma *tag* do tipo *container*, que apenas delimita o início e o final da lista de opções e a forma como elas serão apresentadas.

Quem define o tipo de visualização (lista ou lista suspensa) é o atributo *size*. Se contiver o valor "1", ele indicará ao componente a apresentação de uma lista suspensa e, caso contenha um valor maior do que um, indicará a apresentação de uma lista com o número de elementos visíveis igual ao valor de *size* – somente nesse tipo de visualização é possível realizar múltiplas seleções de valores.

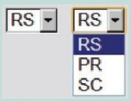
Dentro da *tag* **<select>**, indicaremos por meio da *tag* **<option>** a relação de opções que serão apresentadas. Cada opção poderá indicar, no atributo *value*,

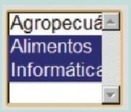
o valor que será submetido à aplicação quando o formulário for enviado. O Quadro 6.2 ilustra a utilização da *tag* **<select>** em suas diferentes formas de apresentação.

Quadro 6.2: Tipos de componentes da *tag* <select>

Componente/código	Visualização

Lista de seleção única
`<select name="sexo" size=2>`
`<option>Masculino </option>`
`<option>Feminino </option>`
`</select>`
	

Lista de seleção suspensa
`<select name="UF" size=1>`
`<option>RS </option>`
`<option>SC </option>`
`<option>PR </option>`
`</select>`
	

Lista de seleção múltipla
`<select name="cursos" size=3 multiple>`
`<option>Agropecuária </option>`
`<option>Alimentos </option>`
`<option>Informática </option>`
`</select>`
	

Fonte: http://www.cafw.ufsm.br/~bruno/disciplinas/desenvolvimento_web/material/formularios.html

Da mesma forma que na *tag* **<input>**, na *tag* **<select>**, o atributo *name* é importante, pois é através dele que, na aplicação *web*, os dados poderão ser identificados. Observe, no Quadro 6.2, como o atributo *size* modifica a aparência da lista (no segundo caso, em que *size* é igual a 1, a lista se apresenta de forma suspensa). As opções, conforme mencionamos anteriormente, devem ser delimitadas pela *tag* **<option>**.

Textarea – área de texto

Há situações nas quais precisamos definir uma área de texto maior para a aquisição de informações de nosso usuário; uma área com, por exemplo, mais de uma linha para digitação (uma notícia ou uma mensagem de *e-mail*, por exemplo). Nesse caso, recomenda-se a utilização da *tag* **<textarea>**, que delimita um conjunto de caracteres que serão apresentados dentro de uma caixa de edição com um número predefinido de linhas (atributo *rows*) e colunas (atributo *cols*). Cabe ressaltar que essa configuração (linhas e colunas) é simplesmente estética e não está relacionada ao tamanho máximo permitido para a área de texto.

De forma a demonstrar a construção de um formulário utilizando as diferentes opções discutidas até esse momento, organizamos, na Figura 6.4, uma pesquisa avançada que emprega o serviço de busca do Google.

```
<html>
 <head> <title>Formulários</title> </head>
 <body>
  <form action="http://www.google.com/search" name="f" method="get">
    <fieldset> <legend> O que você quer pesquisar? </legend>
       <label for="palavras">Palavras que deseja pesquisar:</label>
        <input type="text" name="as_oq" id="palavras" size="40" />

        <h3>ou</h3>

        <label for="exp">Expressão exata:</label> <br />
        <textarea name="as_epq" rows="4" cols="55" id="exp"></textarea>
    </fieldset><br/>
    <fieldset> <legend> Opções </legend>
        <label for="opc">Idioma:</label>
        <select name="lr" size="1" id="opc">
           <option value="lang_pt">Português</option>
           <option value="lang_en">Inglês</option>
           <option value="lang_es">Espanhol</option>
        </select> <br />

        <label for="onde">Pesquisar onde:</label> <br />
        <select name="as_occt" size="4" id="onde">
           <option value="">Em qualquer lugar da página</option>
           <option value="title">No título da página</option>
           <option value="body">No corpo da página</option>
        </select>
    </fieldset>

    <input type="reset" name="limpar" value="Limpar" />
    <input type="submit" name="pesquisar" value="Pequisar" />
  </form>
 </body>
</html>
```

Figura 6.4: Documento HTML com exemplos de utilização de componentes de formulário
Fonte: Autores

Observe que inserimos algumas *tags* que não havíamos utilizado antes, como **<fieldset>** (conjunto de campos), **<legend>** (legenda) e **<label>** (rótulo). As duas primeiras (**<fieldset>** e **<legend>**) são utilizadas em conjunto para congregar um grupo de campos relacionados. No exemplo demonstrado na Figura 6.4, os campos foram divididos em duas áreas ("o que você quer pesquisar?" e "opções"). Na Figura 6.5, podemos observar o efeito visual de tais *tags* (elas criam uma espécie de quadro legendado em torno dos campos que estão demarcando).

A *tag* **<label>** pode ser utilizada para estruturar os rótulos dos campos. Perceba que ela utiliza um atributo denominado *for* (para) – o que indica "sou um rótulo para...". A informação que completa a frase anterior é o valor do atributo *id* que pode ser utilizado em qualquer *tag* HTML e seu objetivo é identificar um elemento, ao longo do documento, de forma única (nenhum elemento no mesmo documento pode ter o mesmo *id*). Através da relação de um rótulo (*label*) com um elemento, ao clicar sobre o rótulo, o usuário é direcionado para o elemento a ele relacionado de forma que possa começar a digitação.

Para saber mais sobre o serviço de busca do Google, acesse a página de pesquisa avançada: http://www.google.com.br/advanced_search e explore as opções disponíveis. Para conhecer os valores e os nomes dos atributos, estude a URL gerada após a execução do serviço.

No caso da Figura 6.4, utilizamos, novamente, o serviço de busca do Google para organizar nosso formulário. Nesse caso, estamos utilizando quatro parâmetros disponibilizados pelo serviço:

- ***as_oq*** – palavras que deseja pesquisar (ou uma ou outra).

- ***as_epq*** – expressão exata que se deseja pesquisar.

- ***lr*** – idioma (com os valores predefinidos *lang_pt* – português, *lang_en*– inglês e *lang_es* – espanhol).

- ***occt*** – local da pesquisa (com os valores predefinidos *title* – título da página, *body* – corpo da página ou vazio para não considerar um local específico).

