SCOTT GILBERTSON 05.10.10 6:34 PM

EMBED VIDEOS IN YOUR WEB PAGES USING HTML5

HTML5 video is taking the web by storm.

Not only has a very public (and contentious) debate unfolded on the web about the efficacy of presenting videos using HTML5 instead of Flash, but momentum is gathering behind the nascent web standard.

From giant video sites like YouTube to Wikipedia, everyone it seems wants to get their video out of Flash and into native web formats. With Microsoft recently announcing it will support the HTML5 video tag in the coming Internet Explorer 9, expect even more sites to abandon Flash for native video.

So, you want in on the fun? Do you want to use some HTML5 video tags on your site right now? No problem. Fasten your seat belts, as we're about to take a tour of the wonderful world of HTML5 video.

BROWSER SUPPORT FOR HTML5

First, let's deal with some very basic stuff, like where this will work and where it won't. As you can see in the table below, only the latest versions of most browsers support native video playback using HTML5's <video> tag.

HTML5 <video> support by browser:

Fx 3.0	Fx 3.5	IE7	IE8	IE9	Safari 3	Safari 4	Chrome 3+	Opera 10.5
·	✓	·	·	✓	✓	✓	✓	✓

Since Firefox 3.0 and IE 7 & 8 lack any support for HTML5 video, you'll have to come up with a fallback plan for serving video to those users. Depending on what you want to do you, could fallback first to Quicktime and then, failing that, to Flash. That's the strategy used in Video for Everyone (note that as of v0.4, Video for everyone no longer falls back to QuickTime).

To keep things simple we're just going to fall straight from HTML5 to Flash.

FORMATS, CODECS AND TECHNICALITIES

The next thing you need to understand is what is actually happening when you load and play a video file in your web browser. You're probably used to thinking of video as .mp4 or .mov files, but unfortunately it's not that simple. The actual file formats are just containers. Think of them as a bit like a .zip file — they hold other stuff inside.

Each container holds at minimum one video track and, most likely, one audio track. When you watch a movie online, your video player (most likely Flash) decodes both the audio and video and sends the information to your screen and speakers.

Why does this matter? Well, because the process of decoding what's inside the video container file varies. To know how to decode a movie, the player (which is your web browser in the case of HTML5 video) has to know which codec the movie was encoded with.

When it comes to web video there are two codecs to worry about: **H.264** and **Theora**.

There's a huge debate right now among web developers, browser makers and just about everyone else as to which codec is right for the web. We believe that a free, open codec without patent and licensing restrictions is the best solution for the web. Sadly, neither codec exactly fulfills that dream, so for now, let's just skip the whole argument and be practical — we're going to use both codecs.

Why? Well, have a look at the table below, which shows which codecs work where and you'll quickly see that there is no one-size-fits-all-browsers solution. Only Google Chrome can play both H.264 and Theora.

Codec support by browser/platform:

	Firefox	Opera	Chrome	Safari	IE 9	iPhone	Android
Ogg Theora	✓	✓	✓	·	·	·	
H.264	·	·	✓	✓	✓	✓	✓

So, you may be thinking ... if HTML5 video doesn't work in IE7 or IE8 and it means I'm going to have to encode my videos twice, then why bother at all? Well, the best answer is simple: mobile users. If you want iPhone and Android users to be able to see your video, HTML5 is the only way to do it — Flash support is coming to Android sooner or later but for now HTML5 is the only option, and we all know Flash doesn't run on the iPhone or the iPad.

THE HTML5 CODE

Here's how to actually embed your videos. First, we encode video in both .ogv and .mp4 containers. Encoding video is beyond the scope of this article, so instead we suggest you check out Mark Pilgrim's online book Dive Into HTML5, which has a whole chapter devoted to understanding video encoding. Pilgrim's encoding suggestions use free software to get the job done, and in the end you'll have two files — one .mp4 and one .ogv.

Now it's time to unleash those movies in pure HTML glory. Here's the code:

```
<video width="560" height="340" controls> <source src="path/to/myvideo.mp4"
type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'> <source src="path/to/myvideo.ogv"
type='video/ogg; codecs="theora, vorbis"'> </video>
```

Yes, that's it. What we've done here is use the `<video>` tag to specify the dimensions of our video, and to denote that we want to use the browser's default controls. Then, within the video tag, we've added two `<source>` elements which link to our video files.

The "type" attribute of the `<source>` tag helps the browser understand which file it should load. It's a bit of an ugly chunk of code that needs to specify the container format, the video codec and the audio codec.

In this case we've assumed standard .ogv and baseline encoded H.264 video as per Pilgrim's tutorial. See the WHATWG wiki for more information on which video types you can specify.

And there you have it — native web video, no plugins required.

DEALING WITH EVERYONE ELSE

What about IE7, IE8 and older versions of just about any other browser? Well, for those users, we'll fall back on Flash. To do that, we just use an `<embed>` tag within our video tag:

```
<video width="560" height="340" controls> <source src="path/to/myvideo.mp4"
type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'> <source src="path/to/myvideo.ogv"
type='video/ogg; codecs="theora, vorbis"'> <object width="640" height="384"
type="application/x-shockwave-flash" data="path/to/swf/player.swf?
image=placeholder.jpg&file=path/to/myvideo.mp4"> <param name="movie"
value="path/to/swf/player.swf?image=placeholder.jpg&file=path/to/myvideo.mp4" />
</object> </video>
```

Now any browser that doesn't understand the HTML5 video tag will just continue on its way until it hits the object tag, which it should understand (note that the order, mp4 before ogv, is important for iPad support — Apple requires that mp4 be the first video file).

Of course for this to work you need a Flash video container. JW Player is one popular example, or you can roll your own using Adobe's tools. Also remember that we still haven't handled the case of an older version of Firefox with no Flash plugin installed (maybe your users are surfing your tubes with an outdated Linux machine). You can always add good old text-based links to the video files as a catch-all fix for anyone who can't, for whatever reason, see either the HTML5 or Flash versions.

CONCLUSION

Embedding HTML5 video isn't significantly more difficult than using Flash, especially if you've been using H.264 video files in your Flash player — which is exactly what YouTube has done with its HTML5 beta.

While we're concerned about the licensing and patent requirements of H.264, it isn't hard to notice that if you skip Theora and make all non-H.264 fall back to Flash, you've still saved yourself a considerable encoding headache. In fact, that's probably the best practical argument against Mozilla and Opera's refusal to support H.264.

If you'd like to use some of the more advanced aspects of HTML5 video, be sure to check the SublimeVideo player, which offers very nice JavaScript-powered set of custom controls. Also be sure to have a look at Video for Everybody, which makes for more complex code but handles just about every use case you could imagine. And there's a handy Video for Everybody WordPress plugin as well.

See Also:

Building Web Pages With HTML5

Add Semantic Value to Your Pages With HTML5

Who Needs Flash?