

**LIGA DE ENSINO DO RIO GRANDE DO NORTE
CENTRO UNIVERSITÁRIO DO RIO GRANDE DO NORTE
BACHARELADO EM SISTEMAS DE INFORMAÇÕES**

PADRÕES DE DESENVOLVIMENTO COM HTML5

ROBSON COSTA DO NASCIMENTO

PADRÕES DE DESENVOLVIMENTO COM HTML5

**NATAL/RN
2013**

ROBSON COSTA DO NASCIMENTO

PADRÕES DE DESENVOLVIMENTO COM HTML5

Trabalho de Conclusão de Curso
apresentado ao Centro Universitário do
Rio Grande do Norte como requisito final
para obtenção do título de bacharel em
Sistemas de Informação.

Orientador:

Profº. Esp. Romulo Fagundes Catanhede,

**NATAL/RN
2013**

BIBLIOTECA UNI-RN

004 N244p - RB: 37274 (M002088)

Título: Padrões de desenvolvimento com

Autoria: Nascimento, Robson Costa do
2013, ex. 1

**Catálogo na Publicação – Biblioteca do UNI-RN
Setor de Processos Técnicos**

Nascimento, Robson Costa do.

Padrões de desenvolvimento com HTML5. / Robson Costa do
Nascimento. – Natal, 2013.
52 f.

Orientador: Profº. Rômulo Fangundes Catanhede.

Monografia (Graduação em Sistemas de Informação) – Centro
Universitário do Rio Grande do Norte.

1. Sistema de Informação - Monografia. 2. HTML5 - Monografia. 3.
Desenvolvimento Web - Monografia. 4. Compatibilidade - Monografia. I.
Catanhede, Rômulo Fangundes. . II. Título.

RN/UNI-RN/BC

CDU 004

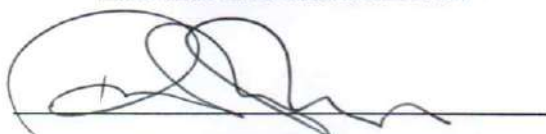
ROBSON COSTA DO NASCIMENTO

PADRÕES DE DESENVOLVIMENTO WEB COM HTML5

Trabalho de Conclusão de Curso
apresentado ao Centro Universitário do
Rio Grande do Norte como requisito
final para obtenção do título de
Bacharel em Sistemas de Informação.

Aprovado em: ____/____/____

BANCA EXAMINADORA



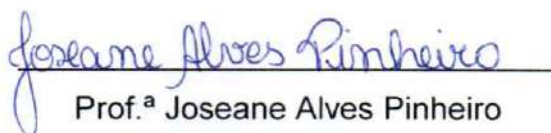
Prof. Romulo Fagundes Catanhede

Orientador



Prof. Alexandre Luiz Galvão Damasceno

Membro



Prof.ª Joseane Alves Pinheiro

Membro

DEDICATÓRIA

Dedico essa monografia aos meus pais,
irmãos, esposa e filhos, grandes
incentivadores nesta minha caminhada.

AGRADECIMENTO

Agradeço primeiramente a Deus, que está comigo em todos os momentos, me dá forças, me concede luz e todas as oportunidades para evoluir nesta vida.

Agradeço a todos os meus professores que durante esses quatro anos me passaram ensinamentos que certamente utilizarei em minha vida profissional.

Agradeço à minha família, pais e irmãos, que me apoiaram em meus anos de estudo e me incentivaram em todos os momentos.

Agradeço ao meu irmão Ivan por me mostrar o quanto era importante a graduação para meu futuro e sempre insistir para que eu aceita-se esse desafio.

Agradeço aos amigos, em especial Hanes Barbosa que foi uma das pessoas que me mostrou a importância de ingressar nesta jornada e a Danilo pelo incentivo que sempre me passou.

Agradeço a minha esposa Edivania que esteve ao meu lado me passando confiança e compartilhando minhas conquistas e dificuldades durante o curso.

Agradeço aos meus filhos Sandy e Gustavo que compreenderam minha ausência em muitos momentos e sempre foram a minha maior razão para terminar essa caminhada.

Agradeço ao Professor Rômulo pela orientação deste trabalho, por suas sugestões, correções e constante trocas de ideias que fomentaram este estudo.

Agradeço aos colegas de faculdade que continuaram acreditando e chegaram ao final do curso, cabe citar: Leandro, Ivaldo, André, Lays, Rafaela, Rafael, Emerson, Caio, Manuel, Daniel, Sthenio e Ronaldo.

Agradeço enfim a todos os que direta e indiretamente contribuíram e me apoiaram desde a elaboração deste trabalho, até a obtenção desta graduação.

Muito obrigado.

RESUMO

O surgimento do HTML5 no ano de 2010 agregou diversos novos recursos ao desenvolvimento web. Entre suas premissas um dos principais objetivos é prover soluções multiplataforma sem a necessidade de sistemas proprietários ou aplicações nativas, bem como dispositivos com configurações específicas. Através desse trabalho foi realizado um estudo sobre as novas especificações do HTML5, mostrando suas principais características, novos elementos como formulários, armazenamento, multimídia, geolocalização, tags e API's entre outros, tomando como base material especializado como livros e artigos sobre o assunto. Com o uso da biblioteca **Modernizr** foram realizados testes dos principais recursos do HTML5 com o intuito de mapear como está atualmente o suporte dos principais navegadores em relação às novas especificações. Através de um aplicativo protótipo contendo cadastros e códigos específicos, foi realizada a prova de conceito que teve como objetivo verificar o comportamento dos novos elementos de formulários e os objetos de armazenamento SQLite e indexedDB nos browsers nativos de três dispositivos móveis com diferentes arquiteturas e sistemas operacionais.

Palavras-Chave: HTML5, Desenvolvimento WEB, Compatibilidade, Padronização.

ABSTRACT

The emergence of HTML5 in 2010 added many new features to web development . Among its premises a major goal is to provide multi-platform solutions without the need for proprietary systems or native applications as well as devices with specific settings . Through this work, a study on the new HTML5 specifications was performed , showing its main features , new elements such as forms , storage , multimedia , geolocation , tags and APIs among others, taking as specialized base such as books and articles on the subject . Using the Modernizr library testing of key features of HTML5 in order to map as is currently the major browsers support regarding new specifications were made . Through a prototype application containing records and specific codes , a proof of concept that aimed to verify the behavior of the new form elements and objects SQLite and IndexedDB storage in native browsers of three mobile devices with different architectures and operating systems was performed .

Palavras-Chave: HTML5, WEB Development, Compatibility, Standard.

LISTA DE FIGURAS

Figura 01 – API's e Elementos do HTML5.....	15
Figura 02 – Estrutura Semântica do HTML5	20
Figura 03 – Complexidade de uma aplicação de comunicação HTTP	23
Figura 04 – Comunicação half-duplex full-duplex	25
Figura 05 – Código de implementação do Web Socket	26
Figura 06 – Exemplos de dados chave/valor salvos localmente	28
Figura 07 – Exemplo de código SessionStorage	28
Figura 08 – Exemplo de código LocalStorage	29
Figura 09 – Exemplo de código WebSQL Storage	30
Figura 10 – Exemplo de código elemento <audio>	31
Figura 11 – Exemplo de código elemento <video>	32
Figura 12 – Figuras feitas a partir do uso do elemento <canvas>	33
Figura 13 – Exemplo de código elemento <canvas>	34
Figura 14 – Exemplo de código SVG Circle	35
Figura 15 – Exemplo de código para solução Offline	36
Figura 16 – Exemplo de código geolocalização	37
Figura 17 – Estrutura WebSQL	45
Figura 18 – Tela de cadastro de pedido do aplicativo utilizado nos testes	46
Figura 19 – Tela de cadastro de usuários do aplicativo utilizado nos testes	47

LISTA DE QUADROS

Quadro 01 – Linha do tempo - Fases de importantes da linguagem HTML.....	12
Quadro 02 – Comparação seções estruturais entre o HTML4 e o HTML5	18
Quadro 03 – Elementos de marcação semântica do HTML5.....	19
Quadro 04 – Novos Elementos de Conteúdo do HTML5	20
Quadro 05 – Compatibilidade dos browsers com os elementos e tags do HTML5..	41
Quadro 06 – Resultado Tablet Galaxy Note	47
Quadro 07 – Resultado Nokia Lumia 920	48
Quadro 08 – Resultado Iphone 4S.....	49

LISTA DE TABELAS

Tabela 01 – Browsers usados nos teste de compatibilidade	40
Tabela 02 – Dispositivos usados na prova de conceito	44

LISTA DE SIGLAS

AJAX - Asynchronous Javascript and XML	12
API - Application Programming Interface	14
DOM - Document Object Model	14
HTML - Linguagem de Marcação de Hypertexto	15
HTTP - Hypertext Transfer Protocol	15
HTTPS - HyperText Transfer Protocol Secure	15
IETF - Internet Engineering Task Force	15
JPG - Joint Photographic Experts Group	20
PNG - Portable Network Graphics	20
SQL - Structured Query Language	20
TCP - Transmission Control Protocol	20
W3C - World Wide Web Consortium	20
WHATWG - Hypertext Application Technology Working Group	20
WWW - World Wide Web	20
XHTML - Extensible Hypertext Markup Language	20

SUMÁRIO

1 INTRODUÇÃO	12
1.1 OBJETIVO GERAL	14
1.2 OBJETIVOS ESPECÍFICOS	14
1.3 METODOLOGIA	14
2 REFERENCIAL TEÓRICO	15
2.1 HTML5	16
2.2 ELEMENTOS ESTRUTURAIS	17
2.2.1 Semânticas dos novos elementos estruturais do HTML5	18
2.2.2 Novos elementos de conteúdo do HTML5	20
2.2.3 Compatibilidade	21
2.2.4 Conectividade	23
2.2.5 Armazenamento	26
2.2.6 Armazenamento por sessão	28
2.2.7 Armazenamento Local	29
2.2.8 Banco de Dados	29
2.2.9 Multimídia e Gráficos	30
2.2.9.1 Áudio e vídeo	31
2.2.9.2 Canvas	33
2.2.9.3 SVG	34
2.2.10 Offline	35
2.2.11 Geolocalização	37
2.3 HTML x FLASH	38
2.4 HTML5 x APLICAÇÕES NATIVAS	39
3 ANÁLISE DE COMPATIBILIDADE DOS NAVEGADORES AO HTML5	40
3.1 TESTES DE COMPATIBILIDADE	40
3.2 CONCLUSÕES DO TESTE DE COMPATIBILIDADE	42
4 ANÁLISE DE ELEMENTOS DE FORMULÁRIO E OBJETOS DE ARMAZENAMENTO EM BROWSERS DE DISPOSITIVOS MÓVEIS	44
4.1 ELEMENTOS UTILIZADOS	44
4.2 APLICAÇÃO	45
4.3 RESULTADO DA PROVA DE CONCEITO	47
5 CONCLUSÃO	50
REFERÊNCIAS	51
ANEXOS	53

1. INTRODUÇÃO

Conforme a W3C, 2011, o HTML é uma abreviação de Hypertext Markup Language - Linguagem de Marcação de Hipertexto. Resumindo em uma frase: o HTML é uma linguagem para publicação de conteúdo (texto, imagem, vídeo, áudio e etc) na Web.