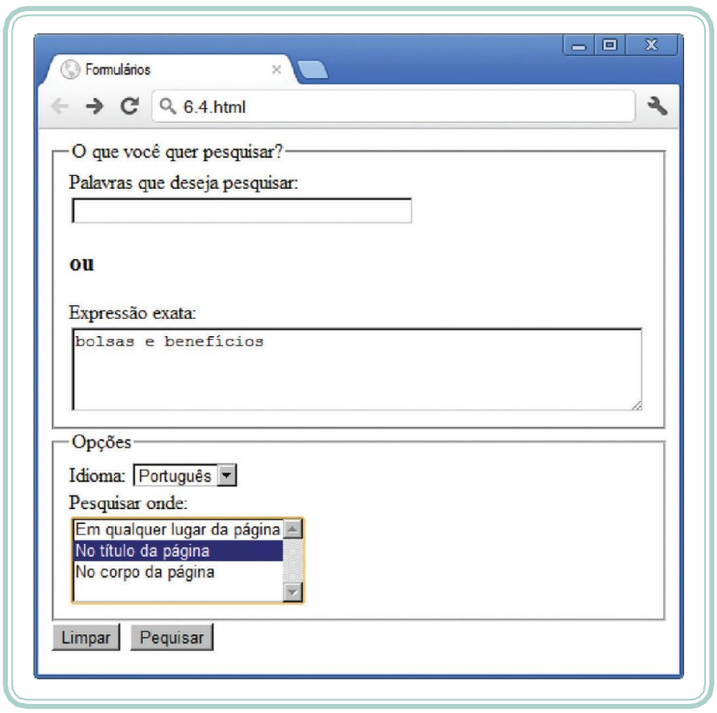


Figura 6.5: Apresentação de um formulário HTML, explorando diferentes elementos
Fonte: Autores

Resumo

Vimos, nesta aula, de que forma um documento HTML pode se comunicar com uma aplicação. Aprendemos como organizar formulários para coletar informações. Ainda, identificamos os principais elementos de um formulário e suas funções e exploramos alguns atributos que permitem alterar a forma como são apresentados e como comportam.

Atividades de aprendizagem

1. Codifique um formulário HTML para coletar um nome de usuário e uma senha (utilize o tipo de elemento adequado para evitar que a senha seja visualizada no momento em que é digitada).

2. Qual a utilidade de um campo de entrada do tipo *submit*?

3. Qual é a aplicação que está sendo chamada no formulário exemplificado na Figura 6.3?

4. Diferencie os métodos de envio de dados *get* e *post*, identificando suas principais vantagens e desvantagens.

5. Codifique um formulário HTML para coletar o nome de um cliente e algumas informações pessoais como: sexo, estado civil e endereço (rua, número, complemento, CEP, cidade, UF).

Aula 7 – Introdução às folhas de estilo em cascata

Objetivos

Conhecer o propósito e as terminologias relacionadas às folhas de estilo.

Compreender como são formadas as regras de folhas de estilo em cascata (CSS).

Vincular folhas de estilo a elementos de documentos HTML.

Explorar alguns recursos de CSS para formatar documentos HTML.

Estrutura e estilo

A separação entre estrutura e estilo, embora pareça óbvio, ainda é algo que nem sempre conseguimos encontrar na internet. A linguagem de marcação que estudamos até aqui – o HTML, deveria ser utilizado apenas para estruturar informações, independente de sua apresentação. Visto que, na medida em que precisamos organizar *layouts* de forma mais elaborada, o HTML não nos oferece as ferramentas adequadas e acabamos por inventar alguma "gambiarra" para alcançar nossos objetivos (quem nunca pensou em fazer um *layout* usando tabelas que atire a primeira pedra).

Folhas de estilo em cascata ou CSS (*Cascading Style Sheet*) é um mecanismo com a função de adicionar estilos (fontes, cores, espaçamentos, bordas, sombras, etc.) aos elementos de documentos codificados em HTML. É provável que você venha a entrar em contato com o termo (X)HTML ou HTML estendido que, na verdade, é uma espécie de HTML com algo mais: o CSS.

As CSS têm por finalidade devolver ao HTML o propósito inicial da linguagem, ou seja, a marcação e a estruturação de conteúdos. Segundo os idealizadores do HTML, não cabe a este fornecer informações ao navegador sobre a apresentação dos elementos – cores de fontes, tamanhos de textos, posicionamento e todo o aspecto visual de um documento não devem ser funções do HTML. Todas as funções de apresentação cabem a CSS, sendo esta a sua finalidade maior. Há uma frase consagrada que circula entre *blogs* e *sites*

especializados em desenvolvimento *web*: “HTML para estruturar e CSS para apresentar” (SILVA, 2007).

Desde a concepção da linguagem HTML, algumas funcionalidades e regras intrínsecas controlavam a apresentação dos documentos. Posteriormente, os *browsers* adicionaram conjuntos de regras que, minimamente, proviam algum estilo aos documentos HTML. No entanto, a primeira proposta de implementação de CSS só ocorreu em 1994 e, somente em 1996, a W3C passou a regulamentar e recomendar de forma oficial o uso de CSS. Para ilustrar as diferenças entre HTML e CSS, leia a conversa demonstrada no Quadro 7.1.

