HTML 5

A LINGUAGEM DE MARCAÇÃO QUE REVOLUCIONOU A WEB

Maurício Samy Silva

Novatec

Editor: Rubens Prates
Capa: Carolina Kuwabata
Revisão gramatical: Débora Facin
Editoração eletrônica: Camila Kuwabata e Carolina Kuwabata

ISBN: 978-85-7522-261-4

Histórico de impressões:

Julho/2011 Primeira edição

Novatec Editora Ltda.
Rua Luís Antônio dos Santos 110
02460-000 – São Paulo, SP – Brasil
Tel.: +55 11 2959-6529
Fax: +55 11 2950-8869
E-mail: novatec@novatec.com.br
Site: www.novatec.com.br
Twitter: twitter.com/novateceditora
Facebook: facebook.com/novatec
LinkedIn: linkedin.com/in/novatec

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

Silva, Maurício Samy
HTML 5 / Maurício Samy Silva. -- São Paulo : Novatec Editora, 2011.

Bibliografia.
ISBN 978-85-7522-261-4

1. HTML (Linguagem de programação para Internet) 2. Internet (Rede de computadores) 3. Websites - Desenvolvimento I. Título.

11-06541 CDD-005.133

Índices para catálogo sistemático:

1. HTML : Linguagem de programação para Internet : Desenvolvimento de sites : Ciência da computação 005.133

OGF20110616

CAPÍTULO 1

Apresentação da HTML5

Neste capítulo, será apresentada a linguagem de marcação HTML5, relatando-se suas origens, finalidades e destinação. Será feito um breve relato histórico de sua evolução, examinando as motivações que resultaram na sua criação, a filosofia que norteia o desenvolvimento das especificações para a linguagem e as principais diferenças conceituais para as versões anteriores da linguagem HTML.

1.1 Introdução

HTML é a sigla em inglês para *HyperText Markup Language*, que, em português, significa linguagem para marcação de hipertexto.

O conceito de hipertexto admite um sem-número de considerações e discussões que fogem ao escopo deste livro. Para o bom entendimento das definições, podemos resumir hipertexto como todo o conteúdo inserido em um documento para a web e que tem como principal característica a possibilidade de se interligar a outros documentos da web. O que torna possível a construção de hipertextos são os links, presentes nas páginas dos sites que estamos acostumados a visitar quando entramos na internet.

Desde a invenção da web por Tim Berners-Lee, a HTML evoluiu por sete versões que são:

- HTML
- HTML +
- HTML 2.0

- HTML 3.0
- HTML 3.2
- HTML 4.0
- HTML 4.01 (versão atual)
- HTML5 (versão em fase de desenvolvimento)

Tecnicamente, o W3C considera oficialmente somente as versões HTML 2.0, HTML 3.2, HTML 4.0, HTML 4.01 e HTML5. As versões HTML e HTML+ são anteriores à criação do W3C e a versão HTML 3.0 não chegou a ser lançada oficialmente, transformando-se na versão HTML 3.2.

A web foi inventada em 1992 por Sir Tim Berners-Lee. Atualmente Tim é diretor do World Wide Web Consortium (W3C), pesquisador sênior do Laboratório da Ciência da Computação e Inteligência Artificial (CSAIL) do Instituto de Tecnologia de Massachusetts (MIT) e professor de Ciência da Computação na Universidade de Southampton, na Inglaterra.

Tim Berners-Lee trabalhava na Seção de Computação da Organização Europeia de Pesquisa Nuclear (CERN), com sede em Genebra, na Suíça, quando iniciou pesquisas visando a descobrir um método que possibilitasse aos cientistas do mundo inteiro compartilhar eletronicamente seus textos e pesquisas e que tivesse a funcionalidade de interligar os documentos. Estavam criadas as noções *web* e de *links* como são conhecidas atualmente.

Em 1990, Tim criou o protótipo de um navegador para rodar em computadores da NeXT, uma companhia fundada em 1985 por Steve Jobs, atual CEO da Apple. Inicialmente, o navegador foi chamado de WorldWideWeb e, posteriormente, renomeado para Nexus, a fim de evitar confusão com a World Wide Web.

O termo inglês *browser* é usado no jargão da internet para designar um programa capaz de ler e apresentar ao usuário os conteúdos de um documento web escrito em linguagem de marcação. Browser vem do verbo *to browse*, que significa folhear casualmente as páginas de um livro e foi traduzido para o português como navegador, gerando a tão bem conhecida expressão "navegar na internet". São exemplos de navegadores o Internet Explorer, o Firefox, o Opera, o Chrome e o Safari, entre outros. Neste livro, adotaremos o termo em sua forma traduzida: navegador.

Tim Berners-Lee acreditava que seria possível interligar hipertextos em computadores diferentes com uso de links globais, também chamados de hiperlinks. Ele desenvolveu um software próprio e um protocolo para recuperar hipertextos,

denominado HTTP. O formato de texto que criou para o HTTP foi chamado de HTML. Tim tomou como base para criação da HTML a especificação SGML, que é um método internacionalmente reconhecido e aceito, contendo normas gerais para a criação de linguagens de marcação. A marcação para hiperlinks conduzindo a documentos que não estivessem em um mesmo computador obviamente não constava das normas para SGML e foi inventada por Tim, demonstrada pela primeira vez em 1990, em uma estação de trabalho NeXT, nos laboratórios da CERN. Estava criado o embrião da World Wide Web, bem como a primeira versão da linguagem HTML para a marcação de hipertextos. A partir daí, a evolução cronológica da HTML deu-se conforme relatado sumariamente a seguir.

1.2 Histórico

Em setembro de 1991, foi criada a lista de discussão eletrônica denominada *WWW-talk*, com o propósito de trocar ideias e experiências sobre a HTML desenvolvida por Tim Berners-Lee. Um dos frequentadores da lista era Dave Raggett, dos laboratórios da Hewlett-Packard, em Bristol, Inglaterra. Dave, empolgado com a nova ideia, desenvolveu suas pesquisas e acabou por escrever a HTML+, uma versão elaborada e enriquecida da HTML original desenvolvida por Tim. Em outubro de 1993, Dave Raggett deu por encerrada as discussões e, no mês seguinte, publicou a versão final da HTML+.

A HTML+ começa com a seguinte afirmação:

> Documentos marcados com HTML+ são constituídos de títulos, parágrafos, listas, tabelas e figuras.

E continua estabelecendo:

> Ao contrário da maioria das tecnologias destinadas à criação de documentos, a HTML+ não se destina a determinar a aparência; assim, nomes e tamanhos de fontes, margens, tabulações, espaçamentos entre os elementos não são funções da linguagem. Fica a cargo dos softwares responsáveis pela renderização dos documentos marcados com HTML+ a maneira como os documentos devam ser apresentados (talvez com base em configurações de preferência do usuário).

Convém salientar com muita ênfase que, desde sua criação, os idealizadores da HTML tiveram a preocupação de retirar da linguagem de marcação qualquer atribuição ou função de apresentação, ou seja, HTML destina-se exclusivamente a estruturar documentos. É nessa destinação que se fundamentam os princípios básicos do desenvolvimento seguindo os Padrões Web.

O ano de 1992 marcou o interesse do Centro Nacional de Aplicações para Supercomputadores (NCSA) da Universidade de Illinois – então representado por Joseph Hardin e Dave Thompson, os quais acabaram por desenvolver um navegador que foi denominado Mosaic. Da NCSA participaram ainda dois expoentes da web: Marc Andreessen, que apresentou a ideia de um elemento para marcação de imagens nos hipertextos – e depois ficou milionário vendendo produtos para a web –, e Eric Bina, um brilhante programador considerado um gênio da web.

Em 1993, Lou Montulli criou o navegador de texto denominado Lynx versão 2.0a e Dave Raggett começou a trabalhar no desenvolvimento do navegador Arena, que se destinava a demonstrar, na prática, a implementação de todas as funcionalidades inventadas e discutidas até então. Ainda nesse ano a Sun Microsystems lançou a versão 1 do navegador Mosaic.

As grandes companhias não se interessaram pela nova invenção, alegando não acreditar que pudessem tirar proveito algum da web e que se tratava de um meio de comunicação restrito e de emprego exclusivo na área acadêmica. Tal pensamento desestimulou as pesquisas que vinham sendo desenvolvidas, pois estas dependiam de patrocínio. As equipes, que até então desenvolviam seus projetos em horas vagas, refrearam o ímpeto inicial por falta de tempo e verba. O trabalho continuou, mas em um ritmo aquém do esperado pelos envolvidos no projeto.

