UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

TIAGO DE CARVALHO

AVALIAÇÃO E COMPARAÇÃO DE RECURSOS UTILIZANDO TECNOLOGIAS HTML5 E *FLASH* PARA CRIAÇÃO DE APLICAÇÕES *WEB*

CRICIÚMA

2013

TIAGO DE CARVALHO

AVALIAÇÃO E COMPARAÇÃO DE RECURSOS UTILIZANDO TECNOLOGIAS HTML5 E *FLASH* PARA CRIAÇÃO DE APLICAÇÕES *WEB*

Trabalho de Conclusão de Curso apresentado para obtenção do grau de Bacharel no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. Esp. Fabrício Giordani

CRICIÚMA

2013

TIAGO DE CARVALHO

AVALIAÇÃO E COMPARAÇÃO DE RECURSOS UTILIZANDO TECNOLOGIAS HTML5 E *FLASH* PARA CRIAÇÃO DE APLICAÇÕES *WEB*

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Criciúma, 27 de Dezembro de 2013

BANCA EXAMINADORA

Prof. Esp. Fabrício Giordani - (UNESC) - Orientador

Prof. MSc. Paulo João Martins - (UNESC)

Prof. MSc. Luciano Antunes - (UNESC)

Dedico este trabalho à minha mãe, pela força que sempre me deu para prosseguir e por nunca ter me deixado desistir.

A toda minha família, por estar comigo nos melhores e piores momentos de minha vida.

AGRADECIMENTOS

A Deus, por estar à frente de todas as minhas conquistas, dando-me saúde, sabedoria e determinação para seguir sempre em frente.

Agradeço também a meus pais, irmãos e amigos, por sempre me incentivarem e me ajudarem a chegar até aqui.

A todos que, de alguma forma, contribuíram para minha formação, meu mais sincero obrigado.

Não poderia deixar de agradecer ao meu orientador, Professor Fabrício Giordani, que, mesmo em dias de mau humor, não deixou de me mostrar o caminho.

E ofereço um agradecimento especial à minha mãe, por sempre ter acreditado em mim.

Não passe horas e horas criando um manual para ensinar ao usuário como usar o seu programa, gaste esse tempo deixando seu programa intuitivo o bastante para que o usuário não precise de um manual.

Robson Feitosa

RESUMO

Com o desenvolvimento muito rápido da *Internet* nos últimos anos, tornaram-se necessárias novas tecnologias que suportassem a demanda dos usuários que buscam recursos na *web*. Visando ajudar os desenvolvedores na escolha dos principais recursos disponíveis foi que se pensou neste trabalho, o qual apresenta um estudo comparativo entre as tecnologias *Hiper Text Markup Language*, versão 5 (HTML5), e *Flash*, sendo que o último domina as produções de vídeo, áudio e animações no mercado. Pretende-se mostrar que o HTML5 pode, futuramente, substituir a tecnologia *Flash*, e, sendo assim, serão apresentados os principais recursos das duas tecnologias para que os desenvolvedores possam comparar as mesmas e decidir qual irá adaptar-se melhor às suas necessidades para aplicações *web*. Para isso, será feito um protótipo, onde serão realizados todos os testes comparativos visando conhecer, com maior propriedade, suas vantagens e desvantagens. O estudo feito foi focado na qualidade de áudio e vídeo, aplicando-se os mesmos parâmetros de avaliação para as duas tecnologias. Com base nos testes objetivos, foi feita uma tabela comparativa, relatando-se, posteriormente, todos os resultados obtidos.

Palavras-chave: Tecnologias. HTML5. *Flash*. Desenvolvedores.

ABSTRACT

With the faster development of the internet in the last years, it was necessary new technologies which were able to give right support for the bigger number of users who make researches at web. To give a tool to the developers at the time to find the main resources available this paper was done, showing a comparative study of the technologies Hiper text Markup Language, HTML5 and Flash, keeping the idea that the last one is one of the most used at the internet business production of video, audio and animations. The main idea showed here will talk about the possibility of change in the future from FLASH to HTML5 technology, and will be presented the main resources of both technologies to the developers can compare and decide which one will be the best to their necessities and web applications. To show how both work a prototype will be made and tested where will be possible to see the pros and contras of the use of each one technologies. The main point of the study is the quality of audio and video applying them the same parameters to both technologies. Based on objective tests a comparative graphic was made talking lately about all the answers acquired.

Key – word: Technologies, HTML5, Flash, developers

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACR	*Absolute Category Rating* (Classificação de Categoria Absoluta)
ACR-HR	*Absolute Category Rating with Hidden Reference* (Classificação de Categoria Absoluta com Referência Oculta)
API	*Application Programming Interface* (Interface de Programação de Aplicativos)
APP	Aplicativo
AS1	*ActionScript*
AS2	*ActionScript* 2.0
AS3	*ActionScript* 3.0
ASP	*Active Server Pages*
AVM	*ActionScript Virtual Machine*
CERN	*Conseil Européen pour La Recherche Nucléaire* (Organização Europeia de Pesquisa Nuclear)
CGI	*Common Gateway Interface*
CPU	*Central Processing Unit* (Central Única de Processamento)
CSS	*Cascading Style Sheets* (Folha de Estilo em Cascata)
DB	Decibéis
dBfs	*Decibels Relative to Full Scale* (Decibéis em Relação à Escala)
DCR	*Degradation Category Rating* (Classificação de Categoria de Degradação)
DMOS	*Differential Mean Opinion Scores* (Parecer Diferencial de Pontuação Média)
DS	*Double Stimulus* (Estímulo Duplo)
DSCQS	*Double Stimulus Continuous Quality Scale* (Escala Contínua de Qualidade com Duplo Estímulo)
DSIS	*Double Stimulus Impairment Scale* (Escala de Imparidade com Duplo Estímulo)
ECMA	*European Computer Manufacturers Association* (Associação Europeia de Fabricantes de Computadores
FPS	*Frames per second* (Quadros por segundo)
GB	*Gigabytes*
GHZ	*Gigahertz*

GIF	*Graphics Interchange Format* (Formato para Intercâmbio de Gráficos)
HTML	*HyperText Markup Language* (Linguagem de Marcação de Hipertexto)
HTTP	*Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto)
ITU	*International Telecommunication Union* (União Internacional de Telecomunicações)
JPEG	*Joint Photographic Expert Group* (Grupo Expert de Junção de Fotografias)
JSP	*JavaServer Pages*
KB	*Kilobytes*
KHZ	*Quilohertz*
LCD	*Liquid Crystal Display* (Tela de Cristal Líquido)
MB	*Megabytes*
PC	*Pair Comparison* (Comparação de pares ou duplas)
PHP	*Hypertext Preprocessor*
RAM	*Random Access Memory* (Memória de Acesso Aleatório)
RTMP	*Real-Time Messaging Protocol* (Protocolo de Mensagem em Tempo Real)
SCACJ	*Stimulus Comparison Adjectival Categorical Judgement* (Estímulo de Comparação de Julgamento de Categoria Adjetiva)
SDSCE	*Simultaneous Double Stimulus for Continuous Evaluation* (Avaliação Contínua com Duplo Estímulo Simultâneo)
SGML	*Standard Generalized Markup Language* (Linguagem Padronizada de Marcação Genérica)
SQL	*Structured Query Language* (Linguagem de Consulta Estruturada)
SS	*Single Stimulus* (Estímulo único ou simples)
SSCQE	*Single Stimulus Continuous Quality Evaluation* (Avaliação Contínua de Qualidade com Estímulo Simples)
SWF	*Shockwave Flash*
TI	Tecnologia da Informação
URL	*Uniform Resource Locator* (Localizador-Padrão de Recursos)
W3C	*World Wide Web Consortium* (Consórcio da Rede Mundial)
WEB	*World Wide Web* (Ampla Rede Mundial)
WHATWG	*Web Hypertext Application Technology Working Group* (Grupo de Trabalho para Tecnologias de Hipertexto em Aplicações para Web)

WEBGL	*Web Graphics Library* (Biblioteca Gráfica da Web)
XHTML	*eXtensible Hypertext Markup Language* (Linguagem Extensível para Marcação de Hipertexto)
XML	*eXtensible Markup Language* (Linguagem de Marcação Extensível)
XP	*Extreme Programming* (Programação Extrema)

SUMÁRIO

1 INTRODUÇÃO

Em 1989, quando Tim Berners-Lee criou a *World Wide Web* [1], ou simplesmente *Web*, ela era apenas um ambiente para publicar documentos no formato de um texto, não havia dinamismo com o usuário, o qual estava limitado a ler, imprimir e selecionar *links*[2] para outros documentos. Logo após, vieram os formulários *Common Gateway Interface*[3] (CGIs), que possibilitavam ao usuário entrar com dados e ter dinamismo com aplicações do banco de dados. (WINCKLER, 2002).

Com a crescente popularização da Internet, a necessidade de novas funcionalidades aumentou. Como as páginas da época já não mais satisfaziam, com o tempo foram incorporadas às páginas novos recursos, como imagens, sons, vídeos, animações etc.

Ao mesmo tempo em que páginas mais robustas foram sendo criadas, novas tecnologias com a ideia de permitir aos desenvolvedores criações mais complexas começaram a aparecer.

No decorrer dessa evolução, surge o *Flash*, sendo um *software* [4], primeiramente, de gráficos vetoriais, para criação de animações que também oferecia suporte para *bitmap*[5] e vídeo. A partir de então, foram sendo adicionados mais e mais recursos ao *software*, tornando-o, assim, o aplicativo mais usado para a criação de *websites* interativos.

Apesar da grande gama de recursos que o *Adobe Flash* e outras tecnologias ofereciam, elas necessitavam que *plug-ins* [6] fossem instalados nos navegadores, mas logo os problemas inerentes aos *plug-ins* começaram a aparecer, pois utilizá-los para fazer algo que deveria ser nativo no próprio navegador tornou visível algumas desvantagens: os *plug-ins* não eram nativos, eram proprietários, seu código não era aberto e não seguiam um padrão. Nesse contexto, surge a proposta do *HiperText Markup Language*[7], versão 5 (HTML5),

---

[1] Ampla Rede Mundial
[2] Caminhos
[3] Não há uma tradução satisfatória, mas a CGI é uma tecnologia que permite a geração de páginas dinâmicas.
[4] Parte lógica do computador, como sistemas e aplicativos
[5] Mapa de *bits*, os quais, por sua vez, são os impulsos elétricos que representam as menores unidades de informação armazenadas ou transmitidas.
[6] Módulos de extensão ou utilitários que trazem mais funcionalidades aos programas
[7] Linguagem para Marcação de Hipertexto

uma forma de centralizar todos os recursos oferecidos pela *Internet* numa única tecnologia nativa. (BATISTA, 2009).

Mesmo usando *plug-ins* e sendo uma tecnologia proprietária, as qualidades do *Adobe Flash* são indiscutíveis, mas desde o anúncio da criação do HTML5 houve uma grande movimentação na *Internet* com notícias do possível fim do *Flash*, que seria substituído pelo HTML5, deixando vários desenvolvedores *Flash* temerosos pelo futuro dessa tecnologia.

O HTML5 é uma forte tecnologia e veio como uma boa opção para criações interativas em substituição ao *Flash,* mas antes de se optar por usar HTML5 ou *Flash* existem vários aspectos que devem ser avaliados como, por exemplo:

a) *plug-in versus* nativo: *Flash* necessita de *plug-in* para rodar nos navegadores além de ser uma tecnologia proprietária, já o HTML5 é nativo nos navegadores e aberto;

b) portabilidade: a Adobe anunciou descontinuar o desenvolvimento do *Flash Player* para dispositivos móveis, mantendo apenas o suporte às configurações já existentes e permitindo que desenvolvedores criem aplicativos (*apps)* nativos com *Adobe Air*, enquanto o HTML5 é compatível com a maioria dos navegadores móveis existentes;

c) animações: *Flash* é uma forte tecnologia na criação de animações, principalmente por trabalhar com gráficos vetoriais, facilmente redimensionáveis e alteráveis por funções; em contrapartida, o HTML5 vem com um novo elemento para criação de animações, o *canvas*, com o qual é possível criar elementos gráficos, composição de imagens e animações robustas, sem a necessidade de utilizar *plug-in*;

d) vídeo: um dos grandes recursos do *Flash* é a possibilidade de se trabalhar com vídeos, sendo uma tecnologia bem robusta nesse quesito, com recursos como criação de vídeos, controle de reprodução, importação, linha do tempo, pontos de sinalização, entres outros; é amplamente usada para a publicação de vídeos na *web*. Na mesma linha, foi incorporado ao HTML5 a *tag*[8] vídeo, a qual permite ao desenvolvedor trabalhar com vídeos usando unicamente o HTML sem a necessidade de incorporar ao código algum arquivo externo de outra tecnologia e, principalmente, sem a necessidade de *plug-ins* para isso.

[8] Estrutura de linguagem de marcação

O HTML5 surgiu com força no mercado e o seu conhecimento tornou-se quase obrigatório para qualquer desenvolvedor *web*, o estudo dessa tecnologia que chega como um divisor de águas se torna relevante pelas mudanças que ela promete na área de desenvolvimento *web*. Apesar de todos os estudos feitos sobre essas duas tecnologias ainda se perguntam:

a) Que tipos de formatos de música, vídeo, imagens etc., são suportados pelas tecnologias?

b) Será que os navegadores mais utilizados são compatíveis com HTML5, mesmo nas versões mais antigas?

c) Em que situações se pode optar por uma tecnologia ou outra?

Ainda não se tem total segurança sobre o desempenho que cada uma pode ter e em quais situações elas poderão ser utilizadas. Tentando esclarecer todas essas dúvidas é que se pensou em fazer um estudo comparativo entre as duas linguagens.

Visando conhecer os principais recursos do HTML5 e do *Flash* na mesma proporção, propõe-se, com este trabalho, orientar os desenvolvedores, não somente aqueles que trabalham com *Flash*, mas todos que atuam com desenvolvimento *web*, fazendo uma comparação entre os principais recursos do *Flash* e do HTML5. Ao identificar os pontos fortes e fracos das duas tecnologias, torna-se possível escolher, com maior propriedade, qual das linguagens poderá atender melhor às necessidades dos desenvolvedores.

1.1 TEMA

Avaliação e comparação de recursos utilizando tecnologias HTML5 e *Flash* para criação de aplicações *web*

1.2 OBJETIVOS

1.2.1 Objetivo geral

Avaliar os recursos disponíveis no HTML5 e no *Flash,* realizando testes comparativos, aplicando diversos problemas nas mesmas condições para as duas

tecnologias e oportunizando, assim, que os desenvolvedores escolham com maior propriedade qual recurso poderá ser usado para implementação de aplicações *web*.

1.2.2 Objetivos específicos

a) Definir as diferenças e semelhanças na tecnologia HTML5 e *Flash*.

b) Descrever as vantagens e desvantagens em relação às duas tecnologias.

c) Analisar os componentes gráficos e de mídia existentes na tecnologia de marcação HTML5 e no *Flash.*

d) Identificar os navegadores com suporte às tecnologias em questão.

e) Desenvolver protótipos para comparar os recursos existentes no HTML5 e no *Flash*.

1.3 JUSTIFICATIVA

Atualmente, as aplicações *web* estão muito evoluídas, os usuários já se habituaram a entrar em uma página e ver várias imagens gráficas, vídeos e animações. Desenvolver páginas com tais recursos não é mais um grande diferencial e, na área de recursos multimídia, o *Adobe Flash* vem dominando o mercado desde que foi criado. Recentemente, a comunidade tecnológica foi surpreendida com notícias de que o *Adobe Flash* poderia vir a terminar. Um dos motivos para as notícias tomarem tais proporções foi o comentário do fundador da *Apple*, Steve Jobs, dizendo que o *Flash* seria substituído pelo HTML5, seguido pelo anúncio da própria *Adobe* comunicando que iria descontinuar o desenvolvimento do *plug-in Flash Player* para plataformas móveis.

Apesar das notícias, o *Flash* continuará a ser desenvolvido para aplicações *web* que rodem em aparelhos *desktop*[9], mas a tendência é a de que venha a entrar em uma briga de mercado com HTML5. Este, além de permitir a criação de animações, vídeos e imagens comumente feitos com *Flash*, possui portabilidade para vários dispositivos, tornando aplicações feitas com essa tecnologia visíveis não somente em aparelhos *desktops*, mas também em celulares

[9] Computador de mesa

e *tablets,* entre outros, sem a necessidade de códigos específicos para cada dispositivo.

Essa nova versão do HTML, o HTML5, foi totalmente reformulada, com a introdução de novas *tags,* como de áudio e vídeo, trazendo assim mais possibilidades aos desenvolvedores. Em relação às animações que caracterizam o grande forte do *Adobe Flash*, foi implementada no HTML5 uma *tag* chamada *canvas*. *Canvas* é um elemento que possibilita especificar uma área da página onde é permitido, utilizando-se *scripts*, desenhar e renderizar imagens, o que aumenta significativamente as possibilidades dos recursos visuais e gráficos e permite fazer criações que até então estavam reservadas para o desenvolvedor *Flash*, com a vantagem de que *canvas* não necessita de nenhum *plug-in* no navegador.

O HTML5 não é apenas mais uma coleção de novos marcadores e, sim, uma tecnologia que provê novos mecanismos para transferência e armazenamento de dados, apresentação de vídeo, som, animações, novas fontes tipográficas de texto e novos modelos de disposição da estrutura gráfica das páginas (*layou*t). A preocupação principal não é com a tecnologia, mas em como as pessoas a utilizam. A tecnologia responsável por estas funcionalidades já existe. Há uma grande variedade de extensões que programam as mesmas funcionalidades, porém, sem uma padronização. (MANSFIELD-DEVINE, 2010).

A área de informática está em constante mudança e evolução, e, para os profissionais que atuam neste setor, é importante estarem sempre atentos às novidades que aparecem no mercado. HTML5 é uma tecnologia que promete bastante no desenvolvimento para *web*, e o estudo dela tornou-se quase obrigatório para a maioria dos desenvolvedores *web*.

Diante das perspectivas para o HTML5, pode ser traçado um paralelo entre as vantagens e desvantagens do padrão, com uma implementação que inclua os principais recursos do HTML5, sendo também implementado em *Flash* para esclarecimento de dúvidas e melhor entendimento das duas tecnologias.

Um estudo comparativo entre as tecnologias HTML5 e *Flash* se torna interessante porque o HTML5 veio como uma nova opção para recursos antes dominados pelo *Flash* na *web*, o que leva a uma grande disputa por território comercial entre as tecnologias, como já é perceptível. Esse estudo comparativo

pode ajudar a orientar os desenvolvedores na escolha de qual é a melhor opção quando forem desenvolver alguma aplicação.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está dividido da seguinte forma: o capítulo 2 começa com uma fundamentaçao teórica, a qual aborda a definição, as principais características, e as diversas vantagens do HTML5 encontradas ao se utilizar os padrões *World Wide Web Consortiun*[10] (W3C). O capítulo trata também da Tecnologia *Flash*, apresentando a definição, principais funções e desempenho dentro da *web*. O capítulo 3 faz uma comparação entre a linguagem HTML5 e *Flash*. O capítulo 4 apresenta uma análise mais detalhada dos resultados obtidos com a comparação. E, por fim, o capítulo 5 apresenta a conclusão do trabalho e sugestões para trabalhos futuros.

[10] Consórcio da Rede Mundial

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, apresenta-se a fundamentação teórica necessária para um melhor entendimento do estudo de caso proposto. Desenvolver um projeto *web* em HTML5 para os navegadores atuais é uma opção que pode ser estudada, mas, antes que se decida por ela, é preciso tomar algumas precauções levando em conta restrições existentes, para contorná-las. A maioria das funcionalidades do HTML5 já é suportada por um ou mais navegadores atuais, porém, existem mecanismos desenvolvidos com a linguagem *javascript* [11] que detectam suporte para as funcionalidades, criando condições de o desenvolvedor oferecer um modo alternativo para a funcionalidade que não for suportada por aquele navegador em questão. (SILVA, 2011).

A principal premissa do HTML5 é melhorar a semântica das páginas e adicionar novos recursos à linguagem para responder melhor aos avanços tecnológicos. Nas próximas seções serão apresentadas as formas de como trabalhar com áudio, vídeo e animações utilizando as tecnologias HTML5 e *Flash*. Da seção 2.1 até o final deste capítulo, o referencial teórico fundamenta este trabalho que visa fazer uma comparação entre os recursos HTML5 e *Flash,* analisando, assim, suas principais funcionalidades. Por fim, a seção 2.4 apresenta os trabalhos correlatos.

2.1 EVOLUÇÕES *WEB*

A *Web* é definida por seu criador, o físico inglês Tim Berners-Lee, como um universo de informações acessíveis globalmente. Um meio interligado de páginas de texto, imagens, sons, vídeos e animações em que se pode interagir. (ARAYA; VIDOTTI, 2010).

Tim Berners-Lee estava no setor de computação do *Conseil Européen pour La Recherche Nucléaire* [12] (CERN) quando começou a fazer pesquisas procurando um modo que permitisse aos cientistas de toda a parte do mundo compartilhar e interligar suas pesquisas eletronicamente. (SILVA, 2011).

---

[11] Linguagem para programação *client-side*
[12] Organização Europeia de Pesquisa Nuclear

Em 1990, Berners-Lee criou um *software* navegador que possibilitava a criação e navegação entre páginas de hipertexto e foi chamado de *Word Wide Web*. Para que a nova tecnologia funcionasse, ele também criou a linguagem de marcação de texto (HTML), o *Hypertext Transfer Protocol*[13] (HTTP), para que os computadores se comunicassem na Internet, e a *Uniform Resource Locator*[14] (URL), um endereço único para cada documento na *Internet*. (ARAYA; VIDOTTI, 2010).

