KAPITOLA 7

Vkládání audia a videa

Audio a video jsou v dnešním světě Internetu jeho velmi důležité součásti. Podcasty, zvukové ukázky a návodová videa jsou všude, a až dosud je bylo možné používat pouze díky zásuvným modulům prohlížečů. HTML5 představuje nové metody vkládání audio- a videosouborů do stránky. V této kapitole prozkoumáme několik metod, které můžeme použít nejenom k vložení audio či video obsahu, ale také k zajištění, že takový obsah bude dostupný také ve starších prohlížečích.

V této kapitole se podíváme na následující dva elementy:[29]

`<audio>` `[<audio src="bici.mp3"></audio>]`

Nativně přehrává v prohlížeči zvuk. *[Ch4, FF3.6, IE9, S3.2, O10.1, IOS3, A2]*

`<video>` `[<video src="navod.m4v"></video>]`

Nativně přehrává v prohlížeči video. *[Ch4, FF3.6, IE9, S3.2, O10.5, IOS3, A2]*

[29] V následujících popisech se podpora prohlížečů zobrazuje v hranatých závorkách pomocí zkratky a nejnižší podporující verze. Použité kódy jsou: Ch pro Google Chrome, FF pro Firefox, IE pro Internet Explorer, O pro Operu, S pro Safari a IOS pro zařízení iOS s mobilním Safari, A pro prohlížeč Androidu.

Ještě než se dostaneme k příkladům, popovídáme si stručně o historii audia a videa na Internetu. Jestliže chceme porozumět tomu, kam míříme, měli bychom také vědět, kde jsme byli dříve.

Stručná historie

Lidé se už dávno dříve snažili na webových stránkách používat audio a video. Začalo to tím, že lidé začali na domovskou stránku svého webu vkládat MIDI soubory a k odkazování na takové soubory používali element `embed`. A to nějak takto:

```
<embed src="soubor.mp3" autostart="true"
 loop="true" controller="true" ></embed>
```

Element `embed` se ale nikdy nestal standardem, takže lidé začali místo něj používat element `object`, který je součástí standardů W3C. K podpoře starších prohlížečů, které elementu `object` nerozuměli, se často používalo vnoření elementu `embed` do elementu `object`. Vypadalo to takto:

```
<object>
   <param name="src" value="simpsonovi.mp3">
   <param name="autoplay" value="false">
   <param name="controller" value="true">
   <embed src="soubor.mp3" autostart="false"
    loop="false" controller="true" ></embed>
</object>
```

Ne každý prohlížeč dokázal tímto způsobem obsah přehrávat a stahovat v jednu chvíli (streamovat), a ne každý server byl řádně konfigurován, aby vše správně obsloužil. Vše se ještě zkomplikovalo, že se na Internetu stalo video ještě populárnějším. Setkávat jste se mohli stále častěji s různými odměnami prezentace audio- a videoobsahu, od RealPlayeru přes Windows Media až po QuickTime. Každá společnost měla svou strategii vůči videu na Internetu a vypadalo to, že každý web používá jinou metodu a formát k dekódování jejich videoobsahu.

Společnost Macromedia (nyní Adobe) si zavčas uvědomila, že jejich Flash Player by mohl být skvělým nástrojem k dodávání audio- a videoobsahu napříč platformami. Flash je nyní nainstalovaný v téměř 97 % prohlížečů. Jakmile tvůrci tohoto obsahu zjistili, že mohou svůj obsah dekódovat a přehrát kdekoliv, tisíce webů se vrhlo na streamování pomocí technologie Flash, a to jak v případě audia, tak videa.

A poté v roce 2007 přišel Apple se svými zařízeními iPhone a iPod Touch a rozhodl se, že nebudou Flash podporovat. Poskytovatelé obsahu odpověděli tím, že videostreamy zpřístupnili tak, aby je bylo možné přehrát rovnou v prohlížeči Mobile Safari. Tato videa, používající kodek H.264, šlo také přehrát v normálním Flash Playeru, jenž poskytovatelům obsahu umožňoval kódovat jen jednou, aby byl obsah přístupný na více platformách.

Tvůrci specifikace HTML5 věří, že prohlížeč by měl podporovat audio a video nativně, než aby se musel spoléhat na smluvně ošetřené zásuvné moduly. A v tuto chvíli začalo audio a video poskytované pomocí HTML5 dávat smysl: aby se s videem a audiem zacházelo jako s prvořadým občanem v ohledu k webovému obsahu.