No mês de maio de 1994, a Spyglass comprou a licença de comercialização da versão aperfeiçoada do navegador Mosaic. Ainda nesse mês de maio, a CERN organizou, em Genebra, a primeira conferência para a World Wide Web com a apresentação da HTML+, criada por Dave Raggett. A conferência contou com a participação de aproximadamente 380 pessoas, a maioria envolvida com atividades e instituições acadêmicas.

No mês de outubro de 1994, Tim Berners-Lee em parceria com a CERN, onde a web foi por ele inventada, criaram o World Wide Web Consortium (W3C), com sede no Laboratório da Ciência da Computação do Massachusetts Institute of Technology (MIT).

Com a criação de novos navegadores, a HTML tornou-se um caos, com cada fabricante inventando novas formas de marcação HTML exclusivas para seus navegadores. Em uma tentativa de organizar a situação, Dan Connolly e colaboradores fizeram um levantamento minucioso de tudo o que existia na HTML e propuseram, em julho de 1994, a especificação HTML 2.0, uma tentativa de consolidar e unificar as diferentes formas HTML de marcação que vinham sendo criadas. Adicionalmente, Dan e sua equipe escreveram a primeira Definição do Tipo de Documento (DTD) para a HTML 2.0, uma espécie de descrição matemática da linguagem.

Em setembro, foi criado pela Força-Tarefa para Engenharia de Internet (IETF) o primeiro grupo de trabalho para a HTML. A IETF é uma organização internacional que congrega interessados nos vários aspectos da internet, com a finalidade de desenvolver normas e tecnologias pertinentes à operação e à evolução da arquitetura da internet. A criação desse grupo de trabalho marcou a abertura do desenvolvimento da HTML para qualquer pessoa interessada, independentemente de pertencer ou não a uma organização ou órgão particular, isto é, a discussão tornou-se aberta ao público em geral.

Marc Andreessen e Jim Clark haviam fundado a Mosaic Communications que, em novembro desse ano, transformou-se na Netscape Communications Corporation. Nascia a Companhia que se tornaria, em pouco tempo, a senhora absoluta da web, impulsionada com a aceitação unânime da nova versão do navegador Mosaic. O sucesso da Netscape deveu-se, entre outros fatores, primordialmente, ao investimento maciço nas funcionalidades de proporcionar acesso à internet, mesmo para os usuários com dispositivos mais precários. A Netscape tornou-se a sensação e muitos ainda acreditam erroneamente que a web foi criada pela Netscape. Por outro lado, a companhia isolou-se em suas pesquisas e desenvolvimentos, adotando a política de inventar HTML própria, sem participação da comunidade interessada e passando ao largo de conferências e encontros públicos relacionados.

No mês de outubro de 1994, foi criado o World Wide Web Consortium (W3C), um consórcio internacional formado por empresas, instituições, pesquisadores, desenvolvedores e público em geral, com a finalidade de desenvolver a web a seu potencial máximo, criando normas e especificações aplicáveis aos diversos segmentos e setores da web, desde tecnologias e softwares até fabricantes e fornecedores.

O W3C tem sua sede principal distribuída em três lugares distintos: nos Laboratórios de Ciência da Computação do MIT, em Massachusetts, nos Estados Unidos, no Instituto Nacional de Pesquisas de Informática e Automação, na França, e na Universidade de Keiko, no Japão, além de escritórios espalhados em várias cidades do mundo. As mais proeminentes e brilhantes cabeças envolvidas com o desenvolvimento para a web foram convidadas a formar o núcleo básico do W3C. Nomes de projeção internacional no envolvimento com a web foram convidados a participar, destacando-se Henrick Frystyk Neilsen, Anselm Baird-Smith, Jay Sekora, Rohit Khare, Dan Connolly, Jim Gettys, Tim Berners-Lee, Susan Hardy, Jim Miller, Dave Raggett, Tom Greene, Arthur Secret, Karen MacArthur, Arnaud le Hors, Håkon Lie, Bert Bos e Chris Lilley, entre outros.

O ano de 1995 assinalou o início de um desenvolvimento frenético de novas marcações para a HTML, com prioridade para a criação de elementos e atributos de apresentação em total desacordo com o propósito inicial da linguagem, qual seja: o de ser uma linguagem exclusivamente de marcação e estruturação de textos. Atributos e elementos para definir tamanhos, tipos e cores de letras dos textos, cores de fundo, texturas e toda uma parafernália relacionada, completamente fora do escopo inicial da HTML.

A versão final da HTML 2.0 foi publicada em 22 de setembro de 1995.

Em março de 1995, Dave Raggett lançou sua proposta para a HTML 3.0, que vem com a primeira sugestão de uma marcação específica para estilização e apresentação, ao mesmo tempo que também propõe a criação do atributo `class`. Surgiu, ainda, a marcação para tabelas, para notas de rodapé e formulários. A marcação para tabelas gerou grande discussão na época, tendo sido efetivada somente na versão seguinte da HTML.

Em setembro, a Netscape propôs o conceito de *frames* em documentos HTML e implementou essa funcionalidade em seu navegador, sem consultas à comunidade, como era prática comum.

Em novembro, a Microsoft lançou a versão 2.0 do Internet Explorer. Bert Bos, Håkon Lie, Dave Raggett, Chris Lilley e colaboradores iniciaram, nesse mesmo mês, a idealização das folhas de estilo em cascata (CSS).

Em dezembro, dissolveu-se o grupo de trabalho para a HTML.

Em fevereiro de 1996, o W3C criou o HTML ERB, um grupo formado por representantes da IBM, Microsoft, Netscape, Novell, Softquad e do W3C Consortium, encarregado de rever e padronizar a HTML com a finalidade de acabar com as implementações proprietárias.

Em dezembro, o HTML ERB transformou-se no Grupo de Trabalho da HTML, que existe até hoje. Esse mês assinalou o início dos estudos para a implementação de uma marcação provisoriamente denominada *Cougar*, a qual, mais tarde, transformar-se-ia no HTML4.

Em janeiro de 1997, o W3C endossou a HTML 3.2 como uma Recomendação oficial. Com essa versão, a HTML incorporou os elementos `table` e `applet`, bem como elementos para marcar subscritos, sobrescritos e texto ao redor de imagens.

Em julho de 1997, foi lançada a versão rascunho para a Cougar, depois denominada HTML4.

Em dezembro desse ano, o W3C endossou a HTML4 como uma Recomendação oficial.

Em dezembro de 1999, o W3C publicou as Recomendações para o HTML 4.01. Essa é a versão atual da HTML.

1.3 Criação da HTML5

Em maio de 2007, o W3C reconsiderou sua decisão de encerrar o desenvolvimento da HTML em favor da XHTML e tornou pública sua decisão de retomar os estudos para o desenvolvimento da HTML5, tomando como base o trabalho que já vinha sendo desenvolvido pelo WHATWG.

WHATWG é a sigla em inglês para *Web Hypertext Application Technology Working Group*, que, em português, significa Grupo de Trabalho para Tecnologias de Hipertexto em Aplicações para Web. O WHATWG foi criado em 2004 por desenvolvedores da Apple, da Fundação Mozilla e do navegador Opera, que, descontentes com os rumos adotados pelo W3C, propuseram-se a desenvolver as especificações para HTML5, Web Forms 2.0 e Web Controls 1.0. Atualmente, o foco único do Grupo de Trabalho é a HTML5, uma vez que a Web Forms 2.0 também foi assimilada pelo W3C e os estudos para Web Controls 1.0 foram interrompidos.

O WHATWG desenvolve a HTML5 em conjunto com o W3C e ambos mantêm em seus sites uma versão das especificações que diferem ligeiramente em pequenos detalhes. A versão do WHATWG é menos restritiva do que a versão do W3C. Por exemplo: em vários itens da especificação, apresenta exemplos ilustrativos e informações sobre suporte da funcionalidade descrita, nos navegadores modernos. Essas informações adicionais não constam da versão do W3C.

