Introducing VTT

WebVTT (Web Video Text Tracks), formerly known as WebSRT, is a W3C community proposal for synchronized video caption playback. It is a time-indexed file format and it is referenced by HTML5 video ***and*** audio elements.

As with many assistive technologies, it would be a mistake to assume that they are only meant as a way to provide for accessibility accommodations. We can enable captions when the ambient noise is too loud to listen to a recorded presentation, we can use chapters to navigate through a long lecture video just like DVD or Blue Ray movies.

Captions can also improve our movies' discoverability. Google indexes the content of our captions. Both YouTube and Google search can report results based on the video captions available for a given file.

WebVTT files provide open captions, independent of the audio or video files they are attached to, they are not "hard coded" into pixels. This also means that creating VTT files requires nothing more than a text editor; although there are more specialized tools to create the captions.

Browser support

Based on Silvia Pfeiffer's post to the VTT community group dated August, 2012, and updated with new information about Firefox, the following browsers support VTT tracks for video and audio:

Browser	First supported	Format Supported	Notes
Internet Explorer	Version 10	VTT and TML	▪ Test Page ▪ Documentation ▪ HTML5 Video Caption Maker ▪ Timed Text Track Information ▪ Timed Text Tracks examples

Google Chrome	Version 18	VTT	▪ Basic tutorial hosted at HTML5 Rocks ▪ Based on Webkit's implementation
Apple Safari	Version 6	VTT	▪ Based on Webkit's implementation
Opera	Since August, 2012	VTT	▪ Documentation at: dev.opera">dev.opera ▪ Based on Webkit's implementation
Firefox	Nightly	VTT	▪ See the Mozilla Developer Documentation for more information ▪ Tested with version 29.0a1 (12/14/2013) ▪ Feature enabled by default in nightly builds

Polyfills and alternatives

I will use one of the many polyfils available for HTML5 Video Tracks. Playr seems to be the most feature complete polyfill for HTML5 video tracks. The downside is 2 more files (one CSS and one JavaScript) to download for the video page but until VTT is widely supported the extra files are worth the effort to create accessible content.

One way to ensure that we only load our polyfill if the browser doesn't support tracks natively is to use Modernizr.load to conditionally load Playr's CSS and JavaScript when the browser does not support HTML5 video tag natively.

```
Modernizr.load([
  {
    // test whether we support video
    test : Modernizr.video,
    // Load the corresponding assetts for the polyfill you want to use
```

```
    // in this case we are using the playr polyfill
    nope : ['playr.js', 'playr.css']
  },
 ])
'playr.js'
```

The code below uses plain JavaScript to test if a browser supports HTML5 video by creating an empty video element and testing for the video's canPlayType property. It will not load the code for a polyfill like the Modernizr example.

```
var h, plink, pscript;
var canPlay = false;

// Create an empty video element
var v = document.createElement('video');
'video'// If the video can playType and can play MP4 video
if(v.canPlayType) {
  // Set canPlay to true
  canPlay = true;
  // Display an alert telling them sot('Can Play HTML5 video')
}
else {
  // Appe'Can Play HTML5 video'// Append Playr CSS and JS to the head of t
  // provide a fallbackTagName('head')[0];
  plink = document.createElement('link');
  plink.setAttribute('href', 'css/playr.css');
  plink.setAttribute('media', 'screen');
  h.appendChild(plink);
  pscript = document.createElement('script');
  pscript.setAttribute('src', 'js/playr.js');
  h.append('pscript');
}
'head'
```

This is the simplest test for video support; a more elaborate version can include support for specific formats and write the tags only for the supported formats. The example below makes the following assumptions:

