JW Player for HTML5

Release 1.0 beta

www.longtailvideo.com

May 20, 2010

Contents

1 HTML5 Video Tag Reference

This guide provides an overview of the functionalities and options provided by the HTML5 <video> tag. It includes a few examples, plus a list of known browser-specific quirks.

1.1 Introduction

Our HTML5 player is designed to gracefully enhance videos embedded with the HTML5 <*video*> tag. For a complete reference, visit the W3C HTML5 video element draft.

1.2 Example

Here is a basic example of a video embedded with the HTML5 video tag:

```
<video height="270" src="/static/bunny.mp4" width="480" controls>
  <a href="/static/bunny.mp4">Download this video</a>
</video>
```

This example will display the video in 480 by 270 pixels with a small controlbar over it. Browsers that do not support the video tag (nor the MP4 video) will render the enclosed download link instead.

1.3 Attributes

The HTML5 <*video*> tag has a short list of useful attributes:

autoplay ()
: Include this attribute (no value) to let the video start as the page gets loaded. Use sparsely, since this might get annoying.

controls ()
: Include this attribute (no value) to display a simple controlbar, provided by the user agent. The looks of this controlbar vary per browser.

height (number)
: Height of the video on the page, in pixels is required.

loop ():
: Include this attribute to let the browser continously repeat playback of the video. At present, no browser honors this attribute. More info below, under *browser quirks*

poster (string)
: The url of a poster frame that is shown before the video starts. Can be any PNG, JPG or GIF image. Is *undefined* by default.

preload (*auto*, *metadata*, *none*)
: This attribute defines whether the video is entire loaded on page load, whether only the metadata is loaded on page load, or whether the video isn't loaded at all. At present, no browser honors this attribute. More info below, under *browser quirks*,

src (string)
: The URL of the video to play. This is not required, since source files can also be listed using <*source*> tags (see below). It is *undefined* by default. See *Supported file formats* for more info on which browser supports which filetype.

width (*number*)
: Width of the video on the page, in pixels. Is required.

Here is another example of an HTML5 video tag, this time using more attributes:

```
<video
  controls
  height="270"
  loop
  poster="/static/bunny.jpg"
  src="/static/bunny.mp4"
  width="480">
  <a href="/static/bunny.mp4">Download the video</a>
</video>
```

This example will display the video with a poster frame and controls. After starting the video, it will repeat playback.

1.4 Multiple Sources

It is possible to provide a video tag with multiple source videos. The browser can then pick whichever file it can playback. This functionality works through the HTML5 <*source*> tag, which can be nested inside the <*video*> tag. A common use case is the provision of two videos, one in OGG for Firefox/Opera and one in MP4 for Chrome/Safari/iPhone/iPad:

```
<video height="270" width="480" controls>
  <source src="/static/bunny.mp4" type="video/mp4">
  <source src="/static/bunny.ogg" type="video/ogg">
  <a href="/static/bunny.mp4">Download the video</a> for local playback.
</video>
```

This example will play the MP4 video in Chrome/Safari and the OGG video in Firefox/Opera. Note that Chrome will play the OGG file if you reverse the order of the two <*source*> tags (Chrome supports both formats). We strongly advise against this though: MP4 files render much more smoothly and seek effortlessly in Chrome, whereas OGG files don't.

Defining codecs

The video type can be specified in more detail by amending the codecs of the mediafile to the mimetype. Here is an example in which we explicitly specify that the first video uses H.264 Baseline video / AAC Low-complexity audio and the second video uses Theora video and Vorbis audio:

```
<video height="270" width="480" controls>
  <source src="/static/bunny.mp4" type="video/mp4; codecs='avc1.42E01E, mp4a.40.'">
  <source src="/static/bunny.ogg" type="video/ogg; codecs='theora, vorbis'">
```

```
 <a href="/static/bunny.mp4">Download the video</a>
</video>
```

This additional level of detail is very useful at such instances where nonstandard codecs are used. For example, the MP4 video could be using H.264 High profile video, which would cause e.g. the iPhone/iPad to not support it. A more extensive overview of supported codecs and mimetypes can be found in *Supported file formats*.