Otázky a odpovědi

Flash již funguje v různých prohlížečích, takže proč použít HTML5?

Jednoduchou odpovědí je fakt, že zde nejsou žádná omezení poskytovatele ohledně toho, co jako vývojář můžete s obsahem dělat po jeho vložení do stránky. K manipulování s elementem můžete použít CSS a JavaScript a nemusíte „kšeftovat" s parametry předávanými nahrávce Flashe. A navíc – tento stav se zlepší, jakmile bude standard šířeji přijímaný.

Kontejnery a kodeky

Pokud mluvíme o videu na webu, pak hovoříme o kontejnerech a kodecích. Myslet můžete na video natočené pomocí digitálního fotoaparátu ve formátu AVI nebo MPEG, ale to by bylo velké zjednodušení. Kontejnery jsou obalem obsahujícím proudy audia, videa a někdy i další metadata, jako jsou titulky. Tyto proudy audia a videa je třeba zakódovat, a to je chvíle, kdy na scénu vstupují kodeky. Video a audio lze zakódovat stovkami různých způsobů, ale když dojde řeč na HTML5, pak se budete bavit jen o několika z nich.

Kodeky videa

Když se díváte na video, váš videopřehrávač jej musí nejdříve dekódovat. Naneštěstí vámi používaný přehrávač nemusí být schopen dekódovat zrovna to video, které chcete přehrát. Některé přehrávače používají software k dekódování videa, který může být pomalý nebo může silně zatěžovat procesor. Jiné přehrávače používají hardwarové dekodéry a jsou tak limitovány pouze tím, co dokážou přehrát.

Dnes existují tři videoformáty, o nichž byste měli vědět, pokud chcete začít používat element HTML5 s názvem video ve vašich projektech: H.264, Theora a VP8.

Kodeky a podporované prohlížeče

- H.264 [IE9, S4, C3, IOS]
- Theora [FF3.5, Ch4, O10]
- VP8 [IE9 (pokud je kodek nainstalovaný), FF4, Ch5, O10.7]

H.264

Tento kodek určený pro videa s vysokou kvalitou obrazu byl standardizován v roce 2003 a vytvořila jej skupinou MPEG. Specifikace H.264 je rozdělena do několika profilů, aby mohla podporovat jak levná zařízení, jako jsou mobilní telefony, tak zároveň video pro zařízení s vysokým rozlišením. Tyto profily sdílejí sadu běžných funkcí, ale high-endové profily nabízejí další možnosti vylepšující kvalitu. Například iPhone i Flash Player mohou oba přehrávat videa zakódovaná pomocí kodeku H.264, ale iPhone podporuje pouze základní profil s nízkou kvalitou, zatímco Flash Player podporuje profil s vysokou kvalitou. Je také možné zakódovat video najednou a poté jej vložit do více profilů, takže vypadá hezky na více platformách.

H.264 je vlastně standardem, neboť jej podporuje jak Microsoft, tak Apple, kteří oba drží licenci. Navíc svá videa převádí na tento kodek také YouTube vlastněný Googlem, takže je lze přehrát na iPhonu i v zařízeních podporujících Adobe Flash Player. Nicméně nejde o otevřenou technologii. Je patentovaná a její využití je předmětem licenčních ujednání. Aby mohli video kódovat tímto kodekem, musí tvůrci obsahu za jeho použití platit licenční poplatek. Tyto poplatky se ale nevztahují na obsah, jenž je volně dostupný koncovým uživatelům.[30]

Zastánci volného softwaru jsou znepokojeni tím, že by vlastníci práv mohli eventuálně od tvůrců obsahu začít požadovat vysoké poplatky. Toto znepokojení vedlo k vytvoření a uvedení alternativních kodeků.

Theora

Theora je volně dostupný kodek vyvinutý Xiph.Org Foundation. Ačkoliv tvůrci obsahu mohou video vytvořit s podobnou kvalitou i v Theoře, výrobci zařízení jej zatím adoptují jen velmi pomalu. Firefox, Chrome a Opera umí přehrát videa kódovaná pomocí Theory na různých platformách bez jakéhokoli přídavného softwaru, ale zařízení s Internet Explorerem, Safari a iOS jej přehrát neumí.

[30] *http://www.reelseo.com/mpeg-la-announces-avc-h264-free-license-lifetime/*

Apple a Microsoft jsou ostražití ohledně „ponorkových patentů“, což je termín používaný k popisu patentů, v jejichž přihlášce je úmyslně zpožděno zveřejnění a vydání patentu, takže se skrývají do doby, než ostatní technologii implementují. Jakmile nastane ten správný čas, z přihlášky patentu se odhalí a začne vyžadovat licenční poplatky od nic netušícího trhu.