Quadro 7.1: HTML e CSS conversando sobre conteúdo e estilo

HTML	CSS
Saudações, CSS. Estou feliz que você esteja aqui, porque quero esclarecer algumas confusões sobre nós.	É mesmo? Que tipo de confusão?
Muitas pessoas pensam que minhas *tags* dizem ao navegador como exibir o conteúdo. Isso não é verdade! Eu trabalho com a estrutura e não com a apresentação.	Ah, é. Eu não gostaria de ver as pessoas lhe dando crédito pelo meu trabalho.
Bem, você vê como as pessoas podem ficar confusas. Afinal, é possível utilizar HTML sem CSS e, ainda assim, obter uma página com visual decente.	“Decente” soa um pouco exagerado, você não acha? Quero dizer, a maneira como a maioria dos navegadores exibe o HTML puro parece um pouco ordinária. As pessoas precisam entender o poder das CSS e como nós podemos, facilmente, dar um ótimo estilo às páginas *web*.
Ei, eu também sou muito poderoso. Ter seu conteúdo estruturado é muito mais importante do que ter algo bonitinho. O estilo é muito superficial; é a estrutura do conteúdo que interessa.	Fala sério! Sem mim, as páginas *web* seriam muito chatas. E não apenas isso, tire a capacidade de adicionar estilo às suas páginas e ninguém as levará a sério. Tudo parecerá malfeito e nada profissional.
Nossa, que ego, heim? Acho que eu não poderia esperar outra coisa de você. Você está apenas tentando fazer uma declaração de moda com todo esse estilo sobre o qual está falando.	Declaração de moda? Um bom *design* e uma boa apresentação podem ter um efeito enorme na legibilidade e na usabilidade das páginas. E você deveria estar contente porque minhas regras de estilo flexíveis permitem que os *designers* façam qualquer tipo de coisa interessante com seus elementos sem bagunçar sua estrutura.
Certo. Na verdade, somos linguagens totalmente diferentes, o que é bom, porque eu não gostaria de ter nenhum de seus *designers* de estilo bagunçando meus elementos de estrutura.	Não se preocupe, vivemos em universos separados.
É, isso é óbvio para mim todas as vezes que olho para a CSS, essa linguagem alienígena.	Tá... e HTML pode ser considerada uma linguagem? Alguém já viu algo mais desajeitado do que aquelas *tags*?
Milhões de escritores *web* discordariam de você. Eu tenho uma linguagem boa e limpa que se encaixa perfeitamente ao conteúdo.	Dê só uma olhada nas CSS. Elas são elegantes e simples e não são sinais de menor e maior patéticos <torno> <de> <tudo>. <Olha>, <gente>, <eu> <posso> <falar> <como> <o> <sr.> <HTML><!>
Ei, sua estúpida! Já ouviu falar das *tags* de fechamento? Veja bem, não importa aonde você for, está cercada por *tags* **<style>**. Boa sorte ao tentar escapar delas!	HAHAHAHA! Eu vou te mostrar....Sabe por quê? Eu posso escapar...

Fonte: Freeman e Freeman, 2006

Regras, seletores e declaração – sintaxe CSS

A motivação e a conceituação inicial devem ter despertado o seu interesse em conhecer um pouco sobre o funcionamento da linguagem CSS. Antes de começar, vamos discutir alguns conceitos relacionados à organização das regras CSS. A unidade básica de uma folha de estilo é uma regra CSS. Ela é a menor porção de código capaz de produzir um efeito estilizado.

Uma regra CSS é composta por duas partes: o seletor e a declaração. A declaração compreende uma propriedade e um valor. Podemos observar, através da Figura 7.1, a sintaxe para declaração de regras CSS e um exemplo no qual definimos uma declaração para o seletor "p".

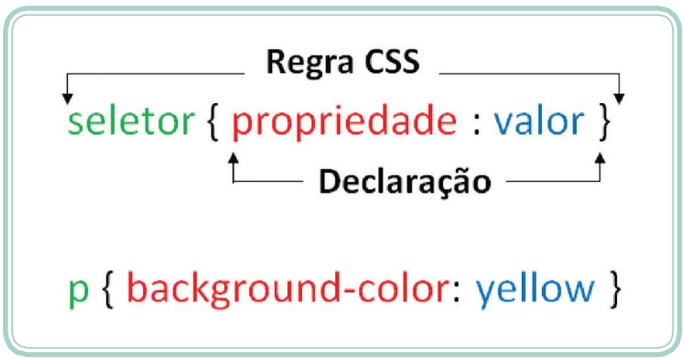


Figura 7.1: Sintaxe da regra CSS
Fonte: Silva, 2007

O seletor é o alvo da regra CSS. Genericamente, pode ser a *tag* do elemento da marcação ou, então, uma entidade capaz de definir com precisão em qual lugar da marcação a regra será aplicada. A declaração, além de determinar os parâmetros de estilização, compreende a propriedade que define qual característica do elemento alvo (determinada pelo seletor) será estilizada e o valor – quantificação ou qualificação da propriedade – (SILVA, 2007). Na Figura 7.1, podemos visualizar um exemplo de aplicação da *tag* **<p>** (parágrafo) um estilo onde a cor de fundo (*backgroud color*) será definida como amarela.

As regras CSS podem conter múltiplas declarações, bastando, para isso, separá-las pelo sinal ";" (ponto-e-vírgula). Quando o valor de uma propriedade for uma palavra composta (separada por espaços) – como o nome da fonte Times New Roman –, devemos informá-lo entre aspas (simples ou duplas). A sintaxe das regras CSS não é *case sensitive*, ou seja, não diferencia letras maiúsculas de letras minúsculas.

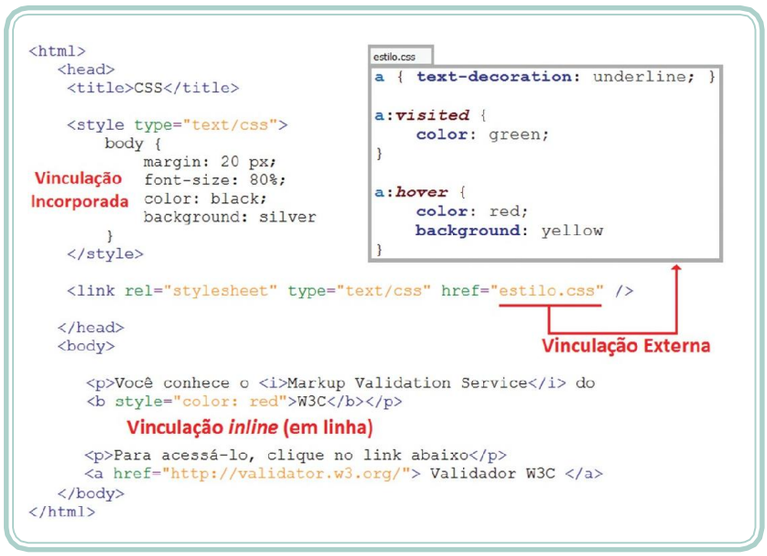
Acoplando CSS em HTML

Existem diferentes formas de se vincular uma folha de estilos a um documento HTML. A primeira delas, conhecida como vinculação *in-line* (em linha), consiste no emprego do atributo *style* diretamente sobre a *tag* HTML que se deseja estilizar. Embora pareça prático em um primeiro momento, esse método torna a manutenção do código bastante difícil e também retira um dos maiores poderes das folha de estilo: o controle centralizado da apresentação. Na Figura 7.2, podemos observar a utilização de CSS, de forma *in-line*, na estilização da *tag* **<b>**, alterando a cor para vermelho.