No dia 19 de janeiro de 2011, Ian Hickson, editor da HTML5, publicou no blog da WHATWG uma matéria informando que a especificação para a HTML5 continuaria a ser desenvolvida exclusivamente pelo W3C, ficando sob responsabilidade do WHATWG a continuidade do desenvolvimento de uma especificação para a HTML geral, isto é, sem sufixo designativo da versão.

1.4 Estrutura da especificação para a HTML5

A especificação para a HTML5 está estruturada em 10 seções, a saber:

1. **Infraestrutura comum**: define terminologia, classes, algoritmos, sintaxes e partes comuns das especificações.
2. **Semântica, estrutura e APIs para documentos HTML**: definem as funcionalidades do DOM HTML e dos elementos HTML em geral.
3. **Elementos HTML**: explicam o significado de cada um dos elementos HTML. São estabelecidas regras de uso dos elementos na marcação, bem como diretrizes de manipulação deles pelos agentes de usuário.
4. **Microformatos**: a especificação para a HTML5 prevê um mecanismo para marcar informações sobre o documento, no formato nome/valor, para serem lidas por máquinas. Essa seção descreve esse mecanismo e os algoritmos capazes de converter documentos HTML em outros formatos. Adicionalmente, nessa seção definem-se alguns vocabulários para Microformatos: informações de contato, calendário de eventos e licenças.
5. **Carregamento de páginas web**: documentos HTML não aparecem do nada – essa seção define as muitas funcionalidades relacionadas ao tratamento de páginas web pelos diferentes dispositivos.
6. **APIs para aplicações web**: descrevem as funcionalidades básicas para desenvolvimento de scripts em aplicações HTML.
7. **Interação com o usuário**: descreve os diferentes mecanismos de interação do usuário com um documento HTML.
8. **APIs para comunicação**: tratam dos mecanismos de comunicação entre aplicações HTML rodando em domínios diferentes e no mesmo cliente.
9. **Sintaxe HTML**: descreve a sintaxe HTML.
10. **Sintaxe XHTML**: descreve a sintaxe XHTML.

A especificação contém ainda apêndices nos quais são estabelecidas regras de renderização para os navegadores, listadas as funcionalidades obsoletas e considerações da Internet Assigned Numbers Authority (IANA), órgão responsável pela coordenação geral de protocolos para a Internet, nomes de domínio (DNS) e endereço IP.

1.5 Princípios de desenvolvimento da HTML5

Em 26 de novembro de 2007, o Grupo de Trabalho da HTML pertencente ao W3C publicou uma nota contendo um conjunto de diretrizes a ser seguido pelo Grupo de Trabalho e que descreve os princípios de desenvolvimento da HTML5. Tal nota está hospedada no site do W3C: `http://www.w3.org/TR/html-design-principles/`.

Os princípios descritos visam a orientar o Grupo de Trabalho que desenvolve a HTML5 nas seguintes áreas:

- Compatibilidade
- Utilidade
- Interoperabilidade
- Acesso universal

Vejamos a seguir cada uma dessas áreas e seus princípios de desenvolvimento para a HTML5.

1.5.1 Compatibilidade

Compatibilidade é um termo que pode ser interpretado de várias maneiras. É frequente o uso dos termos "retrocompatibilidade" e "compatibilidade futura", contudo, dependendo do contexto, nem sempre fica claro o que eles significam. Os princípios de compatibilidade previstos na Nota do W3C esclarecem o termo compatibilidade em diferentes contextos, conforme descritos a seguir.

1.5.1.1 Suporte para conteúdos existentes

Existem milhares de páginas web codificadas em desacordo com as diretrizes da HTML e que são renderizadas a contento e funcionam perfeitamente, pois o tratamento dado pelos navegadores aos erros de marcação é muito complacente.

Historicamente os fabricantes de navegadores cada vez mais implementaram mecanismos de tratamento de erros de marcação capazes de simplesmente ignorá-los. O nível de sofisticação dos mecanismos chegou a tal ponto que os navegadores praticamente "adivinham" o que o desenvolvedor deveria ter feito quando cometeu um erro de marcação e, na maioria dos casos, renderizam o conteúdo como se erros não existissem.

Observe a marcação de uma lista não ordenada com um nível de aninhamento:

```
<ul>
   <li>item 1</li>
      <ul>
         <li>item 1.1
         <li>item 1.2</li>
      </ul>
   <li>item 2
      <ul>
         <li>item 2.1
         <li>item 2.2</li>
```

A marcação em questão contém vários erros, mas, se constar de um documento web servido com o tipo de MIME `text/html`, e qualquer DOCTYPE ou mesmo sem ele, será renderizada normalmente como uma lista de dois itens, cada um deles com dois subitens e respectivos marcadores-padrão.

Em marcação XHTML servida com o tipo de MIME `application/xhtml+xml` o navegador acusará um erro fatal e o usuário será brindado com uma mensagem de erro.

Como sabemos, a tentativa de evoluir a HTML para XHTML fracassou e foi abandonada pelo W3C em favor da HTML5 e esta ao contrário da XHTML, segundo o princípio da compatibilidade, deverá ser desenvolvida de modo a não "quebrar" marcação errada.

Muitos sites usam o elemento `u` para marcar textos sublinhados. Esse elemento foi colocado em desuso pela HTML4; contudo, a HTML5 deve prever suporte para ele.

Esses dois simples exemplos bem ilustram o princípio da compatibilidade, porém não é só marcação errada e elemento em desuso que ele contempla. APIs de especificações HTML antigas, funcionalidades e comportamentos não previstos em especificações e até mesmo funcionalidades proprietárias devem ser consideradas no desenvolvimento da HTML5.

Convém ainda ressaltar que renderizar corretamente marcação errada não implica documento válido.

Segundo esse princípio, a HTML5 está sendo desenvolvida de modo que a marcação mostrada anteriormente renderiza normalmente em HTML5, mas os autores devem evitá-la ao criar documentos novos. Elas serão suportadas em documentos antigos quando migrados para HTML5.

As perguntas que devem ser feitas para se chegar à conclusão sobre o suporte ou não a uma funcionalidade ou comportamento são:

- A quantidade de conteúdos dependente da funcionalidade ou comportamento é considerável?
- A funcionalidade ou comportamento ocorre em sites populares e de grande audiência?
- A funcionalidade ou comportamento é para consumo global ou localizado?
- A funcionalidade ou comportamento é para a web pública ou restrito a dispositivos particulares?
- A funcionalidade ou comportamento funciona em vários dispositivos ou está restrito a determinado dispositivo?

1.5.1.2 Degradação graciosa

Degradação graciosa é a tradução literal para "graceful degradation" cujo significado é o seguinte:

> A criação de um conteúdo deve ser projetada de modo que usuários e agentes de usuários tenham acesso a ele independentemente do dispositivo e condições que estejam usando, ainda que os mais precários possíveis.

O conteúdo é criado implementando-se as mais modernas e avançadas tecnologias com o objetivo de estilizar, incrementar apresentação e usabilidade e tudo mais que achar conveniente.

Nessas condições, o princípio da degradação graciosa estabelece que dispositivos e usuários sem suporte para as tecnologias avançadas não ficarão privados da informação transmitida pelo conteúdo. Por exemplo: projetar um box com bordas arredondadas e sombras proporciona degradação graciosa para usuários com navegador sem suporte para CSS3 que verão o box com bordas não arredondadas e sem sombras, porém terão acesso ao conteúdo nele inserido. Ao projetar um menu dropdown com JavaScript, é preciso garantir que o menu será acessível e funcional para usuários sem suporte ou com JavaScript desabilitado.

Esses dois simples exemplos bem ilustram o princípio da degradação graciosa que, em última análise, significa o seguinte:

> Ao acrescentar funcionalidades cujo suporte é previsto nos mais modernos navegadores, é preciso que os desenvolvedores garantam mecanismos capazes

> de renderizar o conteúdo de forma a não perder a informação transmitida, quando o conteúdo é renderizado em agentes de usuário antigos nem para usuários que navegam com algumas funcionalidades desabilitadas.

Segundo esse princípio, a HTML5 está sendo desenvolvida de modo a fornecer mecanismos capazes de proporcionar uma solução alternativa para a degradação graciosa. Em se tratando de novas funcionalidades, na maioria dos casos, as seguintes considerações são levadas em conta:

- Novos elementos ou atributos, quando não entendidos pelo agente de usuário, devem prever um mecanismo adicional de transmissão de semântica, sem perder sua funcionalidade.
- Novos métodos ou atributos devem ser capazes de ser testados via ECMAScript antes de serem usados.
- Novos elementos ou atributos devem ser semânticos e capazes de receberem estilização mínima.
- Novos elementos capazes de comportarem um mecanismo de renderização sofisticado devem prever uma alternativa de transmissão do seu conteúdo para agentes que não os suportem.