Quando a *Web* foi implementada, as páginas apresentavam apenas textos e *links*, essas eram as famosas páginas estáticas, que permitiam aos usuários apenas ler as informações disponíveis nas páginas e navegar para outros *sites* a partir dos *links* inseridos nos textos, sem oferecer nenhum tipo de interação com os usuários.

Como as páginas estáticas eram insatisfatórias, logo surgiu o padrão CGI, programas executáveis hospedados no servidor, acessáveis por protocolo HTTP, e que podiam ser escritos em qualquer linguagem de programação, como C e *Perl*. A máquina cliente faz uma requisição ao servidor, que passa para um programa CGI, o qual gera uma resposta que é repassada à máquina cliente pelo servidor. Isso possibilitou aos desenvolvedores a criação de documentos interativos e aplicações complexas, com acesso a banco de dados, dando origem às primeiras páginas dinâmicas. (FURLAN, 2012).

Em seguida, apareceram as linguagens de programação *web*, como o *Hipertext Preprocessor*[15] (PHP), *Active Server Pages*[16] (ASP) e *Java Server Pages*[17] (JSP). Essas linguagens eram escritas juntamente com os códigos HTML e o servidor era quem processava o código misto e o transformava em puro HTML, para depois retornar a página para o navegador cliente. Esse modo de desenvolvimento tornou-se rapidamente popular, pois permitia que pessoas com pouco conhecimento em programação construíssem *sites* dinâmicos com certa facilidade. (LALLI; BUENO; ZACHARIAS, 2008).

Até então, as tecnologias atuais não ofereciam recursos multimídia, como áudio e vídeo aos navegadores, o máximo que podia ser feito era “baixar” arquivos de áudio e vídeo para que fossem rodados nos *players* dos computadores. Por um

---

[13] Protocolo de transferência de hipertexto
[14] Localização-padrão de recursos
[15] Sem tradução satisfatória, representa uma linguagem interpretativa livre.
[16] Sem tradução satisfatória, significa linguagem para desenvolvimento de aplicações interativas na *web*.
[17] Sem tradução satisfatória, é uma tecnologia para criação de páginas web geradas dinamicamente.

tempo, essa situação foi aceita naturalmente, já que nem a democratização dos acessos nem a largura de banda ajudavam a tornar a experiência rica para os usuários. (FLATSCHART, 2012).

Com a evolução de todos os setores da computação, a *web* não ficou para trás e, com a constante melhora na largura de banda, houve uma maior democratização de acessos e avanços da *Internet* como meio de negócios. Começou a ser questionado por que não incorporar mais interatividade à navegação com recursos multimídia de áudio, vídeo etc. (FLATSCHART, 2012).

A meteórica evolução da *Internet*, que ficou mais evidente pela grande necessidade de transformar aplicações *desktops* em *web*, tornou impossível a espera do avanço do *eXtensible Hypertext Markup Language*[18] (XHTML) ou do HTML. Com a necessidade de inserir novas funcionalidades, pouco encontradas pela precariedade das linguagens, foram agregados *plug-ins a*os navegadores, possibilitando aos desenvolvedores a utilização de vários novos recursos, como áudio, vídeo e gráficos. Surge, então, a era dos *plug-ins* ou das animações. (BATISTA, 2009).

Os *plug-ins* possibilitavam incorporar aos navegadores funcionalidades que os mesmos não possuíam nativamente. Dentre eles, o de maior sucesso foi o *Flash Player*, nascido da Macromedia, em 1997, e incorporado pela *Adobe*, em 2005, sendo que o mesmo revolucionou as distribuições de material multimídia na *web*. Rapidamente, tornou-se a principal tecnologia na criação de aplicações ricas, e para criação e distribuição de conteúdo que envolvesse jogos, áudio e vídeo, *streaming*[19] e interfaces gráficas interativas. (FLATSCHART, 2012).

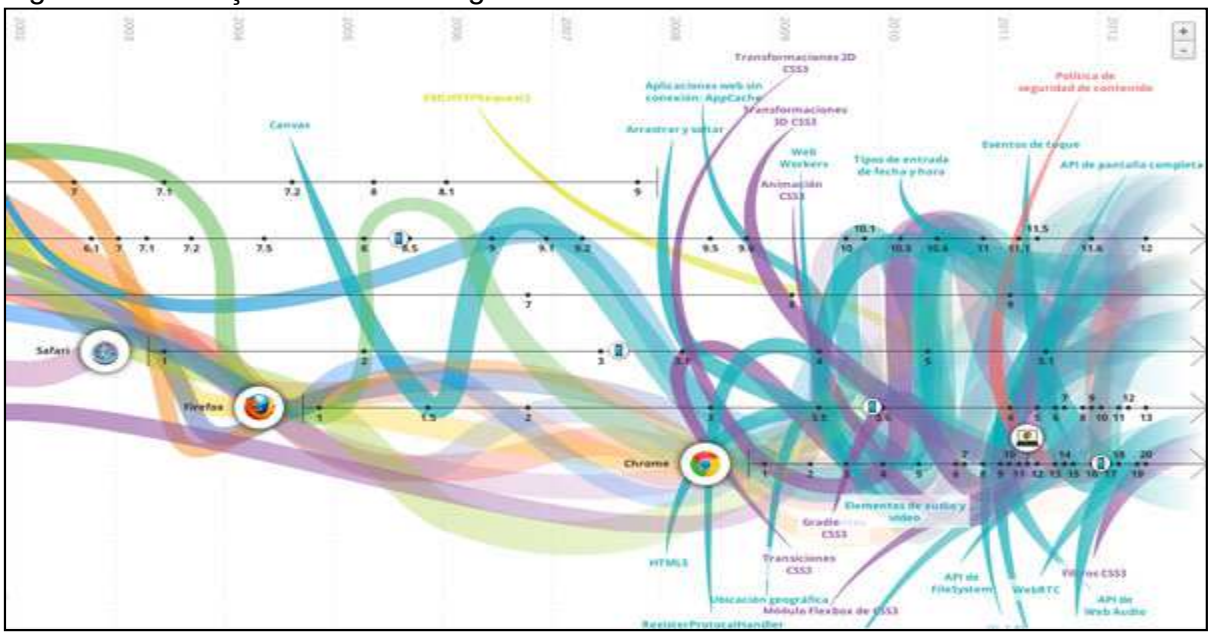
Existem outras tecnologias que possibilitam maior interatividade multimídia aos navegadores, dentre as quais se pode citar o *QuickTime* da *Apple*, o *SilverLight* da *Microsoft*, *Java Applet* da *Sun*, além de inúmeras outras. Apesar da revolução que os *plug-ins* trouxeram no passado, atualmente surgiram dúvidas se é realmente interessante continuar usando *plug-ins* como soluções multimídias e de interatividade. Muitos fatores contribuíram para esse pensamento, mas, provavelmente, os principais são a acelerada evolução do universo *Mobile*[20], a maior procura das empresas por

[18] Linguagem extensível para marcação de hipertexto
[19] Fluxo de mídia ou forma de distribuir informação multimídia em uma rede através de pacotes
[20] Dispositivos móveis, como celulares, *smartphones* e *tablets* ou *Ipads*

soluções nativas nos navegadores e abertas e o surgimento do HTML5, o qual amplia enormemente o leque de possibilidades no desenvolvimento *web*.

A figura 1 mostra a evolução da *web* feita em infográfico:

Figura 1 - Evolução *Web* em infográfico



Fonte: www.tecnoartenews.com

2.2 HTML5

2.2.1 A evolução do HTML ao HTML5

O HTML é usado como linguagem para formatar páginas exibidas na *web* ou em uma intranet. O *HyperText Markup Language* não é uma linguagem de programação e, sim, de marcação de documentos, usada para fazer toda a formatação das páginas *web*. (CARVALHO, 2001).

Sua primeira versão teve como base a linguagem *Standard Generalized Markup Language*[21] *(*SGML), uma linguagem de estruturação de documentos, sendo que muitas das *tags* do HTML foram herdadas dessa linguagem, tais como: <h1> e <h6>, <*head*>, <p>. O grande diferencial entre as duas linguagens é o que caracteriza a *web*: a capacidade de interligações entre documentos que o HTML oferece através da *tag* <a>. (WILLIAM, 2012).

[21] Linguagem padronizada de marcação genérica

Com a maior popularidade do HTML, o grupo W3C definiu os recursos iniciais da linguagem e mais alguns novos que já estavam em uso na *web*, como o HTML 2.0, que era um padrão mínimo suportado por todos os navegadores. Depois dele, por adições próprias da *Netspace* e da *Microsoft*, o padrão HTML tornou-se um pouco confuso, enquanto a versão HTML 3.2 era um meio termo entre a sua antecessora e a sua sucessora, já suportando tabelas, papel de parede nas páginas, *applets*[22] e fluxo de texto ao redor de imagens.

Numa contínua e rápida evolução, logo a W3C especifica um novo padrão HTML, que foi chamado de HTML 4.0, versão que trouxe grandes novidades em relação à sua antecessora. Entre as melhorias identificadas, as mais significativas são: tabelas mais ricas, *forms* melhorados, suporte melhorado a textos, incorporação do *object*, inclusão de *scripts* em páginas; *frames*[23] e *Cascadind Style Sheets*[24] (CSS). A principal novidade desta nova versão foi a adesão das folhas de estilo que separam o conteúdo da apresentação. (W3C, 2012).

Após a versão 4.0, a W3C especifica um novo padrão, o HTML 4.01, o qual, além de outras mudanças, teve como ideia principal a compatibilidade com suas duas versões anteriores, o que deu-se através de três implementações (WILLIAM, 2012):

a) *script* (escrita) – implementação que não permite a utilização de componentes obsoletos no código;

b) *transitional* (transitória) – implementação em que era possível a utilização de componentes obsoletos no código;

c) *frameset* – implementação voltada para a criação de páginas que dispunham de *frames*.

Desenvolvida em paralelo, a W3C especificou também uma recomendação XML para o HTML4, batizada de XHTML 1.0. Esse padrão possuía as mesmas funcionalidades do HTML4, com as três principais implementações, mas com o diferencial de poder ser lido por qualquer navegador, inclusive pelas versões mais antigas. Um documento XHTML 1.0 é também um arquivo XML válido, que qualquer interpretador XML pode ler. (HEY, 2010).

---

[22] Aplicativo executado no contexto de outro programa
[23] Quadros
[24] Folhas de estilo em cascata

A versão seguinte do XHTML 1.0, o XHTML 1.1, nada mais seria que o XHTML 1.0 *Script* reformulado com algumas modificações. Todos os elementos obsoletos que a versão XHTML 1.0 *Script* ainda permite, como *framesets*, alguns atributos como o *lang* e o atributo *name* do elemento âncora, entre outros, foram retirados dessa versão. (W3C, 2010).

Tentando melhorar ainda mais as linguagens HTML e XHTML, a W3C resolve, então, criar uma nova versão, o XHTML 2.0. Essa nova implementação apresentou um problema que causaria uma revolução de proporções inimagináveis, pois a *web* não estava preparada para as transformações que teriam que ser feitas. Ela exigia mudanças no modo de programar e as páginas teriam que ser reconstruídas. E, por não ter adesão do mercado, a W3C resolveu descontinuar o XHTML 2.0. (BATISTA, 2009).

Antes que o XHTML 2.0 fosse descontinuado pela W3C, o *Web Hypertext Application Technology Working Group*[25] (WHATWG) vinha desenvolvendo, em paralelo, o agora tão comentado HTML5. Por vários anos, os dois se ignoraram, cada um trabalhando nas suas próprias implementações. Com o tempo, ficou claro que o HTML5 estava tendo maior adesão no mercado, enquanto o XHTML 2.0 continuava apenas como rascunho, sem ser implementado nos navegadores. Em outubro de 2006, a W3C anunciou que iria trabalhar em conjunto com a WHATWG no desenvolvimento do HTML5. (WILLIAN, 2012).

As últimas versões do HTML não possuíam recursos que possibilitassem aplicações mais robustas, com controle multimídia, sendo deixada essa responsabilidade para outras tecnologias que funcionassem através de *plugins* nos navegadores. Porém, para facilitar a codificação dos desenvolvedores e a navegabilidade dos usuários, é interessante que os navegadores não necessitem de *plugins* para reproduzirem efeitos de mídia. (HEY, 2010).

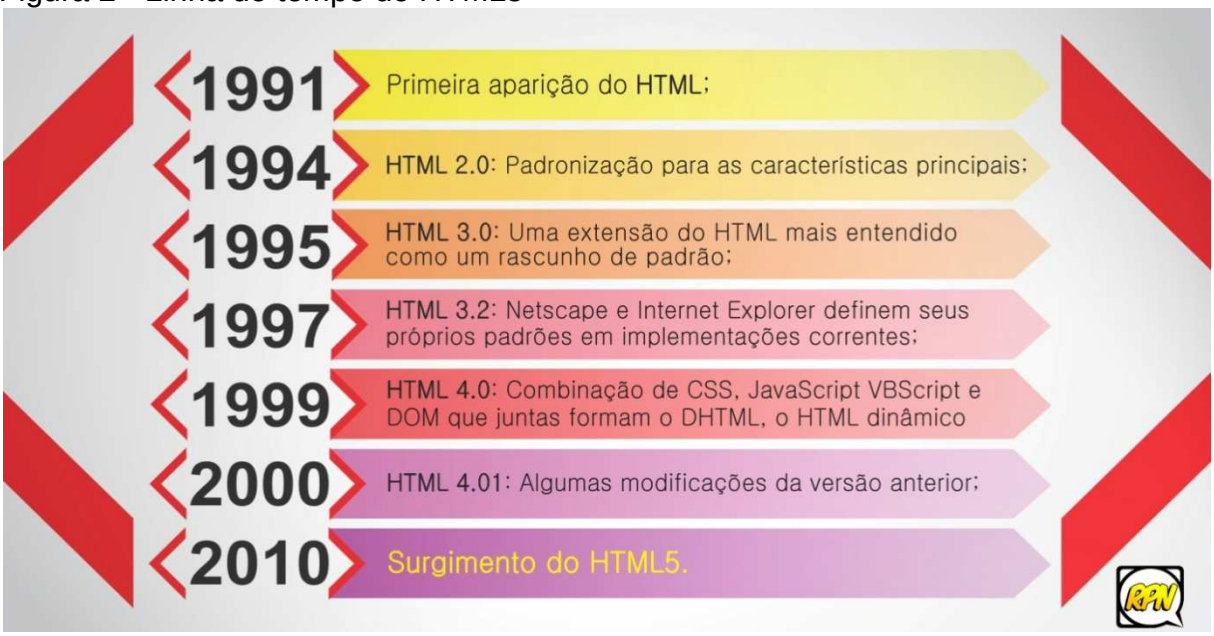
Com esses e outros fatores em mente, o grupo *WHATWG* começa a criação do HTML5, tecnologia que engloba os elementos HTML, a linguagem *JavaScript* e os comandos CSS. A ideia do HTML5 é facilitar a navegação dos usuários e possibilitar páginas mais leves, dinâmicas e seguras, com recursos multimídia que não necessitem de *plug-ins*. A especificação permite uma indexação mais eficiente em mecanismos de busca, exibição de áudio e vídeo nativos nos

[25] Grupo de Trabalho para Tecnologias de Hipertexto em Aplicações para Web

próprios navegadores, execução de aplicações *web offline* e códigos mais legíveis. (HEY, 2010).

Abaixo, pode-se observar parte da evolução HTML na linha do tempo:

Figura 2 - Linha do tempo do HTML5



Fonte: Nunes (2012)

Nas próximas seções aparecem os recursos HTML5 e, em detalhes, os que fornecem as mesmas funções dos recursos *Flash* mais usados na *web*, deixando o leitor ciente sobre quais funcionalidades as duas tecnologias disputam no mercado e servindo de introdução aos componentes que serão comparados neste trabalho, em capítulos posteriores.

2.2.2 Recurso HTML5

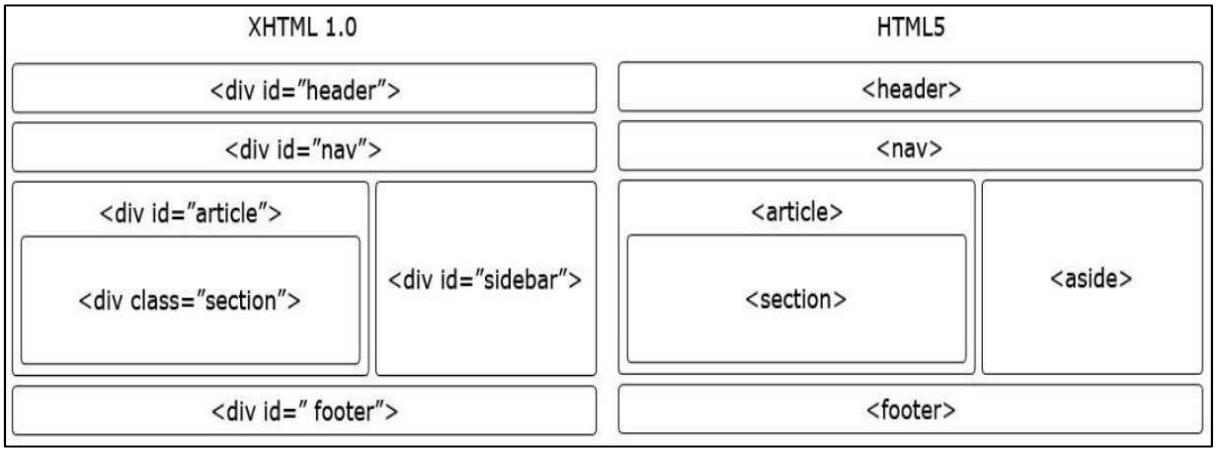
O HTML5 traz várias novidades em relação às suas versões anteriores. Muitas são necessidades que há muito já eram pedidas pelos desenvolvedores, como os elementos *canvas*, áudio e vídeo, enquanto outras são mudanças no que já existia, mas precisava de aprimoramento. Esta seção mostra algumas destas novidades.

Era comum ver *sites* com um longo encadeamento de *divs* [26], diferenciadas apenas por *id,* para tentar organizar o corpo da página, tipo de prática

[26] Tipo de *tag*

que tornava o código de difícil manutenção e poluído. O HTML5 vem com *tags* de estrutura do corpo de página, o que elimina a necessidade de *divs* encadeados. (BATISTA, 2009).

Figura 3 - Comparações entre codificações de um mesmo *layout* em XHTML 1.0 e HTML5



Fonte: Batista (2009)

Através da ilustração, é possível perceber claramente a estrutura do corpo de uma página HTML5, sendo que:

a) cabeçalho (*heade*r): carrega as informações promocionais de um *website*, por exemplo, um logotipo;

b) um menu (*nav*): contém a listagem dos *links* (ligações) disponíveis para outras seções do *website*;

c) um artigo (*article*): divide o *layout* em uma seção de conteúdo;

d) uma seção (*section*): representa um conteúdo específico do *website*, pode ser uma notícia ou um *post*;

e) uma barra lateral (*aside*): uma barra para disposição de informações complementares da página;

f) um rodapé (*footer*): exibe os créditos de criação e direitos autorais de um *website*.

Um dos objetivos da criação do HTML5 era desenvolver funcionalidades que possibilitassem aos desenvolvedores programar páginas e aplicações *web* mais nativas, que dispensassem a necessidade de *plug-ins* de outras tecnologias, como o *Java*, o *SilverLight* e o *Flash*. Dentre essas novas funcionalidades, podem ser citadas:

a) *canvas* – é uma área definida no documento HTML5 onde se podem desenvolver desenhos, jogos, animações, trabalhar com imagens, tudo em tempo real;

b) áudio – possibilita a inserção de áudio em páginas *web,* descartando a necessidade de *plug-ins;*

c) vídeo – possibilita a inserção de vídeos em páginas *web,* descartando a necessidade de *plug-ins;*

d) geolocalização – o HTML5 possui uma *Application Programming Interface*[27] (API) que permite localizar o posicionamento geográfico do usuário;

e) *drag-and-drop* – funcionalidade que permite ao usuário arrastar um elemento e soltá-lo em outro lugar, aumentando, assim, a interatividade do usuário com a página ou aplicação;

A *tag input* recebe vários tipos de entradas que facilitam o desenvolvimento, pois melhoram o controle de dados e ajudam nas validações. Seguem os novos tipos de dados de entradas: c*olor* (cor), d*ate* (data), d*atetime, datetime-local, email, month, number, range, search, tel, time, url e week.*

Além dos recursos citados, outros foram acrescidos nessa nova versão da HTML, mas não serão apresentados todos neste trabalho porque o objetivo é estudar os principais recursos em que o HTML5 e o *Flash* mais se assemelham e verificar quando e em quais condições um pode ser mais eficiente que o outro.

2.2.3 Multimídia

O suporte a comandos multimídias é um dos grandes diferenciais do HTML5. Ele permite que recursos como áudio e vídeo, somente possíveis nos navegares graças a *plug-ins* de tecnologias proprietárias, a exemplo de *Flash, SilverLight* e *Java*, sejam incorporados aos mesmos de forma nativa e código aberto.

2.2.3.1 Áudio

Para que pudessem ser inseridos sons nas criações *web*, os desenvolvedores tinham que recorrer *a plugins* de tecnologias proprietárias, como o

[27] Interface de programação de aplicação

Flash e o *Java*. No HTML5, foi implantada uma nova *tag*, a *tag áudio*, que possibilita a inserção de sons, nativo no próprio *browse*r.

Um elemento de Áudio representa um fluxo de som ou de áudio. Em HTML5, a principal forma de incluir sons e áudios às páginas é através do elemento <áudio>. Este elemento permite que seja enviado, ao navegador do cliente, uma mensagem de erro ou outra tecnologia de reprodução caso o mesmo não possua suporte ao elemento. (MICROSOFT DEVELOPER NETWORK, 2012).