Hipertexto é 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.

A HTML teve seu desenvolvimento iniciado em 1991 por Tim Berners-Lee, Físico britânico, cientista da computação e criador da World Wide Web (Rede Mundial de Computadores - Internet) e durante esses anos foi se aperfeiçoando através de suas especificações e versões, criando padrões e tendo cada vez mais funcionalidades agregadas. O quadro 01 mostra a linha do tempo com os fatos mais relevantes da trajetória do HTML.

Quadro 01 – Linha do tempo contendo as fases de importantes da linguagem HTML

1991	Criada a lista de discussão denominada WWW-talk com o propósito de trocar ideias e experiências sobre a HTML desenvolvida por Tim Berners-Lee
1992	Com a criação de novos navegadores e a falta de uma padronização, Dan Connolly e colaboradores fizeram um levantamento minucioso de tudo que existia no HTML
1994	Proposta a especificação HTML 2.0, uma tentativa de consolidar e unificar as diferentes formas de HTML. Foi criada uma força tarefa que tinha a frente à IETF, organização internacional que congrega interessados nos vários aspectos da internet, com a finalidade de desenvolver normas e tecnologias pertinentes a operação e a evolução da arquitetura da internet. Criada a Word Wide Web Consortium(W3C), com a finalidade de desenvolver a web a seu potencial máximo, criando normas

	e especificações aplicáveis aos diversos segmentos e setores da web.
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 o HTML com a finalidade de acabar com as implementações proprietárias. Em dezembro o HTML ERB transformou-se no Grupo de Trabalho HTML, que existe até hoje. Esse mês assinalou o início dos estudos para a implementação de uma marcação provisória denominada <i>Cougar</i>, a qual, mas tarde, transformar-se-ia no HTML4.
1997	No mês de janeiro o W3C endossou a HTML 3.2 como uma recomendação oficial Em julho foi lançada a versão rascunho para a Cougar, depois denominada HTML4. Em dezembro do mesmo ano, o W3C endossou a HTML 4 como uma Recomendação oficial.
1999	Em dezembro de 1999, o W3C publicou as Recomendações para o HTML 4.01.
2010	Surgiu a especificação do HTML5

Fonte: Próprio autor (2013)

A HTML5, é a nova especificação da W3C que tem características modulares, trazendo novos elementos (tags) orientados para auxiliar no desenvolvimento web baseado em padrões compatíveis com todos dispositivos, na marcação semântica do código e na manipulação dos elementos CSS e JavaScript através da definição de APIs da arquitetura web.

Esse trabalho busca traçar um perfil da nova versão da linguagem no sentido de prover conhecimento das especificações mais importantes, ao mesmo tempo em

que mostrar as novas especificações do HTML5 para o desenvolvimento de sites e aplicativos.

1.1. OBJETIVO GERAL

Esse trabalho tem como objetivo fornecer informações sobre o HTML5, conceitos e princípios utilizados para sua especificação, nova estrutura, elementos, tags e API's, visando demonstrar de maneira objetiva o que de novo foi agregado ao desenvolvimento web e como está sendo utilizado atualmente.

1.2. OBJETIVOS ESPECÍFICOS

- Fornecer informações relacionadas a linguagem HTML5, fazendo uma breve descrição dos principais elementos já contidos na sua especificação, com referencias de especialistas e exemplos.
- Realizar testes de comportamento das novas funcionalidades para os elementos de formulários e armazenamento local em dispositivos móveis.
- Identificar, através de ferramenta de análise, como está a compatibilidade e suporte atual nos navegadores.

1.3. METODOLOGIA

Para atingir os objetivos, a metodologia utilizada foi o estudo bibliografico sobre HTML5, levantando informações a fim de entender todas suas principais especificações já em uso, bem como aquelas que ainda estão em processo de amadurecimento.

Em um segundo momento foi realizado um estudo de caso com as funcionalidades do html5 e seu comportamento nos browsers mais utilizados pelos usuários da internet, a fim de mapear como está a compatibilidade atual.

Por fim foi utilizado um aplicativo protótipo a fim de comprovar o comportamento dos elementos relacionados a formulários e armazenamento nos navegadores de dispositivos móveis.

2. REFERENCIAL TEÓRICO

A web foi criada com o objetivo de compartilhar a informação entre os seus usuários. Essa informação por sua vez deve estar disponível a qualquer hora para ser consumida e reutilizada.

O HTML5 juntamente com o CSS3 leva o desenvolvimento para web para um novo patamar. Um patamar onde a informação está lá esperando para ser utilizada quantas vezes for necessária, onde for necessária e por qualquer meio de acesso. Seja esse meio de acesso um simples sistema de leitura de tela ou um dispositivo ultrarrevolucionário. (EIS, 2010, p. 14)

A W3C, 2010, faz questão de deixar claro que o HTML5 é uma evolução das versões anteriores e não uma revolução. Com isso existe a ideia de aproveitar ao máximo todas as especificações já existentes, readequando-as, sendo possível adicionar novas funcionalidades.

A W3C, 2010, cita ainda que as revoluções às vezes mudam o mundo para melhor. Na maioria das vezes, no entanto, é melhor desenvolver um projeto existente, em vez de jogá-lo fora. Desta forma, os autores não têm que aprender novos modelos e o conteúdo por um período mais prolongado.

Em seu desenvolvimento o HTML5 tem como um dos seus principais objetivos fornecer soluções que possam ser acessadas em qualquer dispositivo ou navegador ao qual o usuário tenha acesso. A figura 01 mostra um resumo das API's e Elementos do HTML5.

Figura 01 – API's e Elementos do HTML5



Fonte: Próprio Autor (2013)

2.1. HTML5

Para a W3C, 2007, o HTML5 estabelece como um grande diferencial em relação a versão anterior as regras de marcação que são usadas no HTML5 e no XHTML. Também são definidas APIs que formam a base da arquitetura web, conhecidas como DOM.

Document Object Model (Modelo de Documentos de Objetos) é uma tecnologia para interação com objetos HTML, XHTML e XML. A ideia por trás de DOM é poder modificar o HTML de uma página web de forma dinâmica mediante uso da biblioteca Javascript, por exemplo.

A especificação da HTML5 está definida em 10 seções:

- Infraestrutura comum;
- Semântica, estrutura e APIs para documentos HTML;
- Elementos HTML;
- Microformatos;
- Carregamento de páginas web;
- APIs para aplicações web;
- Interação com o usuário;
- APIs para comunicação;
- Sintaxe HTML;
- Sintaxe XHTML.

Outra evolução a ser citada consiste no HTML5 oferecer ferramentas para o CSS e o JavaScript desempenharem seu papel de forma otimizada, permitindo por meio de suas APIs a manipulação de características destes elementos, de forma que os websites ou aplicações continuem leves e funcionais.

Um dos principais objetivos do HTML5 é facilitar a manipulação do elemento possibilitando o desenvolvedor modificar as características dos objetos de forma intrusiva e de maneira que seja transparente ao usuário final.

Neste contexto também foram criadas novas tags, ao mesmo tempo em que a função de outras foram modificadas.

2.2. Elementos Estruturais

A especificação para o HTML5 estabelece um Modelo de Conteúdo que vem a ser a descrição do tipo de conteúdo que se espera ser inserido em cada elemento. Dessa forma, as categorias de conteúdos foram divididas em:

- Metadados;
- Fluxo;
- Seção;
- Cabeçalhos;
- Frase;
- Incorporado;
- Interativo;
- Formulários.

Os elementos de formulários podem pertencer às categorias de fluxo, de fase e/ou interativa, além de também pertencerem a uma categoria específica denominada “formulários” que se divide nas subcategorias:

- Listas;
- Rótulos;
- Envio;
- Limpeza.

A especificação para o HTML5 define uma sintaxe que é compatível tanto com a linguagem HTML quanto com ao padrão XHTML. Os documentos que usam a sintaxe HTML com o tipo MIME “text/HTML” estão em conformidade com implementações atualmente em uso.

Outra grande novidade do HTML5 é a possibilidade de atribuir às diversas seções estruturais do documento um elemento próprio, nativo da linguagem. No quadro 02 é mostrado um comparativo da estrutura de seções do HTML5 com a versão anterior.

2.2. Elementos Estruturais

A especificação para o HTML5 estabelece um Modelo de Conteúdo que vem a ser a descrição do tipo de conteúdo que se espera ser inserido em cada elemento. Dessa forma, as categorias de conteúdos foram divididas em:

- Metadados;
- Fluxo;
- Seção;
- Cabeçalhos;
- Frase;
- Incorporado;
- Interativo;
- Formulários.

Os elementos de formulários podem pertencer às categorias de fluxo, de fase e/ou interativa, além de também pertencerem a uma categoria específica denominada “formulários” que se divide nas subcategorias:

- Listas;
- Rótulos;
- Envio;
- Limpeza.

A especificação para o HTML5 define uma sintaxe que é compatível tanto com a linguagem HTML quanto com ao padrão XHTML. Os documentos que usam a sintaxe HTML com o tipo MIME “text/HTML” estão em conformidade com implementações atualmente em uso.

Outra grande novidade do HTML5 é a possibilidade de atribuir às diversas seções estruturais do documento um elemento próprio, nativo da linguagem. No quadro 02 é mostrado um comparativo da estrutura de seções do HTML5 com a versão anterior.

Quadro02 - Comparação seções estruturais entre o HTML4 e o HTML5

HTML4	HTML5
<code><body></code>	<code><body></code>
<code><div id="topo"></div></code>	<code><header></header></code>
<code><div id="navegacao"></div></code>	<code><nav></nav></code>
<code><div id="principal"></div></code>	<code><article></article></code>
<code><div id="lateral"></div></code>	<code><aside></aside></code>
<code><div id="rodape"></div></code>	<code><footer></footer></code>
<code></body></code>	<code></body></code>

Fonte: Próprio Autor (2013)

Nota-se que antes do HTML5 as seções eram divididas usando o elemento `<div>` que não tem nenhum valor semântico.

2.2.1. Semânticas dos novos elementos estruturais do HTML5

Segundo LUBBERS, 2011, HTML5 define uma nova marcação semântica para descrever o conteúdo de um elemento.

Utilizar a marcação semântica não fornece quaisquer benefícios imediatos para o usuário final, mas simplificar o design de uma página em HTML, além torná-las mais legíveis e acessíveis por máquina. Um exemplo dos seus benefícios são os motores de busca que definitivamente aproveitarão esses elementos para rastrear e indexar páginas de uma maneira mais eficiente.

Conforme LAWSONET, 2010, antigamente para marcar a delimitação de estruturas em uma página - topo, rodapé, blocos de navegação - o único elemento disponível era `<div>`. Leitores de tela, como o JAWS, possuem uma funcionalidade denominada navegação pelo teclado (keyboard navigation) que possibilita ao usuário navegar com facilidade pelos cabeçalhos, listas e tabelas desde que convenientemente marcadas com suas respectivas tags HTML.