VP8

VP8 od Googlu je zcela volný kodek nevyžadující žádné poplatky s kvalitou podobnou kodeku H.264. Podporují jej Mozilla, Google Chrome a Opera, a Microsoft slibuje, že také Internet Explorer 9 bude kodek podporovat, pokud si jej uživatel sám nainstaluje. Také jej podporuje Adobe Flash Player, díky čemuž se stává zajímavou alternativou. Nepodporují jej Safari a zařízení s iOS, což znamená, že ačkoliv je kodek volně dostupný, tvůrci obsahu, jež chtějí svůj videoobsah doručovat na iPhony a iPady, stále musejí použít kodek H.264.

Kodeky pro audio

Jako by situace v případě kodeků pro video nebyla už dost zamotaná, i v případě audia se budeme potýkat se soupeřícími standardy.

Kodeky a podporované prohlížeče

- AAC [S4, Ch3, IOS]
- MP3 [IE9, S4, Ch3, IOS]
- Vorbis (OGG) [FF3, Ch4, O10]

Advanced Audio Coding (AAC)

Tento zvukový formát používá Apple ve svém projektu iTunes Store. Je navržen tak, aby měl vyšší kvalitu zvuku než MP3 za stejné velikosti souboru, a také nabízí více profilů audia podobně jako H.264. Podobně jako H.264 není volně dostupný a jsou s ním spojené licenční poplatky.

Všechny produkty společnosti Apple přehrávají soubory ve formátu AAC. Tak také činí Adobe Flash Player a open source přehrávač VLC.

Vorbis (OGG)

Tento open source formát bez poplatků podporuje Firefox, Opera a Chrome. Užitečný je v také kombinaci s videokodeky Theora a VP8. Soubory Vorbisu mají velmi dobrou zvukovou kvalitu, ale nejsou široce podporovány hardwarovými hudebními přehrávači.

MP3

Formát MP3, ačkoliv je extrémně běžný a populární, nepodporuje Firefox a Opera, protože se na něj také vztahují patenty. Podporují jej Safari a Google Chrome.

Kodeky videa a kodeky audia vyžadují, abyste je pro distribuci a přehrání sbalili. Promluvme si nyní o kontejnerech videa.

Kontejnery a kodeky společně

Kontejner je soubor s metadaty identifikující a prokládající audio- nebo video-soubory. Kontejner ve skutečnosti neobsahuje žádnou informaci o tom, jak jsou data kódovaná. V základu kontejner „obaluje" proud audia a videa. Kontejnery mohou často obsahovat jakoukoli kombinaci kódovaného videa, ale tyto kombinace uvidíme až při práci s videem na webu:

- Kontejner OGG obsahuje video kódované v Theoře a audio kódované ve Vorbisu a funguje ve Firefoxu, Chrome a Opeře.
- Kontejner MP4 obsahuje video kódované v H.264 a audio kódované pomocí Vorbisu a funguje v Safari a Chrome. Také jej lze přehrát pomocí Adobe Flash Playeru a na iPhonech, iPodech a iPadech.
- Kontejner WebM obsahuje video kódované pomocí VP8 a audio kódované ve Vorbisu a funguje ve Firefoxu, Chrome, Opeře a v Adobe Flash Playeru.

Vzhledem k tomu, že se Google a Mozilla přesouvají směrem k VP8 a WebM, můžeme z tohoto mixu vlastně klidně vynechat Theoru. Ale podle toho, jak to vypadá, budeme stále muset naše video kódovat dvakrát – jednou pro uživatele Applu (jejichž podíl v případě osobních počítačů je malý, ale neplatí to o podílu mezi mobilními zařízeními) a poté znovu pro uživatele Firefoxu a Opery, neboť oba tyto prohlížeče odmítají přehrát H.264.[31]

Je toho hodně. Ale nyní, když jsme pochopili historii a z ní vyplývající omezení, podívejme se blíže na současnou implementaci. Začneme s audiem.

[31] *http://lists.whatwg.org/pipermail/whatwg-whatwg.org/2009-June/020620.html*

15 Pracujeme s audiem

Společnost *AwesomeCo* vyvíjí web obsahující nezpoplatněné zvukové smyčky určené k použití ve videoukázkách, a rádi by viděli demostránku s modelovou kolekcí jednotlivých smyček. Výsledkem bude seznam zvukových smyček a návštěvník bude schopen si každou z nich rychle vyzkoušet. V rámci tohoto projektu se nebudeme zabývat hledáním smyček, protože zvukový inženýr klienta nám již poskytnul potřebné ukázky jak v MP3, tak ve formátu OGG. Několik informací o tom, jak kódovat vaše vlastní audiosoubory, najdete v příloze C.

