Usando a tag vídeo no HTML5

Usando a tag video no HTML5

Leia em 7 minutos – Nando Vieira

Por muitos anos, Flash foi a alternativa mais viável para quem precisava adicionar vídeos e músicas nas páginas Web. Isso porque ele estava presente em quase todos os navegadores e, finalmente, tínhamos um formato que podia ser usado sem maiores problemas. Bastava adicionar uma chamada para um plugin e a coisa "simplesmente" funcionava.

```
<object width="426" height="240">
  <param name="movie" value="http://www.youtube.com/v/-TdbsZg6CC4"></param>
  <param name="allowFullScreen" value="true"></param>
  <param name="allowscriptaccess" value="always"></param>
  <embed src="http://www.youtube.com/v/-TdbsZg6CC4"
    type="application/x-shockwave-flash"
    allowscriptaccess="always"
    allowfullscreen="true"
    width="426"
    height="240">
  </embed>
</object>
```

Devido às inconsistências dos navegadores, era preciso adicionar tanto o elemento <object> quanto o elemento <embed>, duplicando muitos atributos. Isso sem falar que plugins como o Flash causam muitas instabilidades e, eventualmente, fecham o navegador.
Agora, com o HTML5, podemos finalmente servir conteúdo multimídia de forma confiável, através dos elementos <video> e <audio> e suas APIs de JavaScript, sem ter que criar exceções para os diferentes navegadores.
Aquele código necessário para que o Flash funcione pode ser substituído, em teoria, por um muito mais simples:

```
<video controls src="video.mp4" width="426" height="240"></video>
```

O ponto é que usar o atributo src não ajuda muito se você quer atingir um número maior de navegadores. Por isso, é muito mais comum vermos a tag <video> sendo usada com outro elemento chamado de <source>.
O elemento <source> pode ser usado múltiplas vezes e permite definir um formato de vídeo para cada navegador, fazendo com que você tenha um alcance maior.

```
<video width="426" height="240">
  <source src="video.mp4" type="video/mp4">
  <source src="video.webm" type="video/webm">
</video>
```

Uma coisa interessante da tag <video> é que você pode definir um **fallback** caso o vídeo não consiga ser exibido. Um dos motivos seria a falta de um formato adequado para aquele navegador em particular. Você exibir uma imagem, um link para download ou até mesmo o vídeo através do Flash.

Usando a tag vídeo no HTML5

```
<video width="426" height="240">
  <source src="video.mp4" type="video/mp4">
  <source src="video.webm" type="video/webm">

  <object width="426" height="240">
    <param name="movie" value="http://www.youtube.com/v/-TdbsZg6CC4"></param>
    <param name="allowFullScreen" value="true"></param>
    <param name="allowscriptaccess" value="always"></param>
    <embed src="http://www.youtube.com/v/-TdbsZg6CC4"
      type="application/x-shockwave-flash"
      allowscriptaccess="always"
      allowfullscreen="true"
      width="426"
      height="240">
    </embed>
  </object>
</video>
```

Existem diversos codecs possíveis e você precisará exportar pelo menos duas versões para ter um alcance maior. Veja uma tabela de quais formatos são aceitos em cada navegador sem a necessidade de ter que instalar plugins/codecs.

Navegador	Formatos aceitos
Internet Explorer	.mp4, .m4v
Safari	.mp4, .m4v
Google Chrome	.mp4, .m4v, .webm, .ogv
Firefox	.webm, .ogv
Opera	.webm, .ogv

Seguindo a tabela acima, você precisaria apenas de dois formatos para ter um alcance maior: .mp4 e .webm. Perceba que tirei o formato Ogg (.ogv) da jogada; a relação entre qualidade vídeo e tamanho do arquivo não é tão boa quanto o formato WebM (a própria Mozilla recomenda o uso de WebM).
À seguir você conhecerá um pouco mais sobre cada um dos codecs e aprenderá exportar arquivos usando o FFMPEG.

Conhecendo os codecs

Como era de se esperar, diferentes navegadores precisam de diferentes tipos de formatos. Isso é necessário porque existem questões relacionadas às patentes. Se o Firefox quisesse suportar o formato .mp4, por exemplo, precisaria elimitar a licença de software livre completamente, além de ter que rastrear cada cópia instalada do navegador.