Contudo, tal facilidade não é possível quando se trata de navegar, por exemplo, entre blocos de navegação (marcados com o elemento `div`), porque uma vez marcados com um elemento não semântico não há como agrupá-los e incluí-los em um esquema diferenciado de navegação.

HTML5 fornece um conjunto de novos elementos de seção para serem utilizados pelos navegadores modernos. O Quadro 03 mostra tipos de marcação semântica da nova versão do HTML.

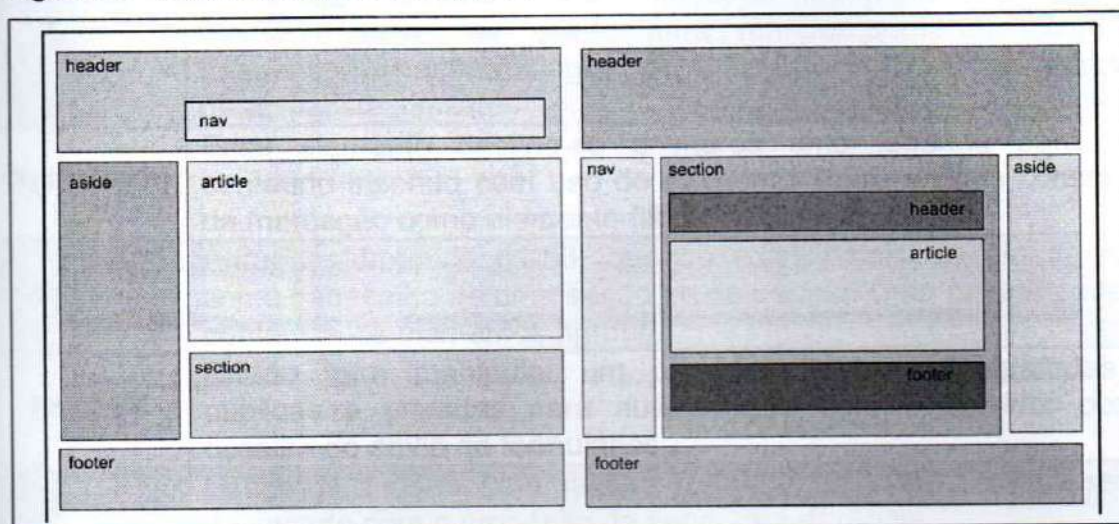
Quadro 03 – Elementos de marcação semântica do HTML5

article	Destina-se a marca um conteúdo de forma que o mesmo possa a princípio ser reusado e distribuído de forma independente, por exemplo, RSS, bloco de texto, um artigo ou ainda um mini-aplicativo embutido no conteúdo (Widget).
aside	Define um bloco de conteúdo que faz referência a um conteúdo principal que o cerca, pode está em uma barra lateral exibindo informações contextuais, assuntos relacionados, publicidade ou um grupo de navegação secundária.
footer	Destina-se a marcar o rodapé de uma seção ou da página como um todo e deve conter informações sobre o conteúdo da seção ou da página, com o seu autor, links para documentos relacionados, direitos autorais e similares.
header	Esse elemento serve para marcar o cabeçalho de uma seção ou da página como um todo e deve conter os elementos h1-h6, bem como informações sobre o conteúdo da seção.
nav	Esse elemento destina-se a marcar uma seção da página que contenha links para outras páginas ou para outras partes dentro da própria página, ou seja, uma seção contendo links para navegação externa ou interna.
section	Esse elemento destina-se a marcar, genericamente, uma seção na página. Entende-se como seção um agrupamento de conteúdos tratando de um mesmo tema e tipicamente contendo um cabeçalho.

Fonte: Portal Maujor (2010)

Tais elementos de marcação tornam os documentos baseados em HTML5 mais claros e acessíveis, uma vez que foi incluído um conjunto mais rico de marcadores, conforme estrutura mostrada na figura 02.

Figura 02 – Estrutura Semântica do HTML5



Fonte: Reproduzida do Livro "HTML5 and Web Future of the Web, Smashing Magazine" (2010)

2.2.2. Novos elementos de conteúdo do HTML5

No HTML5 alguns novos elementos de conteúdo foram incorporados enquanto outros sofreram readequações para melhor aproveitamento por parte dos desenvolvedores em seus sites e aplicações, conforme detalhamento no quadro 04.

Quadro 04 – Novos Elementos de Conteúdo do HTML5

command	Esse elemento marca um comando a ser invocado pelo usuário, podendo esse fazer parte de um menu de contexto ou uma barra de ferramentas criada com o elemento <code>menu</code> . Neste elemento são admitidos os seguintes atributos globais: <code>type</code> , <code>label</code> , <code>icon</code> , <code>disabled</code> , <code>checked</code> , <code>radiogroup</code> e <code>title</code> .
menu	Destina-se a marcar uma lista de comando e é usado como elemento contêiner para elementos <code>command</code> . O elemento <code>menu</code> admite os seguintes atributos: atributos globais, <code>type</code> e <code>label</code> .
datalist	Criado para ser usado em conjunto com o elemento <code>input</code> , fornece uma lista de sugestões para preenchimento de um campo, permitindo dessa forma a seleção por parte do usuário de uma das opções de uma lista predefinida. Para tal elemento foi criado no HTML5 o atributo <code>list</code> .
details	Esse elemento destina-se a fornecer informações textuais ou controles de formulário sobre determinada página ou sobre um conteúdo específico da página.

summary	Esse elemento foi criado para ser utilizado em conjunto com o elemento details e destina-se a fornecer um rótulo ou breve sumário das informações adicionais oferecidas por details.
figure	Serve de container para conteúdos independentes. Esse elemento deve ser usado para marcar conteúdos como imagens, ilustrações, diagramas, fotos, gráficos, vídeos, linhas de códigos e similares.
figcaption	Esse elemento destina-se a marca uma legenda para o conteúdo inserido com uso do elemento figure e deve constar da marcação como elemento-filho do elemento figure.
hgroup	Agrupa elementos h1-h6, quando existir mais de um deles como cabeçalho de uma seção ou da página. Uma página pode conter vários elementos hgroup.
keygem	Criado para manipular um par de chaves criptografadas, públicas e privadas para autenticação com o servidor por ocasião do envio de formulários.
mark	Usado para marca uma palavra ou trecho de texto que deva ser destacado com o propósito de servir de referência.
meter	Destina-se a marcar uma medida scalar compreendida entre determinados limites conhecidos.
output	Serve de container para resultados de cálculos.
progress	Usado para marcar o andamento (ou progresso) de uma tarefa, podendo ser feita de duas formas distintas: indeterminada e determinada.
source	Elemento que permite que se definam múltiplas alternativas para arquivos de mídia. Deve ser usados coo elementos-filho dos elementos áudio e vídeo apontando para diferentes arquivos alternativos para uma ou duas mídias.
time	Esse elemento tem a finalidade de fornecer uma codificação para a data-hora de modo que máquinas possam ler o código e processar, por exemplo, agendando eventos em um calendário ou lembrando datas de aniversário.
track	Destina-se a definir trilhas externas para os elementos audio e vídeo, podendo personalizar trilhas que são gravadas em arquivos externos e inseri-las no vídeo.

Fonte: Próprio Autor (2013)

2.2.3. Compatibilidade

Segundo SILVA, 2011, o HTML5 tem seus princípios de desenvolvimento baseados nos seguintes pontos:

- Compatibilidade;
- Utilidade;
- Interoperabilidade;

- Acesso Universal.

A compatibilidade com versões anteriores e a quaisquer dispositivos é certamente um dos principais objetivos do HTML5. Nesse contexto, em seu desenvolvimento deve ser considerado suporte a conteúdos existentes, elementos de versões anteriores ainda que em desuso, marcações erradas, APIs de especificações antigas, funcionalidades e comportamentos não previstos e até mesmo funcionalidades proprietárias.

O WHATWG, grupo formado por pessoas que se dedicam na evolução do HTML, tem mantido o foco para manter a retrocompatibilidade. Nenhum site deverá ser refeito totalmente para se adequar aos novos conceitos e regras das novas versões do HTML.

Essa lógica é muito parecida com as premissas dos mecanismos capazes de proporcionar uma solução alternativa para a degradação graciosa, conforme citação abaixo.

A criação de um conteúdo deve ser projetada de modo que usuários e agentes de usuários tenham acesso a ele independente dos dispositivos e condições que estejam usando, ainda que os mais precários possíveis (SILVA, 2010, p. 33).

Quando acrescentar funcionalidades em que exista suporte apenas nos navegadores mais atuais, os desenvolvedores precisarão garantir mecanismos capazes de renderizar o conteúdo de forma a não perder a informação transmitida em agentes de usuários, ainda que alguma funcionalidade esteja desabilitada.

Na prática nem tudo conseguirá seguir a risca a premissa de "compatibilidade absoluta" conforme premissas do HTML5. Um exemplo são as novas API's que são dependentes da linguagem de script para o seu funcionamento e não funcionam em browsers que não tenham suporte a mesma.

Conforme Silva, 2011, outro princípio seguido pelo HTML5 é "não reinventar a roda", ou seja, se existe uma tecnologia que já está implementada e sendo usada largamente, não tem sentido se criar uma nova funcionalidade para atender àqueles casos. A ideia é que é mais fácil ampliar funcionalidade já existente e testada do que partir do zero criando uma nova funcionalidade com o mesmo propósito.

A W3C, 2010, cita ainda como princípios do HTML5 a garantia que sua marcação possa ser usada de modo efetivo, solucione problemas reais, efetue

comunicação com segurança entre documentos, tenha independência de mídia, suporte multilíngua e garanta meios alternativos de acesso (acessibilidade).

2.2.4. Conectividade

Conforme EIJI et al, 2010, a web tem sido construída com base no conhecido paradigma de solicitação/resposta de HTTP. Um cliente carrega uma página da web e em seguida nada acontece até que o usuário clique na próxima página.