Outra maneira de utilizarmos CSS em um documento HTML é de forma incorporada. Para tanto, utilizamo-nos da *tag* **<style>** que deve ser declarada na seção de cabeçalho do documento, pois está indicando configurações que o documento, como um todo, irá adotar. Essa forma de vinculação apresenta uma vantagem em relação à anterior: as regras estão centralizadas em um único ponto do documento. No entanto, ela não permite o compartilhamento de regras entre os documentos de um mesmo *site*. A Figura 7.2 ilustra a utilização de CSS, de forma incorporada, alterando um conjunto de propriedades da *tag* **<body>**.

A forma mais eficiente de utilização de CSS, em documentos HTML, é a interligação externa: vinculamos ao documento HTML um arquivo externo (normalmente com extensão CSS) e tal arquivo pode ser compartilhado por diferentes documentos do mesmo *site* (ou de outros). A vinculação externa de uma folha de estilos pode ser feita por meio da *tag* **<link>**, que é utilizada para associar um arquivo externo ao documento HTML em questão.

A vinculação de um arquivo externo – por se tratar de uma configuração para o documento – também deve ser feita na seção de cabeçalho (*tag* **<head>**). Na Figura 7.2, podemos observar a inclusão do arquivo “estilo.css” por meio da *tag* **<link>**. Na mesma figura, está demonstrado o conteúdo do arquivo “estilo.css”, que redefine o formato da *tag* **<a>** aplicando efeitos diferentes em *links* que estão recebendo o *mouse* (evento *hover*) e em *links* já visitados (evento *visited*).




```
<html>
   <head>
    <title>CSS</title>

    <style type="text/css">
        body {
            margin: 20 px;
            font-size: 80%;
            color: black;
            background: silver
        }
    </style>

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

   </head>
   <body>

      <p>Você conhece o <i>Markup Validation Service</i> do
      <b style="color: red">W3C</b></p>

      <p>Para acessá-lo, clique no link abaixo</p>
      <a href="http://validator.w3.org/"> Validador W3C </a>
   </body>
</html>
```

Figura 7.2: Utilização de CSS no documento HTML
Fonte: Autores

Classes, identificadores e pseudoclasses

A utilização de regras CSS não está restrita aos seletores do tipo *tags* HTML. Existem três tipos de seletores que merecem nossa atenção: classes (*class*), identificadores (*id*) e pseudoclasses. Um seletor do tipo classe permite-nos alguns recursos interessantes. Por meio dele, podemos aplicar um mesmo estilo a diferentes elementos e também indicar que um mesmo elemento obedece a diferentes estilos.

Para definir um seletor do tipo *class*, devemos iniciar a especificação do mesmo com o sinal de “.” (ponto final). A utilização de tal seletor, em elementos do documento HTML, ocorre por meio do atributo *class* que é aceito pela maioria das *tags* HTML. Na Figura 7.3, o seletor do tipo *class* está sendo utilizado no subtítulo e na mensagem “Obrigado e volte sempre!”. Perceba que, nesta última, duas classes estão sendo utilizadas (destacado e caixa alta). Para que mais de uma classe seja utilizada sobre o mesmo elemento, basta separar seus nomes por um espaço em branco.

Um seletor do tipo identificador é utilizado com o atributo *id* o qual identifica, de forma única e exclusiva, um determinado elemento no documento HTML. A declaração de um seletor do tipo *id* inicia-se com o sinal de “#” (tralha ou sustenido) seguido do seu nome. A especificidade de um seletor *id* é maior do que a de um seletor *class*. Dessa forma, se, em um mesmo elemento HTML, forem utilizados os dois atributos, serão aplicadas as regras de estilo

do identificador. Na Figura 7.3, o seletor *id* é utilizado na *tag* **<p>** que delimita a expressão "Essa é a mensagem que preparei".

Há, também, um tipo especial de seletor denominado de pseudoclasse, que é utilizado para alcançar elementos que não são especificados por meio de uma marca. Por exemplo, como poderíamos aplicar efeitos diferentes em *links* ainda não visitados, já visitados ou aquele em que o usuário está com o cursor do *mouse* em cima? A resposta são as pseudoclasses. Sua declaração pode ser visualizada, na Figura 7.2, junto ao arquivo "estilo.css". Observe que duas pseudoclasses estão sendo utilizadas para oferecer efeitos diferenciados para a *tag* **<a>:** *visited* (que permite estilizar um *link* já visitado) e *hover* (que permite estilizar um *link* no momento em que o cursor do *mouse* está sobre ele).

As pseudoclasses são predefinidas pela linguagem CSS. Para conhecer a lista completa, acesse: http://www.w3schools.com/css/css_pseudo_classes.asp

```
<html>
   <head>
    <title>CSS</title>
    <style type="text/css">
       .destacado {
        color: blue;
        font-style: italic;
       }

       #mensagem {
        align: center;
        color: green
       }

       .caixaAlta {
        text-transform: uppercase
       }
    </style>
   </head>
   <body>
      <h1>Este é o título</h1>
      <h2 class="destacado"> Este é o subtítulo</h2>
      <p id="mensagem"> Essa é a mensagem que preparei </p>
      <font class="destacado caixaAlta">Obrigado e volte sempre!</font>
   </body>
</html>
```

Figura 7.3: Utilização de classes e identificadores
Fonte: Autores

As possibilidades que a linguagem CSS nos oferece não terminam por aqui. Existem, ainda, outras formas de organização das regras utilizando recursos mais avançados (seletores compostos, seletores de atributos, seletores descendentes, seletores filhos, seletores irmãos e pseudoelementos). A melhor forma de conhecer mais sobre a linguagem CSS é a partir de problemas práticos ou situações reais em que você deseja fazer algo, mas não sabe como. A internet, nesse caso, será a melhor e a mais rápida referência.