A *tag* <áudio> é um elemento de HTML5 que é introduzido diretamente no código HTML; nela, o atributo *src* indica o caminho do áudio que será reproduzido. Cada navegador possui um *player* interno que pode ser utilizado definindo o atributo *controls* da *tag* <áudio> como *true;* o *designer* e o funcionamento dos *players* internos variam para cada navegador e, caso o desenvolvedor queira, poderá criar um navegador padrão, mas precisará usar códigos *JavaScript* para isso. (MICROSOFT DEVELOPER NETWORK, 2012).

Para inserir áudios e sons em aplicações e páginas HTML5, basta inserir no código HTML a seguinte instrução:

```
1 <audio src=”musica.mp3” controls>
2 </audio>
```

Onde: *áudio* é o nome da *tag* usada para reproduzir áudios e sons; *src* é o atributo que especifica o caminho onde se encontra o áudio a ser reproduzido e *controls* é o atributo que chama o *player* interno padrão do navegador.

É possível que o navegador do cliente não possa reproduzir o formato de áudio do arquivo passado ao elemento áudio*.* Para isso, o elemento áudio possui um elemento filho chamado *source,* com o qual é possível especificar formatos de áudio alternativos para reprodução, como se pode observar no exemplo:

```
1 <audio controls="true" autoplay="true">
2 <source src="mus.oga" />
3 <source src="mus.mp3" />
4 <source src="mus.wma" />
5 </áudio>
```

A tabela 1, a seguir, apresenta o suporte aos formatos de áudio nos principais navegadores:

Tabela 1 - Formato de áudio e suporte pelos principais navegadores

O elemento <áudio> suporta três formatos de sons até o presente momento: *MP3*, *WAV* e *OGG*.

Navegadores	***MP3***	***WAV***	***OGG***
Internet Explorer 10	Sim	Não	Não
Firefox 25.0	Sim	Sim	Sim
Chrome 30	Sim	Sim	Sim
Safari 7.0	Sim	Sim	Não
Opera	Não	Sim	Sim

Fonte: Autor

O novo elemento *áudio* do HTML5 possui vários atributos que podem ser usados para definir o funcionamento do *player* de acordo com o interesse do desenvolvedor. São eles (W3C, 2010):

a) autoplay – atributo que instrui ao elemento áudio que comece a reprodução do arquivo de áudio assim que o mesmo for carregado;

b) preload – esse atributo apresenta informações se o carregamento do fluxo de áudio junto com a página é desejável e assume três valores:

- "*none*": instrui ao elemento á*udio* que o usuário não necessita do fluxo de áudio e a diminuição de tráfego desnecessário é desejável;
- "*metadata*": instrui ao elemento á*udio* que o usuário não necessita do fluxo de áudio, mas o carregamento dos metadados (duração, etc.) é desejável;
- "auto": instrui ao elemento *áudio* que o usuário necessita do fluxo de áudio e que seu carregamento com a página é desejável;

c) controls – atributo que instrui ao elemento de á*udio* que o usuário usará o *player* interno, disponibilizado pelo próprio navegador, para interagir com o áudio;

d) *loop* – instrui o elemento *áudio* a reiniciar a reprodução do áudio assim que a mesma acabar;

e) *mediagroup* – instrui o elemento áudio *a* vincular vários fluxos de áudios juntos;

f) src – este atributo instrui ao elemento áudio o caminho para o arquivo de áudio.

2.2.3.2 Vídeo

O elemento vídeo possui os mesmos atributos que o elemento áudio, além de alguns a mais, sendo que ele é usado, principalmente, para a reprodução de vídeos, embora também possa conter imagens e áudios associados. (HEY, 2010).

Este elemento em HTML5 é um padrão crescente que não está vinculado a nenhum formato de vídeo específico. O principal problema que afeta sua utilização é, provavelmente, o fato de não existir um padrão de formato que seja suportado por todos os navegadores. (MICROSOFT DEVELOPER NETWORK, 2012).

O elemento vídeo é muito parecido com o elemento áudio: ele é iniciado através da instrução <vídeo>, direto no código HTML. Com o atributo *controls*, é apresentado ao usuário um *player* interno do navegador, para que o mesmo possa controlar a reprodução do vídeo.

A sintaxe usada para chamar o elemento vídeo no HTML5 é a seguinte:

```
1 <video src="musica.mp4" controls autoplay>
2 <p>Seu navegador não suporta vídeo HTML5</p>
3 </vídeo>
```

Onde: *src* é o caminho para o arquivo que será reproduzido, o atributo *controls* define que o navegador irá mostrar seus controles de reprodução internos, que variam de navegador para navegador, e o atributo *autoplay* instrui que o vídeo comece a reproduzir assim que for carregado.

O elemento *vídeo* do HTML5 possui alguns atributos a mais em relação ao elemento *áudio*, quais sejam (W3C, 2010):

a) autoplay – atributo que instrui ao elemento *vídeo* que comece reprodução do arquivo de vídeo assim que o mesmo for carregado;

b) preload – esse atributo do HTML5 representa informação sobre se o carregamento do arquivo de vídeo, juntamente com página, é desejável, e assume três valores:

- "*none*": instrui ao elemento vídeo que o usuário não necessita do arquivo de vídeo e a diminuição de trafego desnecessário; melhor dizendo, indica que o vídeo não deve ser pré-carregado;

- “*metadata*”: instrui ao elemento vídeo que o usuário não necessita do arquivo de vídeo, mas o carregamento dos metadados (duração etc.) é desejável;
- “auto”: este valor instrui ao elemento vídeo que o usuário necessita do arquivo de vídeo e que seu carregamento com a página é desejável;

c) controls – este atributo instrui ao elemento de vídeo que o usuário usará o *player* interno disponibilizado pelo próprio navegador para interagir com o vídeo;

d) loop – instrui ao elemento vídeo reiniciar a reprodução do vídeo assim que o mesmo acabar;

e) height – define a altura do *player* de vídeo em *pixels*;

f) width – define a largura do *player* de vídeo em *pixels*;

g) poster – o endereço de uma imagem para que o elemento vídeo mostre no *player* enquanto não houver dados de vídeo disponível;

h) muted – atributo que instrui ao elemento vídeo como reproduzir o vídeo com o fluxo de áudio desativado;

i) mediagroup – instrui ao elemento vídeo *como* vincular vários vídeos e/ou fluxos de áudios juntos;

j) src – este atributo instrui ao elemento vídeo o caminho para o arquivo de vídeo.

Assim como no elemento de *áudio*, para o caso de o navegador do cliente não ser capaz de ler o formato do arquivo de vídeo a ser reproduzido, é possível especificar vários formatos de arquivos com a *tag* <*source*>, de modo que o *player* selecione aquele que for mais compatível, conforme mostra o trecho abaixo:

```
1 <video controls="true" poster=”img.jpg”>
2 <source src="video.mp4" type=”video/mp4”/>
3 <source src="video.webm" type=”video/webm” />
4 <source src="video.ogv" type=”video/ogg” />
5 </video>
```

A tabela 2, a seguir, apresenta o suporte aos formatos de vídeo nos principais navegadores:

Tabela 2 - Formato de vídeo e suporte pelos principais navegadores

O elemento <vídeo> suporta três formatos de sons até o presente momento: *OGG*, *MPEG* 4 e *WEBM*

Navegadores	***OGG***	***MPEG 4***	***WEBM***
Internet Explorer 10	Não	Sim	Não
Firefox 25.0	Sim	Sim	Sim
Chrome 30	Sim	Sim	Sim
Safari 7.0	Não	Sim	Não
Opera	Sim	Não	Sim

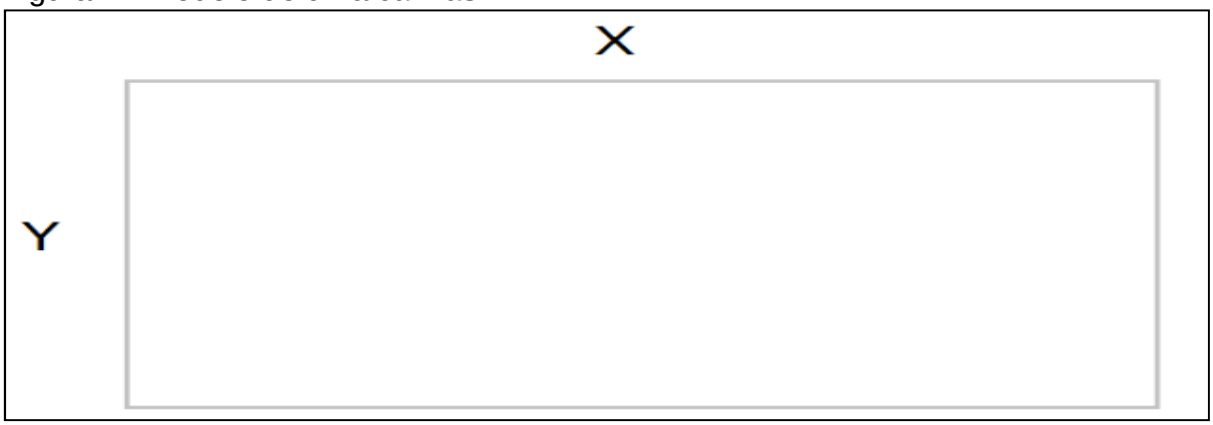
Fonte: www.w3schools.com

2.2.4 Animações (*canvas*)

Pode-se traduzir a palavra *canvas* como tela, e é isso exatamente que esse elemento HTML5 representa: uma tela onde os desenvolvedores, através de códigos *JavaScript*, podem desenhar. (CRIA WEB, 2010). Ele permite desenhar gráficos, criar animações, desenvolver jogos, renderizar imagens dinamicamente, fazer composição de fotos etc. (WHATWG, 2012).

Na figura 5 aparece o modelo de uma *canvas*, que nada mais é do que um espaço onde se podem desenhar elementos específicos.

Figura 4 - Modelo de uma *canvas*



Fonte: Douglas (2012)

Em um mesmo documento HTML5, podem ser declarados vários elementos *canvas*: basta especificar um *id* próprio a cada *canvas* na página e o desenho será direcionado ao *canvas* definido via *JavaScript.* Para poder criar algo, é necessário definir o contexto do *canvas* através de uma referência, o que permite a utilização de métodos e procedimentos para desenho e manipulação de imagens no elemento. (SHERIDAN, 2012).

Para se começar a trabalhar com *canvas*, é preciso, primeiro, inserir o elemento ao documento HTML5. Sua sintaxe é simples:

```
1 <canvas id=”meucanvas” width=”200” height=”100”>
2  Seu navegador não é compatível com o elemento canvas.
3 <br>
4 Por favor, utilize um navegador com suporte a canvas
5 </canvas>
```

Onde: o atributo *id* é um nome único que será utilizado mais tarde para que o elemento possa ser encontrado via *JavaScript,* os atributos *width* e *height* representam, respectivamente, a altura e largura do elemento, ou seja, a área de trabalho onde poderão ser feito os desenhos. Caso esses dois atributos não sejam especificados, a tela terá, inicialmente, 300 *pixels* de largura e 150 *pixels* de altura. Também é possível usar folha de estilo (CSS) para definir altura e largura do elemento *canvas* e aspectos da tela.

Agora, usando *JavaScript*, se pode, efetivamente, desenhar em *canvas*. Utilizando os diversos métodos e procedimentos existentes que permitem a criação de formar e traçados, é possível construir desenhos complexos. Quando se inicia *canvas*, ele está totalmente em branco, sendo preciso, então, informar o contexto em que se irá trabalhar, contexto esse que possui os métodos de desenho. (CRIAR WEB, 2010). Eis um exemplo:

```
1 //Recebe o elemento canvas
2 var canvas = document.getElementById('meucanvas')
3 //Acesso ao contexto de '2d' deste canvas,
  necessário para desenhar
4 var contexto = canvas.getConext('2d')
5 //Desenhando no contexto do canvas
6 context.fillRect(50,0,10,150)
```

Os comandos *JavaSript* de desenho não são suportados por todos os navegadores, então é necessário que, antes de começar a escrever os *scripts*, seja verificado se o navegador em questão está apto a interpretar os métodos utilizados. O código a seguir mostra como fazer as verificações necessárias para o navegador não tentar interpretar scripts desconhecidos caso ele não tenha suporte a comandos de desenho. (CRIAR WEB, 2010):

```
1  //Recebe o elemento canvas
2  var elemento = document.getElementById('meucanvas');
3  //Comprovação sobre se encontra um elemento
4  //e pode-se extrair seu contexto com getContext(),
   que indica
5  //compatibilidade com canvas
6  if (elemento && elemento.getContext) {
7  //Acesso ao contexto de '2d' deste canvas,
   necessário para desenhar
8  var contexto = elemento.getContext('2d');
9  if (contexto) {
10 //Se tem o contexto 2d é que tudo foi bem e pode-se
   começar a desenhar
11 //Começa desenhando um retângulo
12 contexto.fillRect(0, 0, 150, 100);
13 //muda a cor de estilo de desenho a vermelho
14 contexto.fillStyle = '#cc0000';
15 //desenha outro retângulo
16 contexto.fillRect(10, 10, 100, 70);
17 }
18 }
```

2.2.5 Execução *offline*

A tecnologia do HTML5 também possibilita a execução de aplicações *web* no modo *offline*[28]. Com o intuito de que aplicações *web* e todos os seus recursos pudessem ser utilizados em navegadores desconectados da Internet, o HTML5 tornou possível a especificação de um arquivo manifesto, no qual são listados todos os arquivos necessários que o navegador cliente precisa ter em cache para o caso de uma aplicação no modo *offline*. Esta especificação também os torna responsáveis pela interpretação e execução dos aplicativos. (LOBO FILHO, 2010).

[28] Desligado ou desconectado

O código abaixo mostra um exemplo simples de como incluir o manifesto no documento HTML5:

```
1 <!DOCTYPE html>
2 <html manifest=”manifesto.manifest”>
3 <head>
4 <!--conteudo do head-->
5 </head>
6 <body>
7 <!--conteudo do body-->
8 </body>
9 </html>
```

Sendo que o atributo *manifest*, do elemento <*hmtl*>, indica ao navegador que a aplicação possui um manifesto e informa o caminho onde este arquivo encontra-se.

A partir do momento em que o manifesto for carregado ou atualizado, ele executará uma atualização do objeto *applicationCache.* O modo como cada navegador vai reagir ao manifesto dependerá de como o mesmo foi implementado. (ELEMAR, 2010).

Exemplo de um arquivo manifesto:

```
1 CACHE MANIFEST
2 CACHE:
3 index.html
4 page1.html
5
6 FALLBACK:
7 page2.html fallback-page2.html
8
9 NETWORK:
10 *
11
12 # VERSION 1
```

Na primeira linha, acrescenta-se uma *string* fixa, que é obrigatória, o *CACHE MANIFEST*, e depois se divide o arquivo em três seções: *CACHE, FALLBACK e NETWORK* (ELEMAR, 2010), as quais são explicitadas na sequência:

a) na seção CACHE, informa-se ao navegador todos os arquivos que devem ser mantidos em cache para que a aplicação execute *offline*;

b) na seção *FALLBACK*, informam-se opções alternativas de arquivos para serem executados quando a aplicação estiver *offline*. No exemplo, o manifesto indica ao navegador que, em modo *offline*, quando a aplicação pedir o arquivo page2.html, ele deverá usar o arquivo *fallback-page2*.html, que está em *cache;*

c) já a seção *NETWORK* indica ao navegador quais os arquivos que não devem ser mantidos em *cache*. Também é possível usar a máscara todos (*), a qual indica que todos os arquivos que não forem *CACHE* ou *FALLBACK* são *NETWORK.*

O HTML5 também disponibiliza duas APIs de armazenamento *client-side* que podem ser bem úteis na criação de aplicações com execução *offline*, a *Web Storage* e a *Web* SQL[29] *Databases*. (ELEMAR, 2010).

A *Web Storage* oferece a opção de duas formas de armazenamento*: sessionStorage e localStorage* (ELEMAR, 2010):

a) na *sessionStorage*, os dados criados ficarão disponíveis apenas para aquela janela e até que a mesma seja fechada; se o usuário abrir outra janela, no mesmo endereço, os dados daquela seção não estarão mais disponíveis;

b) já o *localStorage* trabalha conforme o endereço. Se o usuário alterar algum dado em um determinado *site*, essa alteração estará disponível para todas as janelas que estiverem no *site*. Além disso, o usuário pode desligar seu computador, ligá-lo novamente e entrar no mesmo endereço que as alterações continuarão disponíveis, sendo que os dados somente serão deletados caso o autor especifique isso.

A *Web SQL Databases* é uma API que permite armazenamento e acesso a dados. Como o nome sugere, ela trabalha exatamente como um banco de dados, com criação de tabelas, inserção, edição e exclusão de dados, tudo através de comandos SQL. Essa API não especifica um tamanho limite para o banco, embora seja possível, na hora da criação do banco, que o desenvolvedor defina um tamanho limite que o banco poderá ocupar; ainda assim, cabe ao navegador aprovar o tamanho máximo. (ELEMAR, 2010).

---

[29] *Structured Query Language* ou Linguagem de Consulta Estruturada

2.3 *FLASH*

O *Flash* foi criado com o intuito de que as animações de *web* fossem rapidamente carregadas e executadas. Por ser um programa de imagens vetoriais, gera arquivos que descarregam mais rápido que arquivos *bitmaps, Graphics Interchange Format* [30] (GIFs) e *Joint Photographic Expert Group* [31] (JPEGs). Seu formato *streaming* permite que vídeos comecem a executar na *web* enquanto ainda estão sendo carregados. Além disso, o *Flash* possibilita que se inclua interatividade de alto nível aos *sites*, sem a necessidade de códigos complexos. (LOPES, 2012).

O *Flash*, depois de anos presente na *Web*, já está incluído em cerca de 98% dos computadores existentes. O seu maior sucesso é decorrente do seu alto poder multimídia, o qual se limita apenas pela criatividade dos criadores, e isso aliado ao fato de que seus arquivos transferidos pela Internet são pequenos. (FURLAN, 2009).

O que permite ao *Flash* construir animações com pouco peso é o fato de trabalhar com gráficos vetoriais, sendo que existem dois tipos de imagens gráficas, os mapas de *bits* e as imagens vetoriais:

a) mapa de *bits* – este tipo imagem gráfica é composta por milhões de pontos (*pixels*), onde cada ponto possui uma cor determinada. As informações destas imagens são gravadas individualmente para cada ponto, ou seja, as informações dos pontos (cor, coordenadas, transparência, saturação etc.) são armazenadas individualmente. Este tipo de gráfico é suscetível a alterações de tamanho e resolução, perdendo facilmente qualidade ao se modificarem suas dimensões;

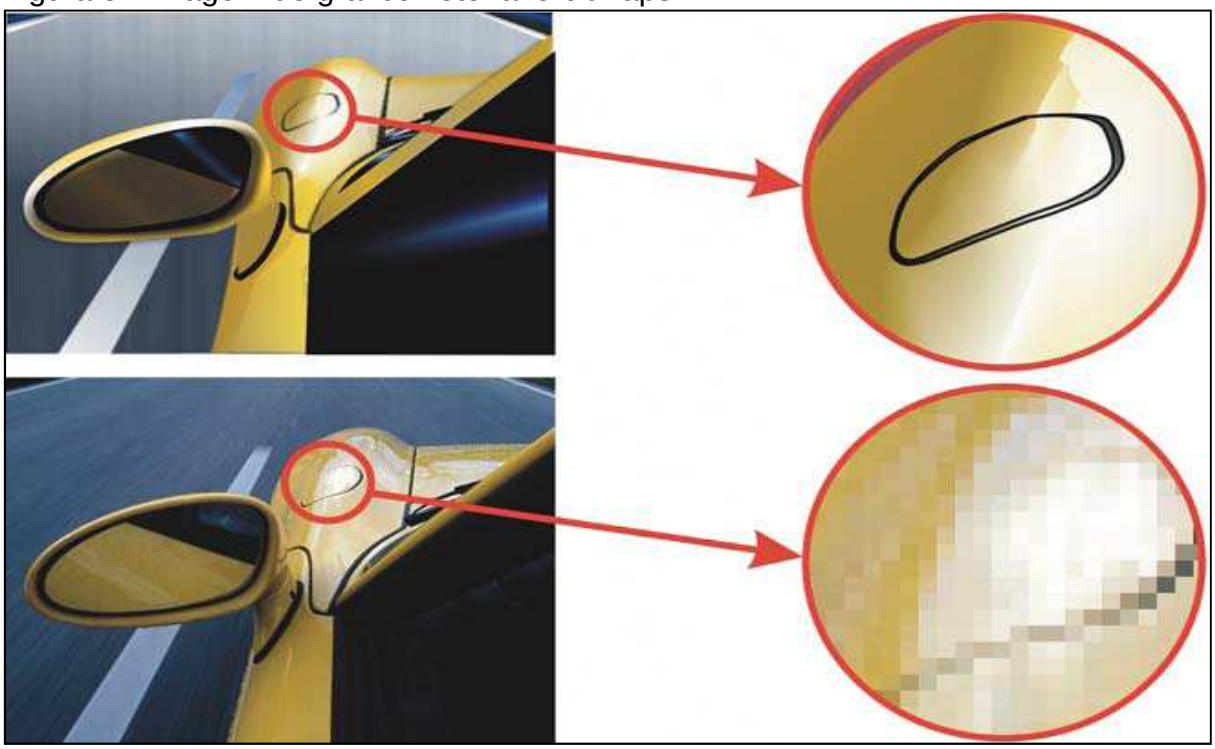
b) imagens vetoriais – são aquelas definidas por formas geométricas, como retângulos, círculos etc. Tais imagens são representadas por fórmulas matemáticas e o processador terá de calculá-las para que a placa gráfica as compreenda. Independente do *zoom* ou da resolução, este tipo de gráfico não altera a sua qualidade; mesmo aproximando muito a imagem ela não pixializa, já que os objetos são definidos automaticamente para cada grau de proximidade.