Contudo, esse princípio não é absoluto no desenvolvimento da HTML5. Em alguns casos, a nova funcionalidade simplesmente não irá funcionar em determinada classe de agentes de usuário ou ainda é impossível de se prover degradação graciosa. Por exemplo: as novas APIs que dependem de linguagem de script para funcionar não funcionarão em agentes de usuário sem suporte para o script.

Os exemplos a seguir ilustram o princípio da degradação graciosa aplicado ao desenvolvimento da HTML5:

- Os novos elementos `canvas` e `audio` destinados a implementar multimídia em um documento preveem um mecanismo de apresentação de conteúdo alternativo para navegadores que não suportam esses elementos, como mostrado na marcação a seguir:

```
<canvas>Insira aqui o conteúdo alternativo</canvas>
<audio>Insira aqui o conteúdo alternativo</audio>
```

- O novo método `getElementsByClassName()` é muito mais rápido do que seu equivalente tradicional desenvolvido com ECMAScript. Segundo o princípio da degradação graciosa, mecanismos de verificação de suporte para esse método podem ser usados para servir o método equivalente para agentes que não o suportam.

1.5.1.3 Não reinventar a roda

Segundo esse princípio, se uma tecnologia já está implementada e é largamente usada para atender a determinados casos, não há necessidade de se criar uma nova funcionalidade para atender àqueles casos. Mesmo porque é mais fácil ampliar funcionalidades já existentes e testadas do que partir do zero criando nova funcionalidade com o mesmo propósito.

O exemplo clássico da aplicação desse princípio é a adoção dos elementos `marquee` e do atributo `contenteditable` que já existem, não constam das especificações para a HTML, mas agora passam a fazer parte da HTML5.

1.5.1.4 Pavimentar os caminhos existentes

Segundo esse princípio, se uma prática é amplamente usada pelos desenvolvedores, é preferível adotá-la a proibi-la ou inventar algo novo para substituí-la.

É comum a prática de os autores usarem, em documentos HTML, a marcação `<br/>` em detrimento de `<br>`. Não há nenhum dano em se permitir a continuidade dessa prática.

1.5.1.5 Evolução em lugar de revolução

Revoluções, por vezes, mudam o mundo para melhor. Contudo, na maioria dos casos, é melhor aperfeiçoar práticas existentes do que simplesmente descartá-las. Segundo esse princípio, se uma técnica é usada pelos desenvolvedores, é preferível aperfeiçoá-la e adotá-la em vez de obrigar os autores a estudar e aprender novas técnicas.

1.5.2 Utilidade

Utilidade visa a garantir que a marcação HTML possa ser usada de modo efetivo para todos os fins a que ela se destina.

1.5.2.1 Solucionar problemas reais

Alterações nas especificações devem ser feitas para solucionar problemas atuais reais. Abstrações e ilações teóricas sobre problemas existentes devem ser descartadas em favor de uma solução pragmática e efetiva. Todos os problemas atuais devem ser estudados e resolvidos, se possível.

1.5.2.2 Prioridades

Prioridades é um princípio que considera uma cadeia de interessados quando há conflitos em relação à implementação de uma funcionalidade na HTML5.

Nesses casos, ao se criar uma funcionalidade, os usuários serão considerados em primeiro lugar. Seguem-se a estes os implementadores das funcionalidades que são seguidos pelos que criam especificações; em último lugar, considera-se a pureza teórica da questão.

1.5.2.3 Segurança

Segurança é um princípio que visa a assegurar que as funcionalidades da HTML5 não ofereçam brechas que possibilitam violar as normas de segurança do modelo web. Preferencialmente as considerações sobre segurança devem ser tratadas diretamente nas especificações.

Por exemplo: a comunicação entre documentos de diferentes sites é muito útil; contudo, implementar tal funcionalidade sem restrições pode colocar em risco a segurança dos dados transmitidos. A funcionalidade para comunicação entre documentos está projetada na HTML5 paralelamente a mecanismos que impedem a violação das regras de segurança.

1.5.2.4 Separação de camadas

A HTML foi projetada para separar a marcação estrutural da apresentação. Nesse contexto, marcação que expresse estrutura é preferida sobre marcação para apresentação. Contudo, marcação estruturada é um meio para se alcançar um fim tal como ocorre quando consideramos o funcionamento da funcionalidade independentemente do tipo de mídia do usuário.

Esse princípio estabelece que não são necessárias considerações profundas e detalhamento refinado sobre questões relacionadas à semântica se o fim pode ser alcançado por uma via mais simples. Definir um padrão razoável de semântica para diferentes tipos de mídia é suficiente. Deve ser feito um balanço ponderado entre semântica refinada e uso prático antes de se chegar a uma conclusão. Nomes de elementos e atributos da marcação devem ser práticos e pragmáticos em lugar de verbosos e muito detalhados.

Por exemplo: o elemento `article` define um artigo individual sem detalhar a maneira como ele deve ser renderizado. Em determinado contexto, um artigo pode ser único na página e disposto em múltiplas colunas ao passo que em outro contexto possa coexistir com vários outros artigos multicolunares em uma mesma página.

Os elementos `b` e `i` são largamente usados. É muito melhor estabelecer novas regras de renderização para eles, nos diferentes tipos de mídia, do que simplesmente bani-los das especificações.

1.5.2.5 Consistência do DOM

O princípio da consistência do DOM, como o próprio nome sugere, visa a criar funcionalidades na HTML5 que permitam a produção de uma árvore do documento consistente e facilmente acessível às linguagens de script e programas em geral. Discrepâncias são admitidas desde que necessárias a contemplar dispositivos do passado, mas, nesses casos, as diferenças devem ser minimizadas.

1.5.3 Interoperabilidade

Interoperabilidade visa a aumentar as chances de uma implementação HTML funcionar nos mais variados dispositivos e sistemas.

1.5.3.1 Comportamentos definidos

Esse princípio visa a que as funcionalidades, quando for o caso, comportem-se de maneira bem definida em lugar de deixar livremente por conta dos agentes de usuário a tarefa de estabelecer como será o comportamento. Contudo, deve-se permitir que as implementações tenham certa liberdade para incrementar algum comportamento em áreas como de interface do usuário e da qualidade de renderização.

1.5.3.2 Evitar soluções complexas

Esse princípio dá preferência ao uso de soluções simples. Funcionalidades simples são mais fáceis de serem implementadas pelos agentes de usuário, além de facilitarem a interoperabilidade e o entendimento pelos autores. No entanto, simplicidade não deve ser desculpa para se ferir outros princípios de desenvolvimento da HTML5.

1.5.3.3 Tratamento de erros

Esse princípio estabelece que os erros de marcação devem ser tratados de modo a não frustrar o usuário apresentando-lhe uma mensagem de erro. Nos casos de erro é preferível um mecanismo de degradação graciosa a uma mensagem de erro fatal.

1.5.4 Acesso universal

Acesso universal visa a garantir o acesso às funcionalidades da HTML5 pelo maior número possível de dispositivos e sistema segundo os princípios a seguir:

1.5.4.1 Independência de mídia

As funcionalidades da HTML5 devem, sempre que possível, funcionar nos mais variados tipos de plataformas, dispositivos e mídias. Isso não significa que determinada funcionalidade deva ser descartada porque não é suportada por essa ou aquela plataforma. Por exemplo: funcionalidades de interação não devem ser descartadas pelo fato de não serem suportadas em mídia impressa, como ocorre com o elemento `a` que simplesmente não funciona quando impresso.

1.5.4.2 Suporte multilíngua

Essa funcionalidade visa a facilitar a produção de documentos nos diferentes idiomas existentes no mundo.

1.5.4.3 Acessibilidade

Essa funcionalidade visa a facilitar o acesso ao conteúdo independentemente do dispositivo ou das necessidades especiais do usuário. Isso não significa que uma funcionalidade deva ser descartada se um grupo de usuários não tiver acesso a ela, mas se devem prever meios alternativos de acesso. É o caso do elemento `img` que não deve ser banido porque usuários cegos não conseguem ver imagens. Usa-se o atributo `alt` para fornecer um meio alternativo de acesso à imagem para aqueles usuários.