- You have encoded a video in all three formats (webm, mp4 and ogg)
- You are testing for support for HTML5 video in general and specific formats
- If HTML5 video is not supported you have a flash-based fallback solution

```
var canPlay = false;
// Get the video by selecting the video tag
var v = document.getElementsByTagName('video');
'video'// Optionally add video attribtues as needed
// At a minimum set height, width and controls
// as shown belowttribute('height', '640');
v.setAttribute('width', '480');
v.setAttribute('control', 'control');

// If t'height'// If the video can playType and can play MP4 videoype &amp
  // append the a/no/'video/webm'// append the appropriate source trackrce
  webm.setAttribute("source", "myvideo.webm");
  webm.setAttribute("type", "video/webm");
}
else if (v.canPlayType &amp;&amp; v.canPlayType('video/mp4; codecs="avc1.4
  // append the appropriate sourc/no/"source"// append the appropriate sou
  var mp4 = v.appendChild(source);
  mp4.setAttribute("source", "myvideo.mp4");
  mp4.setAttribute("type", "'video/mp4; codecs="avc1.42E01E, mp4a.40.2"'"
}
```

Also note that we're testing for specific audio and video codec combinations. WebM supports a single combination of video and audio codecs but MP4 supports multiple profiles, not all of which are supported in HTML5 video. See What are the different profiles supported by MPEG-4 Video? for an introduction to the different profiles supported by MPEG4.

Players and how they interact with polyfills

Playr is by no means the only polyfil or the only player that supports VTT. It is the one that I found the most feature complete for what I needed. The selection below represents a set of players and polyfills available.

- video.js player
- jwplayer

- MediaElementJS player
- LeanBack player
- js_videosub polyfill
- Captionator polyfill
- vtt.js by the Mozilla Foundation

Different types of VTT tracks and their structures

Captioning Tracks

> Captioning is text that appears on a video, which contains dialogue and audio cues such as music or sound effects that occur off-screen. The purpose of captioning is to make video content accessible to those who are deaf or hard of hearing, and for other situations in which the audio cannot be heard due to noise or a need for silence.
>
> Captions can be either open (always visible, aka “burned in”) or closed, but closed is more common because it lets each viewer decide whether they want the captions to be turned on or off.
>
> From http://www.cpcweb.com/faq/

The simplest and most often used type of text track, captions provide alternative text content for people with visual disabilities, for people who choose to play the video without audio, and others.

Depending on the player you may have open captions, where the captions are always visible on screen, or closed captions where you have to manually activate the display of captions; Either open or closed, the captions are independent of the content they are attached to.

```
WEBVTT (1)

railroad (2)
```

```
00:00:10.000 --> 00:00:12.500 (3) [Optional Settings] (4)
Left uninspired by the crust of railroad earth (5)

manuscript
00:00:13.200 --> 00:00:16.900
that touched the lead to the pages of your manuscript.
```

Explanation of the cue above:

1. WEBVTT must be the first item on the file, on the first line and in a line of its own. Optionally there may be lines of metadata. This section must be followed by a blank line
2. The name of the cue. This is also optional
3. Immediately below the name of the cue come the beginning and end time for the cue expressed in hours:minutes:seconds:milliseconds format. **Hours, Minutes and Seconds must have 2 digits and be padded with zeros if necessary. Mil liseconds must have 3 digits and be zero padded if not long enough**
4. Optional Cue Settings separated from the time one or more SPACE or TAB characters
5. The text for the cue

Subtitles Tracks

Subtitle Tracks are similar to Caption Tracks but are not meant to address accessibility issues as Captions are. Subtitle tracks are used primarily to convey the dialogue in a language other than the one being spoken in the video. Take, for example a Japanese movie where the subtitles translate the content to English.

Subtitles are not expected to convey additional non-verbal cues. Once again, subtitles are only meant to provide a translation of the words being spoken although some delivery formats such as Blue Ray do not follow this recommendation.

What's the difference between captions and subtitles?

> The main difference is that subtitles usually only transcribe the spoken dialog, and are mainly aimed at people who are not hearing impaired, but lack fluency in the spoken language.