1.5 Browser Quirks

The W3C HTML5 video format is still in draft. Therefore, browser might have implemented older versions of this draft or filled the gaps with custom implementations. Especially around subjects like event broadcasting and (pseudo)streaming, things are still very buggy. Here are a couple issues worth noting:

No single codec

The biggest issue to be aware of - it's actually hard to have missed this in the press - is that there is no single video format supported by all browsers. Firefox and Opera support the *OGG* format, Internet Explorer (9), Safari and the iPhone/iPad support the *MP4* (H264) format. Chrome does both. Our HTML5 player works around this issue by including a *Flash player fallback* for Firefox/Opera and current versions of IE (6/7/8). Including only an *OGG* in a video tag is not recommended; most browsers (and Flash) would not be able to render it.

Always preloading

A second browser quirk to be aware of is that any browser that currently supports HTML5 will actually preload your video on page load. The tag's *preload* attribute is not honored. This is probably a temporary issue, but it is an important one: unwatched preloaded video is wasted bandwidth. Our HTML5 player works around this issue by clearing the source attribute of a video tag op page load and reinserting it after the user has clicked *play*.

No looping

Third, none of the browsers that currently implement HTML5 video support the *loop* attribute. Our player works around this issue by parsing out the *loop* attribute, mapping it to the player's *repeat* option and using scripting to re-start a video after it has completed.

2 Embedding the player

Embedding the HTML5 player into a page is essentially a two-step process: first, include the JWPlayer javascripts and second, invoke the JW Player method upon certain video tag.

2.1 Basic example

To Embed the HTML5 Player include the *jquery.js* and *jquery.jwplayer.js* scripts into the <*head*> of a page. This can be done with the following code:

```
<script type="text/javascript"
  src="http://ajax.googleapis.com/ajax/libs/jquery/1.4/jquery.min.js"></script>
<script type="text/javascript" src="/scripts/jquery.jwplayer.js"></script>
```

If you already use the jQuery library on your page, make sure to not include it twice. The JW Player works with jQuery version 1.3 and above.

Second, instantiate the player over one or more video tags. Here is an example:

```
<video id="myplayer" src="/files/bunny.mp4" width="480" height="270"></video>

<script type="text/javascript">
  $('#myplayer').jwplayer({
    flashplayer:'/files/player.swf',
    skin:'/files/five.xml'
  });
</script>
```

The player here is instantiated using two options, for telling it where the skin (graphics) and the flashplayer (for browsers that don't support HTML5) are located. There are several more options available: *here is the full list*.

Note that the skin option is required for the beta version of this player.

2.2 Fallbacks

When the player detects that none of the source videos can be played, it can fall back to an instance of the JW Player for Flash. The player will look exactly the same, since the flashplayer loads the same skin element. This is a very exciting feature, since it allows you to use the HTML5 video tag today without losing support for such browsers as Internet Explorer 6, 7 and 8.

If Adobe Flash is not supported by the browser (or if you disabled the *flashplayer option*), the player will fall back to only displaying the poster image. When clicking this image, the video file will be downloaded, allowing users to playback the video with a local player.

2.3 Multiple Sources

Video tags with multiple nested <*source*> tags are also supported by the player. A typical setup is one where an H264/AAC video is provided for playback in Safari/Chrome and a Theora/Vorbis video is provided for playback in Firefox/Opera:

```
<video width="480" height="270" id="myplayer">
  <source src="/files/bunny.mp4" type="video/mp4" />
  <source src="/files/bunny.ogv" type="video/ogg" />
</video>
```

The player tries to playback the <*source*> videos in order of appearance. So with the above example, if a browsers supports both H264/AAC and Theora/Vorbis (e.g. Google Chrome), the first option (H264/AAC) will be played.

If no mimetypes and/or codecs are provided, the player assumes H264/AAC for files with .mp4 and .mov extensions and Theora/Vorbis for files with .ogv and .ogg extensions. See *Supported file formats* for a full list of supported media formats.