1.6 Diferenças entre HTML5 e HTML4

As principais diferenças entre a HTML5 e a HTML4 têm suas origens no fato de a HTML5 estar sendo desenvolvida com o propósito de substituir tanto a HTML criada nos anos 90 quanto a XHTML que foi uma tentativa frustrada de reformular a HTML4 como uma aplicação XML.

1.6.1 Introdução

A especificação para a HTML5 iniciada no ano de 2004 e incorporada pelo W3C no ano de 2007 tem como objetivo geral estudar e resolver problemas relacionados à implementação de um HTML contemporâneo e ao mesmo tempo compatível com conteúdos existentes. Para cumprir esse objetivo, considera os seguintes pontos:

- Definição de uma linguagem única denominada HTML5 que pode ser escrita tanto com a sintaxe HTML quanto com a sintaxe XML.
- Definição de modelos de processamento detalhados com vista a promover a interoperabilidade da linguagem.
- Aperfeiçoamento da marcação dos documentos.
- Criação de marcação e APIs para novas tecnologias, tais como as aplicações Web.

Nesse contexto, podemos agrupar as diferenças entre as duas linguagens nas seguintes áreas:

1.6.1.1 Retrocompatibilidade

A HTML5 está sendo desenvolvida em dois níveis de requisitos de conformidades. Requisitos para autores e requisitos para agentes de usuário. Considere, para ilustrar a diferença nessa área, os elementos `isindex` e `plaintext`. De acordo com a HTML4, esses elementos estão em desuso (deprecated) e seriam banidos das próximas especificações para a HTML. Segundo os princípios de desenvolvimento da HTML5, esses elementos perdem sua condição de "em desuso" e passam a ser considerados elementos obsoletos (obsoletes). A diferença é que eles não serão banidos da especificação. Continuarão constando como requisitos para agentes de usuário e seu uso pelos autores é desaconselhado.

Na prática isso significa que os fabricantes de agentes de usuário devem implementar funcionalidades que permitam reconhecer e renderizar apropriadamente aqueles elementos nos dispositivos futuros, muito embora o uso deles, pelos autores, seja desaconselhado.

Na HTML5 não existem elementos em desuso (deprecated) com o sentido que a palavra tem na HTML4.

O mesmo acontece com outros elementos em desuso segundo a HTML4, como os elementos `font`, `u`, `s` etc.

1.6.1.2 Modelo de desenvolvimento

Nessa área a diferença é que a HTML5 não será considerada concluída até que haja pelo menos duas implementações completas das funcionalidades da especificação. Será criada uma suíte de testes para verificar o funcionamento de todas as funcionalidades nas duas implementações. O objetivo aqui é assegurar que a especificação pode ser implementada e usada pelos autores tão logo esteja finalizada.

Para as versões anteriores da linguagem HTML, a especificação final era aprovada por um comitê do W3C antes mesmo de ser totalmente implementada.

1.6.2 Sintaxe

A especificação para a HTML5 define uma sintaxe que é compatível tanto com a HTML quanto com a XHTML. Os documentos usando a sintaxe HTML com o tipo de MIME `text/html` são parseados segundo regras compatíveis com as atuais implementações populares.

Observe a seguir um exemplo de marcação HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html lang="pt-br">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <title>Título da página</title>
        <link rel="stylesheet" type="text/css" href="/estilos/main.css">
    </head>
    <body>
        <h1>Minha página HTML4</h1>
    </body>
</html>
```

A seguir, a marcação HTML5 compatível com a sintaxe HTML:

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
        <title> Título da página </title>
        <link rel="stylesheet" href="/estilos/main.css">
    </head>
    <body>
        <h1>Minha página HTML5</h1>
    </body>
</html
```

1.6.2.1 DOCTYPE

A declaração `DOCTYPE` (tipo de documento) deve ser feita na primeira linha da marcação HTML para garantir renderização no modo standard. Para maiores informações, ver: `http://maujor.com/w3ctuto/qatips/doctype.html`. Nada deverá existir acima da declaração de DOCTYPE nem mesmo uma linha em branco.

Existem doze diferentes DOCTYPES para uso em documentos HTML e XHTML e todos eles são verbosos e difíceis de se guardar na memória. Todos eles fazem referência e contêm um link apontando para um documento com as regras de sintaxe para a versão HTML ou XHTML na qual o documento foi escrito.

Em HTML5 o `DOCTYPE` foi simplificado, não havendo mais necessidade do link, e o `DOCTYPE` existe com a única função de habilitar a renderização em modo standard para documentos escritos com a sintaxe HTML. Os navegadores já entendem a declaração `DOCTYPE` segundo a sintaxe HTML5.

1.6.2.2 Codificação de caracteres

A codificação de caracteres em um documento web pode ser feita de três maneiras distintas:

- Com uso do parâmetro `charset` no cabeçalho HTTP `Content-Type`.
- Uso de um caractere Unicode Byte Order Mark (BOM) no início do documento.
- Uso do elemento `meta` com o atributo `charset`.

Em HTML5 a declaração da codificação de caracteres com uso do elemento `meta` foi simplificada, pois não há mais a necessidade dos atributos `http-equiv` e `content`. Contudo, convém notar que a sintaxe HTML4 para esse elemento também é válida em HTML5 e, se você a preferir, pode usá-la sem problemas. Para maiores informações, ver: `http://www.w3.org/International/O-charset`.

1.6.2.3 Elemento link

Em HTML5 o atributo `type` para links a folhas de estilos e para scripts é dispensável, mas a declaração desse atributo pode ser usada, opcionalmente, em HTML5.

1.7 A HTML5 nos navegadores atuais

Desenvolver um projeto web em HTML5 para os navegadores atuais é uma opção que pode ser considerada; caso você decida por ela, é preciso levar em conta algumas restrições e mecanismos existentes para contorná-las.

A maioria, mas não todas as funcionalidades da HTML5, já é suportada por um ou mais navegadores atuais; contudo, existem mecanismos desenvolvidos com a linguagem JavaScript que permitem detectar suporte para as funcionalidades criando condições de o desenvolvedor oferecer um conteúdo alternativo para uma funcionalidade não suportada por esse ou aquele navegador. Vejamos a seguir algumas considerações sobre esse tema.

1.7.1 Os novos elementos e o Internet Explorer

A HTML5 criou vários elementos novos que não são estilizáveis com uso das CSS nos navegadores Internet Explorer nas versões anteriores à versão 9, pois esses navegadores, ao contrário dos navegadores standards, não aplicam regras CSS a elementos que não conhecem.

Em uma marcação HTML, se você resolver criar um novo elemento chamado `xpto` (ainda que ele não seja válido) e escrever uma regra CSS como mostrada a seguir, o conteúdo inserido no elemento será normalmente estilizado na cor vermelha com fundo na cor preta, exceto nos IE8 e anteriores.

```
xpto { color: red; background-color: black; }
```

Se você pretende servir seu documento HTML5 para aqueles navegadores, precisa inserir um mecanismo capaz de fazer com que eles reconheçam os novos elementos

da linguagem. Tal mecanismo é oferecido pelo método `createElement()` da JavaScript, conforme mostrado a seguir:

```
<!--[if lte IE 8]>
   <script>
      document.createElement("xpto");
   </script>
<![endif]-->
```

Existem aproximadamente trinta novos elementos na HTML5 e uma alternativa para o script, criando todos esses elementos individualmente da forma como mostrada anteriormente; foi desenvolvida por Remy Sharp que apresentou o script em uma matéria no seu blog, disponível para consulta em `http://remysharp.com/2009/01/07/html5-enabling-script/`.