Propriedades

As propriedades, conforme já estudamos, formam, em conjunto com seus valores, a unidade mais básica de definição de uma regra. Propriedades são como palavras reservadas da sintaxe CSS e são predefinidas (MARCONDES, 2005). A lista de propriedades da linguagem é bastante extensa; vejamos algumas:

Fonte e efeitos sobre o texto

- ***font-family*** – nome da fonte (o tipo de letra).
- ***font-style*** – estilo itálico (*italic*) ou oblíquo (*oblique*).
- ***font-weight*** – largura da fonte (ex.: negrito – *bold*).
- ***text-decoration*** – especifica uma decoração para o texto, como sublinhado (*underline*), riscado (*line-through*), sublinhado invertido (*overline*).
- ***text-transform*** – define se o texto será apresentado em letras maiúsculas (*uppercase*), minúsculas (*lowercase*), com as iniciais em maiúsculas (*capitalize*).
- ***font-size*** – especifica o tamanho da fonte (veja mais sobre unidades de medida na seção seguinte).
- ***letter-spacing*** – espaçamento entre letras.
- ***word-spacing*** – espaçamento entre palavras.

Cores e fundos

- ***color*** – cor do texto. Pode ser especificada em RGB hexadecimal (ex.: #FF00CC), RGB decimal (ex.: RGB [255, 0, 100]), RGB percentual (ex.: RGB [100%, 0%, 50%]) ou por meio do nome da cor (ex.: *red*).
- ***background-color*** – cor de fundo.
- ***background-image*** – imagem de fundo. Em CSS, para especificar um arquivo externo, como uma imagem, utilizamos a função URL (ex.: URL "./imagens/fundo.jpg").
- ***background-repeat*** – define como a imagem de fundo será repetida: *repeat* (repete tanto verticalmente quanto horizontalmente), *no-repeat*

(exibe a imagem apenas uma vez), *repeat-x* (repetição apenas no eixo x – horizontal), *repeat-y* (repetição apenas no eixo y – vertical).

- ***background-attachment*** – determina o comportamento da imagem em relação à rolagem da tela (página ou camada). Pode ser: *fixed* (a imagem não acompanha a rolagem) ou *scrool* (a imagem movimenta-se de acordo com a rolagem da tela).

- ***background-position*** – posicionamento da apresentação da imagem. Essa propriedade precisa de dois valores: o primeiro para identificar o posicionamento vertical (*top*, *bottom* e *center*) e o segundo para identificar o posicionamento horizontal (*left*, *right* e *center*).

Parágrafos

- ***text-align*** – alinhamento do texto: *left* (à esquerda), *right* (à direita), *center* (centralizado) e *justify* (justificado).

- ***text-indent*** – recuo de primeira linha.

- ***line-height*** – espaçamento entre linhas.

Bordas

- ***border-width*** – espessura da borda: *thin* (fina), *medium* (média), *thick* (grossa). Também, é possível acessar propriedades específicas de cada uma das bordas. Ex.: *border-left-width* – altera a espessura da borda da esquerda (outras opções são: *border-right*, *border-top* e *border-bottom*).

Para saber mais sobre a lista completa de propriedades, acesse o *site* do W3C: http://www.w3.org/TR/CSS/

- ***border-color*** – cor da borda.

- ***border-style*** – estilo ou decoração: *hidden*, (oculta), *solid* (linha simples sem efeito), *ridge* (3D), double (dupla), *dotted* (linha pontilhada), *dashed* (linha tracejada), *outlet* (3D em alto-relevo), *inset* (3D em baixo relevo), *none* (invisível).

Unidades de medida

As propriedades que listamos, na seção anterior, podem ser qualificadas (quando seu valor é uma informação textual) ou quantificadas (quando seu valor é, por exemplo, algo mensurável ou algo que se utiliza de valores). O tamanho de uma fonte, a espessura de uma borda ou as margens de um

parágrafo são exemplos de propriedades que podem ser quantificadas. O CSS trabalha com diferentes unidades de medida. É importante que as conheçamos para ajustar os valores de nossas propriedades de forma adequada.

As unidades de medida utilizadas em CSS se dividem em absolutas e relativas. As unidades absolutas são aquelas que não dependem de nenhum outro valor de referência, ou seja, são definitivas (SILVA, 2007).

- **in** – polegada.
- **cm** – centímetro.
- **mm** – milímetro.
- **pt** – ponto (1 pt = 1/72 polegadas).
- **pc** – pica (1 pc = 12 pt).

Unidades de medidas relativas são aquelas que se utilizam de um valor de referência anteriormente definido. São de três tipos:

- **em** – calculada em relação ao tamanho de fonte predefinido (1 em é igual ao tamanho da fonte definido para o elemento a ser estilizado).
- **ex** – calculada em função da altura da letra xis (x) minúscula da fonte adotada.
- **px** – refere-se a um *pixel*. No entanto, como diferentes dispositivos utilizam diferentes resoluções de tela, o tamanho do *pixel* pode variar de dispositivo para dispositivo.

A Figura 7.4 explora a utilização de algumas unidades de medida de CSS. Observemos seus efeitos:

```
<html>
 <head>
  <title>CSS</title>

  <style type="text/css">
    .normal { font-size:1em }
    .dobro  { font-size:2em }
    .borda_G { border-width:3mm; border-style:solid }
    .borda_P { border-width:2px; border-style:solid }
    p { letter-spacing:2ex }

  </style>

 </head>
 <body>
   <font size="2">
      <font class="normal"> Fonte normal </font>
      <font class="dobro"> Fonte grande </font>
   </font>

   <p class="dobro borda_G">  Texto espassado com
   borda grossa </p>
   <p class="borda_P"> Texto normal com borda fina </p>
 </body>
</html>
```

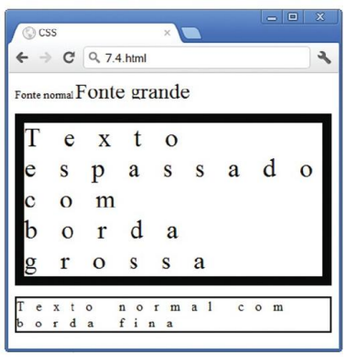


Figura 7.4: Exemplos de unidades de medida em CSS
Fonte: Autores

Resumo

Durante esta aula, apresentamos o recurso de folhas de estilo em cascata, utilizado para dar forma e visual aos nossos documentos HTML. Conhecemos a linguagem CSS e compreendemos de que forma as regras de estilo devem ser organizadas. Também, utilizamos algumas propriedades e explorarmos as unidades de medida utilizadas para quantificá-las.