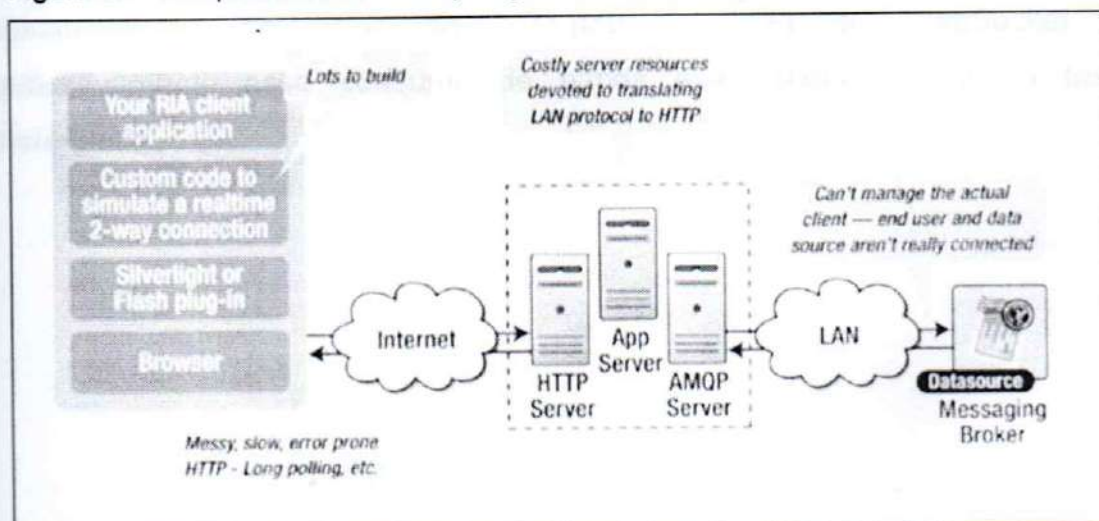
Por volta de 2005, o AJAX (Javascript Assíncrono e XML) começou a deixar a web mais dinâmica. Mesmo assim, toda a comunicação HTTP era direcionada pelo cliente, o que exigia interação do usuário ou sondagem periódica para carregar novos dados do servidor.

KITAMURA, 2013, cita que um dos problemas mais comuns para criar a ilusão de uma conexão iniciada pelo servidor é a chamada sondagem longa. Sondagem longa significa que o cliente abre uma conexão com o servidor e está permanecerá aberta até que a resposta seja enviada.

No entanto neste contexto surgiu um problema que consiste na sobrecarga de HTTP, que não é adequada para aplicativos em que uma baixa latência por parte da rede seja imprescindível ao seu funcionamento adequado.

A figura 03 mostra a complexa arquitetura necessária para a utilização do modo tradicional de comunicação via HTTP.

Figura 03 – Complexidade de uma aplicação com comunicação HTTP



Fonte: Reproduzida do livro HTML5 Programming, Apress (2013)

WebSocket é uma API originariamente incluída na especificação para a HTML5 e posteriormente desvinculada e entregue para desenvolvimento pelo W3C.

Conforme SILVA, 2011, WebSocket destina-se a definir um protocolo e suas funcionalidades que possibilitem a comunicação de um cliente web com um servidor remoto.

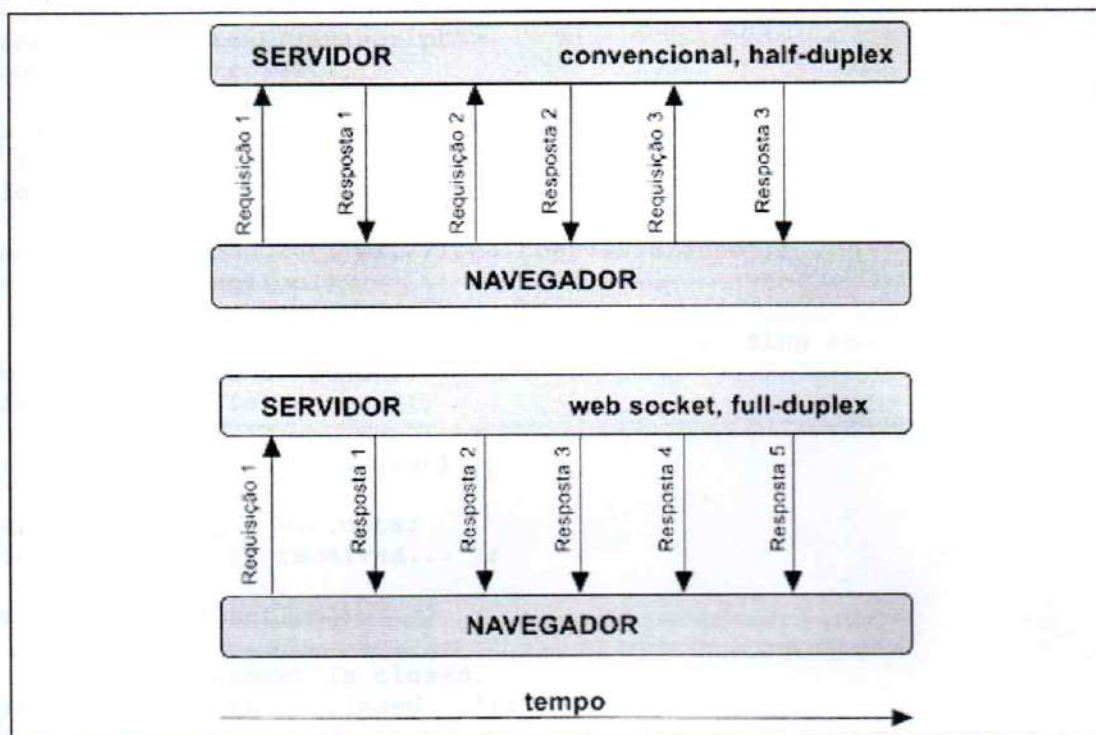
Tal API criou Protocolos de Controle de Transmissão (TCP) para socket denominados "ws" e "wss" destinados a estabelecer comunicação com o servidor com o uso de uma conexão normal e segura, a exemplo dos protocolos HTTP e HTTPS.

Uma vez criada a conexão ela permanece aberta possibilitando a comunicação permanente, full-duplex, entre cliente e servidor e só será fechada por determinação explícita do desenvolvedor.

Uma conexão com socket é full-duplex, ou seja, permite a comunicação simultânea em ambos os sentidos: partindo do cliente em direção ao servidor e partindo do servidor em direção ao cliente. Em uma transmissão half-duplex, com o uso de protocolo HTTP, por exemplo, cada solicitação do cliente ao servidor e respectiva resposta é independente e implica geração de novo tráfego, conforme mostrado na figura 04.

Para uma aplicação que necessite consultar o servidor repetidamente para verificar atualizações em tempo real, o tráfego envolvido na comunicação pode requerer um alto consumo de banda com o respectivo aumento da taxa de transferência. É o caso, por exemplo, de transmissão ao vivo de eventos, de jogos multiusuários e de salas de bate-papo. A API do WebSocket reduz significativamente esse consumo de banda e conseqüentemente a taxa de transferência.

Figura 04 – Comunicação half-duplex x full-duplex



Fonte: Reproduzida do livro HTML5 – A linguagem de Marcação que Revolucionou a WEB, Novatec (2011)

Segundo KITAMURA, 2013, outra vantagem ao se usar o WebSocket é a redução drástica de latência na comunicação. Latência em comunicação bidirecional é o tempo de espera pela resposta.

Em resumo o Web Sockets pode torna uma aplicação mais rápida, eficiente e escalável.

Cada nova tecnologia tem um novo conjunto de problemas. No caso do WebSocket um desses problemas trata-se da compatibilidade com os servidores Proxy.

O protocolo WebSocket usa o sistema de upgrade HTTP (que normalmente é usado para HTTP/SSL) para fazer "upgrade" de uma conexão HTTP para uma conexão WebSocket. Alguns servidores proxy não gostam disso e abandonarão a conexão. Assim, mesmo se um determinado cliente usar o protocolo WebSocket, talvez não seja possível estabelecer uma conexão. (KITAMURA, 2013, p.43).

A figura 05 mostra um exemplo de código de utilização do WebSocket no HTML5.

Figura 05 – Código de implementação do WebSocket

```
<script type="text/javascript">
functionWebSocketTest()
{
  if ("WebSocket" in window)
  {
    alert("WebSocket is supported by your Browser!");
    // Let us open a web socket
    var ws = new WebSocket("ws://localhost:9998/echo");
    ws.onopen = function()
    {
      // Web Socket is connected, send data using send()
      ws.send("Message to send");
      alert("Message is sent...");
    };
    ws.onmessage = function (evt)
    {
      var received_msg = evt.data;
      alert("Message is received...");
    };
    ws.onclose = function()
    {
      // websocket is closed.
      alert("Connection is closed...");
    };
  }
  else
  {
    // The browser doesn't support WebSocket
    alert("WebSocket NOT supported by your Browser!");
  }
}
</script>
```

Fonte: tutorialspoint.com (2012)

2.2.5. Armazenamento

Aplicações sempre tiveram a necessidade de armazenamento local de dados. Até o HTML4 quando se tinha a necessidade de armazenar no agente de usuário dados adotava-se os cookies. Eles permitem armazenar o status de um menu javascript que precisava ser mantido entre as páginas, lembrar o nome do usuário, o histórico de operações realizadas por ele ou a última vez que ele visitou nosso site.

Com o aumento da complexidade das aplicações baseadas em web, as limitações dos cookies ficaram evidenciadas. Citam-se as três principais:

1. Interface complexa: o código para armazenar cookies envolve complexos cálculos com datas e controle do nome de domínio;
2. Limite de armazenamento: alguns agentes de usuário permitiam o armazenamento de no máximo 20 cookies, com apenas 4KB cada;

3. Problemas relacionados a segurança.

Conforme MARK, 2010, a linguagem HTML5 fornece mecanismos que possibilita sites guardar informação na máquina do usuário e recuperá-la mais tarde. O conceito é semelhante ao usado pelos cookies, porém é projetado para trabalhar com grandes quantidades de informações.

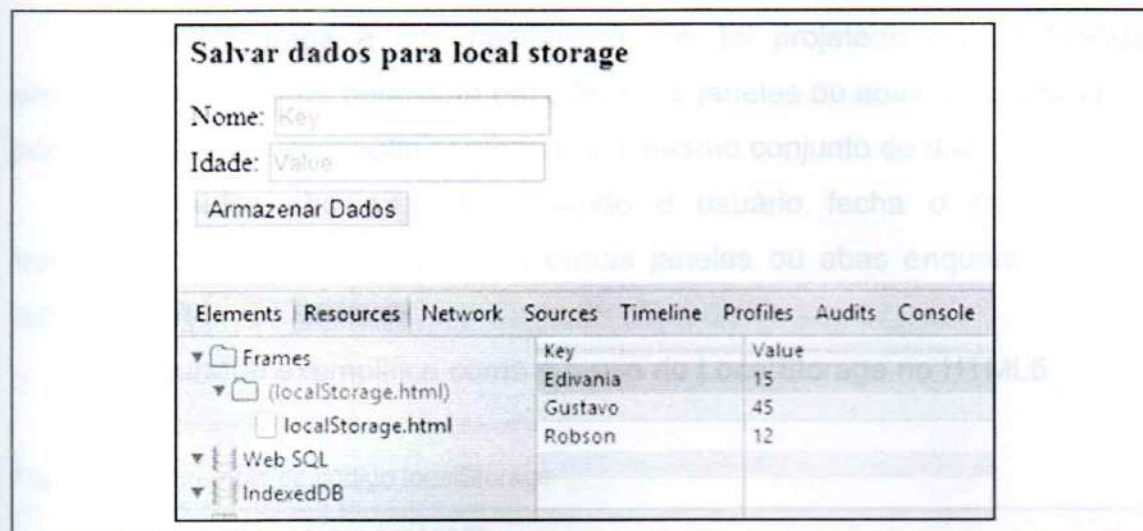
A Web Storage é uma API para armazenamento de dados, persistentes em clientes web e que tem a finalidade de corrigir a forma precária dos cookies. Para tanto o HTML5 disponibiliza mais de um objeto para ser prover armazenamento, são eles:

- Local Storage;
- Web SQL Storage;
- SessionStorage;
- Indexed DB.