---

[30] Formato para intercâmbio de gráficos
[31] Grupo *expert* de junção de fotografias

A figura 5, abaixo, apresenta uma imagem composta de gráficos vetoriais e, a seguir, de *bitmaps*:

Figura 5 - Imagem de gráfico vetorial e bitmaps



Fonte: Adobe (2012)

2.3.1 *ActionScript*

Antes de se começar a falar dos recursos *Flash*, será feita uma breve explanação sobre a linguagem *ActionScript,* recurso amplamente presente nas criações *Flash* e que será muito citado mais adiante.

ActionScript nada mais é que uma linguagem de programação padronizada pelo *European Computer Manufacturers Association*[32] (ECMA), sob a denominação de *ECMAscript* em função de ser baseada em *scripts*. Ela possibilita ao desenvolvedor manipular melhor os dados, maior interatividade, criação de efeitos visuais via código e deixa as aplicações muito mais suaves e leves. Essa linguagem é executada com uma máquina virtual, a *ActionScript Virtual Machine* (AVM), que está incorporada no *Flash Player*. Os arquivos *Shockwave Flash* (SWF) têm o código incorporado para, então, serem executados pelo *Flash Player*. (ADOBE, 2012).

[32] Associação Europeia de Fabricantes de Computadores

Atualmente, a *ActionScript* está na sua terceira versão, batizada de *ActionScript* 3.0 (AS3), mas a sua versão anterior ainda é amplamente utilizada. Quando se inicia um projeto no *Adobe Flash* CS6, ele permite que se criem animações com *ActionScript* 2.0 ou 3.0. A principal diferença entre os dois está no fato de o AS3 ser totalmente orientado a objetos; com o AS2, os desenvolvedores colocavam o código em vários lugares, como na *timeline, movieclips*, botões, enquanto no AS3 o código é colocado nos métodos das classes. (ADOBE, 2012).

Outro ponto importante é que, junto com o AS3, foi criada uma nova versão para a máquina virtual AVM. A AVM1 executa os *scripts* AS1 e AS2, enquanto a AVM2, além de executar exclusivamente conteúdo AS3, também utiliza um novo conjunto de instruções de código de *bytes* e melhora, de forma substancial, o desempenho. Por isso, não é possível misturar códigos AS2 e AS3 em um mesmo projeto, sendo que as versões mais recentes do *Flash Player* oferecem suporte a AVM1 para serem compatíveis com aplicações que sejam herança de versões mais antigas. (ADOBE, 2012).

2.3.2 Animações

2.3.2.1 Palco

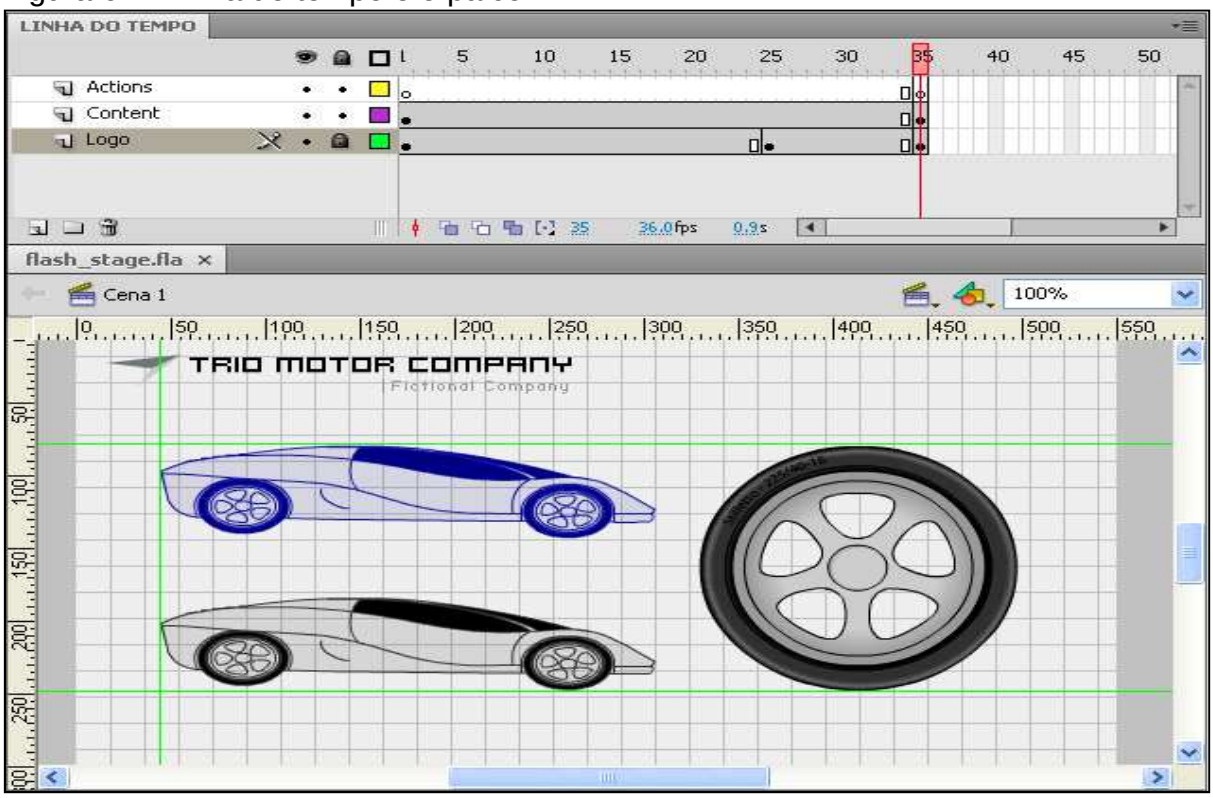
Palco é o retângulo branco onde será colocado o material gráfico no documento *Flash*. Ele representa o espaço onde a animação é exibida, sendo que somente o que está no palco será visto no momento em que a aplicação estiver rodando. (ADOBE, 2012).

Para facilitar o desenvolvimento, é possível aplicar *zoom* no palco, deixando-o todo na tela, ou simplesmente escolher um local específico, onde se queira trabalhar, e aumentá-lo. O quanto de *zoom* pode ser aplicado é definido pelo tamanho do documento e pela resolução do monitor. A aplicação de mais e menos *zoom* no palco é de 2000% e 8%, respectivamente. (ADOBE, 2012).

Para melhorar o posicionamento dos objetos no palco, o *Flash* disponibiliza réguas, as quais ficam nas laterais superior e esquerda e trabalham com um padrão de medida em *pixels* que o desenvolvedor, se desejar, pode alterar para uma medida que mais lhe ajude. Quando se traz algum objeto para o palco,

são mostradas as linhas de dimensão do objeto na régua. O desenvolvedor ainda pode criar linhas-guias apenas arrastando-as da régua até o palco. (ADOBE, 2012).

Figura 6 - A linha do tempo e o palco



Fonte: Adobe (2012)

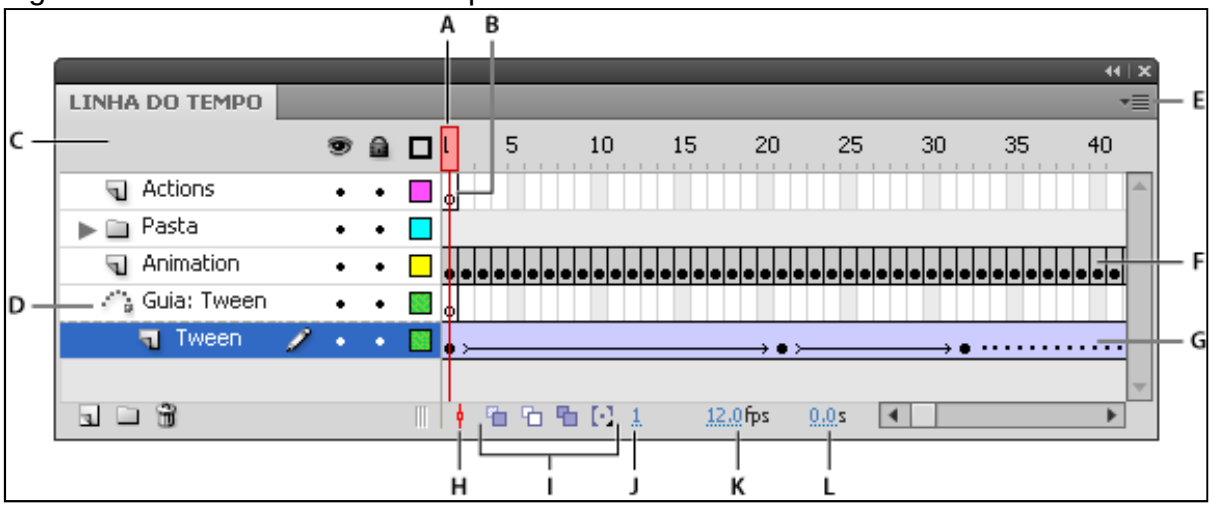
2.3.2.2 *TimeLine* (linha do tempo)

Um componente importante e que sempre esteve presente nas ferramentas de desenvolvimento *Flash* é a *Timeline* ou Linha do Tempo. Ela permite que o desenvolvedor controle a animação em cada quadro e camada através do tempo. Os documentos *Flash* trabalham com vários quadros separados pelo tempo. É possível definir as camadas como várias faixas de filmes, todas empilhadas e contendo, cada uma, sua própria imagem, que aparece no palco. (ADOBE, 2012).

As camadas são empilhadas no lado esquerdo da Linha do Tempo, sendo que os quadros de cada camada ficam em uma linha ao lado da própria camada. O cabeçalho contém a numeração dos quadros. O indicador de reprodução define qual o quadro mostrado no palco, o qual se move para esquerda e para a direita na Linha do Tempo. Os componentes mais importantes na Linha do tempo são as camadas, os quadros e o indicador de reprodução (ADOBE, 2012).

A Linha do tempo possui vários controles de seção que permitem ocultar, mostrar, bloquear ou desbloquear camadas; também permite mostrar seu conteúdo como contorno. Pode-se mudar o local de um quadro na camada ou transferi-lo para outra camada, assim como também é possível mudar a ordem das camadas. (ADOBE, 2012).

Figura 7 - Partes da linha do tempo



Fonte: Adobe

2.3.2.3 Ferramentas

O *Flash* disponibiliza diversas ferramentas para a criação de formas livres, linhas e caminhos:

a) *selection tool* (ferramenta de seleção) – ferramenta usada para selecionar e mover os objetos presentes no palco e que também permite deformação de formas;

b) *subselection tool* (ferramenta de sub-seleção) – ferramenta utilizada para a seleção de vértices e mudança de suas propriedades;

c) *free transform tool* (ferramenta de transformação livre) – permite aumentar, diminuir e rotacionar objetos selecionados;

d) *gradient transform tool* (ferramenta de transformação gradiente) – usada para controlar o efeito de preenchimento de uma forma;

e) *line tool* (linha) – possibilita a criação de linhas;

f) *lasso tool* (laço) – usada para a seleção de partes específicas de um desenho;

g) *pen tool* (caneta) – ferramenta que possibilita a criação de formar vetoriais;

h) *text tool* (texto) – para criação de textos e alteração de características e conteúdo de textos já existentes;

i) *oval tool* (oval) – para a criação de círculos ou eclipses;

j) *rectangle tool* (retângulo) – para a criação de retângulos, quadrados;

k) *polystar tool* – permite o desenho de estrelas e polígonos;

l) *pencil tool* (lápis) – ferramenta para desenhos à mão livre através do *mouse*;

m)*brush tool* (pincel) – também permite desenhos à mão livre, mas de forma relativamente mais fácil que o Pencil Tool (Lápis);

n) *paint bucket tool* (balde de tinta) – permite pintar os desenhos e formas;

o) *eyedropper tool* (conta gotas) – permite selecionar a cor de algum desenho existente e aplicá-la em outro desenho;

p) *eraser tool* (borracha) – permite apagar todo ou partes específicas de formas e desenhos;

q) *hand tool* (mão) – serve para mover o palco;

r) *zoom tool* (lupa) – permite dar *zoom* aumentando e diminuindo um ponto específico dos desenhos;

s) *stroke color* – define a cor das linhas e das formas;

t) *fill color* – define a cor de preenchimento das formas.

2.3.2.4 Biblioteca

O *Flash* disponibiliza uma biblioteca onde são armazenados e organizados todos os elementos, incluindo desenhos, *bitmaps*, formas, sons e vídeos. Ela permite classificar os itens por tipo, separá-los em pastas específicas e saber com que frequência estão sendo utilizados. (CARVALHO, 2012).

Quando se importa algum item, é possível escolher trazê-lo tanto para o palco quanto para a biblioteca, porém, ao importar um item para o palco, o mesmo também será adicionado à biblioteca, e, da mesma forma, isso acontecerá com qualquer elemento que for criado. (CARVALHO, 2012).

2.3.3 Áudio

O *Adobe Flash* oferece uma gama de possibilidades para se trabalhar com áudio. Possibilita animações que sincronizam com sons a partir da linha do tempo, ou reprodução de sons independentes da linha do tempo. Fazer com que sons apareçam e desapareçam gradativamente, ou seja, adicionado a botões. (*ADOBE*, 2010).

O *Flash* apresenta dois tipos de áudios: os de eventos e os de fluxo. No áudio de eventos, para que o som possa ser reproduzido, é necessário que todo o áudio seja baixado. Já os áudios de fluxo podem ser reproduzidos tão logo haja dados suficientes para isso; áudios de fluxo são muito usados em *sites*. (*ADOBE*, 2010).

O *Flash* possibilita pleno controle dos arquivos carregados, utilizando-se de comportamentos pré-gravados ou componentes de mídia. Comportamentos de mídia possibilitam autocontrole dos sons, permitindo comandos como interromper, pausar, retroceder etc. Além disso, os desenvolvedores podem usar as linguagens *ActionScrip*t 2.0 ou *ActionScript* 3.0 para trabalhar com os áudios. (ADOBE, 2010).

Estes são os formatos de áudio suportados pelo *Flash* (ADOBE, 2010):

a) ASND (*Windows* ou *Macintosh*), que é o formato de som nativo do Adobe® Soundbooth™;

b) WAV (somente *Windows*);

c) AIFF (somente *Macintosh);*

d) MP3 (*Windows* ou *Macintosh*);

e) AIFF (*Windows* ou *Macintosh*);

f) Sound Designer® II (somente *Macintosh*);

g) Sound Only QuickTime Movies (Windows ou *Macintosh*);

h) *Sun* AU (*Windows* ou *Macintosh*);

i) Sons do *System* 7 (somente *Macintosh*);

j) WAV (*Windows* ou *Macintosh*).

2.3.4 Vídeo

A tecnologia *Flash* é amplamente usada na *web* para incorporar vídeos digitais a *sites*. Seus formatos FLV e F4V (H.264), trazem amplas possibilidades aos

desenvolvedores, como controle interativo, união de dados a vídeos, gráficos e sons, sendo que os vídeos presentes na *Internet* podem ser vistos por praticamente todos os usuários *web*. (*ADOBE*, 2011).

É possível trabalhar com vídeos em *Flash* de três maneiras (ADOBE, 2011):

a) *download* progressivo de um servidor *web* – é a forma mais utilizada de trabalhar com vídeos em *Flash*, já que deixa o arquivo SWF pequeno, pois mantém o arquivo de vídeo separado do arquivo SWF final. É possível fornecer ao usuário controles e, assim, possibilitar uma maior interação usando o componente FLV*PlayBack* ou a linguagem *ActionScrip*t;

b) fluxo de vídeos com *Adobe Flash* Mídia Server – o *Adobe Flash* Mídia Server é uma solução para a criação de vídeos em tempo real. Ele usa o protocolo *Real-Time Messaging Protocol*[33] (RTMP), protocolo para aplicação em tempo real no servidor. Este método também mantém o arquivo de vídeo separado do arquivo SWF e possibilita a utilização do componente FLV*PlayBack* e de *ActionScript*, para tornar a experiência do usuário mais interativa;

c) incorporar vídeo ao documento *Flash* – o *Flash* permite que os vídeos sejam incorporados diretamente aos arquivos SWF do *Flash*, mas isso acarreta um aumento substancial do arquivo e é recomendado apenas para vídeos de, no máximo, 10 segundos, pois a sincronia do áudio com o vídeo pode ficar ruim quando vídeos maiores são incorporados ao documento.

Um modo de trabalhar com arquivos de vídeo no ambiente *Flash* é utilizar o componente FLV*PlayBack* ou comandos *ActionScript* e reproduzir dinamicamente arquivos externos. É possível ainda usá-los em conjunto. Chamando um arquivo FLV ou F4V através de um *link* URL, e adicionando o componente *FLVPlayBack* ou comandos *ActionScript* ao documento, é possível introduzir controles na reprodução do vídeo. Os *links* dos vídeos utilizados podem ser de arquivos em pastas locais ou disponibilizados na *web*.

A forma de se controlar um arquivo de vídeo incorporado a um documento *Flash* é controlar a linha do tempo onde o vídeo se encontra. Por exemplo, se um vídeo estiver na linha do tempo principal de um documento, e se quiser parar a reprodução do vídeo, chama-se uma ação de parar para essa linha do tempo. Assim, se um objeto de vídeo estiver dentro de um símbolo de *clip* de vídeo,

[33] Protocolo de Mensagem em Tempo Real

controlando a linha de tempo desse símbolo é possível controlar o objeto de vídeo. (ADOBE, 2011).

O Flash ainda possibilita que arquivos FLV definidos com configuração *ActionScript* 2.0, sejam controlados com comportamentos. Comportamentos são *scripts* pré-definidos que introduzem comandos *ActionScript* ao documento sem a necessidade da criação de código *ActionScript.* Contudo, somente podem ser utilizados em documentos *ActionScript* 2.0, pois não estão disponíveis em *ActionScript* 3.0. (*ADOBE*, 2011).

Segundo as especificações, o *Flash* trabalha com três *codecs* de vídeo diferentes (ADOBE, 2011):

a) H.264 – *codec* de vídeo incorporado ao *Flash Player* desde a versão 9.0.r115; os arquivos de vídeo *Flash* que funcionam com esse *codec* possuem uma qualidade maior quando comparados com os outros dois *codecs*, com uma taxa de *bits* substancialmente melhor, exigindo também mais recursos computacionais para reprodução;

b) ON2 VP6 – este é o *codec* de vídeo recomendado pelo *Adobe* para o desenvolvimento de arquivos *Flash* que serão rodados no *Flash Player* 8 ou em versões posteriores. Ele possui qualidade melhor que o *codec Sorenson Spark,* apesar de oferecer a mesma taxa de quadros; por isso mesmo, necessita de um maior poder computacional para reprodução e é perceptivelmente mais demorado ao codificar;

c) *SORENSON SPARK* – com qualidade inferior aos outros dois *codecs*, é recomendado para rodar em computadores mais antigos e com poucos recursos computacionais, e, também, quando for rodar na versão do *Flash Player* 7 ou anteriores.

A tabela 3 mostra os *codecs* suportados pelo *Flash* em suas respectivas versões:

Tabela 3 - *Codecs* suportados pelo *Adobe Flash*

Codec	**Versão SWF (Versão de Publicação)**	**Versão do *Flash Player* (necessária para reprodução)**
Sorenson Spark	6	6, 7, 8
	7	7, 8, 9, 10
ON2 VP6	6,7,8	8, 9, 10
H.264	9.2 ou posterior	9.2 ou posterior

Fonte: Adobe (2011)

2.4 TRABALHOS CORRELATOS

Esta seção visa apresentar trabalhos relacionados ao trabalho desenvolvido na presente pesquisa.

2.4.1 Estudo de viabilidade do HTML5 para desenvolvimento *web*

Este trabalho foi desenvolvido na Universidade Estadual de Maringá por, Daniel Fuverki Hey, em 2010.

A proposta da pesquisa era um estudo de viabilidade do HTML5 para desenvolvimentos na *web*. É feito um estudo detalhado sobre a tecnologia HTML5, revelando as diferentes situações para emprego de ferramentas de programação mais adequadas. O trabalho aborda os atributos gerais da linguagem, descrevendo o funcionamento dos novos elementos estruturais de marcação, conteúdo multimídia e aplicações disponíveis nos navegadores. Através da análise dos motores de renderização *web* e características de aplicabilidade de cada ferramenta, foi realizada uma avaliação comparativa das tecnologias atuais, de modo a determinar pontos críticos em projetos *web*, buscando diferentes alternativas de implementação, a fim de realizar uma breve análise com prós e contras das linguagens disponíveis (HEY, 2010).

2.4.2 Estudo comparativo entre as linguagens de programação PHP, ASP e JSP

O trabalho desenvolvido por Graziela Barnabé (2010) trata-se de um estudo comparativo entre as tecnologias de programação PHP, ASP e JSP.

A *Internet* é ferramenta indispensável no processo de modernização e vem evoluindo a cada dia. As linguagens PHP, ASP e JSP possuem cada uma suas particularidades, vantagens, desvantagens, recursos e ambientes específicos. Com base em tais informações, pretende-se analisar as linguagens citadas acima, apresentando o quanto elas podem auxiliar os desenvolvedores *web* e em quais aspectos cada uma se destaca, tornando assim, mais clara a utilização destas em projetos voltados à Internet.

Atualmente, podem ser encontradas várias linguagens de programação, mas é preciso saber qual é a melhor opção na hora de desenvolver uma aplicação

web. A escolha equivocada pode fazer o programador levar muito mais tempo do que ele planejava e, muitas vezes, tornar o projeto inviável. Diante disso, torna-se conveniente o estudo das linguagens para *web*, fazendo um estudo comparativo, mostrando os principais recursos, vantagens e desvantagens, e o que cada uma delas oferece para desenvolver as aplicações *web* de forma eficiente.

