Web Video Text Tracks Format (WebVTT)

Web Video Text Tracks Format (**WebVTT**) is a format for displaying timed text tracks (such as subtitles or captions) using the `<track>` element. The primary purpose of WebVTT files is to add text overlays to a `<video>`. WebVTT is a text based format, which must be encoded using UTF-8. Where you can use spaces you can also use tabs. There is also a small API available to represent and manage these tracks and the data needed to perform the playback of the text at the correct times.

WebVTT files

The MIME type of WebVTT is `text/vtt`.

A WebVTT file (`.vtt`) contains cues, which can be either a single line or multiple lines, as shown below:

```
WEBVTT

00:01.000 --> 00:04.000
- Never drink liquid nitrogen.

00:05.000 --> 00:09.000
- It will perforate your stomach.
- You could die.
```

WebVTT body

The structure of a WebVTT consists of the following components, some of them optional, in this order:

- An optional byte order mark (BOM).
- The string " `WEBVTT` ".

- An optional text header to the right of `WEBVTT`.
    - There must be at least one space after `WEBVTT`.
    - You could use this to add a description to the file.
    - You may use anything in the text header except newlines or the string "`-->`".
- A blank line, which is equivalent to two consecutive newlines.
- Zero or more cues or comments.
- Zero or more blank lines.

Examples

- Simplest possible WebVTT file

```
WEBVTT
```

- Very simple WebVTT file with a text header

```
WEBVTT - This file has no cues.
```

- Common WebVTT example with a header and cues

```
WEBVTT - This file has cues.

14
00:01:14.815 --> 00:01:18.114
- What?
- Where are we now?

15
00:01:18.171 --> 00:01:20.991
- This is big bat country.

16
00:01:21.058 --> 00:01:23.868
- [ Bats Screeching ]
- They won't get in your hair. They're after the bugs.
```

Inner structure of a WebVTT file

Let's re-examine one of our previous examples, and look at the cue structure in a bit more detail.

```
WEBVTT

00:01.000 --> 00:04.000
- Never drink liquid nitrogen.

00:05.000 --> 00:09.000
- It will perforate your stomach.
- You could die.
```

In the case of each cue:

- The first line is started with a time, which is the starting time for showing the text that appears underneath.
- On the same line, we then have a string of " `-->` ".
- We finish the first line with a second time, which is the ending time for showing the associated text.
- We can then have one or more lines that start with a hyphen (-), each containing part of the text track to be shown.

We can also place comments in our `.vtt` file, to help us remember important information about the parts of our file. These should be on separate lines, starting with the string `NOTE`. You'll find more about these in the below section.

It is important to not use "extra" blank lines within a cue, for example between the timings line and the cue payload. WebVTT is line based; a blank line will close the cue.

WebVTT comments

Comments are an optional component that can be used to add information to a WebVTT file. Comments are intended for those reading the file and are not seen by users. Comments may contain newlines but cannot contain a blank line, which is equivalent to two consecutive newlines. A blank line signifies the end of a comment.

A comment cannot contain the string `-->`, the ampersand character (`&`), or the less-than sign (`<`). If you wish to use such characters, you need to escape them using for example `&amp;` for ampersand and `&lt;` for less-than. It is also recommended that you use the greater-than escape sequence (`&gt;`) instead of the greater-than character (`>`) to avoid confusion with tags.

A comment consists of three parts:

- The string `NOTE`.
- A space or a newline.
- Zero or more characters other than those noted above.

Examples

- Common WebVTT example

```
NOTE This is a comment
```

- Multi-line comment

```
NOTE
One comment that is spanning
more than one line.

NOTE You can also make a comment
across more than one line this way.
```

- Common comment usage

```
WEBVTT - Translation of that film I like

NOTE
This translation was done by Kyle so that
some friends can watch it with their parents.

1
00:02:15.000 --> 00:02:20.000
- Ta en kopp varmt te.
- Det är inte varmt.

2
00:02:20.000 --> 00:02:25.000
- Har en kopp te.
- Det smakar som te.

NOTE This last line may not translate well.

3
00:02:25.000 --> 00:02:30.000
- Ta en kopp
```

Styling WebVTT cues

You can style WebVTT cues by looking for elements which match the `::cue` pseudo-element.