> Closed captions are aimed at the deaf and hearing impaired, who need additional non-verbal audio cues (such as “GUN SHOT” or “SPOOKY MUSIC”) to be transcribed in the text. Closed captions are also useful for situations in which video is being shown but the sound is muted or difficult to hear, such as for a noisy bar, convention floor, video signage & billboards, etc.
>
> From http://www.cpcweb.com/faq/

Other than the content for each type of track, HTML5 video structures the track element the same way. In the example below, the only difference are the kind attributes for each track.

```
<!-- This is the captions track -->
<track kind="captions" lang="en" sr<track kind="captions" lang="en" srclan
<track kind="subtitles" lang="es" srclang="es" label="Espa&ntilde;ol" src=
```

Chapter Tracks

Chapter tracks help you navigate through the video by associating certain “chapters” with time codes. This will let you navigate to different sections of your video using some sort of visual cue.

In the example below, chapter 1 is 10 second long and titled Introduction to HTML5.

```
WEBVTT (1)

Chapter 1 (2)
00:00:01.000 --> 00:00:10.000 (3)
Introduction to HTML5(4)

Chapter 2
00:00:10.001 --> 00:00:15.000
Introduction to HTML5
```

Explanation of the chapter above:

1. WEBVTT must be the first item on the file, on the first line and in a line of its own. It must be followed by a blank line
2. The name of the chapter
3. Immediately below the name of the chapter come the beginning and end time expressed in hours:minutes:seconds:milliseconds format. **Hours, Minutes and Seconds must have 2 digits and be padded with zeros if necessary. Milliseconds must have 3 digits and be zero padded if not long enough**
4. The title of the chapter

Description Tracks

Description tracks are used primarily as an asisstive technology helper, these tracks will be read by asisstive technology devices for people with visual disabilities (blind or low vission). The cues can be arbitrarily long as long as they don't contain empty lines (they would signal the beginning of a new cue)

```
VTT - Description for Sintel trailer

Sintel's Search -- beginning of the search
00:00:01.000 --> 00:00:52.000
Woman walks up a mountain
Fights an unknown man
Smoking man (covering full frame) speaks
Little dragon flies towards the woman before a larger dragon snatches it a
```

Metadata Tracks

Metadata Tracks are used to convey any additional information (such as base64 encoded images, JSON, additional text or any additional text-based file format) the developer needs to include in the page based on time indexes. A web app can listen for cue events, extract the text of each cue as it fires, parse the data and then use the results to make DOM changes (or perform other JavaScript or CSS tasks) synchronised with media playback.

```
WEBVTT - Example metadata track containing JSON payload

multiCell
```

```
00:01:15.200 --> 00:02:18.800
{
"title": "Multi-celled organisms",
"description": "Multi-celled organisms have different types of cells that
  Most life that can be seen with the naked eye is multi-cellular. These c
"src": "multiCell.jpg",
"href": "http://en.wikipedia.org/wiki/Multicellular"
}

insects
00:02:18.800 --> 00:03:01.600
{
"title": "Insects",
"description": "Insects are the most diverse group of animals on the plane
  number of current species range from two million to 50 million. The firs
  400 million years ago, identifiable by a hard exoskeleton, three-part bo
  and antennae.",
"src": "insects.jpg",
"href": "http://en.wikipedia.org/wiki/Insects"
}
```

We can then use Javascript to parse the track content and do something with the track's content.

```
textTrack.oncuechange = function (){
  // "this" is a textTrack
  var cue = this.activeCues[0]; // assuming there is only one active cue
  var obj = JSON.parse(cue.text);
  // do something
}
```

Getting the captions to work

We can build our caption file using the text above as an example, and this is the most common way to caption a video for accessibility.

> We can also build multiple caption tracks as well as a variety of other tracks.

Most polyfills will support a subset of the full VTT specification, Playr, the polyfill I've selected for these examples, supports captions, descriptions and chapter tracks.

Building the tracks

(built with information from http://thenewcode.com/584/Creating-And-Validating-WebVTT-Subtitles)

There are no programs that support VTT as a native captioning format. However there are plenty of programs that will create SRT captions, which is very similar to VTT (we'll discuss the differences later in this section).

Choose whatever tool will work best for you to generate the SRT file; then follow the instructions below to convert them to VTT files.