Vytvoření základního seznamu

Zvukový inženýr nám poskytnul čtyři ukázky: bicí, varhany, basu a kytaru. Každou z těchto ukázek budete potřebovat popsat pomocí značek HTML. Zde je zdrojový kód pro smyčku bicích:

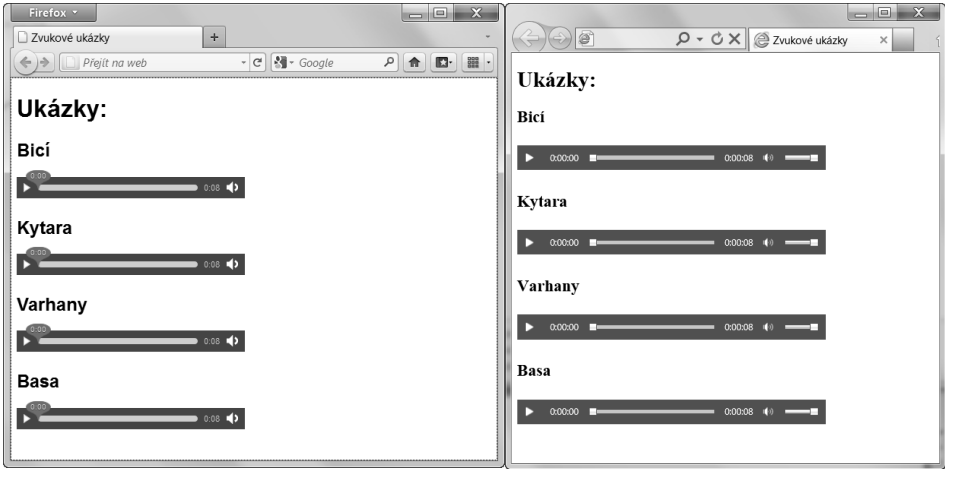
```
<article class="ukazka">
   <header><h2>Bicí</h2></header>
   <audio id="bici" controls>
      <source src="zvuky/ogg/bici.ogg" type="audio/ogg">
      <source src="zvuky/mp3/bici.mp3" type="audio/mpeg">
      <a href="zvuky/mp3/bici.mp3">Stáhnout bici.mp3</a>
   </audio>
</article>
```

Nejdříve definujeme element `audio` a řekneme mu, že chceme zobrazit určité ovládací prvky. Dále definujeme více zdrojů souboru. Nejdříve definujeme verze ukázky v MP3 a formátu OGG, a poté zobrazíme odkaz umožňující návštěvníkovi přímo si stáhnout soubor MP3, pokud by prohlížeč nepodporoval element `audio`.

Tento zcela zásadní kousek kódu bude fungovat v Chrome, Safari a Firefoxu. Vložme ho tedy dovnitř šablony HTML5 se třemi dalšími ukázkami zvuků.

```
<article class="ukazka">
   <header><h2>Bicí</h2></header>
   <audio id="bici" controls>
      <source src="zvuky/ogg/bici.ogg" type="audio/ogg">
      <source src="zvuky/mp3/bici.mp3" type="audio/mpeg">
      <a href="zvuky/mp3/bici.mp3">Stáhnout bici.mp3</a>
   </audio>
</article>
```

```
<article class="ukazka">
   <header><h2>Kytara</h2></header>
   <audio id="kytara" controls>
      <source src="zvuky/ogg/kytara.ogg" type="audio/ogg">
      <source src="zvuky/mp3/kytara.mp3" type="audio/mpeg">
      <a href="zvuky/mp3/kytara.mp3">Stáhnout kytara.mp3</a>
   </audio>
</article>
<article class="ukazka">
   <header><h2>Varhany</h2></header>
   <audio id="varhany" controls>
      <source src="zvuky/ogg/varhany.ogg" type="audio/ogg">
      <source src="zvuky/mp3/varhany.mp3" type="audio/mpeg">
      <a href="zvuky/mp3/varhany.mp3">Stáhnout varhany.mp3</a>
   </audio>
</article>
<article class="ukazka">
   <header><h2>Basa</h2></header>
   <audio id="basa" controls>
      <source src="zvuky/ogg/basa.ogg" type="audio/ogg">
      <source src="zvuky/mp3/basa.mp3" type="audio/mpeg">
      <a href="zvuky/mp3/basa.mp3">Stáhnout basa.mp3</a>
   </audio>
</article>
```



Obrázek 7.1: Naše stránka ve Firefoxu 4 (vlevo) a Internet Exploreru 9 (vpravo)

Když stránku otevřeme v prohlížeči, který je kompatibilní s HTML5, každá položka seznamu bude obsahovat svůj vlastní audiopřehrávač, jak můžete vidět na obrázku 7.1. Prohlížeč se sám postará o přehrání zvuku, jakmile stisknete tlačítko pro přehrání.