Within site CSS

```
video::cue {
  background-image: linear-gradient(to bottom, dimgray, lightgray);
  color: papayawhip;
}

video::cue(b) {
  color: peachpuff;
}
```

Here, all video elements are styled to use a gray linear gradient as their backgrounds, with a foreground color of `"papayawhip"`. In addition, text boldfaced using the `<b>` element are colored `"peachpuff"`.

The HTML snippet below actually handles displaying the media itself.

```
<video controls autoplay src="video.webm">
  <track default src="track.vtt" />
</video>
```

Within the WebVTT file itself

You can also define the style directly in the WebVTT file. In this case, you insert your CSS rules into the file with each rule preceded by the string `"STYLE"` all by itself on a line of text, as shown below:

```
WEBVTT

STYLE
::cue {
  background-image: linear-gradient(to bottom, dimgray, lightgray);
  color: papayawhip;
}
/* Style blocks cannot use blank lines nor "dash dash greater than" */

NOTE comment blocks can be used between style blocks.

STYLE
::cue(b) {
  color: peachpuff;
}

00:00:00.000 --> 00:00:10.000
- Hello <b>world</b>.

NOTE style blocks cannot appear after the first cue.
```

We can also use identifiers inside WebVTT file, which can be used for defining a new style for some particular cues in the file. The example where we wanted the transcription text to be red highlighted and the other part to remain normal, we can define it as follows using CSS. Where it must be noted that the CSS uses escape sequences the way they are used in HTML pages:

```
WEBVTT

1
00:00.000 --> 00:02.000
That's an, an, that's an L!

crédit de transcription
00:04.000 --> 00:05.000
Transcrit par Célestes™
```

```
::cue(#\31) {
  color: lime;
}
::cue(#crédit\ de\ transcription) {
  color: red;
}
```

Positioning of text tracks is also supported, by including positioning information after the timings in a cue, as seen below (see Cue settings for more information):

```
WEBVTT

00:00:00.000 --> 00:00:04.000 position:10%,line-left align:left size:35%
Where did he go?

00:00:03.000 --> 00:00:06.500 position:90% align:right size:35%
I think he went down this lane.

00:00:04.000 --> 00:00:06.500 position:45%,line-right align:center size:35%
What are you waiting for?
```

WebVTT cues

A cue is a single subtitle block that has a single start time, end time, and textual payload. A cue consists of five components:

- An optional cue identifier followed by a newline.
- Cue timings.
- Optional cue settings with at least one space before the first and between each setting.
- A single newline.
- The cue payload text.

Here is an example of a cue:

```
1 - Title Crawl
00:00:05.000 --> 00:00:10.000 line:0 position:20% size:60% align:start
Some time ago in a place rather distant....
```

Cue identifier

The identifier is a name that identifies the cue. It can be used to reference the cue from a script. It must not contain a newline and cannot contain the string " `-->` ". It must end with a single new line. They do not have to be unique, although it is common to number them (e.g., 1, 2, 3).

Here are a few examples:

- A basic cue identifier

  ```
  1 - Title Crawl
  ```

- Common usage of identifiers

```
WEBVTT

1
00:00:22.230 --> 00:00:24.606
This is the first subtitle.

2
00:00:30.739 --> 00:00:34.074
This is the second.

3
00:00:34.159 --> 00:00:35.743
Third
```

Cue timings

A cue timing indicates when the cue is shown. It has a start and end time which are represented by timestamps. The end time must be greater than the start time, and the start time must be greater than or equal to all previous start times. Cues may have overlapping timings.

If the WebVTT file is being used for chapters (`<track>` `kind` is `chapters`) then the file cannot have overlapping timings.

Each cue timing contains five components:

- Timestamp for start time.
- At least one space.
- The string " `-->` ".
- At least one space.
- Timestamp for end time, which must be greater than the start time.

The timestamps must be in one of two formats:

- `mm:ss.ttt`

- `hh:mm:ss.ttt`

Where the components are defined as follows:

`hh`

Represents hours and must be at least two digits. It can be greater than two digits (e.g., `9999:00:00.000`).

`mm`

Represents minutes and must be between 00 and 59, inclusive.

`ss`

Represents seconds and must be between 00 and 59, inclusive.