2.4.3 Comparativo entre as linguagens de programação *Java* e *Ruby* para projetos que utilizem *Extreme Programming*

O trabalho apresentado por Alan Kloh e Rodolfo Cordeiro Burla de Aguiar (2007) traz um estudo comparativo entre as linguagens *Java* e *Ruby*, com o *framework*[34] *Rails*, no que se refere à utilização da metodologia de desenvolvimento *Extreme Programming*[35] (XP). Esse comparativo foi realizado através de uma pesquisa de tecnologias para as linguagens, a fim de identificar as vantagens e desvantagens das mesmas. Para que esse objetivo fosse atingido, foi necessário estudar o funcionamento e tecnologias disponíveis para cada uma das linguagens e escolher aquelas que se acreditava serem mais úteis para a adoção do XP. Com isso, estudaram-se alguns *frameworks* para *Java* e *Ruby*.

Para o estudo, também foram criados dois *blogs*, visando comparar a qualidade do produto final, o tempo de duração do processo de desenvolvimento e a facilidade de sua utilização. Os *blogs* deveriam ser simples e apresentar as mesmas funcionalidades, já que o objetivo era comparar as duas linguagens. Baseados nos resultados obtidos, acredita-se que a utilização do *framework Ruby on Rails*[36], para o desenvolvimento de aplicações *web* desse porte, é o mais recomendado.

2.4.4 Desenvolvimento de uma metodologia para avaliação de desempenho gráfico 3D de plataformas com suporte ao *WebGL*

O trabalho desenvolvido por Anderson Roberto Slivinski (2011) veio com a ideia de criar uma metodologia para avaliação de desempenho gráfico 3D de

---

[34] Conjunto de classes que incorpora um projeto abstrato de soluções para uma família de problemas relacionados.
[35] Programação extrema
[36] *Framework* livre

plataformas com suporte à *Web Graphics Library*[37] (WebGL). Com a popularização da Internet, os navegadores *web* passaram a serem as aplicações mais utilizadas nos computadores pessoais e nos dispositivos portáteis, como *smartphones* e *tablets,* embora os mesmos não tenham sido, inicialmente, planejados para servir como plataforma de aplicações. A fim de contornar este problema, diversas tecnologias foram criadas para fornecer melhores ambientes de desenvolvimentos aos desenvolvedores, como o HTML5 e o WebGL.

O estudo de Slivinski, então, tem como objetivo realizar uma revisão sistemática para identificar trabalhos que apresentem alguma metodologia ou técnica para a avaliação de desempenho gráfico 3D. Ao fim do trabalho, foi desenvolvida uma metodologia para servir de auxílio a programadores que desejam implementar uma ferramenta para análise da performance gráfica 3D em plataformas que suportem o WebGL.

[37] Biblioteca Gráfica da Web, que reúne comandos para aplicações gráficas

3 DESENVOLVIMENTO DO ESTUDO COMPARATIVO

Neste capítulo, chega-se ao objetivo do projeto, que é a comparação dos recursos HTML5 e *Flash*.

É importante enfatizar que o objetivo não é a comparação das tecnologias *Flash* e HTML5, pois as duas em si foram criadas com propósitos diferentes e não possuem o mesmo objetivo como solução final. O HTML5 é uma linguagem de marcação de texto, que abrange a linguagem *front-end JavaScript* e os documentos de apresentação em folha de estilo *CSS*, criada basicamente para o desenvolvimento de páginas *web*, enquanto o *Flash* é uma ferramenta para a criação de animações vetoriais que utiliza códigos *ActionScript* para tornar suas aplicações mais robustas. Ainda assim, elas possuem alguns recursos semelhantes, e esses, sim, podem ser avaliados e comparados.

Tendo todos esses fatores em mente, através de métricas bem definidas deseja-se fazer a avaliação de alguns recursos separadamente e, então, com os resultados obtidos, poder definir vantagens e desvantagens de cada um deles.

3.1 METODOLOGIA DE COMPARAÇÃO

Para conclusão deste trabalho foi necessário utilizar algumas metodologias que avaliassem as características de cada recurso das tecnologias em questão. Essas características referem-se a questões de qualidade e desempenho, como qualidade de áudio e vídeo, uso de memória e processador entre outros. Há diversos fatores referentes à análise dos recursos tecnológicos e, para esta pesquisa, foram escolhidos os mais relevantes para os objetivos propostos. Assim, esta seção começa definindo as métricas que serão utilizadas para fazer a parte prática deste trabalho.

Antes de serem iniciados os testes subjetivos e objetivos, foi necessário adquirir algumas ferramentas que auxiliassem nos testes que serão abordados nas próximas seções. Para poder colocar as sequências de vídeos nos formatos desejados, foram utilizados os programas *Sorenson Squeeze Trial* e o *Free Studio.* Para separar as sequências de áudio em arquivos de 10 a 25 segundos de duração, foi utilizado o *Wondershare Video Editor*, e, para o monitoramento de memória e processador, o programa *Process Explorer*.

3.2 COMPARAÇÃO DOS RECURSOS

Para as comparações, foram escolhidos os navegadores mais utilizados atualmente na *web*: *Internet Explorer*, *Mozilla Firefox* e *Google Chrome*.

3.2.1 Qualidade de vídeo

O modo para realizar testes subjetivos de avaliação de qualidade de vídeo baseia-se na escolha de observadores que assistiram alguns vídeos e atribuíram notas em relação à qualidade visual dos mesmos. A *International Telecommunication Union*[38] (ITU) recomenda vários métodos de avaliação subjetiva para os testes de qualidade de vídeo. Tais métodos definem parâmetros importantes para uma avaliação de qualidade de vídeo, dentre os quais estão: o número mínimo de participantes, as instruções para esses participantes, as condições de visualização, e o modo e sequência nos quais os vídeos escolhidos devem ser apresentados. (GONÇALVES, 2012).

Diversos fatores, desde o ambiente até a luminância do monitor, podem alterar de forma relevante os resultados das avaliações subjetivas de qualidade de vídeo. (GONÇALVES, 2012).

A tabela 4, abaixo, mostra as condições recomendadas pela ITU para testes subjetivos de avaliação de vídeo:

Tabela 4 - Codificações recomendadas pela ITU

Parâmetro	Valor
Distância de visualização	1-8H(2)
Pico de luminância do ecrã	100-200cd/m²
Rácio entre a luminância do ecrã inativo e pico de luminância	≤ 0.05
Rácio entre a luminância do fundo da imagem no ecrã e o pico de luminância da imagem	≤ 0.2
Crominância do fundo/plano	D_{65}(3)
Iluminação de fundo da sala	≤ 20 lux
Máximo ângulo de observação relativamente à normal	30º

Fonte: ITU

Para a escolha dos participantes dos testes, a ITU-T P.910 recomenda que tenham, pelo menos, 15 não especialistas, sendo os participantes divididos em especialistas e não especialistas. Os especialistas seriam avaliadores que possuem

[38] União Internacional de Telecomunicações

familiaridade com técnicas e avaliação de qualidade de imagens, enquanto os não especialistas, avaliadores sem nenhum conhecimento mais aprofundado, que avaliem os vídeos de forma mais apreciativa. Também é necessário que todos os avaliadores tenham uma visão normal, ou normalizada com ajuda de lentes, e que possam distinguir cores. (GONÇALVES, 2012).

Antes de iniciarem os testes propriamente ditos, é realizada uma seção de treino, para que os avaliadores fiquem familiarizados com a melhor e a pior qualidade dos vídeos apresentados. Também é explicado o funcionamento do método utilizado na avaliação, assim como o objetivo do mesmo. (GONÇALVES, 2012).

O grupo ITU recomenda que os testes não ultrapassem o tempo de 30 minutos, para que cansaço e impaciência não afetem os participantes. Os vídeos, geralmente, possuem tempo de 10 segundos, mas, dependendo do método utilizado, eles podem ter uma duração maior. No intervalo dos vídeos, deve ser mostrada uma imagem cinzenta ao avaliador, de 2 ou 3 segundos; no fim das sequências de vídeo, o participante atribui uma nota aos mesmos. (GONÇALVES, 2012).

As formas pelas quais são apresentadas as sequências de vídeos aos avaliadores podem ser divididas em três tipos: *Single Stimulus*[39] (SS), *Double Stimulus*[40] (DS) e métodos de comparação.

No método *Single Stimulus,* os vídeos avaliados são apresentados individualmente, um após o outro. No método *Double Stimulus,* os vídeos são apresentados em duplas, sendo que um é a referência com qualidade superior e o outro, sua forma degradada. Já nos métodos de comparação, os vídeos ainda são mostrados em duplas, mas sem um vídeo referência. Na próxima seção, serão explicados alguns dos métodos utilizados.

3.2.1.1 Métodos *single stimulus*

a) *Absolute Category Rating*[41] (ACR) – neste método, os participantes avaliam os vídeos um de cada vez, utilizando uma escala de qualidade. (GONÇALVES, 2012). Um exemplo desta escala é mostrado na tabela 5, a seguir:

---

[39] Estímulo único ou simples
[40] Estímulo duplo
[41] Classificação de Categoria Absoluta

Tabela 5 - Escala de qualidade

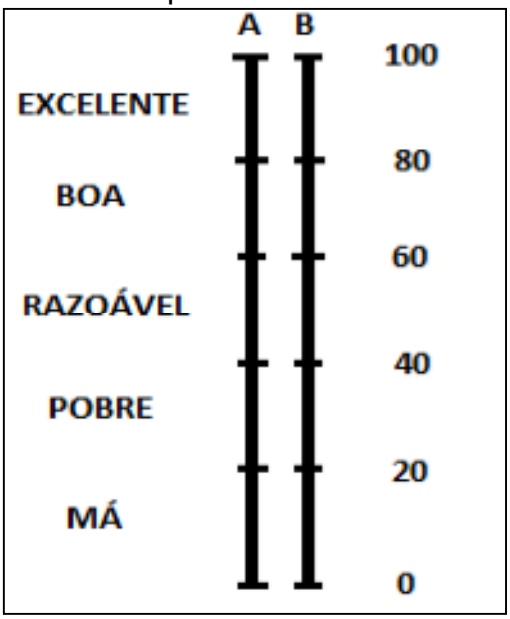
5	Excelente
4	Boa
3	Razoável
2	Ruim
1	Péssima

Fonte: ITU

b) *Absolute Category Rating with Hidden Reference*[42] (ACR-HR) – este método de avaliação utiliza, para comparação de qualidade de áudio, valores de *Differential Mean Opinion Scores*[43] (DMOS), valor que é obtido por meio da diferença entre a pontuação de vídeos com degradação e vídeos referência. O método recebe o nome de “Referência Escondida”, pois os vídeos referência são mostrados no meio da avaliação, sem que o participante tenha conhecimento disso. (GONÇALVES, 2012);

c) *Single Stimulus Continuous Quality Evaluation*[44] (SSCQE) – com este método, a avaliação de qualidade de vídeo é feita ao mesmo tempo em que este é reproduzido. Com o modo de qualificação *Slider*, o participante vai alterando a pontuação de qualidade em cada instante do vídeo. A pontuação dada ao vídeo vem de uma escala contínua de qualidade, dividida em cinco partes (figura 8):

Figura 8 - Escala contínua de qualidade normalizada



Fonte: Gonçalves (2012)

[42] Classificação de Categoria Absoluta com Referência Oculta
[43] Parecer Diferencial de Pontuação Média
[44] Avaliação contínua de qualidade com estímulo simples

3.2.1.2 Métodos *double stimulus*

a) *Degradation Category Rating*[45] (DCR) – também é conhecido como *Double Stimulus Impairment Scale*[46] (DSIS), onde os vídeos são mostrados em duplas, um sendo a referência (vídeo original) e o outro, uma versão da referência com degradações. A versão degradada é a condição de teste à qual o participante que está avaliando deve atribuir uma pontuação, utilizando a escala de artefatos de cinco níveis, a qual vai de muito incômodo (valor 1) a imperceptível (valor 5). Para isto, o vídeo de referência deve ter uma qualidade maior ou igual à condição de teste (vídeo degradado).

A tabela 6 apresenta os níveis de valores da escala de artefatos:

Tabela 6 - Escala de artefatos

5	Imperceptível
4	Perceptível
3	Ligeiramente Incômodo
2	Incômodo
1	Muito Incômodo

Fonte: ITU

Este método oferece duas variantes de avaliação:

- variante I – a dupla, vídeo de referência e vídeo degradado, é mostrada ao mesmo tempo, uma única vez, e, então, o participante pontua a condição de teste;
- variante II – a dupla, vídeo de referência e vídeo degradado, é mostrada ao mesmo tempo, no mínimo duas vezes seguidas, e, então, o participante pontua a condição de teste.

b) *Double Stimulus Continuous Quality Scale*[47] (DSCQS) – esse método assemelha-se ao anterior, embora com algumas diferenças: os participantes atribuem a pontuação aos vídeos sem saber qual é a referência e qual o degradado; além disso, a pontuação dada utiliza a escala de qualidade contínua. (GONÇALVES, 2012).

Este método também oferece duas variantes, como se observa a seguir:

---

[45] Classificação de Categoria de Degradação
[46] Escala de imparidade com duplo estímulo
[47] Escala contínua de qualidade com duplo estímulo

- variante I – o participante pode ficar alternando entre o vídeo referência e o degradado até que esteja seguro sobre qual pontuação atribuir às sequências;
- variante II – essa variante é para múltiplos avaliadores: os vídeos são apresentados duas ou mais vezes e os participantes atribuem pontuação para as duas sequências.

c) *Simultaneous Double Stimulus for Continuous Evaluation*[48] (SDSCE) – nesse método, a pontuação é atribuída com uma escala contínua de qualidade, por meio de uma referência e uma condição de teste na qual são mostrados em simultâneo e o participante é informado sobre qual é a referência. (GONÇALVES, 2012).

3.2.1.3 Métodos de comparação

a) *Pair Comparison*[49] (PC) – neste método não existe um vídeo referência: as duas sequências são consideradas condições de teste, e é pedido ao participante que indique qual condição possui melhor qualidade. Este método também não tem uma escala de qualidade. (GONÇALVES, 2012);

b) *Stimulus Comparison Adjectival Categorical Judgement*[50] (SCACJ) - semelhante ao *Pair Comparison*, neste tipo de comparação são apresentadas duas sequências de vídeo, avaliadas simultaneamente, sem que haja uma referência; diferentemente do método *Pair Comparison*, a avaliação das sequências é feita por meio de um escala de qualidade, na qual o avaliador informa qual das duas imagens tem melhor e qual tem pior qualidade e o quanto melhor ou pior. (GONÇALVES, 2012).

Tabela 7 - Escala de comparação SCACJ

-3	Muito Pior
-2	Pior
-1	Ligeiramente Pior
0	Igual
1	Ligeiramente Melhor
2	Melhor
3	Muito Melhor

Fonte: ITU

[48] Avaliação contínua com duplo estímulo simultâneo
[49] Comparação de pares ou duplas
[50] Estímulo de Comparação de Julgamento de Categoria Adjetiva

Para os testes subjetivos de qualidade de vídeo, escolheu-se o método de comparação SCACJ por este atender amplamente às necessidades do presente estudo. Como um dos objetivos desse trabalho é a comparação das qualidades de vídeo disponibilizadas pelas tecnologias, este método é claramente aplicável, pois não especifica uma referência, o que, para o estudo, é totalmente dispensável. Além disso, possui uma forma clara de pontuação, que ajuda a mensurar a diferença de qualidade entre as tecnologias, diferentemente do método PC, que dificulta medir a diferença entre qualidade entre as tecnologias.

3.2.2 Qualidade de áudio

Assim como na avaliação de qualidade subjetiva de vídeo, existem recomendações da ITU para avaliações subjetivas de áudio aceitáveis. Tais recomendações informam condições que tornam a avaliação o mais fiel possível à realidade, lembrando que fatores externos ou testes mal conduzidos podem levar a resultados não confiáveis. Nas próximas seções, serão mostradas as recomendações utilizadas neste trabalho.

Nas avaliações subjetivas de qualidade de áudio, é fundamental que os participantes sejam todos especialistas, ou seja, que todos tenham experiência em encontrar degradações nas sequências de áudio, condição que se torna mais importante na medida em que se espera mais qualidade das sequências. (BARBEDO, 2004).

O número de avaliadores difere dependendo dos resultados que se desejam obter: quanto mais avaliadores melhores, mais confiáveis os resultados dos testes. Quando as condições dos testes são cuidadosamente controladas, observa-se que 20 avaliadores são mais que o suficiente para conclusões confiáveis. Pensando na diminuição de custo e tempo, é importante que as análises dos dados obtidos venham sendo feitas no decorrer das avaliações, de modo que, ao se alcançar resultados significativos, o teste realizado termine. (BARBEDO, 2004).

Antes de iniciarem as avaliações, os participantes devem estar familiarizados com a escala de pontuação e sobre como usá-la, com o método de avaliação, com os equipamentos utilizados e com o ambiente onde serão feitos os testes. Devem entender, também, o que está sendo estudado e seu objetivo. Se

possível, nessa etapa que antecede os testes formais, os participantes devem ser agrupados para que possam discutir sobre degradações que venham a detectar. Esta interação entre eles pode ajudar a especialistas menos experientes na hora da avaliação. (BARBEDO, 2004).

Antes de serem iniciadas as avaliações formais, cada participante terá uma apresentação instruindo sobre o funcionamento das avaliações, de preferência acompanhada de documento escrito, podendo, ainda, serem feitas comparações de ilustração. Já que a memória auricular de longa e média duração não é confiável, o teste deve explorar somente a memória de curta duração. Por isso, recomenda-se utilizar métodos de chaveamento de estímulos quase instantâneos, pois estímulos instantâneos podem causar distorções. O mais indicado é a utilização de chaveamento com tempo de 40ms[51]. É importante que as avaliações não tenham mais que 20 ou 30 minutos de duração e que os julgamentos não passem de 15 por seção. Testes longos podem ser muito cansativos, interferindo na confiabilidade dos resultados. (BARBEDO, 2004).

Em testes nos quais as sequências apresentam uma qualidade muito aproximada, pode ser necessário fazer testes especiais com “âncoras”, com a ideia clara de avaliar a capacidade dos participantes. Para isso, estas “âncoras” devem ser criadas para que apenas ouvintes especializados possam detectar degradações, sendo imperceptíveis para ouvintes comuns. As sequências de áudio não devem ser agradáveis demais nem muito desagradáveis, evitando que os avaliadores percam a concentração nos testes. O nível dos sinais gravados deverá ter o valor de -18dBfs[52] e a duração dos áudios, em torno de 10 a 25 segundos. (BARBEDO, 2004).

Em testes de avaliação subjetiva de áudio sempre é utilizado um par de sequências para comparação. Esta forma de avaliação difere das avaliações subjetivas de qualidade de voz, nas quais se costumam usar testes do tipo absoluto. Isso ocorre porque degradações de sinais de áudio processados são muito mais difíceis de perceber e a comparações entre duas sequências melhora muito a sensibilidade dos participantes a qualquer degradação existente. (BARBEDO, 2004).

---

[51] 40 milissegundos
[52] dBfs é a sigla de *Decibels Relative to Full Scale* ou decibéis em relação à escala

Para os testes de qualidade subjetiva de áudio, podem ser utilizados dois tipos de escalas, pois o objetivo geral dos testes é que irá definir qual das escalas é mais apropriada e, portanto, deverá ser usada. Para testes de qualidade de som ou perda, é interessante utilizar a escala de cinco graus, ilustrada na tabela 8:

Tabela 8 - Escala de cinco graus

Qualidade		**Descrição**	
5	Excelente	5	Imperceptível
4	Bom	4	Perceptível, mas não incômodo
3	Fraco	3	Ligeiramente incômodo
2	Ruim	2	Incômodo
1	Péssimo	1	Muito incômodo

Fonte: ITU

Em testes comparativos, é possível tanto utilizar a escala de cinco graus quanto a escala de comparação de sete graus. Ainda assim, é necessário saber o objetivo geral dos testes praticados, pois as duas escalas não são equivalentes e não trazem os mesmos resultados. A tabela 9 traz a escala de comparação de sete graus:

Tabela 9 - Escala de comparação

Comparação	
3	Muito melhor
2	Melhor
1	Ligeiramente melhor
0	Igual
–1	Ligeiramente pior
–2	Pior
–3	Muito pior

Fonte: ITU

Escolheu-se, para a realização dos testes de qualidade subjetiva de áudio, utilizar a escala de sete graus de comparação, pois aplica-se melhor aos objetivos deste trabalho. Como estão sendo comparados recursos semelhantes, de duas tecnologias distintas, usar uma escala de comparação torna-se a escolha óbvia. A partir daqui, serão descritos todos os detalhes dos testes de qualidade subjetiva de áudio realizados na presente pesquisa.

3.2.3 Medições objetivas

A realização de medições baseia-se no ato de obter informações sobre alguns dos atributos de uma entidade, sendo que esta entidade pode ser um objeto, como uma casa, ou um evento, como um projeto de *software*. Essas entidades possuem propriedades chamadas atributos, como, por exemplo, a altura de um prédio ou o tempo de carregamento de um programa. Quando se faz alguma medição, não se medem as entidades, mas, sim, seus atributos; logo, para efetuar medições, é necessário que se usem as nomenclaturas corretas e que sejam indicados quais atributos serão quantificados. (ROSA, 2009).

Por meio dos atributos, é possível determinar características e descrever uma entidade, sendo que os mesmos podem ser definidos por números, imagens ou símbolos. Sem análises sobre os atributos de uma entidade, tomar alguma decisão sobre esta se torna inviável. (ROSA, 2009).

Em informática, um *benchmark* é a execução de um ou mais programas, com o intuito de avaliar o desempenho de uma entidade. É comumente usado para medições e comparações, e está mais associado a avaliações de características de *hardware*[53] como, por exemplo, desempenho da *Central Processing Unit*[54] (CPU), podendo, também, ser aplicado a *software*. (ROSA, 2009).