Pokud stránku otevřete ve starších verzích Internet Exploreru, zobrazí se odkaz ke stažení, neboť tento prohlížeč nerozumí elementu `audio`. To volá po nouzovém řešení, ale podívejme se nejdříve na to, zda bychom nemohli provést něco lepšího.

Nouzové řešení

Nouzové řešení podpory elementu `audio` obsahuje rovnou sám element. Definovali jsme více zdrojů našeho zvuku za pomoci elementu `source` a poskytli jsme odkazy ke stažení zvukového souboru. Pokud prohlížeč neumí vykreslit element `audio`, zobrazí alespoň odkaz, který jsme umístili do elementu `audio`. Můžeme jít ale o krok dále a po definování našich zdrojů použít jako nouzové řešení technologii Flash.

Nicméně to nemusí být ten nejlepší přístup. V takovém případě totiž očekáváte prohlížeč podporující element audio, ale nepodporující formáty, v nichž jste soubory poskytnuli. Například byste se mohli rozhodnout, že za váš čas nestojí poskytnout zvuk ve více formátech. Navíc specifikace HTML5 zvlášť zmiňuje, že nouzového řešení podpory elementu `audio` se nesmí používat k umístění obsahu, který by mohli přečíst čtečky obrazovky.

Nejjednodušším řešením je posunout odkaz na stažení mimo element `audio` a použít JavaScript k jeho skrytí. Například takto:

```
<article class="ukazka">
   <header><h2>Bicí</h2></header>
   <audio id="bici" controls>
      <source src="zvuky/ogg/bici.ogg" type="audio/ogg">
      <source src="zvuky/mp3/bici.mp3" type="audio/mpeg">
   </audio>
   <a href="zvuky/mp3/bici.mp3">Stáhnout bici.mp3</a>
</article>
```

A poté budeme potřebovat detekci podpory elementu `audio` a skrytí odkazů. To provedeme vytvořením nového elementu `audio` pomocí JavaScriptu a následným testováním, abychom viděli, zda odpovídá metodě `canPlayType()`. Kód by vypadal takto:

```
var prehrajeAudioSoubory = !!(document.createElement('audio' )
                                        .canPlayType);
```

Odezvu posoudíme a poté skryjeme jakékoli odkazy vnořené na částí našich ukázek.

```
$(function(){
   var prehrajeAudioSoubory = !!(document.createElement('audio' )
                                          .canPlayType);
   if(prehrajeAudioSoubory){
      $(".ukazka a" ).hide();
   };
});
```

Nouzová řešení elementu `audio` jsou relativně jednoduchá a někteří z vašich uživatelů mohou ve skutečnosti ocenit, že mají možnost jednoduše soubor stáhnout.

Nativní přehrání zvuku v prohlížeči je ale pouze začátek. Prohlížeče teprve začínají podporovat javascriptové API pro HTML5 určené audiu a videu, o čemž si můžete přečíst v poznámce na straně 139.

16 Vkládání videa

Společnost *AwesomeCo* chce na svém webu předvést výuková videa a chce, aby se videa přehrávala na tolika zařízeních, kolik je jen možné, zvláště pak na iPadu. Pro použití v ukázce jsme obdrželi dvě videa ze série „Triky ve Photoshopu", které využijeme k vytvoření prototypu. Naštěstí jsme dostali videosoubory kódované ve formátech H.264, Theora a VP8, takže se můžeme zaměřit pouze na tvorbu stránky.[32]

Element `video` funguje úplně stejně jako element `audio`. Prostě poskytneme naše zdroje a Chrome, Firefox, Safari, iPhone, iPad a Internet Explorer 9 zobrazí video, aniž by potřebovaly jakékoli další zásuvné moduly. Zdrojový kód našeho prvního videosouboru, `01_blur` vypadá takto:

```
<article>
   <header><h2>Saturace pomocí nástroje Rozmazat</h2></header>
   <video controls>
      <source src="video/h264/01_blur.mp4">
      <source src="video/theora/01_blur.ogv">
      <source src="video/webm/01_blur.webm">
      <p>Váš prohlížeč nepodporuje element video.</p>
   </video>
</article>
```

Element `video` definujeme včetně zobrazení ovládacích prvků (atribut `controls`). Zároveň sdělujeme, že video by se nemělo přehrát automaticky, protože jsme elementu `video` nepřidělili atribut `autoplay`. V této chvíli se naše video přehraje v množství prohlížečů a naši uživatelé uvidí videopřehrávač podobný tomu, který je zobrazený na obrázku 7.2.