Converting SRT to VTT

Due to their close relationship, conversion from .srt into .vtt is very simple. A typical .srt file will look something like this:

```
1
00:01:21,700 --> 00:01:24,675
Life on the road is something
I was raised to embrace.</pre>
```

The process is little more than a find-and-replace:

1. Add WEBVTT to the first line of the file
2. Convert the comma before the millisecond mark in every timestamp to a decimal point
3. Add styling markup to the subtitle text if needed

   - Special characters must be escaped as in HTML (&, <, >)
   - You can use CSS classes defined in your CSS file by using >c.XXX<

4. See the section Cue Payload Tags for more information about the specific tags you can use to style your content

The resulting VTT file will look like this:

```
WEBVTT

Life
01:21.700 --> 01:24.675
Life on the road is something
I was <i>raised</i> to embrace.</pre>
```

Save the file with a .vtt extension and link to it from a element in your video.

Validating A VTT File

It is not hard to make mistakes when creating a VTT track fille. Fortunately there is an online validator to help with authoring.

It is essentially a two step process:

- Paste the text of your VTT file
- Select the type of track you're working on

The results will display automatically.

Optional Cue Settings

Cues can also be styled and moved around the screen relative to the borders of the video. The table below summarizes the settings avalable for cues.

Vertical Alignment

Name: vertical

Values: rl (right to left) - lr (left to right)

What is used for: Vertical text alignment for languages that can be read from top to bottom

Example: vertical:lr (makes the cue display vertically from left to right)

Line Placement / Top Alignment

Name: line

Value [-][0 or larger] (negative or possitive number) or [0-100]%

What is used for: Absolute references to a particular line number the cue is to be displayed on.

What is used for: Percentage value indicating the position relative to the top of the frame (when using percentages)

- Line numbers are based on the size of the first line of the cue.
- A negative number counts from the bottom of the frame* Positive numbers from the top

Cue Box Size

Name: size

Value: [0-100]%

What it's used for: Indicates the size of the cue box. The value is given as a percentage of the width of the frame

Text Align

Name: align

Values: start | middle | end

What it's used for: Specifies the alignment of the text within the cue. The keywords are relative to the text direction and are the same alignment keywords used in SVG

The alignment values are similar to those used in SVG. For users of CSS that uses a different terminology, the equivalency is:

- Start alignment: The cue box's left side (for horizontal cues) or top side (otherwise) is aligned at the text position.
- Middle alignment: The cue box is centered at the text position.
- End alignment: The cue box's right side (for horizontal cues) or bottom side (otherwise) is aligned at the text position.

Note: if no cue settings are set, the positioning default to the middle, at the bottom of the frame.

Cue positioning

Name: position

Value [0-100]%

What is used for:

Percentage value indicating the horizontal alignment relative to the edge of the frame where the text begins (e.g. the left edge in English)

The value is dependent on the alignment of the cue:

- For left aligned or start aligned cues: 0%.
- For middle aligned cues: 50%.
- For right aligned or end aligned cues: 100%.

Note: Since the default value of the text track cue text alignment is middle, if there is no text track cue text alignment setting for a cue, the text track cue text position defaults to 50%.

Note: Even for horizontal cues with right-to-left paragraph direction text, the cue box is positioned from the left edge of the video frame. This allows defining a rendering space template which can be filled with either left-to-right or right-to-left paragraph direction text. If you define such a cue box template with start or end aligned text, make sure to control its size unless you want text to flip from one side of the video frame to the other.

Cue Payload Tags

These are additional tracks that will allow you to customize the appearance of your tracks. ‘’‘You cannot use payload tags with chapter tracks’’’

Timestamp Tags (Karaoke Style and Paint On Caption Text)

Using timestamp tags can build Karaoke Style tracks. You build the track by inserting the correct time stamp where you want the text to change, subject to the following restrictions:

- The timestamp must be greater that the cue’s start timestamp, greater than any previous timestamp in the cue payload, and less than the cue’s end timestamp.

```
VTT - Example Karaoke Style Track

1
00:16.500 --> 00:18.500
When the moon <00:17.500>hits your eye

2
00:00:18.500 --> 00:00:20.500
Like a <00:19.000>big-a <00:19.500>pizza <00:20.000>pie

3
00:00:20.500 --> 00:00:21.500
That's <00:00:21.000>amore</pre>
```

In the example above:

- The active text is the text between the timestamp and the next timestamp or to the end of the payload if there is not another timestamp in the payload.
- Any text before the active text in the payload is previous text .
- Any text beyond the active text is future text. We can use the previous and future tracks to create the Karaoke experience.