O armazenamento local da linguagem HTML5 é realizado de maneira bem simples, baseado em pares compostos por chaves/valores, conforme figura 06. Desta forma você pode armazenar dados em uma chave nomeada de forma permanente (Local Storage – armazenamento local) ou de forma temporária (SessionStorage – armazenamento por sessão), utilizando-se da mesma chave para realizar a recuperação dos valores. A chave nomeada é uma string e seu valor pode ser em qualquer formato suportado pelo JavaScript.

No HTML5 poder ser utilizado ainda uma estrutura de banco de dados para trabalhar com armazenamento. O WebSQL Database permite criar e acessar bancos de dados no lado do cliente usando uma linguagem SQL, o que será visto mais a frente.

Figura 06 – Exemplos de dados chave/valor salvos localmente



Fonte: Próprio Autor (2013)

2.2.6. Armazenamento por sessão

Esse mecanismo foi projetado para armazenar dados para transação em um documento HTML, criando uma área de armazenamento independente, sendo possível existirem múltiplas transações em diferentes janelas ou abas do navegador ao mesmo tempo. Os dados pertencentes a um mesmo documento são excluídos quando o usuário fecha a janela ou aba do navegador, não havendo comunicação nem transmissão entre outras janelas ou abas.

Esse comportamento resolve uma limitação dos cookies: o fato dos dados só serem excluídos com o fechamento do navegador, desde que não tenha sido prevista uma vida útil para eles. A figura 07 mostra um exemplo de código utilizado para a implementação do armazenamento em sessão.

Figura 07– Exemplo de código sessionStorage

```
<script type="text/javascript">
if(sessionStorage.hits ){
sessionStorage.hits = Number(sessionStorage.hits) +1;
}else{
sessionStorage.hits = 1;
}
document.write("Total Hits :" + sessionStorage.hits );
</script>
```

Fonte: tutorialspoint.com (2012)

2.2.7. Armazenamento Local

O `localStorage` é um mecanismo que foi projetado com a finalidade de armazenar dados que persistem em diferentes janelas ou abas do navegador, sendo possível que uma transação compartilhe o mesmo conjunto de dados.

Os dados são excluídos quando o usuário fecha o navegador e são transmitidos e se comunicam com outras janelas ou abas enquanto o navegador estiver aberto.

A figura 08 exemplifica como é o uso do `LocalStorage` no HTML5.

Figura 08 – Exemplo de código `localStorage`

```
<script type="text/javascript">
if(localStorage.hits ){
localStorage.hits = Number(localStorage.hits) +1;
}else{
localStorage.hits = 1;
}
document.write("Total Hits :" + localStorage.hits );
</script>
```

Fonte: tutorialspoint.com (2012)

2.2.8. Banco de Dados

Outra forma de armazenamento prevista na especificação do HTML5 é em banco de dados. Em um primeiro momento usou-se o `WebSQL Storage`, implementada nos principais navegadores, exceto o Mozilla que questionava a segurança e inconsistência que a tornava imprópria para a especificação.

A W3C reconheceu os argumentos dados pelo Mozilla e em 2010 adotou a especificação denominada `IndexedDB`, abandonando a `WebSQL Storage` que foi retirado da sua especificação.

A Mozilla Developer Network (MDN), 2013, conceitua o `IndexedDB` como uma API para o armazenamento do lado do cliente de quantidades significativas de dados estruturados e para pesquisas de alto desempenho sobre esses dados usando índices.

Enquanto armazenamento DOM é útil para guardar pequenas quantidades de dados (5Mb), o `IndexedDB` é direcionada para o armazenamento de grandes quantidades de dados estruturados.

Conforme documento da W3C (2010), a API segue os seguintes conceitos:

- Requisições tantas síncronas quanto assíncronas;

- Modelo de banco de dados transicional;
- Armazenamento baseado em chave/valor;
- Requisição contínua durante todo o processo de transação;
- Uso do de eventos do DOM;
- Disponível offline.

Na figura 09 pode-se observar um exemplo de implementação para armazenamento de dados usando o WebSQL Storage.

Figura 09 – Exemplo de código WebSQLStorage

```
<script type="text/javascript">
  var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
  var msg;
  db.transaction(function (tx) {
    tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id
unique, log)');
    tx.executeSql('INSERT INTO LOGS (id, log) VALUES (1,
"foobar")');
    tx.executeSql('INSERT INTO LOGS (id, log) VALUES (2,
"logmsg")');
    msg = '<p>Log message created and row inserted.</p>';
    document.querySelector('#status').innerHTML = msg;
  });

  db.transaction(function (tx) {
    tx.executeSql('SELECT * FROM LOGS', [], function (tx, results) {
      var len = results.rows.length, i;
      msg = "<p>Found rows: " + len + "</p>";
      document.querySelector('#status').innerHTML += msg;
      for (i = 0; i < len; i++) {
        msg = "<p><b>" + results.rows.item(i).log + "</b></p>";
        document.querySelector('#status').innerHTML += msg;
      }
    }, null);
  });
</script>
```

Fonte: tutorialspoint.com (2012)

2.2.9. Multimídia e Gráficos

O suporte nativo ao processamento de gráficos, conteúdo multimídia e jogos é uma demanda crescente de mercado. Dotar de capacidade de renderizar elementos de internet rica nos navegadores, agrega grande ganho de acessibilidade e comodidade aos usuários, eliminando o requisito de uso de tecnologias adicionais ou complementares para visualização de conteúdo.

Um dos aspectos do HTML5 são os recursos nativos de multimídia que ele oferece. Todos os navegadores oferecem controles internos de vídeo (play / pause, procurar, controles de volume), assim como fazem as barras de rolagem , controles de formulários etc. O fato dos recursos de multimídia serem nativos ao invés de tratados por um plugin é vantajoso para acessibilidade porque a mídia é parte do browser (BRUCE, 2011, p. 45).

Conforme a W3C, 2011, com o HTML5, os elementos de multimídia ganham uma nova importância, visto que através de suas APIs é possível ter controle total de funções, criando seus próprios controladores e combinando-os com outras tecnologias como Canvas, SVG, CSS ou WebGL.

2.2.9.1. Áudio e vídeo

Conforme HOGAN, 2010, áudio e vídeo é uma parte importante da Internet moderna. Podcasts, previews de áudio e até mesmo como fazer vídeos são informações que estão por toda parte, e até agora, eles só foram realmente utilizáveis com o uso de plug-ins.

Segundo DYER, 2012, os elementos de áudio e vídeo introduzem vários novos atributos que determinam como o navegador irá apresentar o conteúdo de mídia para o usuário final.

Para SILVA, 2011, o elemento áudio destina-se a incorporar um som ou um stream de áudio. De uma forma bem simples, apenas declarando o caminho, é possível incorporar um som em uma página web, sem a necessidade de plugins como Flash, Quick Time, Windows Media Player, entre outros.

A figura 10 mostra um exemplo de uso do elemento <audio>.

Figura 10 – Exemplo de código elemento <audio>

```
<!DOCTYPE HTML>
<html>
<body>
<audio controls autoplay>
<source src="/html5/audio.ogg" type="audio/ogg" />
<source src="/html5/audio.wav" type="audio/wav" />
  Your browser does not support the <audio> element.
</audio>
</body>
</html>
```

Fonte: tutorialspoint.com (2012)

SILVA, 2011, salienta que antes do HTML5 para se assistir um vídeo na web era necessário um plugin de terceiro como QuickTime, RealPlayer ou Flash. O

HTML5 define uma forma padrão para incorporar vídeos em uma página web, usando o elemento <video>.

Segundo GILBERTSON, 2010, a adoção de vídeo nativo entre os navegadores providos pelo HTML5 foi rapidamente disseminada. A figura 11 mostra um exemplo de como pode ser implementado o elemento <video>.

Figura 11 – Exemplo de código elemento <video>

```
<!DOCTYPE HTML>
<head>
<script type="text/javascript">
functionPlayVideo(){
var v = document.getElementsByTagName("video")[0];
v.play();
}
</script>
</head>
<html>
<body>
<form>
<video width="300" height="200" src="/html5/foo.mp4">
    Your browser does not support the <video> element.
</video>
<input type="button" onclick="PlayVideo();" value="Play"/>
</form>
</body>
</html>
```

Fonte: tutorialspoint.com (2012)

Um dos problemas enfrentado pela nova especificação é a diferença de formatos de áudios e vídeos suportados pelos navegadores.

No que se refere a vídeos, por exemplo, enquanto os navegadores Internet Explorer, Safari, Chrome e dispositivos móveis (iPhone, iPad, Androide Windows Phone) suportam o codec de vídeo h. 264, o Firefox e o Opera tem suporte ao codec de vídeo VP8, que é colocado dentro do contêiner do WebM.

Já no tocante ao áudio mais uma vez existem diferenças entre dois grupos de navegadores. Enquanto o Internet Explorer, Chrome e Safari suportam formato mp3, Firefox e Opera apoiam o codec Vorbis dentro de um recipiente Ogg.

Para solucionar essas diferenças vários arquivos podem ser especificados para disponibilizar vídeo ou áudio codificado em formatos diferentes, para navegadores diferentes, ativando o codec a ser usado.

2.2.9.2. Canvas

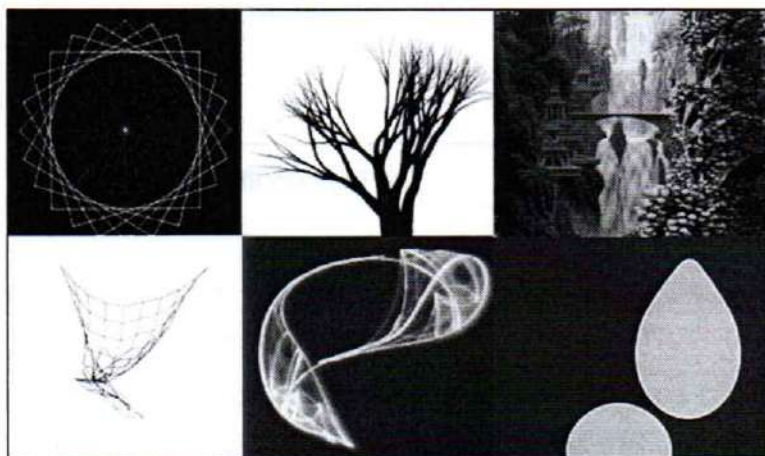
A W3C, 2011, diz que a API do canvas serve para desenhar formas, definir caminhos, criar gradientes e aplicar transformações.

Ainda segundo a W3C, 2011, o elemento `<canvas>` HTML5 é usado para desenhar gráficos, em tempo real, através de script (geralmente JavaScript) em uma área retangular dentro de uma página HTML.