Obsloužit můžete ale také uživatele Internet Exploreru 8 a starších verzí. Aby vše fungovalo, použijeme Flash.

[32] Pokud se chcete dozvědět něco více o kódování vašich vlastních videosouborů, podívejte se do přílohy C.

Obrázek 7.2: Naše video zobrazené ve videopřehrávači HTML5 v prohlížeči Firefox 5

Nouzové řešení

Aby bylo možné zároveň použít nouzové řešení založené na technologii Flash a také použít element HTML5 `video`, do elementu `video` umístíme kód elementu `object` pro video ve Flashi. Web Video For Everybody[33] podává přehled tohoto procesu docela do detailu, ale my půjdeme za hranici této základní implementace.

FlowPlayer[34] je přehrávač založený na technologii Flash, který dokáže přehrát vaše video kódované pomocí H.264. Open source verzi přehrávače stáhneme

[33] *http://videoforeverybody*

[34] *http://flowplayer.org/download/index.html*

a soubory `flowplayer-x.x.x.swf` a `flowplayer-controls-x.x.x.swf` umístíme v rámci našeho projektu do složky `swf`, abychom měli vše hezky roztříděné.

Poté do elementu kód umístíme následující kód, a to hned za náš poslední element `source`:

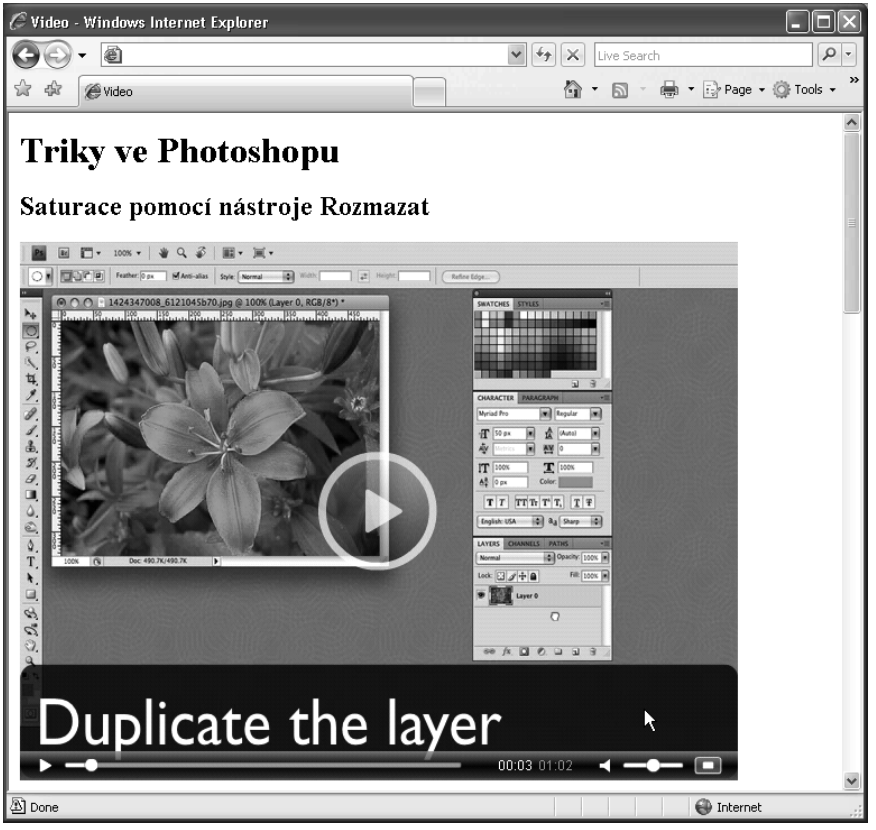
```
<object width="640" height="480"
   type="application/x-shockwave-flash"
   data="swf/flowplayer-3.2.2.swf" >
   <param name="movie" value="swf/flowplayer-3.2.2.swf" />
   <param name="allowfullscreen" value="true" />
   <param name="flashvars"
      value='config={"clip" :{"url" :"../video/h264/01_blur.mp4",
         "autoPlay" :false,
         "autoBuffering" :true
         }
      }' />
   <img src="video/nahled/01_blur.png"
      width="640" height="264" alt="Poster Image"
      title="Není možné přehrát video." />
</object>
```