O script de Remy Sharp encontra-se hospedado no Google em endereço público; podemos criar em nossos documentos um *hotlink* para o script, inserindo-o na seção `head` do documento, como mostrado a seguir:

```
<!--[if lte IE 8]>
   <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

1.7.2 Os novos elementos e o modelo de renderização

Outro aspecto importante a considerar, e esse válido para todos os navegadores, é que os novos elementos, todos eles, renderizam seus conteúdos como elementos inline, pois os modelos de conteúdo da HTML5 são prática e conceitualmente diferentes dos modelos das HTML anteriores.

Se quisermos que um ou mais dos novos elementos sejam renderizados segundo o conceito nível de bloco da HTML4, devemos declarar explicitamente essa condição em regra CSS, conforme mostrado a seguir:

```
xpto { display: block; }
```

1.7.3 Biblioteca JavaScript Modernizr

Modernizr é uma pequena biblioteca JavaScript que está sendo constantemente atualizada por seus autores, destinada a detectar suporte nativo pelos navegadores para as funcionalidades das novas tecnologias para a web. As funcionalidades contempladas pela biblioteca são aquelas que estão sendo implementadas pelas especificações para a HTML5 e para as CSS3.

A biblioteca foi desenvolvida por Faruk Ateş com a colaboração de Paul Irish e encontra-se disponível para download em www.modernizr.com.

O script da biblioteca cria um objeto JavaScript denominado Modernizr e várias propriedades para esse objeto. As propriedades e métodos criados têm nomes idênticos aos das novas funcionalidades a detectar. Observe os exemplos a seguir:

```
Modernizr.canvas;          // propriedade para verificar suporte ao novo elemento canvas
Modernizr.audio;           // propriedade para verificar suporte ao novo elemento audio
Modernizr.borderradius;    // propriedade para verificar suporte a bordas arredondadas das CSS3
Modernizr.multiplebgs;     // propriedade para verificar suporte a múltiplos backgrounds das CSS3
```

Uma vez linkada a biblioteca a um documento web, podemos testar o suporte a uma nova funcionalidade, conforme exemplo a seguir, em que testamos o suporte para o elemento canvas.

```
if (Modernizr.canvas) {
    // script para uso de canvas
} else {
    alert("Lamento, seu navegador não suporta canvas");
}
```

A biblioteca cria e adiciona classes ao elemento html, oferecendo ao desenvolvedor a possibilidade de estilizar diferenciadamente um ou mais elementos na página conforme o navegador ofereça ou não suporte a determinada funcionalidade.

Suponha o seguinte cenário para exemplificar o uso dessas classes:

Se o navegador oferecer suporte para múltiplos backgrounds, os parágrafos do documento serão na cor vermelha; se não suportar, serão na cor azul.

A biblioteca identifica o suporte pelo navegador e acrescenta a classe com o nome multiplebgs ao elemento html. Não havendo suporte, o nome da classe acrescentada é no-multiplebgs e a folha de estilos para resolver o problema proposto no cenário criado é:

```
<style>
    .multiplebgs p { color: red; }
    .no-multiplebgs p { color: blue; }
</style>
```

Notar que o acréscimo da classe, como mostrado, possibilita uma autêntica condicional na folha de estilos, pois as regras mostradas cumprem a mesma finalidade da condicional mostrada no script a seguir.

```
<script>
    var para = document.getElementsByTagName("p");
    if (Modernizr.multiplebgs) {
        for (var i=0; i<para.length; i++) {
            para[i].style.color = "red";
        }
    } else {
        for (var i=0; i<para.length; i++) {
            para[i].style.color = "blue";
        }
    }
</script>
```

Outra funcionalidade importante da biblioteca é que ela agregou o script desenvolvido por Remy Sharp, mostrado no item 1.1.8, que faz com que o navegador Internet Explorer, nas versões 8 e anteriores, reconheça, para fins de aplicação de estilização, os novos elementos da HTML5. Assim, o uso da biblioteca dispensa o uso do script de Remy Sharp.

Atualmente não existe um endereço público de hospedagem da biblioteca para incorporá-la aos nossos documentos com uso de um *hotlink*. Você deverá fazer o download da biblioteca, hospedá-la no seu servidor e linká-la às suas páginas, conforme mostrado a seguir:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    ...
    <script src="path/modernizr-v.n.min.js"></script>
</head>
<body>
...
```

Maiores detalhes sobre a biblioteca, incluindo a relação de todos os objetos, propriedades e classes por ela criadas encontram-se em `http://www.modernizr.com/docs/`.

1.8 Templates HTML5

A sintaxe HTML5 é bastante flexível, pois a marcação pode ser escrita segundo as regras da HTML4 que admitem, por exemplo, que valores de atributo sejam ou não marcados com aspas ou apóstrofes, que algumas tags, como `p` e `li`, não sejam fechadas e uma série de outras regras que dão liberdade de escolha da marcação conforme

a preferência pessoal do autor. Escrever HTML5 com base nessas regras significa produzir um documento HTML5, ou em conformidade com a sintaxe HTML.

Por outro lado, a HTML5 pode também seguir a sintaxe XML e ser escrita segundo suas regras que são bem rígidas, pois não admitem, por exemplo, valores de atributos sem aspas ou apóstrofes e tags sem fechamento. Se você desenvolve XHTML, deve estar bem familiarizado com essa sintaxe. Escrever HTML5 com base nas regras da XML significa produzir um documento XHTML5, ou em conformidade com a sintaxe XML.

Nessa seção apresentaremos as formas de escrever HTML5 segundo as diversas sintaxes permitidas pela linguagem, desenvolvendo alguns templates ilustrativos.

Antes de passarmos aos templates, façamos algumas considerações sobre a marcação XHTML.

Seja você um desenvolvedor experiente, seja um iniciante que cria conteúdos com uso de marcação HTML, é quase certo que, assim como eu, desenvolve seus documentos segundo as regras da XHTML.

Existem milhões de páginas na Internet desenvolvidas com essa marcação; isso porque quem as criou estava se preparando para o futuro, pois até o início de 2007 o W3C alardeava que o futuro da HTML era a XHTML e que não haveria novas versões da HTML.

O objetivo do W3C ao criar a XHTML foi proporcionar meios aos desenvolvedores de incluir as funcionalidades da XML nos documentos para a web. Em março de 2007, Tim Berners-Lee publicou um press release afirmando o seguinte:

> Após a publicação do HTML4 e seguindo as conclusões do Workshop 1998, o W3C focou na transformação do HTML em um formato com base em XML chamado XHTML devido aos benefícios próprios do formato XML. A primeira recomendação XHTML completa foi publicada no início de 2000. Contudo, devido ao maciço legado do conteúdo Web, com base em variações do HTML, os fabricantes de navegadores começaram vagarosamente a adotar o XHTML. Isso resultou em pouca motivação para os desenvolvedores de conteúdos adotar o XHTML em seu ambiente tradicional de desenvolvimento. Expoentes das comunidades de desenvolvedores e de design apelaram ao W3C para a necessidade urgente de uma revisão dos seus compromissos com o HTML, acrescentando novas funcionalidades (a começar com a Norma para HTML4) em uma maneira que seja consistente com as práticas da comunidade e ao mesmo tempo retrocompatíveis. O W3C garantirá interoperabilidade por meio da implementação de suítes de testes robustas e de serviços de validação para as futuras tecnologias e que estarão à disposição da comunidade.

> O W3C com o forte suporte dos seus Membros e com mais recursos (tanto em pessoal quanto em hardware) tem o prazer de retomar os trabalhos para o HTML. O W3C modificou os Estatutos do Grupo de Trabalho do HTML com o propósito de permitir a efetiva participação dos fabricantes de navegadores, projetistas de aplicações e desenvolvedores de conteúdos, pois depende do trabalho conjunto dessas pessoas o sucesso do HTML do futuro.

Estava decretada a morte da XHTML em favor da HTML5. Contudo, o legado da XHTML permanece e você que já se acostumou com a sintaxe rígida da marcação não terá de rever o que aprendeu, pois ela continua perfeitamente válida na HTML5, assim como é válida também a sintaxe HTML4 muito menos rígida e mais complacente do que a da XHTML.

Um documento XHTML para ser puro e poder se beneficiar das funcionalidades da XML – e foi para isso que a XHTML foi criada – deve ser servido com o tipo de MIME `application/xml` ou `application/xhtml+xml`. Contudo, a maioria dos documentos XHTML na web é servida com o tipo MIME para HTML `text/html`, o que significa que não são conformes com a XML.

A tentativa de o W3C evoluir a HTML para conformidade com XML fracassou, porque páginas XHTML servidas como XML que contenham qualquer violação com a sintaxe XML causarão um erro fatal de parseamento e a página simplesmente não será renderizada.

Por outro lado, apesar da rigidez com o tratamento de erros, a sintaxe XHTML foi bem recebida pelos desenvolvedores e é amplamente usada.