2.4 Multiple Players

Technically, instantiating the player is a matter of calling the *jwplayer()* function on a jQuery selector. For example, here we apply the player to all <video> elements on the page, setting the skin and mute configuration options:

```
<video src="/files/bunny.mp4" width="480" height="270"></video>
<video src="/files/corrie.mp4" width="480" height="270"></video>

<script type="text/javascript">
  $('video').jwplayer({
    mute: true,
    skin:'/files/five.xml'
  });
</script>
```

Here is another example, in which two players are instantiated, each with their own settings:

```
<video id="video1" src="/files/bunny.mp4" width="480" height="270"></video>
<video id="video2" src="/files/corrie.mp4" width="480" height="270"></video>

<script type="text/javascript">
  $('#video1').jwplayer({
    autostart:true,
    skin:'/files/five.xml'
  });
  $('#video2').jwplayer({
    autostart:false,
    skin:'/files/stormtrooper.xml'
  });
</script>
```

Full jQuery selector support is available. However, make sure your video tag contains an ID attribute if you want to interact with it using the API.

3 Supported file formats

The HTML5 player can play a wide range of media formats. Since the player includes an Adobe Flash based fallback, the list of formats is not restricted to formats the browsers can play, but also extends to formats the Flash Plugin can play. The exact list of supported formats varies per browser and per device.

3.1 Video formats

Generally, the following video formats are supported by the player:

H264 video / AAC audio in the MP4 container
: Supported for Safari 4+, Chrome 3+, Internet Explorer 9+ and as part of the the Flash fallback.

H264 video / AAC Audio in the MOV container.
: Supported for Safari 4+ and as part of the the Flash fallback.

Theora video / Vorbis audio in the OGG container
: Supported for Firefox 3.5+, Opera 10.5+, Chrome 4+ and as part of the the Flash fallback.

VP6 video / MP3 audio in the FLV container
: Supported as part of the the Flash fallback.

Spark (H263) video / MP3 audio in the FLV container
: Supported as part of the the Flash fallback.

Note the use of the word **generally**, since support also depends upon such variables as the dimensions of the video and/or the encoding profile used. For example, the iPhone supports H264 video, but only in the Baseline profile up to 640 by 480 pixels.

3.2 Determining Support

There are three parameters the player uses to determine if a certain video is supported on a certain device: *file extension*, *mimetype* and *codecs*. A mimetype + codecs listing is the most accurate way for the player to determine if a format is supported. In case only a mimetype or a file extension is available, the player presumes certain codecs to check if support is available.

Under water, the player uses the *video.canPlayType()* method to ask a browser if it can play a video. In other words, there's no *if(IE) {} elseif(Firefox) {}* things going on.

Codecs

A codec listing looks like this:

```
<video width="400" height="200" id="player">
  <source src="/static/video.mp4" type="video/mp4; codecs='avc1.42E01E, mp4a.40.2'">
  <source src="/static/video.ogv" type="video/ogg; codecs='theora, vorbis'">
</video>
```

Here's a list of frequently-used codecs:

avc1.42E01E, mp4a.40.2
: Baseline profile H264 video / Low-complexity AAC audio.

avc1.4D401E, mp4a.40.2
: Main profile H264 video / Low-complexity AAC audio.

avc1.64001E, mp4a.40.5
: High profile H264 video / Low-complexity AAC audio.

theora, vorbis
: Theora video and Vorbis audio.

Mimetypes

The mimetype tells the user agent what video container is being used. It's more fuzzy than a list of codecs, since most containers contain audio and video in multiple codecs. Below is what just a mimetype listing looks like:

```
<video width="400" height="200" id="player">
  <source src="/static/video.mp4" type="video/mp4">
  <source src="/static/video.ogv" type="video/ogg">
</video>
```

Here is a list of frequently-used mimetypes and the codecs the player presumes:

video/mp4
: MP4 video. The player presumes H264 baseline video and AAC low-complexity audio.

video/ogg
: OGG video. The player presumes Theora video and Vorbis audio.

video/quicktime
: Quicktime video. The player presumes H264 baseline video and AAC low-complexity audio.

video/flv
: Flash video. The player presumes no codec, but switches straight to Flash fallback.

Extensions

A file extension is even fuzzier than a mimetype, since several extensions are sometimes used for content with a single mimetype. The most frequent use case for only a file extension is when a video is embedded without <*source*> tags:

```
<video width="400" height="200" id="player" src="/static/video.mp4"></video>
```

Again, a list of frequently used file extensions and the mimetype/codecs the player presumes:

.mp4, .m4v, .f4v:
: The player presumes H264 baseline video and AAC low-complexity audio in an MP4 container.

.ogg, .ogv
: OGG video. The player presumes Theora video and Vorbis audio in an OGG container.