Em resumo, a utilização de *benchmark* para avaliação de *software* é a execução de um ou mais aplicativos nas mesmas condições, visando encontrar resultados a partir do desempenho destes aplicativos. *Benchmarks* possuem testes comparativos entre sistemas, dispositivos físicos etc., sendo muito úteis para compreender como um sistema se comporta nas mais variadas condições. (ROSA, 2009).

Desempenho também é um aspecto de qualidade. Uma forma de se obter o desempenho de um *software* é analisar o consumo de memória e processador que um aplicativo consome em execução. (ROSA, 2009). Para as medições objetivas, pretende-se abordar testes de desempenho. Por meio de comparação das medições do consumo de memória e processador, serão obtidos dados que mostrem quais das duas tecnologias utiliza melhor os recursos de *hardware*.

---

[53] Parte física do computador, como peças e equipamentos
[54] Central única de processamento

4 EXECUÇÃO E RESULTADOS DAS COMPARAÇÕES

A partir deste ponto, serão descritos os testes subjetivos de qualidade de vídeo e áudio, bem como os testes objetivos de consumo de processamento e memória e respectivos resultados.

A versão do *Flash Player* utilizada para os testes é a 11.9.900.117.

Os navegadores usados são o *Google Chrome 30*, o *Mozilla Firefox 25* e o *Internet Explorer* 10.

Todas as análises e comparações foram realizadas em um *notebook* da marca *Evolute*, cujas configurações podem ser observadas na tabela 10:

Tabela 10 – Configuração do Computador usado

Computador
Sistema Operacional *Windows* 7 *Pro* – 64 *bits*
Processador *Intel Core* i3-M370 2.40GHz[55]
Memória RAM[56] 4GB[57] DDR3-1333
Placa de vídeo Integrada *Intel* HD *Graphics* 256 Mb[58]
Monitor LCD[59] *Widescreen*[60] 14'' resolução 1366 x 768
Placa de Áudio *Intel Ibex Peak* PCH - *High Definition Audio Controller*[61]

Fonte: Autor

4.1 TESTES SUBJETIVOS DE VÍDEO

Para os testes subjetivos de vídeo, foi utilizado o *trailer* do jogo *Assassin's Creed IV Black Flag*.

O vídeo foi dividido em 12 sequências, as quais continham tanto vídeos simples, com pouca movimentação e poucas alterações de cenas, quanto vídeos complexos, com muito movimento.

A figura 9, a seguir, ilustra algumas destas sequências:

[55] 1 *gigahertz* é a medida equivalente a um bilhão de ciclos por segundo
[56] *Random Access Memory* ou memória de acesso aleatório
[57] 1 *gigabit* é a medida equivalente a 1024 *megabits* (Mb)
[58] 1 *megabit* equivale a 1024 *kilobits* (Kb), sendo que cada Kb equivale a 1024 *bytes*
[59] *Liquid Crystal Display* ou tela *de* cristal líquido
[60] Tela cujo formato é mais largo, semelhante a telas de cinema
[61] Controle de Áudio de Alta Definição

Figura 9 - Sequências do vídeo do jogo utilizado nos testes

Fonte: Autor

Para a codificação das sequências, foram utilizados dois *softwares*: o *Sorenson Squeeze*, versão *Trial*, e o *Free Studio.* O *Sorenson Squeeze* foi escolhido por ser a única ferramenta encontrada que converte arquivos de áudio para o formato FLV com codec *Sorenson Spark* além de ser uma ferramenta muito robusta, o *Free Studio* foi escolhido por ser gratuito, de fácil utilização e por complementar as deficiências do *Sorenson Squeeze.*

Todos os vídeos foram convertidos para os formatos/*codecs* utilizados nos testes, sendo *MP4/h.2464*, que é suportado pelas duas tecnologias, *ON2 VP6* e *Sorenson Spark*, que são os *codecs* suportados pelo Flash, e *Wave* e *OGV*, que são formatos suportados pelo HTML5.

Por meio dos programas de conversão, puderam ser configurados detalhes das sequências para que fatores como a Taxa de Transferência de *Bits* (*bit rate)* e *Frames* por Segundo (FPS) fossem iguais entre os vídeos comparados e não influenciassem na diferença da qualidade final. Para os testes de qualidade subjetiva de vídeo, utilizou-se o método de comparação SCACJ, com escala de qualidade de sete níveis, por adequar-se perfeitamente aos objetivos deste trabalho.

Os vídeos foram configurados da seguinte forma.

a) *MP4/H.264* – esse formato/*codec* de vídeo, por ser o único formato suportado pelas duas tecnologias, teve duas configurações: a primeira, de melhor qualidade, teve resolução de 640 x 360, FPS de 24 *frames*, *bit rate* de 2000kbps[62], com uma variação máxima de até 2400kbps. A segunda configuração, de qualidade média, teve resolução 640 x 360, FPS de 24 *frames* e *bit rate* de 900kbps;

b) *Flv/On2 vp6 e Webm/vp8* – esses dois formatos/*codec* tiveram resolução de 640 x 360, FPS de 24 *frames* e *bit rate* de 900kbps;

c) *Flv/Sorenson Spark e Ogv/Theora* – foram configurados com resolução de 640 x 360, FPS de 24 *frames* e *bit rate* de 600kbps.

Os testes foram projetados para os navegadores *Google Chrome*, *Mozilla FireFox* e *Internet Explorer*. No início da seção de testes foi realizada uma introdução, explicando os objetivos dos mesmos e como seriam feitas as avaliações, sendo que participante acompanhava a explicação por meio de um documento explicativo que lhe tinha sido entregue. Após a introdução, efetuou-se um pré-teste, visando familiarizar o participante com a melhor e a pior qualidade das sequências apresentadas. Acerca da visão dos participantes, todos apresentavam uma visão considerada normal ou normalizada com a utilização de óculos ou lentes.

Com relação aos testes executados no navegador *Google Chrome,* não foi possível avaliar a tecnologia *HTML5* para vídeos no formato *MP4.* Mesmo sendo especificado pela Google que seu navegador oferece suporte nativo ao formato, tentativas de executar vídeos com o formato citado no navegador *Chrome* mostraram-se inviáveis ou impossíveis, pois a maioria dos computadores com o referido navegador apresentavam imagens distorcidas, enquanto alguns nem mesmo reconheciam os vídeos.

Como o objetivo era comparar os recursos das tecnologias *Flash* e *HTML5*, as comparações foram feitas de modo que os vídeos tivessem uma qualidade equiparada, sem favorecer nenhuma das tecnologias. Como o único formato/*codec* reconhecido pelas duas tecnologias é o *Mp4/h.264*, para os testes com outros padrões de vídeo foram escolhidos formatos/codecs com qualidade mais aproximada, codificando-os nas mesmas condições.

As comparações foram efetuadas da seguinte forma: os vídeos em

[62] *Kilobits* por segundo

Mp4/h.264 foram comparados nas duas tecnologias, utilizando-se duas configurações, uma de alta e outra de média qualidade.

Os vídeos *Webm/vp8*, do *HTML5*, foram comparados com os vídeos *Flv/vp6,* do *Flash,* e os vídeos *Ogv/theora,* do *HTML5,* comparados com os vídeos *Flv/spark*, do *Flash*.

A tabela 11 apresenta a legenda atribuída à pontuação de vídeo:

Tabela 11 - Pontuação de vídeo

Comparação	
3	Muito melhor
2	Melhor
1	Ligeiramente melhor
0	Igual

Fonte: Autor

Em relação aos resultados obtidos nos teste de vídeo, as tabelas mostram os formatos/*codecs* que foram comparados entre si nos diferentes navegadores.

Ressalta-se que os participantes dos testes foram numerados de 1 a 15 e cada um atribuiu uma nota aos formatos apresentados.

O número abaixo do participante representa o quanto este considerou determinado formato/*codec* melhor, de acordo com a escala ilustrada na tabela anterior.

As tabelas 12 e 13 apresentam os resultados alcançados nos testes de vídeo efetuados no navegador *Google Chrome*:

Tabela 12 - Comparação de vídeo 1

Participantes / Formato/*codec*	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
WebM/vp8			0	0		0	0	1	2	0	2	1	1	1	0	8 pts
Flv/vp6	1	1	0	0	1	0	0			0						3 pts

Fonte: Autor

Tabela 13 - Comparação de vídeo 2

Participantes / Formato/*codec*	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total

Ogv/theora					0	0	0						0			0 pts
Flv/spark	1	2	2	1	0	0	0	1	1	2	1	3	0	1	2	17 pts

Fonte: Autor

As tabelas 14, 15, 16 e 17, a seguir, mostram os resultados obtidos com o navegador *Mozzilla FireFox:*

Tabela 14 - Comparação de vídeo 3

Participantes / Formato/codec	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
Html5/h.264 qualidade boa		0	2	0		0	0	1	0	0	1	0	0	0	0	4 pts
Flash/h.264 qualidade boa	1	0		0	1	0	0		0	0		0	0	0	0	2 pts

Fonte: Autor

Tabela 15 - Comparação de vídeo 4

Participantes / Formato/codec	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
Html5/h.264 qualidade media			0	0		0		0		1	0	0	0	0	0	1 pts
Flash/h.264 qualidade media	1	1	0	0	2	0	1	0	1							6 pts

Fonte: Autor

Tabela 16 - Comparação de vídeo 5

Participantes / Formato/*codec*	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
Webm/vp8	1		0	1	0	0	0	2	1	0	0	2	1	0	0	8 pts
Flash/vp6		1	0		0	0	0			0	0			0	0	1 pts

Fonte: Autor

Tabela 17 - Comparação de vídeo 6

Participantes / Formato/*codec*	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
Ogv/theora					0	0					0				1	1
Flash/spark	2	1	3	2	0	0	2	1	2	1	0	1	2	1		18 pts

Fonte: Autor

Nas tabelas 18 e 19, podem ser observados os resultados dos testes de vídeo realizados no navegador *Internet Explorer*:

Tabela 18 - Comparação de vídeo 7

Participantes / Formato/*codec*	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
Html5/h.264 qualidade boa		0	0	0	0	0		1	1	0	0	0	0	0		2 pts
Flash/h.264 qualidade boa	1	0	0	0	0	0	1			0	0	0	0	0	1	3 pts

Fonte: Autor

Tabela 19 - Comparação de vídeo 8

Participantes / Formato/*codec*	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
Html5/h.264 qualidade media	0		1	0		0		0	0	0	0	2	0	0	0	3 pts
Flash/h.264 qualidade media	0	2		0	2	0	2	0	0	0	0		0	0	0	4 pts

Fonte: Autor

4.2 TESTES SUBJETIVOS DE ÁUDIO

Nos testes de qualidade subjetiva de áudio foram usadas duas sequências de música: a primeira, chamada Só Rezo*,* e a segunda, Espero a Minha Vez. Ambas foram compostas por Di Ferreira e Gee Rocha, nomes artísticos, e são tocadas pela banda NX Zero. A música Só Rezo foi dividida em 12 sequências, enquanto a música Espero a Minha Vez, em 13 sequências, todas com 20 segundos de duração.

Para satisfazer a condição de participantes especialistas para os testes subjetivos de áudio, foi-se no estúdio de música Maesttro Estúdio de Ensaios, localizado na cidade de Criciúma no endereço, rua Engº Fiúza da Rocha, 320, para a busca do material humano qualificado. Utilizou-se o programa *Wondershare Video Converter Ultimate* [63] para codificar as sequências de áudio nos formatos e configurações necessários à realização dos testes subjetivos de áudio. Utilizou-se essa ferramenta por ser de fácil manuseio, intuitiva e principalmente por possibilitar a conversão para todos os formatos de áudio utilizados nos testes.

Os formatos utilizados foram *Wave*, *Aiff* e *Mp3*, suportados pela tecnologia *Flash*, e *Wave, Ogg* e *Mp3*, suportados pela tecnologia HTML5.

É importante ressaltar que, apesar de o HTML5 possibilitar o uso dos três formatos citados, os navegadores precisam oferecer suporte nativo aos

[63] Programa que converte arquivos de áudio e vídeo a diferentes formatos

mesmos. No momento, apenas dois dos navegadores estudados, o *Google Chrome* e o *Mozilla FireFox*, têm suporte nativo para todos os formatos, enquanto o *Internet Explorer* possui suporte nativo somente para o formato *Mp3,* impossibilitando a utilização dos outros formatos com as *tags* de áudio do HTML5.

Para que todos os áudios estivessem nas mesmas condições, as sequências comparadas foram configuradas com qualidade idêntica de *bit rate* e *taxa de amostragem*, dois fatores que têm muita influência na qualidade final dos áudios, permitindo restringir as comparações a influências apenas das duas tecnologias em estudo.

Desta forma, os áudios foram assim configurados:

a) nos formatos *Wave* e *Aiff*, que trabalham sem perdas de dados e, portanto, sem perda de qualidade, foram utilizados *bit rates* de 256kbps, taxa de amostragem de 44100Hz[64] e 2 canais de áudio;

b) os formatos *MP3* e *Ogg,* que utilizam perda de dados para diminuir o tamanho do arquivo, prejudicando a qualidade do áudio, tiveram um *bit rate* de 128kbps, taxa de amostragem de 44100Hz e 2 canais de áudio.

Assim como nos testes de vídeo, os testes de áudio foram efetuados nos navegadores *Google Chrome*, *Internet Explore* e *Mozilla FireFox*.

Junto aos participantes, foi feita uma introdução apresentando os objetivos e o modo de avaliação. Os testes foram realizados apenas com participantes especialistas, os quais, antes de iniciarem as seções, foram submetidos a um pré-teste para verificar a percepção auditiva. Para os testes, foi utilizada a escala de qualidade de sete níveis, já descrita no capítulo anterior.

A configuração do Fone de Ouvido utilizado para os testes de qualidade subjetiva de áudio pode ser observada na tabela 20:

Tabela 20 - Equipamento de testes qualidade de áudio

Fone de Ouvido	
Marca	Sennheiser 151238. Preto HD 202
Cabo	3 metros
Conector	3,5 mm (P2) estéreo
Frequência	18Hz – 18kHz[65]

[64] *Hertz* ou unidade de frequência por segundo
[65] 1 *quilohertz* equivale a mil *hertz*

Impedância	32 ohms[66]
Sensibilidade	115dB

Fonte: Autor

Para poder testar os áudios sem favorecer nenhuma das tecnologias, foi preciso levar em conta que os formatos de áudio suportados pelo HTML5 e pelo *Flash* dividem-se em dois tipos básicos: o *lossless*, compressão sem perda de dados, representado pelos formatos *Wave* e *Aiff*, e o *lossy*, compressão com perda de dados, representado pelos formatos *Mp3* e *Ogg*. Vale ressaltar que, para esta pesquisa específica, não era interessante comparar áudios *lossless* com áudios *lossy*, pois, como o foco concentra-se nas tecnologias em estudo, tal comparação poderia trazer dados influenciados pelo tipo de compressão, tornando a pesquisa inconclusiva.

Os formatos comparados foram escolhidos de forma que não cruzassem tipos *lossless* com *lossy,* ficando assim organizados: o formato *Wave*, do HTML5, com o *Wave*, do *Flash*, o formato *Wave*, do HTML5, com o *Aiff*, do *Flash*, o formato *Mp3*, do HTML5, com o *Mp3*, do *Flash*, e o formato *Ogg*, do HTML5, com o *Mp3*, do *Flash*.

A legenda de pontuação para comparação de áudio está expressa na tabela 21:

Tabela 21 - Comparação de áudio

Comparação	
3	Muito melhor
2	Melhor
1	Ligeiramente melhor
0	Igual

Fonte: Autor

Na sequência, as tabelas mostram os resultados obtidos nos testes subjetivos de áudio nos diferentes navegadores. Do mesmo modo que as tabelas de resultados dos testes subjetivos de vídeos, os testes de áudio foram realizados com 15 participantes, nesse caso, todos especialistas, número suficiente para obter resultados significativos.

As tabelas 22, 23, 24 e 25 apresentam a pontuação alcançada pelos formatos de áudio no navegador *Google Chrome*:

[66] Unidade de medida de resistência elétrica

Tabela 22 - Comparação de áudio 1

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/wave	0	0	0	0		2			2	2		1	0	1	1	9 pts
Flash/wave	0	0	0	0	2		1	3			1		0			7 pts

Fonte: Autor

Tabela 23 - Comparação de áudio 2

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/wave		0	0	1	0	1		1	2	0	2	0			1	8 pts
Flash/aiff	1	0	0		0		1			0		0	1	2		5 pts

Fonte: Autor

Tabela 24 - Comparação de áudio 3

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/Mp3	2	1		1	1	0	1			0		1	1			8 pts
Flash/Mp3			1			0		3	1	0	1			3	1	10 pts

Fonte: Autor

Tabela 25 - Comparação de áudio 4

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/Ogg		2	1	0	1	0	2	0		2	1		2		0	11 pts
Flash/mp3	1			0		0			2			1		2	0	6 pts

Fonte: Autor

As tabelas 26, 27, 28 e 29 demonstram os resultados alcançados no navegador *Mozilla FireFox*:

Tabela 26 - Comparação de áudio 5

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/wave		2	0	3	0	3	2		0			0			0	10 pts
Flash/wave	1		0		0			1	0	2	1	0	1	2	0	8 pts

Fonte: Autor

Tabela 27 - Comparação de áudio 6

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/wave	0	0	0	1				2	0		2	0		1		6 pts
Flash/aiff					1	0	2			3			3		1	10 pts

Fonte: Autor

Tabela 28 - Comparação de áudio 7

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/Mp3	2	1		0	2	0	2	0	2	3	1	1	2	0	0	16 pts
Flash/Mp3			2	0		0		0						0	0	2 pts

Fonte: Autor

Tabela 29 - Comparação de áudio 8

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/Ogg		0	0	2	0	0	2	2			1	0				7 pts
Flash/mp3	1	0	0		0	0			3	2		0	1	2	1	10 pts

Fonte: Autor

A tabela 30 traz a pontuação alcançada pelo formato de áudio suportado pelo navegador *Internet Explorer*:

Tabela 30 - Comparação de áudio 9

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/Mp3	1	2			0			0	0	2		1	1		0	7 pts
Flash/Mp3			2	1	0	1	2	0	0		1			2	0	9 pts

Fonte: Autor

4.3 TESTES OBJETIVOS DE CONSUMO DE CPU E MEMÓRIA

Nesta seção são apresentados os resultados dos testes objetivos de consumo de processador e memória das duas tecnologias.

Os testes foram realizados por meio dos conceitos de *benchmark*. Assim, utilizando o mesmo ambiente, foram alcançados os resultados para o consumo das duas tecnologias em estudo, HTML5 e *Flash*. Os resultados encontrados nesta seção irão complementar os testes subjetivos de áudio e vídeo mostrados anteriormente e ajudarão desenvolvedores e profissionais de tecnologias da informação (TI) quanto à utilização da tecnologia que melhor se adapte aos seus projetos.

Nos testes, foram utilizados uma sequência de áudio e outra de vídeo, avaliadas 3 vezes cada uma, no mesmo ambiente e nos três navegadores estudados, *Google Chrome*, *Mozilla FireFox* e *Internet Explorer*.

A configuração do vídeo e áudio utilizada foi a seguinte:

a) vídeo – para os navegadores *Internet Explorer* e *Mozilla FireFox*, foi utilizado um vídeo no formato *Mp4*, *codec H.264,* com resolução de 640 x 360, FPS de 30 *frames* e *bit rate* de 1500kbps, enquanto para o navegador *Google Chrome* foi utilizado um vídeo no formato *Webm*, *codec Theora*, FPS de 30 *frames* e *bit rate* de 1500kbps;

b) áudio – para todos os navegadores foi utilizada uma sequência no formato *Mp3*, com *bit rate* de 128kbps, taxa de amostragem de 44100Hz e 2 canais de áudio.

Os resultados foram obtidos com a ajuda do programa *Process Explorer*, criado por Mark Russinovich e, atualmente, propriedade da *Microsoft*. Esse aplicativo atende perfeitamente às necessidades dos testes objetivos, pois mostra, de maneira clara, todos os processos executados e o quanto cada um está consumindo de memória e processador em tempo de execução.

Em relação ao navegador *Google Chrome*, os resultados encontrados estão demonstrados nas tabelas 31 e 32:

Tabela 31 - Consumo de processador e memória 1

HTML5						
Recursos	Reprodução de Áudio consumo			Reprodução de Vídeo consumo		
Processador (%)	1.96	1.90	1.98	4.73	4.49	4.91
Memória (KB)	40.752	40.992	37.340	55.690	56.936	56.644

Fonte: Autor

Tabela 32 - Consumo de processador e memória 2

Flash						
Recursos	Reprodução de Áudio consumo			Reprodução de Vídeo consumo		
Processador (%)	1.91	1.88	1.97	6.73	7.01	7.11
Memória (KB)	72.720	73.068	72.816	54.850	60.144	59.763

Fonte: Autor

Em relação ao navegador *Mozilla FireFox*, as tabelas 33 e 34 apresentam os resultados obtidos:

Tabela 33 - Consumo de processador e memória 3

HTML5						
Recursos	Reprodução de Áudio consumo			Reprodução de Vídeo consumo		
Processador (%)	1.77	1.55	1.60	4.12	3.91	4.36
Memória (KB)	113.620	115.904	115.652	145.052	151.763	142.144

Fonte: Autor

Tabela 34 - Consumo de processador e memória 4

Flash						
Recursos	Reprodução de Áudio consumo			Reprodução de Vídeo consumo		
Processador (%)	2.65	2.58	2.75	7.72	7.96	8.07
Memória (KB)	68.740	68.568	68.708	55.556	68.064	69.036

Fonte: Autor

Quanto aos resultados alcançados utilizando o navegador *Internet Explorer*, estes estão expressos nas tabelas 35 e 36:

Tabela 35 - Consumo de processador e memória 5

HTML5						
Recursos	Reprodução de Áudio consumo			Reprodução de Vídeo consumo		
Processador (%)	1.08	1.32	1.30	3.96	4.01	3.98
Memória (KB)	54.672	60.044	58.776	82.896	93.936	94.768

Fonte: Autor

Tabela 36 - Consumo de processador e memória 6

Flash						
Recursos	Reprodução de Áudio consumo			Reprodução de Vídeo consumo		

Processador (%)	2.53	2.65	2.58	8.01	7.97	8.17
Memória (KB)	108.696	117.704	117.052	106.496	105.168	103.208

Fonte: Autor

O navegador *Mozilla FireFox* não separa a execução dos vídeos e áudios reproduzidos pelo HTML5 em processos únicos, sendo que o consumo de memória e processador das sequências, neste navegador, mistura-se com consumo do próprio navegador. Deste modo, quando se analisam os resultados obtidos, é importante ter em mente que o consumo da tecnologia HTML5, no navegador *FireFox*, não é decorrente apenas da execução das sequências, mas representa uma soma de toda a execução do navegador, o qual, somente por estar aberto, já consome, em média, 110.000Kb de memória. No que diz respeito ao *Flash*, esse consumo é distinto e possível de ser diferenciado, como demonstram os resultados dos testes realizados.