Atividades de aprendizagem

1. Você deve ter percebido que muitos atributos ou mesmo *tags* da linguagem HTML acabam se sobrepondo aos efeitos de propriedades em CSS. Por que é interessante que a estrutura e o formato estejam separados? Há alguma vantagem evidente na utilização de CSS?

2. Descreva o que são regras, seletores, propriedades e valores, considerando os conceitos discutidos ao longo desta aula.

3. Cite exemplos e explique o que são propriedades qualitativas e propriedades quantitativas.

4. Utilizando sintaxe CSS, reescreva o quarto exercício da Aula 3 sobre HTML (“Os Loucos e o Cocô”).

5. As unidades de medida de propriedades quantitativas podem ser absolutas ou relativas. Diferencie-as e cite situações em que uma opção seria mais indicada que a outra.

Aula 8 – Camadas

Objetivos

Compreender o modelo de camadas.

Organizar *layouts*, dividindo-os em seções lógicas.

Conhecer e utilizar propriedades de regras de estilo para camadas.

O modelo de camadas (*box model*)

Em nossa quinta aula, quando falamos sobre tabulação de dados, é possível que você tenha pensado em utilizar uma tabela sem bordas para organizar um *layout*. Afinal, seria muito natural imaginar, por exemplo, uma linha inicial para cabeçalho (talvez dividida em duas células – uma para o logotipo e outra para algum tipo de *banner*), uma segunda linha para uma barra de navegação ou menu institucional. O conteúdo em si poderia ser apresentado na terceira linha (talvez dividida em duas colunas para separar o menu do conteúdo) e, por fim, uma linha final com o rodapé da página. A Figura 8.1 ilustra essa situação.

```
<html>
 <head>
  <title>Layout com Tabela</title> </head>
 <body>
   <table width="100%">
     <tr>
      <td width="200" bgcolor="yellow"> Logotipo </td>
      <td width="700" bgcolor="magenta"> Banner </td>
     </tr>
     <tr>
      <td colspan="2" align="center" bgcolor="red">
           Barra de Navegação </td>
     </tr>
     <tr>
      <td bgcolor="olive"> Menu </td>
      <td bgcolor="white" align="center">
           <h1>Área de Conteúdo</h1> </td>
     </tr>
     <tr>
      <td colspan="2" align="right" bgcolor="cyan" >
           Rodapé </td>
     </tr>
   </table>
 </body>
</html>
```

Figura 8.1: Organização de *layout* utilizando uma tabela
Fonte: Autores

Pensando dessa forma, estamos "desvirtuando" o propósito inicial de uma linguagem de marcação (estruturar o conteúdo) e estamos utilizando seus recursos para dar forma (visual) ao conteúdo. Usar tabela para "fatiar" *layouts*

é mais comum do que você imagina. No entanto, é uma prática completamente equivocada. Você deve estar se perguntando: “Ok! Mas, como posso ‘fatiar’ um *layout* sem usar tabelas?” Essa é, justamente, a resposta que ajudaremos você a encontrar ao longo desta aula.

Inicialmente, vamos abordar o modelo de camadas ou modelo de caixas (*box model*). Uma caixa ou um quadro é uma área lógica do documento HTML com características próprias. Uma caixa funciona como um *container*, ou seja, permite que existam, dentro de si, outras caixas (caixas descendentes). Por meio da estruturação de blocos lógicos, é possível organizar a apresentação de diferentes elementos ao longo do documento HTML. Para compreender melhor como isso pode ser feito, vamos estudar a anatomia de uma caixa/quadro. A Figura 8.2 ilustra suas diferentes propriedades:

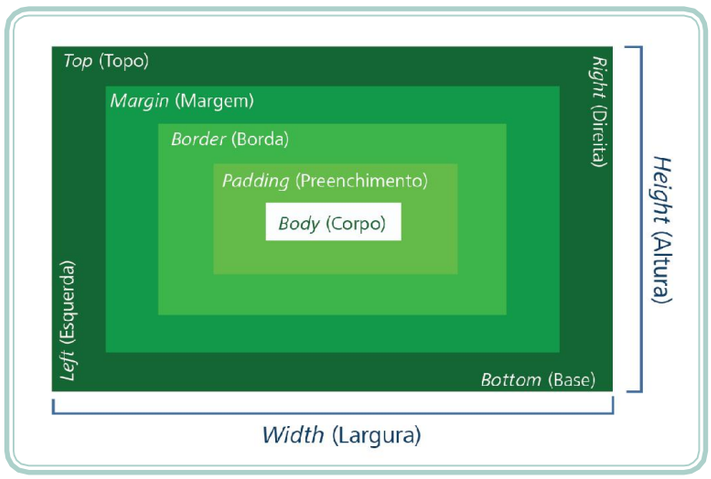


Figura 8.2: Estrutura de um quadro
Fonte: Manzano e Toledo, 2008

Perceba a existência de diferentes propriedades que regulam o comportamento visual de uma caixa. A área de conteúdo ou corpo está ao centro da figura e ela tem seus atributos de altura e largura regulados pelas propriedades *height* e *width*, respectivamente. O corpo é seguido por uma área de preenchimento – cuja espessura pode ser definida pela propriedade *padding* – e, ao redor dessa área, podemos definir bordas (*border*). Além das bordas, ainda há uma área denominada de margem (sempre transparente) – cuja espessura pode ser definida pela propriedade *margin*.

As diferentes propriedades que regulam um quadro podem ter suas dimensões, formatos e estilos especificados para cada uma das orientações (topo, direita, base e esquerda). Essa especificação pode ser feita com um único

comando informando todos os valores na ordem *top*, *right*, *bottom*, *left* (sentido horário) ou individualmente, conforme as propriedades abaixo:

- ***margin*** (margem) – *margin-top* (margem superior), *margin-right* (margem da direita), *margin-bottom* (margem inferior) e *margin-left* (margem da esquerda).

- ***padding*** (enchimento ou espessura) – *padding-top* (enchimento superior), *padding-right* (enchimento da direita), *padding-bottom* (enchimento inferior) e *padding-left* (enchimento da esquerda).

- ***border*** (borda) – *border-top* (borda superior), *border-right* (borda da direita), *border-bottom* (borda inferior) e *border-left* (borda da esquerda).