`ttt`

Represents milliseconds and must be between 000 and 999, inclusive.

Here are a few cue timing examples:

- Basic cue timing examples

```
00:00:22.230 --> 00:00:24.606
00:00:30.739 --> 00:00:34.074
00:00:34.159 --> 00:00:35.743
00:00:35.827 --> 00:00:40.122
```

- Overlapping cue timing examples

```
00:00:00.000 --> 00:00:10.000
00:00:05.000 --> 00:01:00.000
00:00:30.000 --> 00:00:50.000
```

- Non-overlapping cue timing examples

```
00:00:00.000 --> 00:00:10.000
00:00:10.000 --> 00:01:00.581
00:01:00.581 --> 00:02:00.100
00:02:01.000 --> 00:02:01.000
```

Cue settings

Cue settings are optional components used to position where the cue payload text will be displayed over the video. This includes whether the text is displayed horizontally or vertically. There can be zero or more of them, and they can be used in any order so long as each setting is used no more than once.

The cue settings are added to the right of the cue timings. There must be one or more spaces between the cue timing and the first setting and between each setting. A setting's name and value are separated by a colon. The settings are case sensitive so use lower case as shown. There are five cue settings:

`vertical`

Indicates that the text will be displayed vertically rather than horizontally, such as in some Asian languages. There are two possible values:

`rl`

The writing direction is right to left

`lr`

The writing direction is left to right

`line`

If vertical is not set, specifies where the text appears vertically. If vertical is set, line specifies where text appears horizontally. Its value can be:

a line number

The number is the height of the first line of the cue as it appears on the video. Positive numbers indicate top down and negative numbers indicate bottom up.

a percentage

It must be an integer (i.e., no decimals) between 0 and 100 inclusive and must be followed by a percent sign (%).

ine	`vertical` omitted	`vertical:rl`	`vertical:lr`
`ine:0`	top	right	left
`ine:-1`	bottom	left	right
`ine:0%`	top	right	left
`ine:100%`	bottom	left	right

`position`

Specifies where the text will appear horizontally. If vertical is set, position specifies where the text will appear vertically. The value is a percentage, that is an integer (no decimals) between 0 and 100 inclusive followed by a percent sign (%).

osition	`vertical` omitted	`vertical:rl`	`vertical:lr`
`osition:0%`	left	top	top
`osition:100%`	right	bottom	bottom

`size`

Specifies the width of the text area. If vertical is set, size specifies the height of the text area. The value is a percentage, that is an integer (no decimals) between 0 and 100 inclusive followed by a percent sign (%).

ize	`vertical` omitted	`vertical:rl`	`vertical:lr`
`ize:100%`	full width	full height	full height

ize	`vertical` omitted	`vertical:rl`	`vertical:lr`
`ize:50%`	half width	half height	half height

`align`

Specifies the alignment of the text. Text is aligned within the space given by the size cue setting if it is set.

lign	`vertical` omitted	`vertical:rl`	`vertical:lr`
`lign:start`	left	top	top
`lign:center`	centered horizontally	centered vertically	centered vertically
`lign:end`	right	bottom	bottom

Let's study an example of cue setting.

The first line demonstrates no settings. The second line might be used to overlay text on a sign or label. The third line might be used for a title. The last line might be used for an Asian language.

```
00:00:05.000 --> 00:00:10.000
00:00:05.000 --> 00:00:10.000 line:63% position:72% align:start
00:00:05.000 --> 00:00:10.000 line:0 position:20% size:60% align:start
00:00:05.000 --> 00:00:10.000 vertical:rt line:-1 align:end
```

Cue payload

The payload is where the main information or content is located. In normal usage the payload contains the subtitles to be displayed. The payload text may contain newlines but it cannot contain a blank line, which is equivalent to two consecutive newlines. A blank line signifies the end of a cue.

A cue text payload cannot contain the string `-->`, the ampersand character (`&`), or the less-than sign (`<`). Instead use the escape sequence `&amp;` for ampersand and `&lt;` for less-than. It is also recommended that you use the greater-than escape sequence `&gt;` instead of the greater-than character (>) to avoid confusion with tags. If you are using the WebVTT file for metadata these restrictions do not apply.