Usando a tag vídeo no HTML5

MP4 H.264

O container MP4 utiliza o codec de vídeo H.264 e pode usar o MP3 ou AAC como codecs de áudio. Sua relação compressão–qualidade é muito superior aos outros formatos como WebM e Ogg.
Ele pode ser executado sem a necessidade de instalações adicionais no Internet Explorer, Safari e Chrome, mas não funciona no Chromium.
O formatos baseados no MPEG são cobertos por patentes e não podem ser distribuídos livremente, pois exigem o pagamento da licença de uso.
O mime type utilizado para o formato MP4 é video/mp4.

WebM

O formato WebM é baseado em uma versão mais restrita do container Matroska (.mkv). Ele usa o codec de vídeo VP8 e o codec de áudio Vorbis. Ele pode ser executado sem a necessidade de instalações adicionais no Firefox, Chrome e Opera.
O Google já investiu mais de US$250 milhões no WebM, o que dá para mostrar a importância do formato de vídeo para o Google.
O mime type utilizado para o formato WebM é video/webm.

Ogg Theora Vorbis

O container Ogg, que utiliza o **codec de vídeo Theora** e o **codec de áudio Vorbis**, pode ser executado sem a necessidade de instalações adicionais no Firefox, Chrome e Opera. Embora ele funcione neste navegadores, o formato mais recomendado é o WebM, pois ele oferece uma relação compressão–qualidade melhor, além de ser suportado em mais navegadores.
O mime type utilizado para o formato Ogg é video/ogg.

Convertendo vídeos com ffmpeg

Se você quer adicionar vídeos ao seu site, precisará exportar os formatos MP4 e WebM. Isso pode ser feito com o FFMPEG. Para instalá-lo no Mac, você pode utilizar o Homebrew.

```
brew install ffmpeg --with-fdk-aac --with-libvo-aacenc --with-libvorbis --with-libvpx --with-theora
```

Como uso Mac, o formato mais comum de exportação de vídeos dos programas de screencast é o .mov. Por isso, você verá exemplos de como exportar através deste formato. O vídeo que vou converter é o clipe Depois que todos vão embora, do Hateen e tem as seguintes especificações:

- 640x360
- Duração de 3m36
- MP4 H.264 + AAC
- Audio sampling rate de 44100Hz
- 24 FPS
- Video bitrate de 719,08kbps
- 2 canais
- 19,5MB

Usando a tag vídeo no HTML5

O objetivo é exportar este arquivo para o tamanho 426x240, com um tamanho razoável, sem perder muito a qualidade.

Gerando o arquivo .mp4

Você pode converter um .mov em .mp4 sem ter que reencodar o vídeo; isso porque é possível apenas alterar o formato do container, fazendo com que a qualidade e áudio originais sejam mantidos. Por este motivo, o processo é bastante rápido.

```
time ffmpeg -i video.mov -acodec copy -vcodec copy video.mp4
real  0m0.268s
user  0m0.088s
sys   0m0.081s
```

No nosso exemplo, temos que redimensionar o vídeo final, então o comando é um pouco diferente.

```
time ffmpeg -i video.mov -acodec copy -ac 2 -b:a 128k -ar 44100 -b:v 345k -s 426x240 video.mp4
real  0m29.206s
user  1m33.632s
sys   0m1.086s
```

Veja abaixo a descrição de cada parâmetro:

- -acodec: define o codec de áudio utilizado.
- -ac: define a quantidade de canais de áudio.
- -b:a: define o bitrate de áudio.
- -ar: define a frequência do sampling.
- -b:v: define o bitrate do vídeo.
- -s: define as dimensões do vídeo.

Como resultado, tivemos um arquivo com tamanho final de 11.6MB.

Gerando o arquivo .webm

Para converter um .mov em .webm é preciso reexportar o vídeo, tarefa que pode demorar bastante dependendo do seu tamanho.

```
time ffmpeg -i video.mov -acodec libvorbis -ac 2 -b:a 128k -ar 44100 -b:v 345k -s 426x240 video.webm
real  1m12.010s
user  1m16.878s
sys 0m0.560s
```

Como resultado, tivemos um arquivo com tamanho final de 12.2MB.

Você pode ver como ficou a codificação do vídeo no grid que mostra o arquivo original e as versões em MPEG e WebM.