Marcações lógicas

Em nossas aulas sobre HTML, deixamos de fora duas *tags* que, se fossem explicadas naquele momento, não fariam sentido. Tratam-se das *tags* ** ** (*division*) e ****. Essas *tags* são utilizadas para realizar uma marcação lógica, pois não possuem efeito visual algum. Suas funções são: dar significado ao conteúdo marcado. A diferença entre ambas fica no fato de que a *tag* ** ** é utilizada para dividir áreas ou blocos – isso significa que uma divisão pressupõe uma quebra antes e depois da área dividida. Já a *tag* **** é utilizada de forma *in-line* (em linha), sem quebras de texto.

A partir dessas duas *tags*, podemos associar a elas regras CSS, por meio de classes ou identificadores, e alterar as propriedades do modelo de caixa. Vejamos, através da Figura 8.3, um exemplo de utilização de ** ** e **** para demarcar áreas da página onde são aplicados estilos específicos.

Figura 8.3: Marcações lógicas – *div* e *span*
Fonte: Autores

Fatiando um *layout*

Agora que conhecemos um pouco sobre o modelo de caixa e sobre marcações lógicas, vamos tentar organizar um *layout* semelhante àquele apresentado na Figura 8.1. Todavia, buscaremos separar o visual do conteúdo e organizar um *layout* ***tableless*** que explore, adequadamente, os recursos do HTML e do CSS.

A-Z

tableless
Tableless ou "menos tabelas" é uma filosofia ou método de desenvolvimento de páginas difundido pela W3C que prega a ideia de que cada *tag* deve ser utilizada para o propósito ao qual foi criado. Por meio de tal filosofia, altamente difundida, prega-se que a *tag* **<table>** deve ser utilizada para tabulação de dados e não para organização de *layouts*.

Imaginemos que nosso *site* terá quatro áreas distintas: um cabeçalho, uma barra de navegação, o corpo (onde estará o conteúdo) e um rodapé. Cada área, por sua vez, poderá ter subdivisões. O cabeçalho se divide em dois espaços: um para o logotipo (no canto superior esquerdo) e outro espaço maior para um *banner* (à direita do logotipo). A barra de navegação oferece um menu de acesso rápido. A área do corpo será dividida em um menu (à esquerda) e uma área de conteúdo (ao centro). Na base da página, estarão as informações de rodapé. Inicialmente, sem preocupação com a apresentação, vamos tentar organizar as informações estruturando-as em um documento HTML. A Figura 8.4 demonstra a estruturação dos conteúdos utilizando linguagem HTML e identificadores de estilo.

```
<html>
   <head>
    <title>CSS</title>
   </head>
   <body>
      <div id="cabecalho">
        <div id="logo" > <img src="logo.png" /> </div>
        <div id="banner"> <h3> Banner </h3> </div>
      </div>                                                    Cabeçalho

      <div id="barra_navegacao"> Barra de Navegação
         [ <a href="#">Inicio</a> | <a href="#">Meio</a> | <a href="#">Fim</a> ]
      </div>

      <div id="corpo">
        <div id="menu">
            <ul>
               <li>Item de Menu 1</li>
               <li>Item de Menu 2</li>
                  <ul> <li>Submenu 2.1</li> </ul>
               <li>Item de Menu 3</li>
            </ul>
        </div>                                                  Corpo

        <div id="conteudo">
            <h1 align="center">Lorem ipsum</h1>
            <p>Lorem ipsum dolor sit amet ...</p>
            <p>Cras accumsan lacus ut .... </p>
            <p>In aliquam, mauris in placerat ... </p>
        </div>
      </div>

      <div id="rodape"> <b> contato@seusite.com.br </b> </div> Rodapé
   </body>
</html>
```

Figura 8.4: Estrutura do documento HTML
Fonte: Autores

O resultado do processamento desse documento (sem a atribuição de estilos) pode ser visualizado na Figura 8.5. Veja que, mesmo sem o visual, o documento pode ser entendido, uma vez que suas informações estão estruturadas e organizadas.

Figura 8.5: Apresentação do documento HTML sem CSS
Fonte: Autores

De forma a demonstrar a utilização de CSS para alteração do visual da página, vamos adicionar uma seção **<style>** contendo algumas regras que irão definir o comportamento visual dos seletores utilizados para identificar os elementos HTML. O conjunto de regras a ser adicionado pode ser visualizado na Figura 8.6.

```
#cabecalho { background:silver;
             height: 12mm;
             width: 100%;
             padding: 3mm;
             display: table;}

#barra_navegacao { background:yellow;
                   padding: 2mm;}

#corpo { background:white;
         width: 100%;
         display: table; }

#rodape { background:gray;
          text-align: center;}

#logo { background:blue;
        float: left;
        width: 20%;
        text-align: left; }

#banner { background:red;
          float: right;
          width: 80%;
          text-align: center; }

#menu { background:cyan;
        float: left;
        width: 30%;
        overflow: auto;
       }

#conteudo { background:purple;
            margin-left: 30%;
            }
```

Figura 8.6: Estrutura do documento usando regras CSS
Fonte: Autores

Como num passe de mágica, aquele documento originalmente estruturado apenas com *tags* HTML, no momento em que é associado a um conjunto de regras CSS, passa a apresentar-se conforme podemos visualizar na Figura 8.7

(as cores foram mantidas no *layout* apenas para facilitar a visualização das áreas compreendidas por cada divisão).

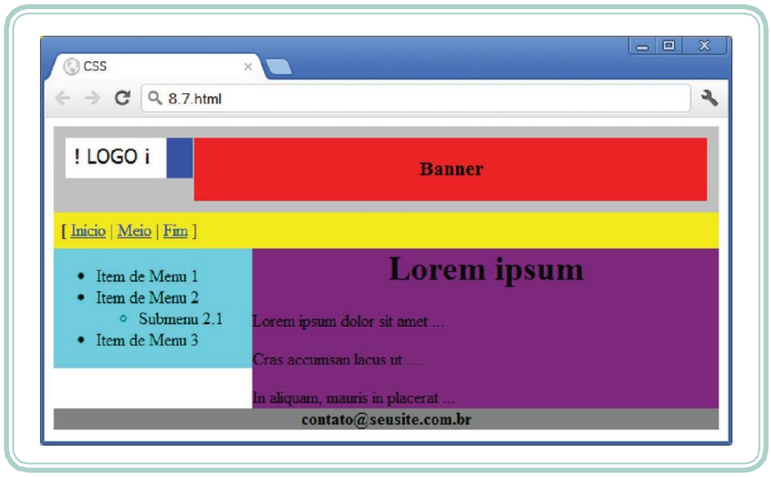


Figura 8.7: Documento HTML utilizando regras CSS para apresentação
Fonte: Autores

Agora, experimente alterar a propriedade *float*, da regra "#menu", de *left* para *right* e o valor de *float*, da regra "#conteudo", de *right* para *left*. Com essa simples alteração, o menu que, antes, era exibido no lado esquerdo, agora, pode ser visualizado no lado direito.

