

Inline WebVTT Styling

Author: evliu@ 08-07-2019

Last updated: 09-20-2019

One-page overview

Summary

The [W3C WebVTT specification](#) allows the ability to embed CSS style sheets within the STYLE block of a WebVTT file. This feature is supported by several other web browsers and video players including Safari and VLC but not Chrome. To ensure a consistent experience across platforms, Chrome should conform to the WebVTT specification by supporting inline styling.

This project is part of a larger effort to improve Accessibility around the Media Experience in Chrome.

Platforms

All platforms supported by Chrome

Team

evliu@google.com

Feature bug

<https://crbug.com/991729>

Code affected

- Text track classes (VTT parser, VTT cue, VTT element, text track, text track loader)
- CSS classes (style resolver, element rule collector, style sheet, selector checker, style engine, CSS property helper, CSS parser context)

Design

WebVTT file parsing

The WebVTT file will be parsed in accordance with the algorithm outlined in the [parsing section](#) of the WebVTT specification. The changes to [vtt_parser.cc](#) will be analogous to the parsing of the REGION sections of a WebVTT file. The specification allows the definition of multiple STYLE blocks in a single WebVTT file. Each STYLE block will be parsed into a style sheet and added to a collection of style sheets that will be associated to a [TextTrack](#). A reference to the text track containing the style sheets will be added to the [VTTElement](#) as well as the background box element.

Rule matching

When collecting rules for matching in the [ElementRuleCollector](#), the ruleset from the stylesheet on the element will also be added for elements that can be targeted by WebVTT CSS selectors. This includes [VTTElements](#) as well as the outer div element that is targeted by the ::cue pseudo-element without arguments.

Style scoping

The style sheets from the WebVTT files are only associated to the cues generated by the same WebVTT file, so they can't apply to elements outside of the text track container or to other text tracks. To support dynamic media queries, the style must be refreshed when values affecting media queries are changed. To accomplish this, a collection of references to text tracks will be added to the [StyleEngine](#). When a value affecting media queries is changed, the rulesets from the style sheets on text tracks that contain media queries are cleared and the style for the document is recalculated.

Selector checking

According to the [WebVTT specifications](#), the CSS from the inline STYLE blocks should apply to a hypothetical document with a single, featureless element acting as the [originating element](#) for the cues. A flag will be added to the [CSSStyleSheet](#) that will indicate whether the style sheet was parsed from a WebVTT file. When matching rules that originated from a WebVTT STYLE block, the selector checker will not match selectors with combinators.

Security

Supporting external resources with '@import' or 'background-image' would be a new ability for media elements and <track> elements to issue network requests as the user watches

the video, which could be a privacy issue. To prevent this, '@import' rules will be disabled for the style sheets parsed from WebVTT files and external background images will only be [consumed](#) if the URL's scheme is "data".

Test cases

Web Platform Tests will be added to validate that the behavior is according to specifications. Because the cue styles can't be retrieved via Javascript, [reftests](#) will be used to compare screenshots of the actual rendered captions to the expected rendering.

1. Verify that rules from author style sheets still apply to videos.
2. Verify that the rules from VTT style sheets are applied after author styles sheets.
3. Verify that rules from VTT style sheets do not apply to other elements in the document.
4. Verify that rules from VTT style sheets do not apply to other text tracks on the same video from different text tracks.
5. Verify that the following selectors work:
 - a. *|*::cue(b)
 - b. *::cue(b)
 - c. |*::cue(b)
 - d. ::cue(b)
6. Verify that the following selectors do not work:
 - a. *
 - b. b
 - c. *::cue(b)
 - d. * > *::cue(b)
 - e. * + *::cue(b)
 - f. video::cue(b)
 - g. html|*::cue(b)
7. Verify that ::cue with no arguments works.
8. Verify that ::cue() works.
9. Verify that style sheets can't be imported via @import.
10. Verify that images can't be imported from an external URL.
11. Verify that images can be defined using a URL with a "data" scheme.
12. Verify that STYLE blocks are parsed correctly and malformed blocks are not parsed.