4.4 CONSOLIDAÇÃO DOS RESULTADOS

Essa seção descreverá, unicamente, os resultados encontrados de forma objetiva, sem maiores explicações, as quais serão apresentadas posteriormente, na conclusão do trabalho.

A descrição a seguir será feita em forma de números, sendo que os pontos representam a qualidade do objeto: quanto mais pontos, melhor sua qualidade.

Nos resultados de vídeo do navegador *Google Chrome*, o *Webm/vp8* recebeu 8 pontos e o *Flv/vp6* recebeu 3 pontos, diferença de 5 pontos a mais para o *Webm/vp.* O *Ogv/theora* recebeu 0 ponto, enquanto o *Flv/spark* recebeu 17 pontos, uma grande diferença de 17 pontos a mais para o *Flv/spark.*

No *FireFox*, o formato/*codec Mp4/h.264*, com boa qualidade, reproduzido com o HTML5, recebeu 4 pontos, ao passo que, reproduzido com *Flash*, o mesmo recebeu 2 pontos, resultando em 2 pontos a mais para a reprodução em HTML5. O *Mp4/h.264*, com média qualidade, reproduzido em HTML5 recebeu 1 ponto; já a reprodução com *Flash* recebeu 6 pontos, o qual levou uma vantagem de 5 pontos. O *Webm/vp8* obteve 8 pontos e o *Flv/vp6* obteve 1 ponto, sensível diferença de 7 pontos a mais para o *Webm/vp8.* Para finalizar, no navegador *FireFox*, o *Ogv/theora* obteve 1

ponto, enquanto o *Flv/spark* obteve 18 pontos, novamente uma diferença considerável de 17 pontos a mais para o *Flv/spark*, assim como ocorreu no *Google Chrome*.

O *Internet Explorer* apresentou os seguintes resultados: o formato/*codec Mp4/h.264*, com boa qualidade, reproduzido no HTML5, recebeu 2 pontos e, quando reproduzido no *Flash*, recebeu 3 pontos, uma diferença mínima de 1 ponto a mais quando reproduzido com *Flash*. O mesmo *Mp4/h.264,* mas agora em média qualidade, obteve 3 pontos com o HTML5 e 4 pontos com o Flash, novamente uma diferença de 1 ponto a mais quando reproduzido com *Flash.*

Nos formatos de áudio no *Google Chrome*, o *Wave*, reproduzido com HTML5, teve 9 pontos, e, com Flash, 7 pontos, 2 pontos a mais para o *Wave/HTML5*. No teste dois, *HTML5/wave* contra *Flash/aiff*, o primeiro teve 8 pontos e o segundo, 5 pontos, 3 pontos a mais para o *HTML5/wave.* No teste do *Mp3,* quando tocado com HTML5, obteve 8 pontos, enquanto com o *Flash*, 10 pontos, 2 a mais para o primeiro. Por fim, na comparação *HTML5/ogv* contra *Flash/mp3*, o primeiro recebeu 11 pontos e o segundo, 6 pontos, resultando em 5 pontos a mais para o primeiro.

Em relação aos resultados de áudio no *Mozilla FireFox*, o *Wave* recebeu 10 pontos com HTML5 e 8 pontos com *Flash*, mínima diferença de 2 pontos a mais para a reprodução com HTML5. Na comparação *HTML5/wave* contra *Flash/aiff*, o primeiro obteve 6 pontos e o segundo, 10 pontos, 4 a mais para o *Flash/aiff*. Nos testes com *Mp*3, o formato recebeu 16 pontos com HTML5 e 2 pontos com o *Flash*, uma diferença considerável de 14 pontos para o HTML5. Na última comparação realizada no navegador *FireFox*, o *Ogg*, do HTML5, obteve 7 pontos; já o *Mp*3, do *Flash*, 10 pontos, resultando em 3 a mais para o *Mp*3.

Como o navegador *Internet Explorer* reconhece, nativamente, apenas o formato *Mp*3, os testes nele efetuados envolveram somente este formato, o qual, no HTML5, recebeu 7 pontos, e, no *Flash*, 9 pontos, mínimos 2 pontos a mais para a segunda tecnologia.

Acerca dos resultados dos testes de consumo de processador e memória, estes foram iniciados com os testes de reprodução de áudio no *Chrome*: o HTML5 usou cerca de 1.94% do processador e em torno de 39Mb de memória, ao passo que o *Flash*, 1.92% de processador e 72Mb de memória, um consumo equivalente de processador e uns 45% a mais de memória consumida pelo *Flash*. As reproduções de vídeo no *Chrome* tiveram um consumo, por parte do HTML5, de

4.71% de processador e 56Mb de memória, enquanto o *Flash* teve consumo de 6.96% de processador e mais ou menos 58Mb de memória, resultando em uso equivalente de memória e diferença em torno de 2% a mais utilizados pelo *Flash* em relação ao processador.

No *FireFox*, os resultados de reprodução de áudio foram de 1.64% de consumo de processador e de 115Mb de memória com HTML5, e 2.66% de consumo de processador e 68Mb de memória com o *Flash*, 1% a mais de consumo de processador pelo *Flash* e 40% a mais de memória usados pelo HTML5. Os testes de consumo de *hardware* no *FireFox*, com reprodução de vídeo, trouxeram como resultado 4.13% de uso de processador e 146Mb de uso memória com HTML5, e 7.91% de uso de processador e 64Mb de memória com o *Flash,* consumo de mais de 3% de processador pelo *Flash* e mais de 56% de memória pelo HTML5.

Finalizando a seção, os resultados obtidos no Internet Explorer demonstram que a reprodução de áudio resultou em 1.23% de uso de processador e por volta de 57Mb de memória pelo HTML5, e 2.58% utilizados de processador e em torno de 114Mb de memória usados pelo *Flash*, diferença de consumo de mais de 1% de processador e 50% de memória pelo *Flash.* O consumo por sequências de vídeo, no *Internet Explorer*, foi de 3.98% de uso de processador e aproximadamente 90Mb de consumo de memória pelo HTML5, contra 8.05% utilizados de processador e 104Mb consumidos de memória pelo *Flash*, o que resulta em um consumo, a maior, de 4% de processador e 13% de memória pelo *Flash*.

5 CONCLUSÃO

Este trabalho teve, como objetivo principal, avaliar os recursos disponíveis no HTML5 e no *Flash,* realizando testes comparativos, aplicando diversos problemas nas mesmas condições para as duas tecnologias e oportunizando, assim, que os desenvolvedores escolham com maior propriedade qual recurso poderá ser usado para implementação de aplicações *web*.

No decorrer da pesquisa, percebeu-se que essas duas tecnologias, com capacidades tão amplas, somente poderiam ser comparadas em um contexto específico. Tal consideração deve-se a dois fatores: primeiro, pelo fato de, mesmo as duas tecnologias tendo vários recursos com funções semelhantes e podendo ser usadas com os mesmo objetivos, elas não têm, em um contexto mais abrangente, o mesmo propósito como solução final; segundo, por ambas possuírem tantos recursos, uma comparação que não trabalhasse com fronteiras bem definidas acabaria levando a resultados não confiáveis.

Analisando-se os novos recursos oferecidos pela versão atual do HTML, o HTML5, e a utilização do *Flash* na *web*, optou-se por concentrar os estudos comparativos nos recursos multimídia de áudio e vídeo das duas tecnologias. Esta concentração tornou-se interessante pelo valor prático dos resultados encontrados, pois o *Flash* destaca-se como a principal tecnologia para a reprodução de áudio e vídeo na *web*, estando presente em cerca de 98% do computadores existentes. Todavia, com o surgimento do HTML5, o *Flash* pode começar a perder mercado e, entre os principais motivos para isso, estão as novas *tags* de áudio e vídeo existentes no HTML5, as quais, além de serem muito simples, não necessitam de *plugin*, já que são nativas nos navegadores.

Desenvolver a fundamentação teórica e o estudo comparativo foi importante para a compreensão da evolução da informática e para o aprimoramento do conhecimento acerca das tecnologias estudadas, pois permitiu compreender a dimensão que poderia tomar a pesquisa e a necessidade de especificar o contexto estudado.

No transcurso da pesquisa, tornou-se fácil perceber que testes subjetivos são simples, embora muito dispendiosos, já que são inúmeras as recomendações da ITU a serem respeitadas, no sentido de alcançar resultados confiáveis. Mesmo

assim, a maior dificuldade é conseguir participantes e realizar os testes com os mesmos, exatamente nas condições recomendadas. Os testes de áudio são ainda mais complexos, pois exigem participantes especialistas, isto é, que possuam conhecimento aprofundado sobre técnicas e avaliação de qualidade nas sequências.

Nos testes de vídeo, o formato/*codec Mp*4/h.264, utilizado nas duas tecnologias, não apresentou grandes diferenças de qualidade, apesar de que, no navegador *Firefox*, o formato/*codec*, em média qualidade, tenha mostrado uma diferença de cinco pontos. Com os resultados muito próximos aos encontrados no *Internet Explorer*, e com a proximidade nos resultados encontrados no próprio navegador *FireFox* com o formato/*codec* em boa qualidade, é possível classificar as duas tecnologias como equivalentes.

Nos testes comparativos utilizando-se os formatos/*codecs Webm*/*vp8* e *Flv/vp8*, percebeu-se que o HTML5, com o *Webm/vp8*, foi melhor que o *Flash* com *Flv/vp6*. Além disso, foram muito interessantes os resultados encontrados nos testes efetuados com *Ogv/theora* e *Flv/spark*, pois, inesperadamente, o formato/*codec,* utilizado pelo *Flash*, mostrou-se muito superior ao utilizado pelo HTML5. A grande diferença obtida chamou a atenção porque o *Ogv/theora* é uma tecnologia muito mais atual e, portanto, deveria apresentar técnicas de compressão mais eficazes que o *Flv/spark*.

Os testes de qualidade de áudio, assim como os testes de vídeo, não mostraram diferenças significativas de qualidade, sendo que a maior parte dos resultados obtidos não favoreceu a nenhuma das tecnologias estudadas. Apesar da proximidade de qualidade de áudio das tecnologias em estudo, observou-se, no navegador *Mozilla FireFox*, uma grande superioridade dos áudios reproduzidos no formatos *Mp3* pela tecnologia HTML5, superioridade essa que não ocorre nos demais navegadores. Por meio deste resultado, recomenda-se que desenvolvedores deem preferência ao HTML5 quando criarem *players* de áudio *Mp3* para tocar no navegador *FireFox.*

Nos resultados obtidos com os testes de consumo de memória e processador, foi fácil perceber que os recursos de *hardware* são melhor utilizados pelo HTML5, principalmente o processador.

No *Google Chrome*, a execução de arquivos de áudio teve uma utilização equivalente do processador pelas duas tecnologias, mas percebeu-se uma grande

diferença quanto ao uso da memória, sendo que o *Flash* teve um consumo de cerca de 42% a mais que o HTML5. Na execução de arquivos de vídeo, enquanto o uso dos recursos de memória foi equivalente, o consumo de processador usado pelo *Flash* foi 2% maior que o utilizado pelo HTML5.

O *Mozilla FireFox* apresentou dados peculiares em relação à memória: o navegador, diferentemente dos seus concorrentes, não se divide em processos, utilizando a mesma memória para todas as funções do navegador, inclusive a execução de arquivos de áudio e vídeo HTML5, o que aumenta, de modo considerável, o valor dos resultados da quantidade de memória usada pelo HTML5 no navegador, ainda mais tendo em vista que o mesmo, somente pelo fato de estar aberto, já consumia em torno de 110MB de memória. Também em relação ao *FireFox*, a utilização de processador é muito melhor pelo HTML5, o qual tem um consumo em torno de 2% menor em reproduções de áudio, e mais de 3% menor em reproduções de vídeo.

Os resultados mais claros quanto ao melhor desempenho do HTML5 pertencem ao navegador *Internet Explorer*, o qual apresentou diferenças de até 4% no uso de processador e cerca de 50% menos no uso de memória.

Acerca dos testes realizados, perceberam-se outros detalhes importantes, além dos resultados propriamente ditos.

Uma questão importante do HTML5 é o suporte dos *codecs* por ele utilizados na reprodução de áudio e vídeo. O fato é que não existe um consenso dos fabricantes de navegadores para o suporte dos formatos usado pela tecnologia, tampouco um formato padrão que seja recomendado pela *W3C*. As discussões sobre o tema ainda são intensas e não parecem estar perto de chegar ao fim. Aliás, tal questão deve ser a principal causa que impede o HTML5 de fortalecer-se na *web* e ser aceita como solução definitiva para muitos projetos, pois é inegável que, neste quesito, o *Flash* é uma opção melhor. A ideia de uma solução nativa e não proprietária pode ser muito interessante e atrativa, mas antes ela precisa ser realmente uma solução, e não mais um problema.

No geral, o HTML5 mostrou-se uma tecnologia melhor que o *Flash* para aplicações que precisam preocupar-se com os recursos de *hardware,* recursos esses muito importantes para aparelhos como *tablets* e celulares. Quanto à qualidade de reprodução áudio e vídeo, as duas tecnologias equipararam-se,

revelando não ser um problema a escolha por qualquer uma delas. Se forem pensadas em aplicações para computadores *desktop*, o *Flash* parece mais preparado, já que não tem necessidade de os navegadores suportarem quaisquer formatos e está presente em boa parte dos computadores existentes.

Como sugestão para trabalhos futuros e como complemento a esta pesquisa, poderão ser analisados, com maior profundidade, os recursos de *canvas* existentes nas duas tecnologias, pois o *Flash* possui um palco com recursos avançados para desenvolvimento de animações e o HTML5 trouxe a nova *tag canvas* que, por meio de código *javascript*, permite a criação de imagens e animações.

REFERÊNCIAS

ADOBE. ***Flash professional***: adicionar vídeo ao *flash*. 2011. Disponível em: <http://help.adobe.com/pt_BR/flash/cs/using/WSb03e830bd6f770ee-70a39d612436d472f4-7ff8.html>. Acesso em: 08 out. 2012.

_____. ***Flash professional***: linha do tempo. Disponível em: <http://help.adobe.com/pt_BR/flash/cs/using/WSd60f23110762d6b883b18f10cb1fe1af6-7f84a.html>. Acesso em: 10 out. 2012.

_____. ***Flash professional***: uso do painel palco e ferramentas. Disponível em:<http://help.adobe.com/pt_BR/flash/cs/using/WSd60f23110762d6b883b18f10cb1fe1af6-7fb8a.html>. Acesso em: 10 out. 2012.

_____. **Programação do *Adobe* ® *Actionscript*® 3.0**. Disponível em: <http://help.adobe.com/pt_BR/ActionScript/3.0_ProgrammingAS3/flash_as3_programming.pdf>. Acesso em: 15 out. 2012.

_____. **Utilização do *adobe flash professional* CS5 e CS5.5**. 2011. Disponível em: <http://help.adobe.com/pt_BR/flash/cs/using/flash_cs5_help.pdf>. Acesso em: 06 out. 2012.

_____. **Utilização do *adobe flash professional* CS5**. Disponível em: <http://www.apostilando.com/pagina.php?cod=1>. Acesso em: 04 out. 2012.

ARAYA, Erm; VIDOTTI, Sabg. **Criação, proteção e uso legal de informação em ambientes da *World Wide Web***. São Paulo: UNESP; São Paulo: Cultura Acadêmica, 2010. Disponível em: <http://xa.yimg.com/kq/groups/15419196/2033306924/name/Criacao>. Acesso em: 22 ago. 2013.

BARBEDO, Jayme Garcia Arnal. **Avaliação objetiva de qualidade de sinais de áudio e voz**. 2004. 163 f. Tese (Doutorado em Engenharia Elétrica e da Computação) – Faculdade de Engenharia Elétrica e da Computação, Campinas/SP, 2004. Disponível em: <http://cutter.unicamp.br/document/?code=vtls000321395>. Acesso em: 15 set. 2013.

BATISTA, Diogo T. C. **O impacto do HTML5 no desenvolvimento para a internet.** 2009. Disponível em: <www.diogocezar.com/files/html5/artigo_html5.pdf>. Acesso em: 10 maio. 2012.

CARVALHO, Alan. **Criando na internet com HTML 4 para *trainees***. Rio de Janeiro: *Book Express*, 2001.

CARVALHO, Flávia Pereira. **Introdução ao adobe flash CS4**. Disponível em: <https://fit.faccat.br/~fpereira/aula_flash/slides_aula_flash_6porfolha.pdf>. Acesso em: 05 out. 2012.

CRIAR WEB. **Introdução a *canvas* do HTML5**. 2010. Disponível em: <http://www.criarweb.com/artigos/introducao-canvas-html5.html>. Acesso em: 06 out. 2012.

ELEMAR. **HTML5**: suporte para *offline*. 2010. Disponível em: <http://elemarjr.net/2010/10/20/html-5-parte-7-suporte-para-offline/>. Acesso em: 17 out. 2012.

FLATSCHART, Fábio. **O fim da era dos *plugins***. Disponível em: <http://imasters.com.br/artigo/25329/tendencias/o-fim-da-era-dos-plugins>. Acesso em: 23 out. 2012.

FURLAN, Marcos Paulo. **Flash CS5.** Disponível em: <http://www.apostilando.com/pagina.php?cod=1>. Acesso em: 04 out. 2012.

GONÇALVES, Márcio. **Ferramenta para avaliação subjetiva da qualidade de vídeo**. 2012. 66 f. Dissertação (Mestrado em Engenharia de Telecomunicações e Informática) – Instituto Universitário de Lisboa, 2012. Disponível em: <https://code.google.com/p/subjective-video-quality-assessment-tool/downloads/detail?name=Relat%C3%B3rio%20M%C3%A1rcioGon%C3%A7alves%20METI_25525.pdf&can=2&q=>. Acesso em: 21 set. 2013.

HEY, D. Fuverki; **Estudo de viabilidade do HTML5 para desenvolvimento web**: Maringá. 2010. HTML e XHTML. Disponível em: <http://www.connection.ufma.br/Apostila_html_e_xhtml.pdf>. Acesso em: 15 out. 2012.

LALLI, Micaroni Felipe; BUENO, Franco Felippe; ZACHARIAS, Keese Guilherme. **Evolução da programação *web***. 2008. Disponível em: <http://battisti.etc.br/unialfa/2011/pdi/evolucao-da-programacao-web.pdf>. Acesso em: 25 maio. 2012.

LOBO FILHO, Roberto Jorge Haddock. **Integração entre HTML5 e JSF 2.0 em aplicações web off-line**. 2010. Disponível em: <https://projetos.inf.ufsc.br/arquivos_projetos/projeto_809/TCC+Artigo.pdf>. Acesso em: 27 maio. 2012.

LOPES, Carla Teixeira. **Introdução ao flash**. Disponível em: <http://www.carlalopes.com/pubs/tutorialFlash.pdf>. Acesso em: 13 out. 2013.

MANSFIELD-DEVINE, Steve. ***Hybrid applications****: divide and conquer: the threats posed by hybrid apps and* HTML5. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1353485810700337>. Acesso em: 16 abr. 2012.

MICROSOFT DEVELOPER NETWORK. **Adicionando um controle de vídeo HTML5 à sua página da *web*.** Disponível em: <http://msdn.microsoft.com/pt-br/library/hh924820(v=vs.85).aspx>. Acesso em: 03 out. 2012.

_____. **Como usar o HTML5 para reproduzir arquivos de vídeo em sua página da *web***. Disponível em: <http://msdn.microsoft.com/pt-br/library/hh924821(v=vs.85).aspx>. Acesso em: 02 out. 2012.

_____. **Introdução ao elemento de áudio do HTML5**. Disponível em: <http://msdn.microsoft.com/pt-br/library/ie/gg589529(v=vs.85).aspx>. Acesso em: 27 set. 2012.

ROSA, Rafael Ledoux. ***Benchmark* de linguagens compiladas, semi-compiladas e interpretadas**. 2009. 104 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade do Estado de Santa Catarina, Joinville, 2009. Disponível em: <http://www.pergamum.udesc.br/dados-bu/000000/00000000000D/00000D77.pdf>. Acesso em: 07 out. 20xx.

SHERIDAN, Malcolm. **Guia do elemento *canvas* do HTML5 para desenvolvedores**. Disponível em: <http://centralhtml5.sourceforge.net/Guia-do-elemento-Canvas-do-HTML5-para-desenvolvedores>. Acesso em: 03 out. 2012.

SILVA, Maurício Samy. **HTML5**: a linguagem de marcação que revolucionou a *web*. São Paulo: Novatec, 2011.

W3C. ***Introduction to HTML 4.0***. Disponível em: <http://www.w3.org/TR/REC-html40-971218/intro/intro.html>. Acesso em: 22 out. 2012.