Vejamos, a seguir, o significado e o efeito de algumas propriedades relacionadas ao posicionamento e à forma de exibição nas áreas divididas de um *layout*:

- ***position*** – determina como serão considerados os comandos de posicionamento (*left*, *right*, *top* e *bottom*) e como um elemento será posicionado em relação aos demais. Admite quatro valores: *absolute* (os valores são considerados a partir de seu elemento pai), *fixed* (posicionamento fixo em relação à janela do navegador, dessa forma o elemento fica sempre visível, não importando a rolagem de tela), *relative* (posicionamento estabelecido a partir do local que o elemento ocupa) e *estatic* (este é o valor padrão e indica que o elemento não possui posicionamento definido).

- ***z-index*** – define a ordem de apresentação da área quando houver sobreposição. Valores mais altos posicionam a área na frente de áreas com valores mais baixos.

- ***overflow*** – especifica o comportamento da área quando seu conteúdo exceder os seus limites. Pode ser *visible* (redimensionando a camada para

comportar o conteúdo), *hidden* (ignorando o conteúdo excedente), scroll (incluindo barras de rolagem) ou *auto* (apresentando barras de rolagem quando necessário).

- ***visibility*** – define se a camada é visível (*visible*) ou oculta (*hidden*). Há, ainda, uma terceira opção: *inherit*, na qual o atributo é herdado da camada pai.

- ***cursor*** – especifica o tipo de ponteiro do *mouse* a ser utilizado sobre a camada (URL, *auto*, *crosshair*, *default*, *pointer*, *move*, *e-resize*, *ne-resize*, *nw-resize*, *n-resize*, *se-resize*, *sw-resize*, *s-resize*, *w-resize*, *text*, *wait*, *help*).

- ***display*** – define a forma de apresentação de um elemento. Por exemplo: a forma de apresentação *table* faz com que a camada se apresente como uma tabela, ou seja, como um quadro com dimensões definidas; já a apresentação *in-line* mostra a camada em uma única linha, sem quebras. Os principais valores são: *in-line*, *block*, *in-line-block*, *list-item*, *table*, *table-cell*.

- ***float*** – define se uma caixa poderá ou não flutuar à esquerda (*left*) ou à direita (*right*) do restante dos elementos de uma mesma área.

- ***clear*** – define os lados de um elemento no qual não são permitidos elementos flutuantes (*left*, *right*, *both*, *none*).

A Figura 8.8 ilustra o comportamento de algumas das propriedades exploradas até esse momento. Observe que a camada 2 se sobressai em relação à camada 1 devido ao valor da propriedade *z-index*. Perceba, também, que a imagem (“logo.png”) flutua sobre a esquerda do texto, o qual se acomoda no espaço restante da área. No caso do quadro amarelo, podemos ver as barras de rolagens apresentadas de forma fixa ao longo do quadro.

```
<html>
 <head>
  <title>CSS</title>
  <style type="text/css">
    .Q1 {
        position: absolute;
        width: 140px;
        height: 130px;
        background-color: green;
        cursor: wait;}

    .Q2 {
        position: absolute;
        width: 140px;
        height: 130px;
        background-color: blue;
        color: white;
        z-index:2;}

    .Q4 {
        position: fixed;
        background-color: yellow;
        width: 250px;
        height: 400px;
        overflow: scroll;}
  </style>
 </head>
 <body>
   <span class="Q1">camada 1</span>
   <span class="Q2" style="left: 50px; top: 50px;"> camada 2</span>
   <span class="Q2" style="left: 100px; top: 100px; visibility:hidden"> camada 3</span>

   <span class="Q4" style="right: 30px; top: 10px;">
        <img src="logo.png" style="float: left; margin: 10px">
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi id massa at
   </span>
 </body>
</html>
```

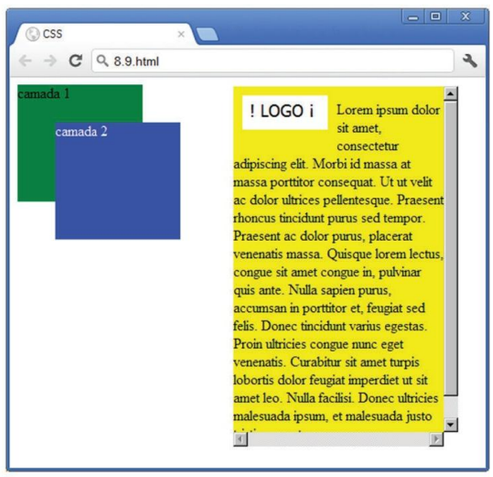


Figura 8.8: Utilização de propriedades de posicionamento e de formato para quadros empregando CSS
Fonte: Autores

Resumo

Nesta aula, compreendemos que, utilizando estilos, podemos alterar a disposição e a organização dos elementos ao longo de um *layout* sem que isso afete o conteúdo de nossa página. Exploramos a anatomia das áreas de divisão e o seu comportamento. Por meio de exemplos, utilizamos algumas propriedades para organizar um *layout* base para uma página *web*.

Com esta aula, finalizamos o estudo dos fundamentos de desenvolvimento *web*. É importante que você tenha presente que este material não é uma referência definitiva sobre o assunto, mas um guia que apresenta algumas ideias e indica caminhos para que você procure se aperfeiçoar.

Atividades de aprendizagem

1. O que são marcações lógicas?

2. Organize a estrutura e o estilo de uma página para apresentar uma notícia. Essa área deve conter uma foto sobre o fato noticiado (do lado esquerdo) e o texto da notícia circundando a foto.

3. Crie um documento HTML utilizando diferentes divisões com posicionamento absoluto e largura/altura fixos. Para cada área explore a propriedade cursor, experimentando os diferentes valores possíveis e observe o resultado.

4. Faça uma pesquisa na internet e descreva o que você entende por:

a) *Cross-browser*.

b) *Hack* CSS.

c) *Web standards*.