.mov
: The player presumes H264 baseline video and AAC low-complexity audio in a Quicktime container.

.flv
: The player presumes a Flash container and tries the Flash fallback.

3.3 Determining Flash Support

The player will try a fallback to Flash if the following three items check out:

- The *flashplayer option* needs to be set.
- The browser has the Flash plugin version 9.0.115 or higher installed (checked with javascript).
- The video is in MP4, Quicktime or Flash format. Codecs are not checked.

There are obvious cases in which the Flash fallback is enabled (such as MP4 video in Firefox), and less obvious cases (such as MP4 video with the H.264 HIGH codec in Chrome). In the latter case, the player profits from the more extensive H.264 support in Flash when compared to the browser. This will only work if you properly specify the used codecs in the <*source*> type attribute.

4 Configuration options

The options of the HTML5 player are what the *flashvars* are for the Flash player: configuration settings for tweaking its layout and behaviour.

Most of these options correspond to certain attributes of the <*video*> tag. Whenever the JW Player is applied to a tag, the options of this tag will automatically be loaded into the player.

4.1 Inserting options

Options can be added to the player by inserting them into a player instantiation code. For example, here we add the *skin* and *autostart* options:

```
$('video').jwplayer({
  skin:'/player/skins/five/five.xml',
  autostart:false
});
```

See *Embedding the player* for more embedding options. Note that the default embedding option (tagging everything with a **jwplayer** class) does not allow you to set specific player options.

4.2 Media options

duration (0):
Duration of the video, in seconds. This can be set to the actual duration of the video, purely to make the controlbar reflect the video duration before it starts playing.

file (undefined):
The video file to play. See *Supported file formats* for a list of supported file formats in the various browsers.

Note: If the player is applied to a <*video*> tag, its **src** attribute is the default value of this option. If the player is applied to a <*video*> tag with multiple <*source*> tags, the first source that can be played back is used.

image (undefined):
The poster image to display before the video starts and after it ends. Can be any JPG, GIF or PNG image.

Note: If the player is applied to a <*video*> tag, its poster attribute is the default value of this option.

4.3 Layout options

For modifying the layout of the player, the following options are available:

controlbar ('bottom'):
Position of the controlbar related to the player. Can be bottom, none or over.

Note: If the player is applied to a <*video*> tag, its controls attribute is the default value of this option (with bottom corresponding to true).

height (295):
Height of the player in pixels. Can be a number between 10 and 9999.

Note: If the player is applied to a <*video*> tag, its **height** attribute is the default value of this option.

skin ('assets/five/five.xml'):
PNG Skin to load. PNG skins are fully supported by both the HTML5 and the Flash player.

Note: The JW Player for HTML5 is fully compatible with the PNG Skinning model of the JW Player for Flash. Any PNG skin built for the Flash player can be used in the HTML5 player. You do need to unzip a skin before it can be loaded into the HTML5 player. Zipped skins are not supported (yet).

screencolor ('000000'):
Background color of the video display. Can be any hexadecimal color value.

stretching ('uniform'):
Stretching mode of the video to the display. Can be:

- *uniform*: scale the video proportionally so that it fits to the display.
- *fill*: scale the video proportionally to entirely fill the display.
- *exactfit*: scale the video disproportionally to exactly fit the display.
- *none*: show the video in its original dimensions.

width (480):
Width of the player in pixels. Can be a number between 10 and 9999.

Note: If the player is applied to a <*video*> tag, its **width** attribute is the default value of this option.

4.4 Behaviour options

For modifying the behaviour of the player, the following options are available:

autostart (false):
Set this to **true** to automatically start the player on page load.

Note: If the player is applied to a <*video*> tag, its **autoplay** attribute is the default value.

debug (false):
Set this to **true** to let the player fire all its events to *console.log()*. This can be read e.g. by Firebug or the Safari error console.

flashplayer ('assets/player.swf'):
Location of the JW Player for Flash that is used for fallback in browsers that do not support HTML5. When set to **false**, the flashplayer fallback is not used.

mute ('false'):
Set this to **true** to mute the video on page load.

repeat (false):
Set this to **true** to continously repeat playback.

Note: If the player is applied to a <*video*> tag, its **loop** attribute is the default value.

volume (90):
Startup volume of the video, can be **0** to **100**.