Conforme ALVAREZ, 2010, para posicionar elementos em `<canvas>` considera-se o eixo de coordenadas em duas dimensões, que começa no canto superior esquerdo da tela. A tela produzida terá dimensões indicadas com os atributos "width" e "height".

Existem vários métodos para fazer desenhos, caixas, círculos, texto e adicionar imagens. A figura 12 mostra o exemplo do que pode ser criado utilizando o elemento `<canvas>` em combinação com Javascriptpt.

Figura 12 – Figuras feitas a partir do uso do elemento `<canvas>`



Fonte: geekproject.com.br (2012)

De acordo com SILVA, 2010, é necessário que o desenvolvedor faça uma análise criteriosa para o uso desse elemento, uma vez que não se deve usá-lo se houver um elemento mais apropriado.

Existem inúmeras possibilidades de criação com o elemento `<canvas>`, incluindo animações, fator que possibilita a personalização de itens com movimento simples, trazendo a independência de formatos animados como o GIF e potencialidade de interação com usuário e jogos. Algumas vantagens são:

- Superfície 2D de alta performance;
- Desenho constante – tudo é um pixel;

- Poder salvar imagem resultante como png ou jpg.

Entre algumas desvantagens estão:

- Não há nós DOM para qualquer coisa que você desenhar. Trata-se de todos os pixels;
- Pobres capacidades de renderização de texto;
- Exige que você desenhe manualmente cada elemento da interface;
- Falta de animação e apoio a acessibilidade.

A figura 13 exibe um exemplo com a utilização do elemento <canvas>.

Figura 13 – Exemplo de código elemento <canvas>

```
<!DOCTYPE HTML>
<html>
<head>
<style>
#mycanvas{
border:1px solid red;
}
</style>
</head>
<body>
```

Fonte: tutorialspoint.com (2012)

2.2.9.3. SVG

De acordo com a W3C, 2011, o SVG (Scalable Vectorial Graphics) é uma SVG como uma plataforma para gráficos bidimensionais. O formato SVG é uma especificação de linguagem baseada em XML para representar arquivos vetoriais estáticos ou animados.

Ela tem duas partes: um formato de arquivo baseado em XML e uma API de programação para aplicações gráficas.

As principais características do SVG incluem formas, texto e gráficos incorporados, com muitos estilos de pintura diferentes. Ele suporta script através de linguagens como ECMA Script e tem suporte completo para animação.

SVG é usado em muitas áreas de negócio, incluindo gráficos para a Web, animações, interfaces de usuário, intercâmbio de gráficos, impressão e saída impressa, aplicações móveis e design de alta qualidade.

O SVG também é aplicado na criação de jogos, podendo misturar-se com o <canvas> para se obter resultados mais completos. As vantagens para o seu uso são:

- Independência de resolução de tela;
- Bom suporte a animação com JavaScript, por exemplo;
- Formato de ficheiro XML facilitando a acessibilidade aos seus documentos.

Quanto às desvantagens pode-se citar:

- Renderização lenta para documentos mais complexos;
- Necessidade de ser combinado com o canvas para soluções de jogos.

A figura 14 exemplifica como usar o SVG Circle.

Figura 14 – Exemplo de código SVG Circle

```
<!DOCTYPE html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Circle</h2>
<svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
<circle id="redcircle" cx="50" cy="50" r="50" fill="red" />
</svg>
</body>
</html>
```

Fonte: tutorialspoint.com (2012)

2.2.10. Offline

Em sua especificação a W3C, 2010, conceitua que a criação de aplicações offline tem a finalidade de possibilitar aos usuários continuar interagindo com uma aplicação ou documento, mesmo que a conexão com a Internet seja momentaneamente descontinuada.

Para torna isso possível é criado um “manifesto” que faz a relação dos arquivos necessários para o funcionamento offline da aplicação.

Segundo SILVA, 2011, o navegador ou dispositivo criará uma cópia em cachê dos arquivos necessários para o uso offline e deles fará uso quando o

usuário estiver em um local sem conexão ou quando houver uma perda de conexão.

O Manifesto é um arquivo de texto puro que informa ao navegador quais são os arquivos necessários e não necessários para uso offline. O manifesto contém um título e três seções nas quais são listados três tipos de arquivos:

- Arquivos necessários para uso offline;
- Arquivos não necessários para uso offline;
- Arquivos alternativos para uso offline.

A atualização com novas versões dos arquivos não acontece mesmo que o usuário utilize a aplicação de modo online, ou seja, conectado a internet. Para que tal atualização ocorra existem três hipóteses:

- O usuário apagar os dados de armazenamento em cachê para a aplicação;
- O desenvolvedor atualizar o arquivo manifesto;
- O desenvolvedor usar o objeto `applicationCache` para verificar o status do cache e atualizá-lo.

O application cache é uma API prevista nas especificações do HTML5 que define métodos, propriedades e eventos destinados a monitorar o estado dos dados armazenados para uma aplicação offline.

A figura 15 mostra um exemplo de como utilizar soluções offline no html5 utilizando o arquivo manifesto.

Figura 15 – Exemplo de código para solução Offline

```
<!DOCTYPE HTML>
<html manifest="clock.appcache">
<head>
<title>Clock</title>
<script src="clock.js"></script>
<link rel="stylesheet" href="clock.css">
</head>
<body>
<p>The time is: <output id="clock"></output></p>
</body>
</html>
```

2.2.11. Geolocalização

A especificação do HTML5 criou uma API para fornecer os dados da localização de um cliente via script.

Segundo PILGRIM, 2011, a API de geolocalização permite compartilhar localização de um determinado usuário em sites confiáveis. A latitude e longitude são disponibilizadas na página via Java Script, que por sua vez pode ser enviado a um servidor web e fazer coisas como encontrar locais ao seu redor ou mostrar sua posição em um mapa.

Conforme a W3C, 2011, obter a localização do dispositivo do usuário na maioria dos casos implica em tomar conhecimento de sua localização, comprometendo assim a sua privacidade. Para tanto a especificação a esse elemento diz que obrigatoriamente a API deve fornecer um mecanismo que permita ao usuário escolher se deseja ou não compartilhar essa informação, ou seja, trata-se de uma tarefa assíncrona uma vez que depende da autorização do usuário para acesso a essa informação.

Ainda é necessário, de acordo com a especificação, um mecanismo que permita a qualquer momento a revogação deste compartilhamento.

A localização é representada por coordenadas que mostram a latitude e a longitude, recebendo um objeto do tipo "Position" e com um atributo "Coordinates" que contém uma série de informações.

É importante deixar claro que a API de Geolocalização apenas retorna informações sobre a localização do usuário, não sendo responsável pela construção de mapas de localização ou tratamento dos dados obtidos, o que deve ser feito com o uso de serviços de mapas de terceiros ou aplicações específicas.

A figura 16 mostra um exemplo de código para se obter as coordenadas de localização pela API.

Figura 16 – Exemplo de código geolocalização

```
<script type="text/javascript">

    $(document).ready(function() {

//Verifica se o Navegador tem suporte para Geolocalização
if(navigator.geolocation){

        //Retorna as informações ou erro
        navigator.geolocation.getCurrentPosition(
        function(position){
            varlat = position.coords.latitude;
            varlon = position.coords.longitude;

// Iniciando o GeoCoder
            vargeocoder = new GClientGeocoder();
            geocoder.getLocations("(" + lat + ', ' + lon + ")", localizacao);

            function localizacao(retorno){

                if (!retorno || retorno.Status.code != 200) {
                    //Tratar o retorno.Status.code aqui
                } else {
                    place = retorno.Placemark[0];
                    alert(place.address);
                }
            }

        })
    }
</script>
```

Fonte: tutorialspoint.com (2012)

2.3. HTML x FLASH

Segundo SILVA, 2011, HTML5 permite a integração de vídeo e áudio em algumas páginas da web sem usar o flash. Assim, o conteúdo multimídia pode ser lido por um navegador, sem qualquer plug-in, em especial.

É possível também incorporar animações e elementos dinâmicos 2D e 3D e criar jogos usando HTML5, sendo essa uma grande evolução uma vez que hoje a maioria dos jogos online são baseados em Flash ou em outras tecnologias proprietárias.

Em novembro de 2011 a empresa ADOBE (desenvolvedora da tecnologia flash) anunciou que descontinuará o flash, apontando como justificativa que seu produto não teria como competir com o HTML5.

“Uma das grandes razões foi a dura competição com o HTML5, que é muito mais suportado em navegadores portáteis do que o Flash pode esperar ser”. (CHAMBERS, Mike, 2011)

Serviços como Google Voice, o YouTube e o Vimeo já migraram suas bases para o HTML5.

2.4. HTML5 x APLICAÇÕES NATIVAS

Uma das grandes discussões em relação ao HTML5 é quanto a sua utilização ao invés de aplicações nativas. Apps nativos são distribuídos através de lojas de aplicativos e mercados controlados por proprietários de plataformas. HTML5 é distribuído através das regras da web aberta.

Muitos recursos do HTML5 imitam o comportamento de aplicações nativas em dispositivos conectados apresentando os seguintes pontos positivos:

- A capacidade de usar elementos de dispositivos de hardware dentro da página web: câmera, microfone, teclado, GPS, etc;
- Otimização dos elementos da página web para interface Multi-Touch;
- Armazenamento em dispositivos que funcionem offline para registro automático de dados ou armazenamento de preferências do usuário;
- Otimização da página web de acordo com o tamanho da tela.

3. ANÁLISE DE COMPATIBILIDADE DOS NAVEGADORES AO HTML5

Segundo W3C, 2011, antigamente, para que uma nova versão do HTML ou do CSS fosse lançada, todas as ideias listadas na especificação deveriam ser testadas e desenvolvidas para então serem publicadas para o uso dos browsers e os desenvolvedores.

Esse método foi mudado com o HTML5 e o CSS3. A partir dessas versões as duas tecnologias foram divididas em módulos. Isso significa que a comunidade de desenvolvedores e os fabricantes de browsers não precisam esperar que todo o padrão seja escrito e publicado para utilizarem as novidades das linguagens.

Por tal motivo permitiu-se que os navegadores avançassem de maneiras distintas no que se referem ao suporte as especificações do HTML5.

Com isso ao implementar uma solução com HTML5 o desenvolvedor precisa realizar testes de compatibilidade para saber se um determinado elemento usado terá funcionamento em todos os browsers afim de não correr o risco de ter problemas de acessibilidade por parte dos usuários aos seus sites e aplicativos web.

Baseado neste foi proposto pelo autor deste trabalho a realização de testes em alguns dos principais elementos do HTML5 para verificar o atual suporte nos principais navegadores.

3.1. TESTES DE COMPATIBILIDADE

Uma vez que a especificação do HTML5 ainda não está fechada e muitas das novas funcionalidades não tem suporte em todos os browsers, foram adotados quatro navegadores para realizar a verificação de como se encontra atualmente o suporte e compatibilidade à nova versão da HTML, conforme tabela 01.