A possible CSS rule to style the content looks like this.

```
::cue:past {
  color:yellow
}

::cue:future {
  text-shadow: black 0 0 1px;
}
```

Timestamp tags can also be used for Paint On captions, which placed independently from each other and don’t erase what was already on the screen. They are written one letter at a time and they appear to ‘paint on’ the screen.

Speaker Semantics

You can use a combination of cue positioning and specific markup on individual cues to further emphazise who is speaking in a given caption or subtitle where appropriate.

```
WEBVTT - Sintel Caption File With Speaker Semantics

Sage
00:00:12.000 --> 00:00:15.000 A:middle T:10%
<v.gatekeeper>What brings you to the land of the gatekeepers?

Searching
00:00:18.500 --> 00:00:20.500 A:middle T:80%
<v.sintel>I'm searching for someone.
```

We can style the speaker semantic classes using CSS. For example we can add a different color for each speaker, something like the example below:

```
video::cue(v.gatekeeper) {
  color:lime;
}

video::cue(v.sintel) {
  color: #ff00ff;
}
```

Addtional Style tags

The following tags require opening and closing tags.

Class tag

Style the contained text using a CSS class.

```
Cue 14 - Class tag example
<c.classname>text</c>
```

Italics tag

Italicize the contained text.

```
Example 15 - Italics tag
<i>text</i>
```

Bold tag

Bold the contained text.

```
Example 16 - Bold tag
<b>text</b>
```

Underline tag

Underline the contained text.

```
Example 17 - Underline tag
<u>text</u>
```

Ruby tag / Ruby text tag

Used together to display ruby characters (i.e. small annotative characters above other characters). Ruby annotations are primarily used in languages with logographic alphabets (Japanese, Chinese, Korean) where a single character may represent a complete word and where the meaning of the character may not be familiar to the reader.

> Ruby characters are small, annotative glosses that can be placed above or to the right of a Chinese character when writing languages with logographic characters such as Chinese or Japanese to show the pronunciation. Typically called just ruby or rubi, such annotations are used as pronunciation guides for characters that are likely to be unfamiliar to the reader.

> From Wikipedia

```
Example 18 - Ruby tag and Ruby text tag
<ruby>WWW<rt>World Wide Web</rt>oui<rt>yes</rt></ruby>
```

Adding the tracks to the video

Either in a supported browser or using one of the polyfills available (like we've chosen to do with Playr) we add `<track>` elements, one for each language of captions that we make available.

There is one non-standard attribute we will add to the video to make it work with Playr. The code below shows what a video track looks with associated an associated caption track for English.

```
<!DOCTYPE html>
<html>
<head>
 <html>e>Sample Captioned Video</title>
  <script src="playr.js"></s</title> <link rel="stylesheet" href="playr.js
</head>
<body>
<video
  id="myvideo"
  controls="controls"
<link rel="stylesheet" href="playr.js">ght="480"
  poster="http://media.w3.org/2010/05/sintel/poster.png"
>
<!--
  These are the three sources. This should cover most of our
  deployed player base
-->
<source src="http://media.w3.org/2010/05/sintel/trailer.mp4" type="video/m
<source src="http://media.w3.org/2010/05/sintel/trailer.webm" type="video/
<source src="http://media.w3.org/2010/05/sintel/trailer.ogv" type="video/o
<source src="http://media.w3.org/2010/05/sintel/trailer.mp4" type="video/m
  This is the captions track
```

```
-->
<track kind="captions" lang="en" srclang="en" label="English" src="sintel
</video>
</body>
</html>
```

The working example is located at http://labs.rivendellweb.net/vtt-demos/basic.html and an example without a polyfill (meant to test native browser support) is located at http://labs.rivendellweb.net/vtt-demos/basic-plain.html