_____. **XHTML™ 1.1**: *module-based* XHTML. 2. ed. 2010. Disponível em: <http://www.w3.org/TR/xhtml11/>. Acesso em: 21 out. 2012.

WHATWG. ***The canvas element***. Disponível em: <http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html>. Acesso em: 27 out. 2012.

WILLIAM, David. **A história do HTML**. Disponível em: <http://www.frontendbrasil.com.br/artigos/a-historia-do-html/>. Acesso em: 19 out.2012.

WINCKLER, Marco. 2002. **Avaliação de usabilidade de *sites web***. Disponível em: <ww.funtec.org.ar/usabilidadsitiosweb.pdf>. Acesso em: 19 set. 2012.

APÊNDICE A - AVALIAÇÃO E COMPARAÇÃO DE RECURSOS UTILIZANDO TECNOLOGIAS HTML5 E FLASH PARA CRIAÇÃO DE APLICAÇÕES WEB

TIAGO DE CARVALHO

DEPARTAMENTO DE CIÊNCIAS DA COMPUTAÇÃO
UNIVERSIDADE DO EXTREMO SUL CATARINENSE (UNESC) – CRICIÚMA, SC - BRAZIL

Abstract. *This work presents a comparative study between the technologies HTML5 and Flash, being that the latter dominates the production of video, audio and animations on the market. You want to compare similar features of the two technologies ,therefore, tests will be conducted with these resources to help developers to decide which technology will better adapt to their needs for web applications.*

Resumo. *Este trabalho apresenta um estudo comparativo entre as tecnologias HTML5 e Flash, sendo que o último domina as produções de vídeo, áudio e animações no mercado. Pretende-se comparar recursos semelhantes das duas tecnologias ,sendo assim, serão feitos testes com esses recursos para ajudar aos desenvolvedores a decidir qual tecnologia irá adaptar-se melhor às suas necessidades para aplicações web. O estudo feito foi focado na qualidade de áudio e vídeo, aplicando-se os mesmos parâmetros de avaliação para as duas tecnologias.*

1. Intrudução

Com a crescente popularização da internet, a necessidade de novas funcionalidades aumentou. Como as páginas da época já não mais satisfaziam, com o tempo foram incorporadas às páginas novos recursos, como imagens, sons, vídeos, animações etc.

No decorrer dessa evolução, surge o *Flash*, sendo um *software*, primeiramente, de gráficos vetoriais, para criação de animações que também oferecia suporte para *bitmap* e vídeo. A partir de então, foram sendo adicionados mais e mais recursos ao *software*, tornando-o, assim, o aplicativo mais usado para a criação de *websites* interativos.

Mesmo usando *plug-ins* e sendo uma tecnologia proprietária, as qualidades do *Adobe Flash* são indiscutíveis. O HTML5 é uma forte tecnologia e veio como uma boa opção para criações interativas em substituição ao *Flash.* Sua premissa é melhorar a semântica das páginas e adicionar novos recursos à linguagem para responder melhor aos avanços tecnológicos.

2. Evolução Web

A meteórica evolução da *Internet*, que ficou mais evidente pela grande necessidade de transformar aplicações *desktops* em *web*, tornou impossível a espera do avanço do *eXtensible Hypertext Markup Language* (XHTML) ou do HTML. Com a necessidade de inserir novas funcionalidades, pouco encontradas pela precariedade das linguagens, foram agregados *plug-ins a*os navegadores, possibilitando aos desenvolvedores a utilização de vários novos recursos, como áudio, vídeo e gráficos. Surge, então, a era dos *plug-ins* ou das animações. (BATISTA, 2009).

Os *plug-ins* possibilitavam incorporar aos navegadores funcionalidades que os mesmos não possuíam nativamente. Dentre eles, o de maior sucesso foi o *Flash Player*, nascido da Macromedia, em 1997, e incorporado pela *Adobe*, em 2005, sendo que o mesmo revolucionou as distribuições de material multimídia na *web*. Rapidamente, tornou-se a principal tecnologia na criação de aplicações ricas, e para criação e distribuição de conteúdo que envolvesse jogos, áudio e vídeo, *streaming* e interfaces gráficas interativas. (FLATSCHART, 2012).

3. HTML5

As versões anteriores ao HTML5 não possuíam recursos que possibilitassem aplicações mais robustas, com controle multimídia, sendo deixada essa responsabilidade para outras tecnologias que funcionassem através de *plugins* nos navegadores. Porém, para facilitar a codificação dos desenvolvedores e a navegabilidade dos usuários, é interessante que os navegadores não necessitem de *plugins* para reproduzirem efeitos de mídia. (HEY, 2010).

Com esses e outros fatores em mente, o grupo *WHATWG* começa a criação do HTML5, tecnologia que engloba os elementos HTML, a linguagem *JavaScript* e os comandos CSS. A ideia do HTML5 é facilitar a navegação dos usuários e possibilitar páginas mais leves, dinâmicas e seguras, com recursos multimídia que não necessitem de *plug-ins*. A especificação permite uma indexação mais eficiente em mecanismos de busca, exibição de áudio e vídeo nativos nos próprios navegadores, execução de aplicações *web offline* e códigos mais legíveis. (HEY, 2010).

4. O Flash

O *Flash* foi criado com o intuito de que as animações de *web* fossem rapidamente carregadas e executadas. Por ser um programa de imagens vetoriais, gera arquivos que descarregam mais rápido que arquivos *bitmaps*, *Graphics Interchange Format* (GIFs) e *Joint Photographic Expert Group* (JPEGs). Seu formato *streaming* permite que vídeos comecem a executar na *web* enquanto ainda estão sendo carregados. Além disso, o *Flash* possibilita que se inclua interatividade de alto nível aos *sites*, sem a necessidade de códigos complexos. (LOPES, 2012).

O que permite ao *Flash* construir animações com pouco peso é o fato de trabalhar com gráficos vetoriais, sendo que existem dois tipos de imagens gráficas, os mapas de *bits* e as imagens vetoriais.

5. Execução e Resultados das Comparações

A partir deste ponto, serão descritos os testes subjetivos de qualidade de vídeo e áudio, bem como os testes objetivos de consumo de processamento e memória e respectivos resultados.

A versão do *Flash Player* utilizada para os testes é a 11.9.900.117.

Os navegadores usados são o *Google Chrome 30*, o *Mozilla Firefox 25* e o *Internet Explorer* 10.

Todas as análises e comparações foram realizadas em um *notebook* da marca *Evolute*, cujas configurações podem ser observadas na tabela 10:

Tabela 37 - Consumo de processamento

Computador
Sistema Operacional *Windows* 7 *Pro* – 64 *bits*
Processador *Intel Core* i3-M370 2.40GHz

Memória RAM 4GB DDR3-1333
Placa de vídeo Integrada *Intel* HD *Graphics* 256 Mb
Monitor LCD *Widescreen* 14'' resolução 1366 x 768
Placa de Áudio *Intel Ibex Peak* PCH - *High Definition Audio Controller*

Fonte: Autor

5.1 Testes subjetivos de vídeo

O vídeo foi dividido em 12 sequências, as quais continham tanto vídeos simples, com pouca movimentação e poucas alterações de cenas, quanto vídeos complexos, com muito movimento.

Todos os vídeos foram convertidos para os formatos/*codecs* utilizados nos testes, sendo *MP4/h.2464*, que é suportado pelas duas tecnologias, *ON2 VP6* e *Sorenson Spark*, que são os *codecs* suportados pelo Flash, e *Wave* e *OGV*, que são formatos suportados pelo HTML5.

Para os testes de qualidade subjetiva de vídeo, utilizou-se o método de comparação SCACJ, com escala de qualidade de sete níveis, por adequar-se perfeitamente aos objetivos deste trabalho.

Os vídeos foram configurados da seguinte forma.

d) *MP4/H.264* – esse formato/*codec* de vídeo, por ser o único formato suportado pelas duas tecnologias, teve duas configurações: a primeira, de melhor qualidade, teve resolução de 640 x 360, FPS de 24 *frames*, *bit rate* de 2000kbps, com uma variação máxima de até 2400kbps. A segunda configuração, de qualidade média, teve resolução 640 x 360, FPS de 24 *frames* e *bit rate* de 900kbps;

e) *Flv/On2 vp6 e Webm/vp8* – esses dois formatos/*codec* tiveram resolução de 640 x 360, FPS de 24 *frames* e *bit rate* de 900kbps;

f) *Flv/Sorenson Spark e Ogv/Theora* – foram configurados com resolução de 640 x 360, FPS de 24 *frames* e *bit rate* de 600kbps.

No início da seção de testes foi realizada uma introdução, explicando os objetivos dos mesmos e como seriam feitas as avaliações, sendo que participante acompanhava a explicação por meio de um documento explicativo que lhe tinha sido entregue. Após a introdução, efetuou-se um pré-teste, visando familiarizar o participante com a melhor e a pior qualidade das sequências apresentadas.

As comparações foram efetuadas da seguinte forma: os vídeos em Mp4/h.264 foram comparados nas duas tecnologias, utilizando-se duas configurações, uma de alta e outra de média qualidade.

Os vídeos *Webm/vp8*, do *HTML5*, foram comparados com os vídeos *Flv/vp6*, do *Flash*, e os vídeos *Ogv/theora*, do *HTML5*, comparados com os vídeos *Flv/spark*, do *Flash*.

A tabela 11 apresenta a legenda atribuída à pontuação de vídeo:

Tabela 38 - Pontuação de vídeo

Comparação	
3	Muito melhor
2	Melhor
1	Ligeiramente melhor
0	Igual

Fonte: autor

Em relação aos resultados obtidos nos teste de vídeo, as tabelas mostram os formatos/*codecs* que foram comparados entre si nos diferentes navegadores.

Ressalta-se que os participantes dos testes foram numerados de 1 a 15 e cada um atribuiu uma nota aos formatos apresentados.

O número abaixo do participante representa o quanto este considerou determinado formato/*codec* melhor, de acordo com a escala ilustrada na tabela anterior.

As tabelas 12 e 13 apresentam os resultados alcançados nos testes de vídeo efetuados no navegador *Google Chrome*:

Tabela 39 - Comparação de vídeo 1

Participantes / Formato/*codec*	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
WebM/vp8			0	0		0	0	1	2	0	2	1	1	1	0	8 pts
Flv/vp6	1	1	0	0	1	0	0			0						3 pts

Fonte: Autor

Tabela 40 - Comparação de vídeo 2

Participantes / Formato/*codec*	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
Ogv/theora					0	0	0						0			0 pts
Flv/spark	1	2	2	1	0	0	0	1	1	2	1	3	0	1	2	17 pts

Fonte: Autor

As tabelas 14, 15, 16 e 17, a seguir, mostram os resultados obtidos com o navegador *Mozzilla FireFox:*

Tabela 41 - Comparação de vídeo 3

Participantes / Formato/codec	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
Html5/h.264 qualidade boa		0	2	0		0	0	1	0	0	1	0	0	0	0	4 pts
Flash/h.264 qualidade boa	1	0		0	1	0	0		0	0		0	0	0	0	2 pts

Fonte: Autor

Tabela 42 - Comparação de vídeo 4

Participantes / Formato/codec	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
Html5/h.264 qualidade media			0	0		0		0		1	0	0	0	0	0	1 pts
Flash/h.264 qualidade media	1	1	0	0	2	0	1	0	1							6 pts

Fonte: Autor

Tabela 43 - Comparação de vídeo 5

Participantes / Formato/*codec*	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total

Webm/vp8	1		0	1	0	0	0	2	1	0	0	2	1	0	0	8 pts
Flash/vp6		1	0		0	0	0			0	0			0	0	1 pts

Fonte: Autor

Tabela 44 - Comparação de vídeo 6

Participantes / Formato/*codec*	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
Ogv/theora					0	0					0				1	1
Flash/spark	2	1	3	2	0	0	2	1	2	1	0	1	2	1		18 pts

Fonte: Autor

Nas tabelas 18 e 19, podem ser observados os resultados dos testes de vídeo realizados no navegador *Internet Explorer*:

Tabela 45 - Comparação de vídeo 7

Participantes / Formato/*codec*	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
Html5/h.264 qualidade boa		0	0	0	0	0		1	1	0	0	0	0	0		2 pts
Flash/h.264 qualidade boa	1	0	0	0	0	0	1			0	0	0	0	0	1	3 pts

Fonte: Autor

Tabela 46 - Comparação de vídeo 8

Participantes / Formato/*codec*	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
Html5/h.264 qualidade media	0		1	0		0		0	0	0	0	2	0	0	0	3 pts
Flash/h.264 qualidade media	0	2		0	2	0	2	0	0	0	0		0	0	0	4 pts

Fonte: Autor

5.2 Testes subjetivos de áudio

Para os testes de áudio foram utilizadas 25 sequencias com duração de 20 segundos cada.

Os formatos utilizados foram *Wave*, *Aiff* e *Mp3*, suportados pela tecnologia *Flash*, e *Wave*, *Ogg* e *Mp3*, suportados pela tecnologia HTML5.

Para que todos os áudios estivessem nas mesmas condições, as sequências comparadas foram configuradas com qualidade idêntica de *bit rate* e *taxa de amostragem*, dois fatores que têm muita influência na qualidade final dos áudios, permitindo restringir as comparações a influências apenas das duas tecnologias em estudo.

Desta forma, os áudios foram assim configurados:

a) nos formatos *Wave* e *Aiff*, que trabalham sem perdas de dados e, portanto, sem perda de qualidade, foram utilizados *bit rates* de 256kbps, taxa de amostragem de 44100Hz e 2 canais de áudio;

b) os formatos *MP3* e *Ogg*, que utilizam perda de dados para diminuir o tamanho do arquivo, prejudicando a qualidade do áudio, tiveram um *bit rate* de 128kbps, taxa de amostragem de 44100Hz e 2 canais de áudio.

Assim como nos testes de vídeo, os testes de áudio foram efetuados nos navegadores *Google Chrome*, *Internet Explore* e *Mozilla FireFox*.

Junto aos participantes, foi feita uma introdução apresentando os objetivos e o modo de avaliação. Os testes foram realizados apenas com participantes especialistas, os quais, antes de iniciarem as seções, foram submetidos a um pré-teste para verificar a percepção auditiva. Para os testes, foi utilizada a escala de qualidade de sete níveis, já descrita no capítulo anterior.

A configuração do Fone de Ouvido utilizado para os testes de qualidade subjetiva de áudio pode ser observada na tabela 20:

Tabela 47 - Equipamento de testes qualidade de áudio

Fone de Ouvido	
Marca	Sennheiser 151238. Preto HD 202
Cabo	3 metros
Conector	3,5 mm (P2) estéreo
Frequência	18Hz – 18kHz
Impedância	32 ohms
Sensibilidade	115dB

Fonte: Autor

Para poder testar os áudios sem favorecer nenhuma das tecnologias, foi preciso levar em conta que os formatos de áudio suportados pelo HTML5 e pelo *Flash* dividem-se em dois tipos básicos: o *lossless*, compressão sem perda de dados, representado pelos formatos *Wave* e *Aiff*, e o *lossy*, compressão com perda de dados, representado pelos formatos *Mp3* e *Ogg*. Vale ressaltar que, para esta pesquisa específica, não era interessante comparar áudios *lossless* com áudios *lossy*, pois, como o foco concentra-se nas tecnologias em estudo, tal comparação poderia trazer dados influenciados pelo tipo de compressão, tornando a pesquisa inconclusiva.

A legenda de pontuação para comparação de áudio está expressa na tabela 21:

Tabela 48 - Comparação de áudio

Comparação	
3	Muito melhor
2	Melhor
1	Ligeiramente melhor
0	Igual

Fonte: Autor

Na sequência, as tabelas mostram os resultados obtidos nos testes subjetivos de áudio nos diferentes navegadores. Do mesmo modo que as tabelas de resultados dos testes subjetivos de vídeos, os testes de áudio foram realizados com 15 participantes, nesse caso, todos especialistas, número suficiente para obter resultados significativos.

As tabelas 22, 23, 24 e 25 apresentam a pontuação alcançada pelos formatos de áudio no navegador *Google Chrome*:

Tabela 49 - Comparação de áudio 1

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total

HTML5/wave	0	0	0	0		2			2	2		1	0	1	1	9 pts
Flash/wave	0	0	0	0	2		1	3			1		0			7 pts

Fonte: Autor

Tabela 50 - Comparação de áudio 2

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/wave		0	0	1	0	1		1	2	0	2	0			1	8 pts
Flash/aiff	1	0	0		0		1			0		0	1	2		5 pts

Fonte: Autor

Tabela 51 - Comparação de áudio 3

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/Mp3	2	1		1	1	0	1			0		1	1			8 pts
Flash/Mp3			1			0		3	1	0	1			3	1	10 pts

Fonte: Autor

Tabela 52 - Comparação de áudio 4

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/Ogg		2	1	0	1	0	2	0		2	1		2		0	11 pts
Flash/mp3	1			0		0			2			1		2	0	6 pts

Fonte: Autor

As tabelas 26, 27, 28 e 29 demonstram os resultados alcançados no navegador *Mozilla FireFox*:

Tabela 53 - Comparação de áudio 5

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/wave		2	0	3	0	3	2		0			0			0	10 pts
Flash/wave	1		0		0			1	0	2	1	0	1	2	0	8 pts

Fonte: Autor

Tabela 54 - Comparação de áudio 6

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/wave	0	0	0	1				2	0		2	0		1		6 pts
Flash/aiff					1	0	2			3			3		1	10 pts

Fonte: Autor

Tabela 55 - Comparação de áudio 7

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/Mp3	2	1		0	2	0	2	0	2	3	1	1	2	0	0	16 pts
Flash/Mp3			2	0		0		0						0	0	2 pts

Fonte: Autor

Tabela 56 - Comparação de áudio 8

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/Ogg		0	0	2	0	0	2	2			1	0				7 pts
Flash/mp3	1	0	0		0	0			3	2		0	1	2	1	10 pts

Fonte: Autor

A tabela 30 traz a pontuação alcançada pelo formato de áudio suportado pelo navegador *Internet Explorer*:

Tabela 57 - Comparação de áudio 9

Participantes / Formato	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	Total
HTML5/Mp3	1	2			0			0	0	2		1	1		0	7 pts
Flash/Mp3			2	1	0	1	2	0	0		1			2	0	9 pts

6. Conclusão

Nos testes de vídeo, o formato/*codec Mp*4/h.264, utilizado nas duas tecnologias, não apresentou grandes diferenças de qualidade, apesar de que, no navegador *Firefox,* o formato/*codec*, em média qualidade, tenha mostrado uma diferença de cinco pontos. Com os resultados muito próximos aos encontrados no *Internet Explorer*, e com a proximidade nos resultados encontrados no próprio navegador *FireFox* com o formato/*codec* em boa qualidade, é possível classificar as duas tecnologias como equivalentes.

Nos testes comparativos utilizando-se os formatos/*codecs Webm*/*vp8* e *Flv/vp8*, percebeu-se que o HTML5, com o *Webm/vp8*, foi melhor que o *Flash* com *Flv/vp6*. Além disso, foram muito interessantes os resultados encontrados nos testes efetuados com *Ogv/theora* e *Flv/spark*, pois, inesperadamente, o formato/*codec,* utilizado pelo *Flash*, mostrou-se muito superior ao utilizado pelo HTML5. A grande diferença obtida chamou a atenção porque o *Ogv/theora* é uma tecnologia muito mais atual e, portanto, deveria apresentar técnicas de compressão mais eficazes que o *Flv/spark*.

Os testes de qualidade de áudio, assim como os testes de vídeo, não mostraram diferenças significativas de qualidade, sendo que a maior parte dos resultados obtidos não favoreceu a nenhuma das tecnologias estudadas. Apesar da proximidade de qualidade de áudio das tecnologias em estudo, observou-se, no navegador *Mozilla FireFox*, uma grande superioridade dos áudios reproduzidos no formatos *Mp3* pela tecnologia HTML5, superioridade essa que não ocorre nos demais navegadores. Por meio deste resultado, recomenda-se que desenvolvedores deem preferência ao HTML5 quando criarem *players* de áudio *Mp3* para tocar no navegador *FireFox*.

Nos resultados obtidos com os testes de consumo de memória e processador, foi fácil perceber que os recursos de *hardware* são melhor utilizados pelo HTML5, principalmente o processador.

Os resultados mais claros quanto ao melhor desempenho do HTML5 pertencem ao navegador *Internet Explorer*, o qual apresentou diferenças de até 4% no uso de processador e cerca de 50% menos no uso de memória.

No geral, o HTML5 mostrou-se uma tecnologia melhor que o *Flash* para aplicações que precisam preocupar-se com os recursos de *hardware,* recursos esses muito importantes para aparelhos como *tablets* e celulares. Quanto à qualidade de reprodução áudio e vídeo, as duas tecnologias equipararam-se, revelando não ser um problema a escolha por qualquer uma delas. Se forem pensadas em aplicações para computadores *desktop*, o *Flash* parece mais preparado, já que não tem necessidade de os navegadores suportarem quaisquer formatos e está presente em boa parte dos computadores existentes.

7. Referências

BATISTA, Diogo T. C. **O impacto do HTML5 no desenvolvimento para a internet.** 2009. Disponível em: <www.diogocezar.com/files/html5/artigo_html5.pdf>. Acesso em: 10 maio. 2012.

FLATSCHART, Fábio. **O fim da era dos *plugins***. Disponível em: <http://imasters.com.br/artigo/25329/tendencias/o-fim-da-era-dos-plugins>. Acesso em: 23 out. 2012.

HEY, D. Fuverki; **Estudo de viabilidade do HTML5 para desenvolvimento web**: Maringá. 2010. HTML e XHTML. Disponível em: <http://www.connection.ufma.br/Apostila_html_e_xhtml.pdf>. Acesso em: 15 out. 2012.

LOPES, Carla Teixeira. **Introdução ao flash**. Disponível em: <http://www.carlalopes.com/pubs/tutorialFlash.pdf>. Acesso em: 13 out. 2012.