Tabela 01 - Browsers usados nos teste de compatibilidade

BROWSER	VERSÃO
Google Chrome	30.0.1599.101
Mozilla Firefox	25.0
Internet Explorer	11.0.9600.16384
Safari	5.1.7

Fonte: Próprio Autor (2013)

A escolha pelos browsers Mozilla Firefox, Google Chrome, Safari da Apple e o Internet Explore é justificada uma vez que os quatro são usados por 95,1% de todos os usuários que navegam na internet, segundo dados do mês de setembro de 2013 fornecidos pelo site w3schools que realiza a medição mensal do uso dos navegadores por parte dos usuários. Cada um possui particularidades e diferentes motores de renderização.

Para a medição da compatibilidade foi utilizada a biblioteca "**Modernizr**" baseada em javascript (ANEXO A) que detecta a disponibilidade de implementações nativas para a nova geração de tecnologias web, isto é, características que se originam das especificações da HTML5 e CSS3. Várias dessas características já estão implementadas, principalmente nos navegadores mais atuais e o **modernizr** nos informar quais dessas características já estão presentes em cada um deles.

Os testes são realizados através da escolha de recursos de cada elemento que se deseja verificar. Para tanto foram selecionadas as principais API's que já estão sendo usadas pela comunidade de desenvolvedores e testadas de forma individual. Os resultados obtidos podem ser vistos no quadro 05.

Quadro05 - Compatibilidade dos browsers com os elementos e tags do HTML5

Elemento	 Chrome	 Firefox	 IE	 Safari
ApplicationCache	☒	☒	☒	☒
Local Storage	☒	☒	☒	☒
SessionStorage	☒	☒	☒	☒
IndexedDB	☒	☒	☒	☒
WebSQL	☒	☒	☒	☒
WebSockets	☒	☒	☒	☒
Áudio	☒	☒	☒	☒
Video	☒	☒	☒	☒
Canvas	☒	☒	☒	☒
SVG	☒	☒	☒	☒
WebGL	☒	☒	☒	☒

Geolocalização	☒	☒	☒	☒
Cross-windowMessaging	☒	☒	☒	☒
OfflineApplications	☒	☒	☒	☒
File API	☒	☒	☒	☒
Touch	☐	☐	☐	☐

Fonte: Próprio Autor (2013)

3.2. CONCLUSÕES DO TESTE DE COMPATIBILIDADE

Após a análise foi verificado que os avanços são cada vez mais evidentes no que se refere ao suporte ao HTML5 por parte dos browsers. O uso cada vez maior de suas especificações por parte dos desenvolvedores faz com que as empresas responsáveis concentrem um grande esforço para possibilitar cada vez uma melhor experiência em suas ferramentas de acesso.

A Microsoft, desenvolvedora do Internet Explorer, criou o HTML5 labs. Um espaço no qual a empresa faz os protótipos de especificações estáveis de entidades de padrões como o W3C, possibilitando ao desenvolvedor ter contato com protótipos como IndexedDB, WebSockets, FileAPI e a API Media Capture e outros.

Na mesma direção podemos encontrar o Mozilla Develor Network que concentra uma seção inteira sobre HTML5 aonde se pode ter acesso aos códigos com testes dos elementos e API'S de acordo com a especificação em curso.

Os principais elementos testados tiveram respostas positivas com pequenas variações principalmente nos objetos de armazenamento.

Conforme os dados obtidos e mostrados no quadro 5, o objeto de armazenamento indexDB só não foi suportado pelo Internet Explorer e pelo Safari, salientando que a o browser da Apple não utiliza em suas especificações esse objeto.

O WebGL, que não é uma especificação do HTML5, mas sim um script para criação de elementos 2D e 3D em conjunto com a API do Canvas só não obteve suporte no Safari.

Já o elemento Touch foi o único que não apresentou suporte em nenhum dos navegadores utilizados nos testes.

Os resultados mostram dos 16 elementos testados, 12 tiveram suporte em todos os browsers, 3 obtiveram resultados parciais e apenas 1 elemento ainda não tem sua especificação suportada pelos navegadores utilizados.

Diante disso, cabem às empresas e desenvolvedores que utilizam as especificações do HTML5 analisar de forma criteriosa quais elementos já estão compatíveis nos principais browsers a fim de garantir a acessibilidade aos conteúdos independente do navegador utilizado pelo usuário.

4. ANÁLISE DE ELEMENTOS DE FORMULÁRIO E OBJETOS DE ARMAZENAMENTO EM BROWSERS DE DISPOSITIVOS MÓVEIS

Segundo KLEINERT, 2011, o HTML5 traz grandes melhorias, tanto para os desenvolvedores que criam formulários quanto para os usuários que os preenchem.

Foram adicionado novos elementos, atributos, tipos de entrada de formulário, validação baseada no navegador, técnicas de estilo CSS3 e o objeto FormData.

A proposta dessa análise é medir, por meio de um protótipo de aplicativo de pedidos de produtos e códigos, como se comporta alguns elementos de formulário e armazenamento local em browsers de dispositivos móveis, como smartphones e tablets.

Conforme tabela 02, foram escolhidos três diferentes dispositivos móveis com arquitetura e sistemas operacionais distintos e neles utilizados os browsers nativos de cada um.

Tabela 02 - Dispositivos usados na prova de conceito

Dispositivo	Browser/versão
Tablet Galaxy Note	Android Browser
Nokia Lumia 920	Internet Explorer Mobile 10
Iphone 4s	Safári 7.0

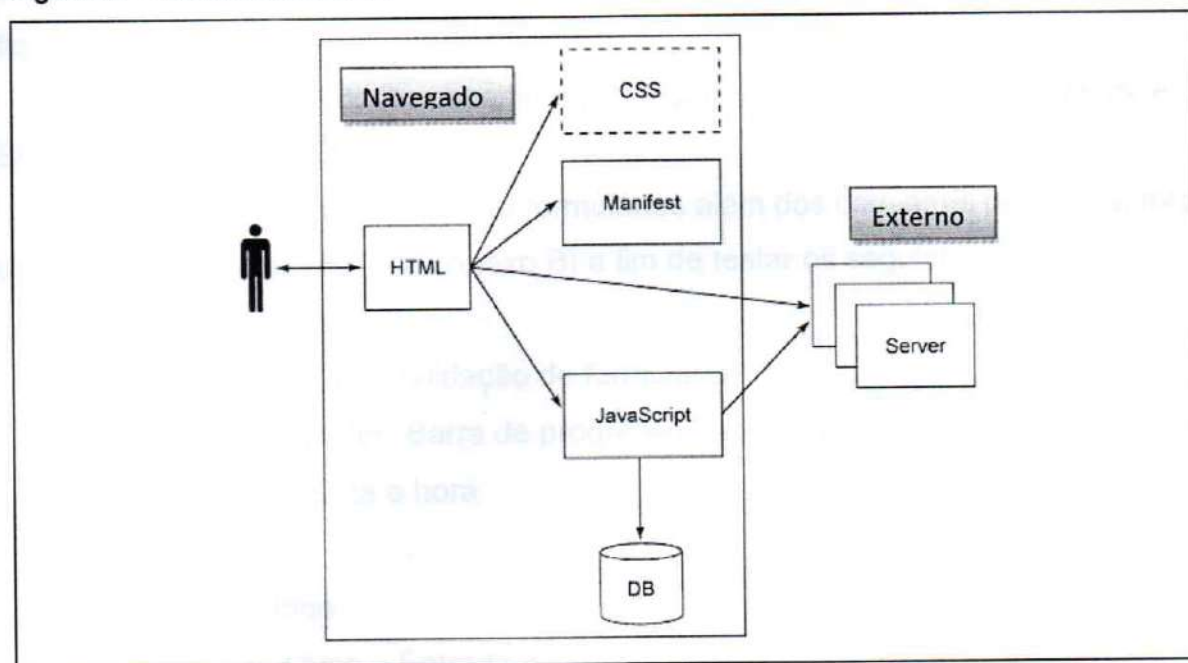
Fonte: Próprio Autor (2013)

4.1. ELEMENTOS UTILIZADOS

Para a prova de armazenamento foram adotados dois objetos: SQLite e o IndexedDB.

O armazenamento local usando SQLite é baseado em browser e teve seu lançamento em 2007 pelo Google. Navegadores baseados no WebKit, incluindo Chrome, Safari, e os navegadores do iPhone e celulares com Android, implementaram uma versão do banco de dados SQLiteEngine (estrutura na figura 17).

Figura 17 – Estrutura WebSQL



Fonte: Próprio Autor (2013)

IndexedDB é um armazenamento de objetos. É diferente de um banco de dados relacional que tem tabelas, linhas e colunas de coleções. É uma tecnologia de banco de dados para navegador capaz de armazenar grandes quantidades de dados estruturados e realizar pesquisas de forma otimizada, utilizando índices.

O armazenamento acontece localmente e quando os dados são requisitados por uma consulta o tempo de resposta é bem menor, uma vez que não é necessário ir até um servidor realizar a consulta.

O IndexedDB é um banco de dados não-relacional, e todas as chamadas são assíncronas com as interações ocorrendo dentro de uma transação.

IndexedDB tem várias vantagens sobre o SQLite. Em primeiro lugar, o seu formato de armazenamento de dados nativo é um objeto JavaScript. Não há necessidade de mapear um objeto JavaScript em uma tabela SQL, a qual se torna um risco de injeção SQL, ataque no qual um código mal-intencionado é inserido em cadeias de caracteres que são passadas posteriormente para uma instância do SQL para análise e execução.

4.2. APLICAÇÃO

Foi criado um protótipo de aplicativo utilizando HTML5 com os novos recursos de formulários e objetos de armazenamento contendo um cadastro de pedidos para

vendas de produtos, conforme figura 17 e um cadastro de clientes demonstrado na figura 18.

Para os testes de armazenamento local utilizamos dois tipos de objetos: SQLite e com indexedDB.

No tocante aos elementos de formulários além dos cadastros já citados, foram utilizados códigos específicos (Anexo B) a fim de testar os seguintes recursos:

- FormValidation – Validação de formulário.
- Progress & Meter- Barra de progresso e medida.
- Data/time – Data e hora
- E-mail
- Tel - Telefone
- Color input type – Entrada de cor
- Number input type – Entrada de números

Figura 18 - Tela de cadastro de pedido do aplicativo utilizado nos testes.

A interface de usuário para o cadastro de produtos é apresentada em uma janela com o título "PEDIDOS DE PRODUTOS".

A seção "Lista de Produtos" contém uma lista de itens:

- 2 x Notebooks
- 3 x Teclado para Notebook
- 10 x Licença windows 8 Professional
- 4 x Monitor 14" Samsung

Abaixo, a seção "Cadastro de Produtos" possui campos para:

- Id:
- Quantidade:
- Nome do Produto:

Na base da seção, há três botões: "create", "update" e "delete".

Fonte: Próprio Autor (2013)

Figura 19 - Tela de cadastro de usuários do utilizado nos testes.

Fonte: Próprio Autor (2013)

4.3. RESULTADO DA PROVA DE CONCEITO

Conforme proposto, a aplicação foi testada em três dispositivos móveis diferentes, cada um usando o browser principal do sistema.