The same example without polyfill support and supporting captions in English and Spanish with the English caption being the default. The default attribute will also display the captions automatically

```
<!DOCTYPE html>
<html>
<head>
 <html>e>Sample Captioned Video</title>
</head>
</head>
<body>
<vide</title>myvideo"
  controls="controls"
  width="640" height="480"
  poster="http://media.w3.org/2010/05/sintel/poster.png"
>
<!--
  These are the three sources. This should cover most of our
  deployed player base
-->
<source src="http://media.w3.org/2010/05/sintel/trailer.mp4" type="video/
<source src="http://media.w3.org/2010/05/sintel/trailer.webm" type="video
<source src="http://media.w3.org/2010/05/sintel/trailer.ogv" type="video/
<source src="http://media.w3.org/2010/05/sintel/trailer.mp4" type="video/
  This is the captions track
-->
<track kind="captions" lang="en" srclang="en" label="English" src="sintel
<track kind="captions" lang="es" srclang="es" label="Spanish" src="sintel
```

```
</video>
</body>
</html>
```

The final example contains multiple caption tracks, subtitles in Spanish and descriptions for the video.

```
<!DOCTYPE html>
<html>
<head>
 <html>e>Sample Captioned Video</title>
</head>
</head>
<body>
<vide</title>myvideo"
  controls="controls"
  width="640" height="480"
  poster="http://media.w3.org/2010/05/sintel/poster.png"
>
<!--
  These are the three sources. This should cover most of our
  deployed player base
-->
<source src="http://media.w3.org/2010/05/sintel/trailer.mp4" type="video/
<source src="http://media.w3.org/2010/05/sintel/trailer.webm" type="video
<source src="http://media.w3.org/2010/05/sintel/trailer.ogv" type="video/
<source src="http://media.w3.org/2010/05/sintel/trailer.mp4" type="video/
  These are the captions track
-->t" default />
<track kind="captions" lang="es" srclang="es" label="Spanish" src="sintel
<track kind="captions" lang="de" srclang="de" label="Spanish" src="sintel
<!--
  This is the subtitles track
-->
<track kind="subtitles" lang="es" srclang<track kind="captions" lang="es"
  This is the subtitles track
-->rack kind="captions" lang="en" srclang="en" label="English" src="sinte
```

```
</body>
</html>
</body><!--
  These are the description tracks
-->
<track kind="captions" lang="en" srclang="en" label="English" src="sintel-
</body>
</html>
```

Text tracks and audio

Text tracks are not limited to working with just video. They work just the same with audio. The example below (taken from http://mattcrouch.net/experiments/music-sync/) provides synchronized captions to an audio track.

Using jQuery, an extract of the audio for the Sintel video and the same captions that we used for the video examples, we change the cues programmatically using the video API to display the cues at the matching time.

As you can see, description tracks would be particularly useful in this case as they would provide a more complete context to the audio.

```
jQuery(document).ready(function() {
// Step below is optional. I don't like taking
// the option from the user and autoplay the video
$('audio').trigger("play");
var audio = document.querySelector("audio");
// log the name of the track we're working with
console.log(audio.textTracks[0]);

audio.textTracks[0].oncuechange = function (){
  $("#output").html(""); // Clear the content of our output region
    if(this.activeCues !== null) {
      for(var i=0;i<this.activeCues.length;i++) {
        if(this.activeCues[i] !== undefined) {
          $("#output").append(this.activeCues[i].text+"<br>");
        }
```

```
      }
    }
  }
});
```

Additional Tutorials And Tools

- Opera: An introduction to webvtt and track
- Mozilla Developer Docs: WebVTT
- MSDN: HTML5 Video Captioning
- http://www.delphiki.com/webvtt/
- Creating and validating WebVTT Subtitles
- http://www.accessiq.org/news/features/2013/03/webvtt-and-captioning-on-the-web
- Test Drive Video Captions
- http://html5labs.interoperabilitybridges.com/prototypes/video-captioning/video-captioning/info
- Microsoft Caption Maker
- Timed Text Track Information
- Timed Text Tracks examples