Observe a marcação XHTML a seguir:

```
<?php header('Content-Type: application/xhtml+xml'); ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Página XHTML</title>
    <style type="text/css" media="all">
        /* regras de estilo */
    </style>
</head>
<body>
    <h1>Página com marcação XHTML</h1>
        <p>Essa página contém marcação XHTML <strong class="destaque">sem erros</strong></p>
```

```
        <p>Foi servida com tipo de MIME <code>application/xhtml+xml</code></p>
</body>
</html>
```

Para demonstrar o exemplo que desenvolvemos, inserimos na primeira linha da página contendo a marcação XHTML a função `header()` do PHP, com a finalidade de servir a página como aplicação XML. O documento está de acordo com a sintaxe XML e será renderizado normalmente em navegadores que suportam XML.

Vamos supor que o desenvolvedor do exemplo tenha se esquecido de usar aspas no valor `destaque` da classe para o elemento `strong` inserido no primeiro parágrafo, como mostrado a seguir.

```
<p>Essa página contém marcação XHTML <strong class=destaque>sem erros</strong></p>
```

Se o documento for servido com o tipo de MIME `text/html`, o erro de marcação será ignorado pelo navegador e a renderização do documento será normal. Contudo, servindo o documento como XML com o tipo de MIME `application/xml` ou `application/xhtml+xml` o erro será fatal e não haverá renderização, conforme figuras 1.1 e 1.2.

Na figura 1.1 mostramos a não renderização da página nos navegadores Firefox e Safari.

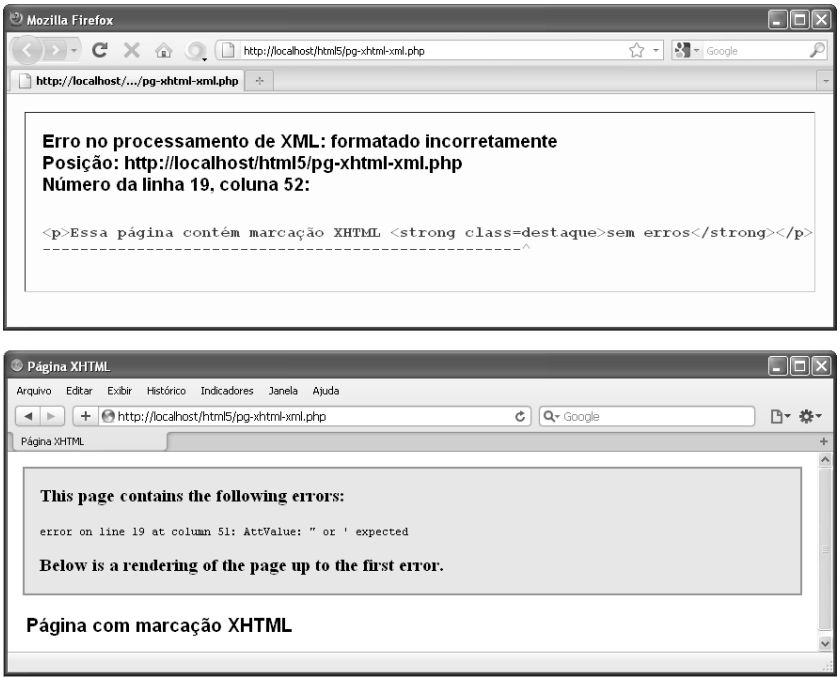

Figura 1.1 – Erro em documento XHTML em navegador standard.

Notar que não há um padrão de apresentação da mensagem de erro, a maneira de apresentá-lo ao usuário difere de um navegador para outro. O navegador Opera, ao apresentar a mensagem de erro, oferece adicionalmente um link para a renderização da página como se o erro não existisse. Por outro lado, o navegador Internet Explorer, que não oferece suporte para XML, comporta-se apresentando ao usuário uma caixa de diálogo com opção de abrir o arquivo com outro programa ou salvá-lo (Figura 1.2).

Figura 1.2 – Erro em documento XHTML no IE.

Essa rigidez no tratamento de erros contribuiu muito para a não adoção da XHTML como XML pelos desenvolvedores. Estima-se que mais de 95% das páginas web contenham pelo menos um erro de marcação. Quem se arriscaria a ter seu site "quebrado" por conta de uma distração ou um mínimo engano involuntário?

1.8.1 Template mínimo

Em HTML5, um documento para ser válido, além do DOCTYPE, deve conter o elemento `title` e nada mais. Assim, um template mínimo da HTML5 pode ser estruturado da seguinte forma:

```
<!DOCTYPE html>
<title>Template</title>
<!-- Aqui os conteudos da pagina -->
```

A marcação mostrada anteriormente valida como HTML5, contudo a marcação a seguir não valida.

```
<!DOCTYPE html>
<title>Template</title>
<!-- Aqui os conteúdos da página -->
```

Notar que na primeira marcação o comentário HTML inserido na última linha do código mostrado contém um texto não acentuado e na segunda marcação o texto foi acentuado.

O uso de caracteres não ASCII (tais como acentuação) na marcação HTML5, mesmo em comentários, obriga que se declare explicitamente a codificação de caracteres. Se isso não for feito, a renderização dos caracteres não ASCII será truncada e o documento não passará no validador. Assim, chegamos ao template mínimo, conforme mostrado a seguir:

```
<!DOCTYPE html>
<title>Template</title>
<meta charset=utf-8>
<!-- Aqui os conteúdos da página -->
```

É evidente que esse template mínimo não deve ser usado em um site real, mas você poderá usá-lo para testes e experiências com a linguagem HTML5.

A sintaxe para DOCTYPE é case insensitive (não distingue minúsculas de maiúsculas), o que significa que qualquer uma das sintaxes seguintes é perfeitamente válida.

```
<!DOCTYPE html>
<!doctype html>
<!DOCTYPE HTML>
<!doctype HTML>
<!DoctyPE hTMl>
```

Apesar da flexibilidade com a sintaxe, aconselho a usar a primeira ou a segunda forma mostrada, que são aquelas preferidas pela maioria dos desenvolvedores. Escolha entre as duas a que mais lhe agrada e siga com ela.

É muito provável que você esteja familiarizado com uma sintaxe HTML para declaração de caracteres no documento com o uso do elemento `meta`, conforme mostrado a seguir.

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

Na sintaxe HTML5, essa declaração foi abreviada para:

```
<meta charset=utf-8>
```

Pelo princípio da compatibilidade, a HTML5 continua a oferecer suporte para a sintaxe da HTML e você, se preferir, pode usá-la nos documentos novos ou deixá-la como está nos documentos existentes migrados para a HTML5.

Tal como na HTML, em HTML5 o uso de aspas ou apóstrofes nos valores dos atributos é facultativo; assim, as duas sintaxes mostradas a seguir são válidas.

```
<meta charset=utf-8>
<meta charset="utf-8">
```

Como já dissemos, a sintaxe HTML5 é compatível tanto com a HTML quanto com a XHTML; portanto, se você está criando um documento XHTML, terá de fechar as tags e usar aspas em nomes de atributo. Deverá escrever como mostrado a seguir:

```
<meta charset="utf-8" />
```

1.8.2 Template HTML5 – versão 1

A seguir, vamos examinar e comentar um template HTML5 com base em sintaxe HTML4.

A seção `head` de um documento HTML5 deve conter obrigatoriamente o elemento `title` e opcionalmente são admitidos os seguintes elementos naquela seção: `base`, `command`, `link`, `meta`, `noscript`, `script`, `style`. Desses sete elementos opcionais apenas o elemento `command` não é previsto nas versões anteriores da linguagem HTML, pois se trata de um novo elemento da HTML5.

Em aplicações que já possuem um título intrínseco, por exemplo, um e-mail escrito em HTML que já tem um campo destinado ao assunto, o elemento `title` pode ser dispensado.