In addition to the three escape sequences mentioned above, there are fours others. They are listed in the table below.

ne	Character	Escape sequence
ɔersand	`&`	`&amp;`
s-than	`<`	`&lt;`
ater-than	`>`	`&gt;`
:-to-right mark	*none*	`&lrm;`
ıt-to-left mark	*none*	`&rlm;`
ı-breaking space		`&nbsp;`

Cue payload text tags

There are a number of tags, such as `<b>`, that can be used. However, if the WebVTT file is used in a `<track>` element where the attribute `kind` is `chapters` then you cannot use tags.

Timestamp tag

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. 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*. This enables karaoke style captions.

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

```
1
00:00:18.500 --> 00:00:20.500
Like a <00:19.000>big-a <00:19.500>pizza <00:20.000>pie

1
00:00:20.500 --> 00:00:21.500
That's <00:00:21.000>amore
```

The following tags are the HTML tags allowed in a cue and require opening and closing tags (e.g., `<b>text</b>`).

Class tag (`<c></c>`)

Style the contained text using a CSS class.

```
<c.classname>text</c>
```

Italics tag (`<i></i>`)

Italicize the contained text.

```
<i>text</i>
```

Bold tag (`<b></b>`)

Bold the contained text.

```
<b>text</b>
```

Underline tag (`<u></u>`)

Underline the contained text.

```
<u>text</u>
```

Ruby tag (`<ruby></ruby>`)

Used with ruby text tags to display ruby characters (i.e., small annotative characters above other characters).

```
<ruby>WWW<rt>World Wide Web</rt>oui<rt>yes</rt></ruby>
```

Ruby text tag (`<rt></rt>`)

Used with ruby tags to display ruby characters (i.e., small annotative characters above other characters).

```
<ruby>WWW<rt>World Wide Web</rt>oui<rt>yes</rt></ruby>
```

Voice tag (`<v></v>`)

Similar to class tag, also used to style the contained text using CSS.

```
<v Bob>text</v>
```

Instance methods and properties

The methods used in WebVTT are those which are used to alter the cue or region as the attributes for both interfaces are different. We can categorize them for better understanding relating to each interface in WebVTT:

VTTCue

The methods which are available in the `VTTCue` interface are:

- `getCueAsHTML()` to get the HTML of that cue.
- A constructor, `VTTCue()` for creating new instances of this interface.

Different properties allowing to read and set the characteristics of the cue, like its position, alignment or size are also available. Check `VTTCue` for a complete list.

VTTRegion

The `VTTRegion` provides methods used for region are listed below along with description of their functionality, especially it allows to adjust the scrolling setting of all nodes present in the given region.

Tutorial on how to write a WebVTT file

There are few steps that can be followed to write a simple webVTT file. Before start, it must be noted that you can make use of a notepad and then save the file as '.vtt' file. Steps are given below:

- Open a notepad.
- The first line of WebVTT is standardized similar to the way some other languages require you to put headers as the file starts to indicate the file type. On the very first line you have to write:

```
WEBVTT
```

- Leave the second line blank, and on the third line the time for first cue is to be specified. For example, for a first cue starting at time 1 second and ending at 5 seconds, it is written as:

```
00:01.000 --> 00:05.000
```

- On the next line you can write the caption for this cue which will run from the first second to the fifth second, inclusive.
- Following the similar steps, a complete WebVTT file for specific video or audio file can be made.

CSS pseudo-classes

CSS pseudo classes allow us to classify the type of object which we want to differentiate from other types of objects. It works in similar manner in WebVTT files as it works in HTML file.

It is one of the good features supported by WebVTT is the localization and use of class elements which can be used in same way they are used in HTML and CSS to classify the style for particular type of objects, but here these are used for styling and classifying the Cues as shown below:

```
WEBVTT

04:02.500 --> 04:05.000
J'ai commencé le basket à l'âge de 13, 14 ans

04:05.001 --> 04:07.800
Sur les <i.foreignphrase><lang en>playground</lang></i>, ici à Montpellier
```

In the above example it can be observed that we can use the identifier and pseudo class name for defining the language of caption, where `<i>` tag is for italics.