No primeiro teste usamos o tablet Galaxy Note com sistema operacional Android JellyBean 4.1. Para tanto foi utilizado o Android Browser em sua versão 4.4 e, como mostrado no quadro 06, obtemos os seguintes resultados:

Quadro 06 – Resultado Tablet Galaxy Note

Features e Elementos	Resultados
Formulário	
Formvalidation	⊗
Progress & Meter	✓
Datalistelement	⊗
Range input type	✓
Email	⊗
Tel	⊗

Date/time input types	☒
Color input type	☒
Number input type	☒
Armazenamento	
IndexedDB	☒
SQLite	☒

Fonte: Próprio Autor (2013)

O segundo dispositivo utilizado no teste foi o Nokia Lumia 920 sistema operacional Windows 8. Neste dispositivo foi utilizado o browser Internet Explorer em sua versão móvel 10.0 no qual obtemos, conforme quadro 07, os seguintes resultados:

Quadro 07 – Resultado Nokia Lumia 920

Features e Elementos	Resultados
Formulário	
Formvalidation	⊗
Progress & Meter	☒
Datalistelement	⊗
Range input type	☒
Email	⊗
Tel	⊗
Date/time input types	⊗
Color input type	⊗
Number input type	☒
Armazenamento	
IndexedDB	☒
SQLite	☒

Fonte: Próprio Autor (2013)

O terceiro dispositivo usado foi Iphone 4S da Apple com sistema IOS 7. Neste dispositivo foi utilizado o browser Safari em sua versão 7.0. Os resultados obtidos são demonstrados no quadro 08.

Quadro 08 – Resultado Iphone 4S

Features e Elementos	Resultados
Formulário	
Formvalidation	⊗
Progress & Meter	✓
Datalistelement	⊗
Range input type	✓
Email	⊗
Tel	⊗
Date/time input types	✓
Color input type	⊗
Number input type	✓
Armazenamento	
IndexedDB	⊗
SQLite	✓

Fonte: Próprio Autor (2013)

O suporte aos recursos de formulário do HTML5 tiveram desempenhos diferentes nos três navegadores e dispositivos testados. Para os itens como "FormValidation" e "DataListElement" não foi gerado nenhum resultado.

Outros recursos tiveram suporte completo dos conjuntos testados, por exemplo: "Progress & Meter", "Date/time input types" e "Range input type".

Quanto ao armazenamento baseado em IndexedDB, o mesmo não tem suporte por parte do browser da Apple, com a justificativa de que já há suporte ao WebSQL. Vale lembrar, porém, que a W3C retirou da especificação do HTML5 o WebSQL uma vez que os desenvolvedores passaram a usar o SQLite como alternativa.

5. CONCLUSÃO

Com base nos estudos, testes e prova realizada fica evidenciado que o HTML5 traz grandes facilidades aos desenvolvedores web através dos seus novos elementos, tags e API's e possibilita uma melhor estruturação de sites e aplicações web.

O fato de não necessitar de plugins e aplicações nativas geram um grande avanço, uma vez que possibilita soluções multiplataformas com independência do dispositivo ou sistemas utilizados.

É importante citar que um dos fatos pelo qual o HTML5 se tornou usual e chamativo, antes inclusive que suas especificações tenham sido finalizadas, foi uma rápida disseminação das suas soluções nos dispositivos moveis.

As novas API's de multimídia, geolocalização, armazenamento, formulários e outras propiciam, entre outras soluções, a criação de jogos e apps tão robustas quanto às concebidas com outras plataformas até então utilizadas, o que despertou ainda mais o interesse na linguagem já nos primeiros anos após o surgimento.

No que se refere a compatibilidade das novas especificações do HTML5 com os principais browsers concluiu-se que esse é de fato um problema que ainda está presente, uma vez que tal diferença no suporte entre eles muitas vezes impossibilita o uso de algumas especificações para sites e aplicações web em produção, porém é importante destacar que ficou evidente durante as pesquisas realizadas, a atenção das empresas desenvolvedoras dos principais navegadores em avançar de forma ágil para que as especificações do HTML5 tenham suporte nos seus navegadores, sem restrições.

Os testes com os elementos selecionados mostraram que poucos itens ainda não funcionam em todos os browsers, o que nos leva a acreditar que em pouco tempo o suporte poderá vir a ser completo e que a linguagem possa ter o uso sem restrições nas soluções webs.

REFERÊNCIAS

DYER, John. MSDN Magazine. **Áudio e vídeo HTML5 para vários navegadores**, 2012. Disponível em <<http://msdn.microsoft.com/pt-br/magazine/hh781023.aspx>>. Acesso em: 16 de nov. 2013.

Eis, Diego. Portal Tableless. **O HTML não vai tomar o lugar do Flash, entretanto, o Flash não será a única opção**, abril 2010. Disponível em <<http://tableless.com.br/contedo-flash-e-html>>. Acesso em: 9 de out. 2013.

Geek Project. **HTML5 – Impressionantes exemplos utilizando Canvas**, abril 2011. Disponível em <<http://www.geekproject.com.br/2011/04/html5-impressionantes-exemplos-utilizando-canvas/>>. Acesso em: 04 de nov. 2013.

HOGAN, Brian P. **HTML5 & CSS3: Develop With Tomorrow's Standards Today**; 1. ed. North Carolina Dallas, Texas: The Pragmatic Programmers, 2010. 248 p. Bibliografia: ISBN 13: 978-1-934356-68-5

HTML5 Labs. <<http://html5labs.interoperabilitybridges.com/>>. Acesso em: 28 de out 2013.

KEITH, Jeremy. **Html5 For Web Designers**; 1. Ed. Korean, French: A Book Apart , 2010. 85 p. Bibliografia: ISBN: 978-0-9844425-0-8

KESSIN, Zachary. **Programming HTML5 Applications**; 1. ed. Gravenstein Highway North, Sebastopol: O'Reilly Media 2012. 128 p. Bibliografia: ISBN: 978-1-449-39908-5

LAWSON, Bruce e FAULKNER Steve. **HTML5 and Accessibility**, maio 2010. Disponível em <<http://msdn.microsoft.com/en-us/magazine/hh204741.aspx>>. Acesso em: 22 de out. 2013.

LUBBERS, Peter. ALBERS, Brian. SALIM, Frank. **Pro HTML5 Programming**; 2. ed. New York: Apress, 2011. 248 p. Bibliografia: ISBN 13: 978-1-4302-3865-2

MDN, Mozilla Developer Network. **IndexedDB**, outubro 2013. Disponível em <<https://developer.mozilla.org/en-US/docs/IndexedDB>>. Acesso em: 03 de nov. 2013.

MDN, Mozilla Developer Network. **HTML5**. Disponível em <<https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5>>. Acesso em: 28 de out. 2013.

Modernizr- Disponível em <<http://modernizr.com/>>. Acesso em: 20 de nov. 2013.

PILGRIM, Mark. **HTML5: Up and Running**; 1. ed. Gravenstein Highway North, Sebastopol: O'Reilly Media, 2010. 224 p. Bibliografia: ISBN 978-0-596-80602-6

SILVA, Maurício Samy. **HTML 5 – A Linguagem de Marcação que Revolucionou a WEB**; 1. ed. São Paulo: Novatec, 2011. 320 p. Bibliografia: ISBN 978-85-7522-261-4.

UBL, Malte e KITAMURA, Eije. **Apresentando WebSockets: trazendo soquetes para a web**, outubro 2010. Disponível em <<http://www.html5rocks.com/pt/tutorials/websockets/basics/>>. Acesso em: 18 de out. 2013.

ANEXOS

ANEXOS A - Exemplo de código usado para testes utilizando a biblioteca Modernizr

```
<script src="modernizr.js" type="text/javascript"></script>

<script type="text/javascript">
    if (Modernizr.applicationcache) {
        alert("ApplicationCache é aceita neste navegador") }
    else {
        alert("ApplicationCache não é aceita neste navegador") }
</script>
<script type="text/javascript">
    if (Modernizr.audio) {
        alert("Áudio é aceita neste navegador") }
    else {
        alert("Áudio não é aceita neste navegador") }
</script>
<script type="text/javascript">
    if (Modernizr.video) {
        alert("Vídeo é aceita neste navegador") }
    else {
        alert("Vídeo não é aceita neste navegador") }
</script>
<script type="text/javascript">
    if (Modernizr.indexeddb) {
        alert("IndexDB é aceita neste navegador") }
    else {
        alert("IndexDB não é aceita neste navegador") }
</script>
<script type="text/javascript">
    if (Modernizr.localstorage) {
        alert("LocalStorage é aceita neste navegador") }
    else {
        alert("LocalStorage não é aceita neste navegador") }
</script>
<script type="text/javascript">
    if (Modernizr.sessionstorage) {
        alert("SessionStorage é aceita neste navegador") }
    else {
        alert("SessionStorage não é aceita neste navegador") }
</script>
<script type="text/javascript">
    if (Modernizr.websqldatabase) {
        alert("WebSQLDatabase é aceita neste navegador") }
    else {
        alert("WebSQLDatabase não é aceita neste navegador") }
</script>
<script type="text/javascript">
    if (Modernizr.geolocation) {
        alert("Geolocalização é aceita neste navegador") }
    else {
        alert("Geolocalização não é aceita neste navegador") }
</script>
</script>
</script>
```

ANEXOS B – Fragmento do código dos cadastros usados para os testes com os objetos de armazenamento

```

<!DOCTYPE HTML>
<html manifest="cadastro.manifest">
  <head>
    <meta name="cadastro" content="width=480; initial-scale=1.0; maximum-
scale=1.0; user-scalable=0;">

    <title>Pedidos de Produtos</title>

    <script type="text/javascript" src="cadastro.js" ></script>
    <link href="principal.css" rel="stylesheet" type="text/css">

  </head>

  <body onLoad="onInit()">

    <div id="principal">
      <div id="topo">
        <br/>
        <h3> PEDIDOS DE PRODUTOS </h3>
      </div>

      <div id="corpo">
        <div id="corpo_lista">
          <div style="margin-left:30px">
            <h3>Lista de Produtos</h3>
            <ul id="itemData" >
            </ul>

          </div>

        </div>

        <div id="form_cadastro">
          <div style="margin-left:30px">
            <h3>Cadastro de Produtos</h3>
            <form name="itemForm">
              <label for="id">Id:</label><input type="text" name="id"
id="id" size=2 disabled="true"/>
              <label for="amount">Quantidade:</label><input type="text"
name="amount" id="amount" size=15/> <br /><br />
              <label for="name">Nome do Produto:</label><input
type="text" name="name" id="name" size=20 /><br /><br />
              <br>
              <input type="button" name="create" value="create"
onClick="onCreate()" />
              <input type="button" name="update" value="update"
onClick="onUpdate()" />
              <input type="button" name="delete" value="delete"
onClick="onDelete()" />
            </form>
          </div>

          <div style="margin-left:30px">
            <h4></h4>

            <div id="status">
              </div>

            <div style="margin-left:30px">
              </div>

            <a href="usuarios.html" title="Cadastro de novo usuário">Cadastrar
Usuários</a>

```