5 Player API

The HTML5 player contains a simple javascript API which can be used to:

- Request the various playback states (*time*, *volume*, *dimensions*).
- Control the player through a number of available methods (*play*, *pause*, *load*).
- Track the player state by listening to certain events (*time*, *volume*, *resize*).

Since the HML5 player is built using the jQuery, its API is using several conventions found in this framework.

5.1 Referencing a player

Before you can interact with the player, you need to be able to reference it (get a hold of it) from your javascript code. If you use a single player on the page, this is very simple:

```
<video width="400" height="300" src="/static/video.mp4"></video>

<script type="text/javascript">
  var player = $.jwplayer();
  player.play();
</script>
```

If you have multiple players on a page, you can reference a single player by giving each player a specific ID. Next, you use a jQuery selector to get to the player you want:

```
<video id="player1" width="400" height="300" src="../video1.mp4"></video>
<video id="player2" width="400" height="300" src="../video2.mp4"></video>

<script type="text/javascript">
  var player = $.jwplayer("#player1");
  player.play();
</script>
```

5.2 Requesting properties

The player contains a number of properties (such as its *volume* or playback *state*) that can be requested at runtime. Here's how to request a player property:

```
<video width="400" height="300" src="/static/video.mp4"></video>

<p onclick="alert($.jwplayer().volume())">Get player volume</p>
```

Here's the full list of available properties:

buffer ()
: The percentage of the videofile that is downloaded to the page. Can be **0** to **100**.

duration ()
: The duration of the video file, in seconds. This will not be available on startup, but as of the moment the metadata of a video is loaded.

fullscreen ()
: The fullscreen state of the player. Can be **true** or **false**.

  Note: Current browsers do not support true fullscreen, like Flash or Silverlight do. The fullscreen mode of the HTML5 player will rather be a full-browser-screen.

height ()
: The height of the player, in pixels.

mute ()
: The sound muting state of the player. Can be **true** (no sound) or **false**.

  Note: *Volume* and *mute* are separate properties. This allows the player to switch to the previously used volume when the player is muted and then unmuted.

state ()
: The current playback state of the player. Can be:

  •**idle**: Video not playing, video not loaded.

  •**buffering**: Video is loading, the player is waiting for enough data to start playback.

  •**playing**: Video is playing.

  •**paused** Video has enough data for playback, but the user has paused the video.

time ()
: The current playback position of the video, in seconds.

volume ()
: The audio volume percentage of the player, can be **0** to **100**.

width ()
The width of the player, in pixels.

5.3 Calling methods

The player exposes a list of methods you can use to control it from javascript (e.g. *play*). Here's how to invoke a player method:

```
<video width="400" height="300" src="/static/video.mp4"></video>

<ul>
  <li> onclick="$.jwplayer().play()">play the video</li>
  <li> onclick="$.jwplayer().pause()">pause the video</li>
  <li> onclick="$.jwplayer().seek(0)">play from the beginning of the video</li>
</ul>
```

Here's the full list of available methods for the player:

fullscreen (state)
Change fullscreen playback. The state can be **true** or **false**.

Note: Current browsers do not support true fullscreen, like Flash or Silverlight do. The fullscreen mode of the HTML5 player will rather be a full-browser-screen.

load (url)
Load a new video into the player. The **url** should be a valid hyperlink to the video file (e.g. **/static/video/holiday.mp4**). The file can be in any *supported media type*.

mute (state)
Change the mute state of the player. The *state* can be **true** or **false**.

pause ()
Pause playback of the video. If the video is already *paused* (or *idle*), this method does nothing.

play ()
Start playback of the video. If the video is already *playing* (or *buffering*), this method does nothing.

resize (width,height)
Resize the player to a certain **width** and **height** (in pixels). Use this to e.g. toggle between a small and large player view like Youtube does.

seek (position)
Seek to and playing the video from a certain *position*, in seconds (e.g. **120** for 2 minutes in). If the player is *paused* or *idle*, it will start playback.

stop ()
Stop playing the video, unload the video and display the poster frame. The player proceeds to the **idle** state.

volume (volume)
Set the player audio volume percentage, a number between 0 and 100. When changing the volume while the player is muted, it will also automatically be unmuted.