Věnujte pozornost následující části:

```
<param name="flashvars"
      value='config={"clip" :{"url" :"../video/h264/01_blur.mp4",
         "autoPlay" :false,
         "autoBuffering" :true
         }
      }' />
```

Zdrojové soubory k videu potřebují být umístěné relativně k místu, kde se nachází FlowPlayer. Jelikož jsme FlowPlayer umístili do složky `swf`, použijeme cestu `../video/h264/01_blur.mp4`, aby přehrávač viděl naše video.

Jakmile stránku zobrazíme v Internet Exploreru, naše video se přehraje, a díky FlowPlayeru tedy není třeba kódovat video v jiném formátu. *A* navíc tak není třeba instalovat Flash. Naši přátelé s Internet Explorerem uvidí to, co je vidět na obrázku 7.3.



Obrázek 7.3: Naše video v Internet Exploreru za použití FlowPlayeru

Samozřejmě stále budeme muset přijít s řešením pro ty, kteří nemají k dispozici nativní podporu videa, a ani nemají nainstalovaný Flash. Proto necháme takové uživatele stáhnout si naše video přidáním sekce s odkazy na stažení.

```
<section class="stazeni">
   <header><h3>Downloads</h3></header>
   <ul>
     <li><a href="video/h264/01_blur.mp4">Formát H264, přehratelné
         na většině platformách</a></li>
     <li><a href="video/theora/01_blur.ogv">Formát OGG</a></li>
     <li><a href="video/webm/01_blur.webm">Formát WebM </a></li>
   </ul>
</section>
```

Ke skrytí těchto videí v případě, kdy není podporována specifikace HTML5, můžeme použít JavaScript. Takto:

```
function prehrajeVideo() {
  return !!document.createElement('video' ).canPlayType;
}
if(prehrajeVideo()){
  $(#video .stazeni).hide();
}
```

Zde využíváme detekční techniku velmi podobnou, kterou jsme použili v předchozím tipu u audia. V našem případě dává větší smysl nechat lidi stáhnout si tato videa, aby je pak mohli použít ve svých iPodech nebo iPadech, takže si je budou moci později sami prohlédnout.

Javascriptové API – Media Content

V této kapitole jsme se lehce dotkli javascriptové API pro elementy `audio` a `video`. Plné API umí detekovat typy audiosouborů, které dokáže prohlížeč přehrát, a poskytuje metody k ovládání přehrávání zvuku.

V předchozím tipu jsme vytvořili stránku s více zvukovými ukázkami. Použít bychom mohli javascriptová API k přehrávání zvuků ve stejnou dobu. Zde je velmi zjednodušené řešení:

```
var element = $("<p><input type='button' value='Přehrát vše'/></p>" )
element.click(function(){
   $("audio" ).each(function(){
      this.play();
   })
});
$("body" ).append(element);
```

Tento kód vytváří tlačítko „Přehrát vše“, po jehož stisknutí dojde k zahájení cyklu přes všechny elementy `audio` na stránce a volání metody `play()` u každého z elementů.

Podobně to můžeme provést i s videem. Existují metody k přehrání a pozastavení elementů a dokonce k dotázání na aktuální čas.

Naneštěstí v době přípravy knihy není toto javascriptové API ještě podporované všude. To by vás ale nemělo odradit od toho, abyste si prohlédli možnosti zmíněné ve specifikaci (*http://www.w3.org/TR/html5/video.html#media-elements*) a zjistili, co vše je možné.

Omezení videa v HTML5

Existují tři velmi závažná omezení, která aktuálně limitují použitelnost videa v HTML5.

Za prvé, video v HTML5 neobsahuje žádná opatření ke streamování videosouborů. Uživatelé si již zvykli na to, že mohou video posouvat na vybraná místa a odtud si je prohlédnout. To je něco, v čem videopřehrávače založené na technologii Flash excelují, a to díky množství snahy, kterou společnost Adobe vložila do technologie Flash jako do platformy pro dodávání videa. Abyste se mohli posunovat ve videu v HTML5, soubor se musí v prohlížeči celý stáhnout. To se ale může časem změnit.