Observe a marcação a seguir:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>Template</title>
    <meta name="description" content="Template HTML5 do livro do Maujor.">
    <meta name="keywords" content="lista de palavras-chave">
```

```
    <meta name="author" content="Mauricio Samy Silva">
    <meta name="generator" content="HTML-Kit 292">
    <meta name="robots" content="all">
    <link rel="stylesheet" href="mais.css">
    <style>
    /* estilos incorporados */
    </style>
<script src="path/modernizr-1.5.min.js"></script>
<script src="scripts.js"></script>
</head>
<body>
    <!-- Aqui os conteúdos da página -->
</body>
</html>
```

O elemento `meta` destina-se a fornecer metadados, ou seja, informações adicionais sobre a página. Informações essas que não podem ser transmitidas com uso dos demais elementos permitidos na seção `head` do documento.

Os atributos previstos nas especificações da HTML5 para o elemento `meta` são: `name`, `http-equiv`, `content` e `charset`, além dos atributos globais. Sempre que o atributo `name` ou `http-equiv` for definido, deve-se definir também o atributo `content`. Nesses casos, o metadado é do tipo par nome/valor no qual o valor do atributo `name` é o nome no par e o valor do atributo `content` é o valor no par.

A quantidade de elementos `meta` a inserir na seção `head` do documento e o tipo de metadado a codificar são funções de cada projeto; no exemplo mostrado, os elementos foram inseridos apenas a título de ilustração, assim aquela seção não pretende ser um guia rígido de metadados.

No exemplo, inserimos na seção `head`, também a título de ilustração, os elementos `link`, `style` e `script`.

Notar que todos os valores de atributos na marcação mostrada estão entre aspas.

1.8.3 Template HTML5 – versão 2

Vamos examinar e comentar a seguir um template HTML5 idêntico ao template mostrado no item anterior, também com base em sintaxe HTML4.

Observe a marcação a seguir:

```
<!DOCTYPE html>
<html lang=pt-br>
```

```
<head>
    <meta charset=utf-8>
    <title>Template</title>
    <meta name=description content="Template HTML5 do livro do Maujor.">
    <meta name=keywords content="lista de palavras-chave">
    <meta name=author content="Mauricio Samy Silva">
    <meta name=generator content="HTML-Kit 292">
    <meta name=robots content=all>
    <link rel=stylesheet href=mais.css>
    <style>
    /* estilos incorporados */
    </style>
    <script src="path/modernizr-1.5.min.js"></script>
    <script src=scripts.js></script>
</head>
<body>
    <!-- Aqui os conteúdos da página -->
</body>
</html>
```

Nessa versão mostramos a sintaxe alternativa que admite valores de atributos sem aspas.

Mas por que conservamos as aspas em alguns valores do atributo `content`? Notar que para os casos em que isso ocorreu o valor do atributo é uma frase ou conjunto de palavras separadas por espaços e, se tirarmos as aspas, o valor do atributo deixa de ser único, o que não é válido. Com aspas, o parser HTML interpreta o valor do atributo como uma só string. Notar que, para o elemento `meta` que fornece informações para os robôs que visitam a página, o valor do atributo `content` está sem aspas, pois tal valor é uma só palavra.

1.8.4 Template XHTML5 – versão 1

Vamos examinar e comentar a seguir um template XHTML5 com base em sintaxe XML e destinado a ser servido com o tipo de MIME `text/html`.

Observe a marcação a seguir:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="utf-8" />
    <title>Template</title>
```

```
    <meta name="description" content="Template HTML5 do livro do Maujor." />
    <meta name="keywords" content="lista de palavras-chave" />
    <meta name="author" content="Mauricio Samy Silva" />
    <meta name="generator" content="HTML-Kit 292" />
    <meta name="robots" content="all" />
    <link rel="stylesheet" href="mais.css">
    <style>
    /* estilos incorporados */
    </style>
    <script src="path/modernizr-1.5.min.js"></script>
    <script src="scripts.js"></script>
</head>
<body>
    <!-- Aqui os conteúdos da página -->
</body>
</html>
```

Nessa versão o template é compatível com a sintaxe XML, pois valores de atributo estão entre aspas e elementos vazios estão fechados. Convém ressaltar que, embora escrito com a sintaxe XML, o template não está apto a se valer das funcionalidades da XML e deverá ser servido com o tipo de MIME `text/html`.

1.8.5 Template XHTML5 – versão 2

Vamos examinar e comentar um template XHTML5 com base em sintaxe XML, destinado a ser servido com o tipo de MIME `application/xml` ou `application/xhtml+xml`.

Observe a marcação a seguir:

```
<!DOCTYPE html>
<html lang="pt-br">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-br">
<head>
    <meta charset="utf-8" />
    <title>Template</title>
    <meta name="description" content="Template HTML5 do livro do Maujor." />
    <meta name="keywords" content="lista de palavras-chave" />
    <meta name="author" content="Mauricio Samy Silva" />
    <meta name="generator" content="HTML-Kit 292" />
    <meta name="robots" content="all" />
    <link rel="stylesheet" href="mais.css" />
    <style>
    /* estilos incorporados */
    </style>
```

```
    <script src="path/modernizr-1.5.min.js"></script>
    <script src="scripts.js"></script>
</head>
<body>
    <!-- Aqui os conteúdos da página -->
</body>
</html>
```

Nessa versão o template é compatível com a sintaxe XML e apto a valer-se das funcionalidades da XML devendo ser servido com o tipo de MIME `application/xml` ou `application/xhtml+xml`.

Em documentos XML, não há necessidade da declaração de DOCTYPE, pois por padrão a renderização acontece em modo standard. Também não há necessidade de declaração da codificação de caracteres, pois por padrão ela é em UTF-8. Contudo, é indispensável que se declare o atributo `xmlns` no elemento-raiz do documento apontando para o arquivo que contém o namespace para XML. Notar ainda que o atributo para definição do principal idioma no qual os conteúdos do documento foram escritos é `xml:lang` e não `lang` como na sintaxe HTML.

1.8.6 Template XHTML5 – versão 3

Com a chegada da HTML5, o Consórcio W3C criou o conceito de marcação poliglota e publicou especificações e diretrizes para a criação de tal marcação.

As especificações definem marcação poliglota assim:

> Documento que usa marcação poliglota é um documento HTML5, que é, ao mesmo tempo, um documento HTML e XML escrito conforme um conjunto de regras definidas. Marcação poliglota é processada e compatível com a HTML e com a XHTML, segundo as regras das especificações para a HTML5.

Nessa versão o template usa uma marcação poliglota, conforme segue:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="pt-br" xml:lang="pt-br">
<head>
    <meta charset="utf-8" />
    <title>Template</title>
    <meta name="description" content="Template HTML5 do livro do Maujor." />
    <meta name="keywords" content="lista de palavras-chave" />
    <meta name="author" content="Mauricio Samy Silva" />
    <meta name="generator" content="HTML-Kit 292" />
```

```
    <meta name="robots" content="all" />
    <link rel="stylesheet" href="mais.css" />
    <style>
    /* estilos incorporados */
    </style>
    <script src="path/modernizr-1.5.min.js"></script>
    <script src="scripts.js"></script>
</head>
<body>
    <!-- Aqui os conteúdos da página -->
</body>
</html>
```

A marcação poliglota deve satisfazer aos seguintes requisitos:

O DOCTYPE, como prevê a XML, deve ser declarado em maiúsculas e o html que se segue ao DOCTYPE deve ser em minúsculas. Lembre-se de que em HTML a sintaxe é insensível ao tamanho de caixa, mas em marcação poliglota é rígida.

Os elementos `html`, `head`, `title` e `body` devem obrigatoriamente constar da marcação.

O elemento `html` deve obrigatoriamente conter os atributos `xmlns`, `lang` e `xml:lang`.

Os namespaces XML permitidos em marcação poliglota são aqueles para as tecnologias SVG e MathML e devem ser declarados nos elementos `svg` e `math` respectivamente, conforme sintaxe mostrada a seguir:

```
<svg xmlns = "http://www.w3.org/2000/svg">...</svg>
<math xmlns="http://www.w3.org/1998/Math/MathML">...</math>
```

O outro namespace permitido é para `xlink` que pode ser declarado tanto no elemento `html` quanto no elemento `svg` que contenha essa funcionalidade, como mostrado a seguir:

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:xlink="http://www.w3.org/1999/xlink" lang="pt-br" xml:lang="pt-br">
```

ou

```
<svg xmlns:xlink="http://www.w3.org/1999/xlink">...</svg>
```

A declaração de caracteres deve constar da seção `head` do documento.

A sintaxe da marcação deve seguir as regras da XML. Para maiores informações sobre sintaxe XML, consulte: `http://www.maujor.com/w3c/xhtml10_2ed.html` e `http://www.maujor.com/w3ctuto/xhtmlfaq.html`.