5.4 Listening to events

The player broadcasts an event whenever one of its properties change (e.g. the playback *time*, *fullscreen* state or *volume*). Listen to these events to build interaction between the player and other elements on the page (e.g. showing a message when the video starts). Here's how to listen to an event:

```
<video width="400" height="300" src="/static/video.mp4"></video>

<p id="message"></p>

<script type="text/javascript">
  function stateListener(event) {
    $('#message').html('The new playback state is '+event.state);
  };
  $.jwplayer.addEventListener('jwplayerPlayerState',stateListener);
</script>
```

Here's the full list of events, and their parameters:

'jwplayerMediaBuffer'
: This event is fired while the video to play is being downloaded. Parameters:

  - **buffer** (*number*): percentage of the video that is downloaded (0 to 100).
  - **player** (*jwplayer*): Reference to the player that sent the event.
  - **version** (*string*): version of the JW Player, e.g. 1.0.877.

'jwplayerMediaComplete'
: The end of the video was reached during playback. Parameters:

  - **player** (*jwplayer*): Reference to the player that sent the event.
  - **version** (*string*): version of the JW Player, e.g. 1.0.877.

'jwplayerMediaError'
: There was an error loading or playing the video (e.g. the video was not found). Parameters:

  - **message** (*string*): The error message.
  - **player** (*jwplayer*): Reference to the player that sent the event.
  - **version** (*string*): version of the JW Player, e.g. 1.0.877.

'jwplayerFullscreen'
: The player has switched from / to fullscreen mode. Parameters:

  - **fullscreen** (*boolean*): The new fullscreen state (if *true*, the player is in fullscreen).
  - **player** (*jwplayer*): Reference to the player that sent the event.
  - **version** (*string*): version of the JW Player, e.g. 1.0.877.

'jwplayerMediaLoaded'
: A new video is loaded into the player. Note that the actual video date is not loaded yet (you should listen to the *jwplayerMediaBuffer* event for that). This event states the loaded file is ready for playback. Parameters:

  - **player** (*jwplayer*): Reference to the player that sent the event.
  - **version** (*string*): version of the JW Player, e.g. 1.0.877.

'jwplayerMediaMeta'
: The player has received metadata about the video it is playing. Parameters:

  - **data** (*object*): an object with key:value pairs of metadata (e.g. duration, height and width).
  - **player** (*jwplayer*): reference to the player that sent the event.
  - **version** (*string*): version of the JW Player, e.g. 1.0.877.

'jwplayerMediaMute'

The mute state of the player just got updated. When muted, the audio is completely dropped. The player will display a muted icon on top of the video. Parameters:

- **mute** (*boolean*): the new mute state (if *true*, the player is silent).
- **player** (*jwplayer*): reference to the player that sent the event.
- **version** (*string*): version of the JW Player, e.g. 1.0.877.

'jwplayerResize'

The player was resized on the page to different dimensions. Parameters:

- **height** (*number*): the new height of the player, in pixels.
- **width** (*number*): the new width of the player, in pixels.
- **player** (*jwplayer*): reference to the player that sent the event.
- **version** (*string*): version of the JW Player, e.g. 1.0.877.

'jwplayerPlayerState'

The playback state of player was changed. Parameters:

- **oldstate** (*idle* ,*buffering* ,*paused* ,*playing*): The playback state the player just switched away from. Can be one of 4 states
- **newstate** (*idle* ,*buffering* ,*paused* ,*playing*): The playback state the player just switched to. Can be one of 4 states
- **player** (*jwplayer*): reference to the player that sent the event.
- **version** (*string*): version of the JW Player, e.g. 1.0.877.

'jwplayerMediaTime'

The position and/or duration of the mediafile have changed. This event typically fires when the mediafile is playing, with a resolution of 10x / second. Parameters:

- **position** (*number*): current playback position in the mediafile, in seconds.
- **duration** (*number*): total duration of the mediafile, in seconds.
- **player** (*jwplayer*): reference to the player that sent the event.
- **version** (*string*): version of the JW Player, e.g. 1.0.877.

'jwplayerMediaVolume'

The audio volume of the player just got updated. Parameters:

- **volume** (*number*): the new volume percentage (0 to 100).
- **player** (*jwplayer*): reference to the player that sent the event.
- **version** (*string*): version of the JW Player, e.g. 1.0.877.