Za druhé, zde neexistuje způsob, jímž spravovat práva. Weby, jako je Hulu,[35] které chtějí předcházet pirátství jejich obsahu, se nemohou na video v HTML5 spolehnout. Flash tedy zůstává v této situaci jediným reálným řešením.

A nakonec, a to je to nejdůležitější, proces kódování videa je finančně a časově náročný. Kvůli potřebě kódovat video do více formátů, je video v HTML5 méně atraktivní. Z toho důvodu můžete vidět mnoho webů poskytujících video v patentovaném formátu H.264, takže je možné je přehrát na iPodu a iPadu za pomoci kombinace elementu HTML5 `video` a Flashe.

Tyto problémy ale HTML5 neohrozí. Jsou zde nicméně věci, na něž je třeba si dát pozor před tím, než budete moci videem v HTML5 nahradit Flash jako technologii k poskytování videa.

Sledujte odvětví zábavy pro dospělé

Odvětví zábavy pro dospělé velmi silně ovlivnilo internetové technologie, od internetových obchodů po vzestup Flashe (*http://chicagopressrelease.com/news/in-tech-world-porn-quietly-leads-the-way*). A mají stejný vliv i nyní v případě HTML5 (*http://news.avn.com/articles/Joone-Points-to-HTML-5-as-Future-of-Web-Content-Delivery-401434.html*). Zařízení jako je iPhone a iPad jsou mnohem osobnější než stolní počítače a notebooky a nefunguje na nich Flash. Mnoho takto orientovaných webů již začalo z tohoto důvodu s přechodem způsobu doručování videa z technologie Flash na HTML5 a kodek H.264. Zajímavé je, že je vůbec netrápí, že současná implementace videa v HTML5 neposkytuje správu práv.

Odvětví zábavy pro dospělé se vůbec nebojí riskovat a ve videu v HTML5 vidí hodně zajímavých výhod vycházejících z jejich zájmu o technologie.

[35] *http://www.hulu.com*

Audio, video a přístupnost

Žádná z nouzových řešení nefungují skutečně dobře v případě uživatelů s omezeními. Vlastně specifikace HTML5 výslovně tuto skutečnost uvádí. Uživatelům se sluchovým omezením nebude nic platná možnost stažení audiosouboru, a uživatelům s vizuálním omezením nebudou mít velký užitek z videosouboru, který si mohou prohlédnout mimo prohlížeč.

Když našim uživatelům poskytujeme obsah, měli bychom poskytnout použitelné alternativy, kdykoliv to je možné. Video- a audiosoubory by měly být opatřeny přepisem, který si lidé mohou prohlédnout. Pokud vytváříte váš vlastní obsah, je přepis snadná věc, pokud s ním počítáte již od začátku, protože může vycházet rovnou z napsaného scénáře. A pokud přepis není k dispozici, zvažte alespoň shrnutí zmiňující důležité části videa.

```
<section class="prepis">
   <h2>Přepis</h2>
   <p>Existující vrstvu přetáhneme na tlačítko Nová u spodního
      okraje palety Vrstvy, čímž vytvoříme novou kopii.</p>
   <p>Poté v nabídce Filtry vyberte příkaz "Gaussovské rozostření".
      Míru rozostření změníme jednoduše tak, že v obrázku přijdeme
      jen o malou část detailu. </p>
   <p>Nyní poklepejte na vrstvu, čímž ji budete moci upravit, a
      změňte režim prolnutí na "Překrýt". Poté můžeme upravit
      množství efektu tak, že posunete jezdce Krytí.</p>
   <p>A výsledkem je lehce rozšířený obrázek.</p>
</section>
```

Text přepisu nebo odkaz směřující na přepis můžete na hlavní stránce s videem skrýt. Pokud jej vytvoříte tak, aby bylo možné jej snadno najít a byl srozumitelný, pak bude skutečně užitečný.

Budoucnost

Prvotřídní podpora audia v prohlížeči vývojářům otevírá ohromné množství nových možností. Webové aplikace tvořené pomocí JavaScriptu mohou snadno spouštět zvukové efekty a výstrahy bez použití Flashe ke vložení audia do stránky. Nativní video umožňuje vytvořit video dostupné v zařízeních, jako jsou iPhony, ale také nám poskytuje otevřenou, standardní metodu interakce s videem pomocí JavaScriptu. A co je nejdůležitější, budeme schopni zacházet s video- a audioklipy tak, jako dokážeme zacházet s obrázky, včetně jejich správného sémantického označení a snadnější identifikace.