The type of pseudo class is determined by the selector it is using and working is similar in nature as it works in HTML. Following CSS pseudo classes can be used:

- `lang` (Language): e.g., `p:lang(it)`.
- `link`: e.g., `a:link`.
- `nth-last-child`: e.g., `p:nth-last-child(2)`.
- `nth-child(n)`: e.g., `p:nth-child(2)`.

Where p and a are the tags which are used in HTML for paragraph and link, respectively and they can be replaced by identifiers which are used for Cues in WebVTT file.

Specifications

Specification
WebVTT: The Web Video Text Tracks Format # the-vttcue-interface

Specification

HTML Standard

# texttrack

WebVTT: The Web Video Text Tracks Format

# the-vttregion-interface

Browser compatibility

api.VTTCue

Report problems with this compatibility data on GitHub

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android
`VTTCue`	23	12	31	12.1	6	25	31	12.1
`VTTCue()` constructor	33	79	31	20	8	33	31	20
`align`	23	79	31	12.1	6	25	31	12.1
`getCueAsHTML`	23	12	31	12.1	6	25	31	12.1
`line`	23	79	31	12.1	6	25	31	12.1
`lineAlign`	No	No	31	No	14.1	No	31	No
`position`	23	79	31	12.1	6	25	31	12.1
`positionAlign`	No	No	31	No	14.1	No	31	No
`region`	No	No	59	No	14.1	No	59	No

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android
`size`	23	79	31	12.1	6	25	31	12.1
`snapToLines`	23	79	31	12.1	6	25	31	12.1
`text`	23	12	31	12.1	6	25	31	12.1
`vertical`	23	79	31	15	6	25	31	14

Tip: you can click/tap on a cell for more information.

Full support No support See implementation notes.

api.TextTrack

Report problems with this compatibility data on GitHub

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android
`TextTrack`	23	12	31	12.1	6	25
`activeCues`	23	12	31	12.1	6	25
`addCue`	23	12	31	12.1	6	25

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android
`cuechange` event	23	12	31	12.1	6	25
`cues`	23	12	31	12.1	6	25
`id`	33	18	31	20	8	33
`inBandMetadataTrackDispatchType`	No	12–18	31	No	8	No
`kind`	23	12	31	12.1	6	25
`label`	23	12	31	12.1	6	25
`language`	23	12	31	12.1	6	25
`mode`	23	12	31	12.1	6	25
`removeCue`	23	12	31	12.1	6	25
`sourceBuffer`	No	No	No	No	8	No

Tip: you can click/tap on a cell for more information.

Full support Partial support No support See implementation notes.

api.VTTRegion

Report problems with this compatibility data on GitHub

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet
`VTTRegion`	No	No	59	No	7	No	59	No	7	N
`VTTRegion()` constructor	No	No	59	No	7	No	59	No	7	N
`id`	No	No	59	No	7	No	59	No	7	N
`lines`	No	No	59	No	12.1	No	59	No	12.2	N
`regionAnchorX`	No	No	59	No	7	No	59	No	7	N
`regionAnchorY`	No	No	59	No	7	No	59	No	7	N
`scroll`	No	No	59	No	7	No	59	No	7	N
`viewportAnchorX`	No	No	59	No	7	No	59	No	7	N
`viewportAnchorY`	No	No	59	No	7	No	59	No	7	N
`width`	No	No	59	No	7	No	59	No	7	N

Tip: you can click/tap on a cell for more information.

Full support   No support

Notes

Prior to Firefox 50, the `AlignSetting` enum (representing possible values for `VTTCue.align`) incorrectly included the value `"middle"` instead of `"center"`. This has been corrected.

WebVTT was implemented in Firefox 24 behind the preference `media.webvtt.enabled`, which is disabled by default; you can enable it by setting this preference to `true`. WebVTT is enabled by default starting in Firefox 31 and can be disabled by setting the preference to `false`.

Prior to Firefox 58, the `REGION` keyword was creating `VTTRegion` objects, but they were not being used. Firefox 58 now fully supports `VTTRegion` and its use; however, this feature is disabled by default behind the preference `media.webvtt.regions.enabled`; set it to `true` to enable region support in Firefox 58. Regions are enabled by default starting in Firefox 59 (see bugs Firefox bug 1338030 and Firefox bug 1415805).

See also

- The CSS `::cue` and `::cue()` pseudo-elements