Embora o grid tenha perdido um pouco de qualidade, dá para perceber que as versões WebM e MPEG são muito parecidas (eu até queria exportar uma versão lossless, mas 34s de vídeo ficou com um tamanho de 950MB).

Usando a tag vídeo no HTML5

Manipulando com JavaScript
Para criar um elemento vídeo, você pode usar a função document.createElement.

```
// create the <video> element
var video = document.createElement("video");

// append to the <body> element
document.body.appendChild(video);
```

Você pode ouvir diversos eventos. Você pode usar a variante on<event>, como onended, mas esse tipo de evento não funciona no Safari. Então, você precisa usar sempre a função addEventListener.

```
video.addEventListener("ended", function(){
  console.log("video has ended");
});
```

Para ver a lista completa de eventos disparados, acesse a documentação no site do W3C. Se você quiser debugar os eventos disparados, faça algo como isso:

```
<video autoplay
onloadstart="console.log(event.type, event)"
onprogress="console.log(event.type, event)"
onsuspend="console.log(event.type, event)"
onabort="console.log(event.type, event)"
onerror="console.log(event.type, event)"
onemptied="console.log(event.type, event)"
onstalled="console.log(event.type, event)"
onloadedmetadata="console.log(event.type, event)"
onloadeddata="console.log(event.type, event)"
oncanplay="console.log(event.type, event)"
oncanplaythrough="console.log(event.type, event)"
onplaying="console.log(event.type, event)"
onwaiting="console.log(event.type, event)"
onseeking="console.log(event.type, event)"
onseeked="console.log(event.type, event)"
onended="console.log(event.type, event)"
ondurationchange="console.log(event.type, event)"
ontimeupdate="console.log(event.type, event)"
onplay="console.log(event.type, event)"
onpause="console.log(event.type, event)">

<source src="video.mp4" type="video/mp4"></source>
</video>
```

Usando a tag vídeo no HTML5

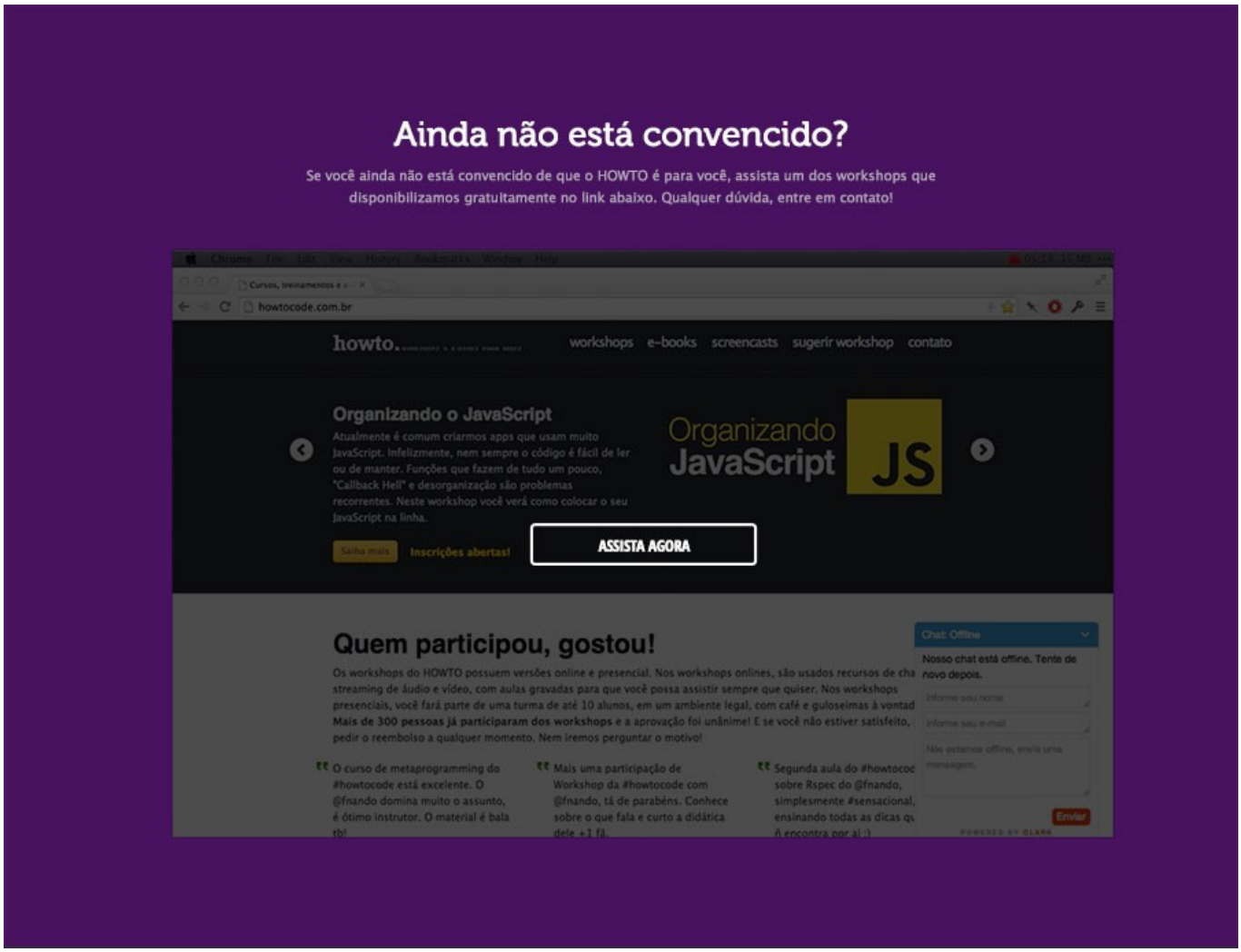
Criando uma interface personalizada

Cada navegador possui sua própria interface do player e ela não pode ser modificada.

Felizmente, como a API de JavaScript é bastante completa, você pode criar o seu tema com CSS e depois aplicar o comportamento com JavaScript. Uma boa alternativa é usar alguma biblioteca como video.js em vez de fazer tudo manualmente.

Exibindo o poster quando o video terminar

No workshop de Interfaces do HOWTO montamos um projeto que precisa da tag <video>. Em vez de usar a interface padrão do navegador, temos apenas um botão de play.



Esse botão deve ser escondido ao ser clicado, mas deve aparecer novamente quando o vídeo acabar. Além disso, ele deve voltar a exibir a imagem defina como poster.

Por incrível que pareça, não existe uma maneira tão simples para fazer isso funcionar, então vamos precisar de um pouco de JavaScript.

Usando a tag vídeo no HTML5

```
var video = document.querySelector("video");
var playButton = document.querySelectory(".play-button");

// disable the right-click context menu.
video.oncontextmenu = function(){ return false; };

// listen to the play event.
video.addEventListener("play", function(){
  playButton.classList.add("hidden");
  video.play();
});

// listen to the stop event.
video.addEventListener("ended", function(){
  playButton.classList.remove("hidden");
  video.src = video.currentSrc;
  video.style.backgroundImage = "url(" + video.poster + ")";
});
```

A ideia por trás desse hack consistem em definir a propriedade video.src com o caminho que foi usado para exibir o vídeo, disponível em video.currentSrc. Esse truque foi suficiente para fazer funcionar em todos os navegadores, com exceção do Safari.

É aí que entra a segunda parte do hack. Ao definir o video.src o vídeo fica transparente, por motivos que fogem à razão. Aí fica fácil! Agora é só adicionar uma imagem de fundo para o poster.

A morte dos plugins?

Muita gente acha que a <video> significou a morte de plugins como Flash e Silverlight, mas eles ainda tem seu uso. O principal deles é distribuir conteúdos protegidos (principalmente com Silverlight).
A tag <video> não permite o uso de DRM, mas existe uma proposta de adicionar seu suporte na própria especificação de HTML. O uso de DRM é um assunto polêmico, mas o fato é que produtoras de filmes não deixarão serviços como Netflix distribuir arquivos que podem ser copiados facilmente (não que não seja possível atualmente, mas exige muito mais trabalho). Então, acho que é melhor ter algo assim na própria especificação que ter que depender de um plugin de terceiro instalado.

Finalizando

Pode parecer que não é tão complexo, mas existe muita coisa em torno da tag <video> que precisa ser levada em consideração, como formatos de arquivos , tema do player e comportamento manipulado por JavaScript. Mas, de um modo geral, usar apenas a tag <video> já é uma realidade.

Fonte: https://nandovieira.com.br/usando-a-tag-video-no-html5