W3C Media & Entertainment Interest Group
Media Capabilities

6 February 2024

Agenda

Date and time
6 February 2024, 15:00-16:00 UTC

IRC
https://irc.w3.org/?channels=#me

IRC Guide
https://www.w3.org/wiki/IRC

Code of Conduct
https://www.w3.org/Consortium/cepc/

Agenda

- Welcome
- Media Capabilities
  - Introduction and current status
  - Text track capabilities
  - Multiple stream decoding
  - Transitions between codec configurations
  - Decoding vs rendering capabilities
  - Other priorities?
- AOB

Media Capabilities API

- Scope includes:
  - Media decoding/playback and encoding/transmission
  - File-based, MSE-based playback, and WebRTC
  - Rendering capabilities (mostly) not in scope
- Returns capability information:
  - supported - can the format be decoded/encoded at all?
  - smooth - can the format be played smoothly (excluding network considerations)?
  - powerEfficient - can the format be decoded/encoded in a way that is power efficient (may mean hardware accelerated in some cases)?
- We'll focus today on file and MSE based decoding/playback, not encoding, and not WebRTC

Media Capabilities API

- Current status:
  - Implemented in Chrome, Safari, Firefox
  - Specification is a W3C Working Draft
  - WG is reviewing and prioritising issues, but has limited capacity
- Open questions for discussion today - with focus on prioritisation and items where help would be welcome:
  - Text track capabilities
  - Multiple stream decoding
  - Transitions between codec configurations
  - Rendering vs decoding capabilities
  - Other features to prioritise

Text track capabilities

Text track capability support

- Should Media Capabilities allow queries for timed text embedded in media files (CMAF tracks, SEI messages)?
- Issues:
  - #177 Accessibility review
  - #157 Text tracks not supported
  - #99 Media Capabilities API support for MPEG CMAF Supplemental Data
- Of the four major browsers (Chrome, Firefox, Edge, Safari), only Safari supports timed text embedded in media files
- Currently, applications would have to test browser name / version
- Discussion in TTWG: https://www.w3.org/2020/07/16-tt-minutes.html#t05

Multiple stream decoding

#113 Multiple stream decoding

- A number of media devices support decoding more than one stream at one time:
  - Some devices may have multiple decoders with identical capabilities
  - Some devices may be able to decode one UHD stream and one HD stream at any time - i.e. the second decoder simply doesn't support UHD ever but can be used at any time regardless of what the first decoder is doing
  - Some devices may have dynamic capabilities, e.g., they can decode one UHD or two HDs at the same time but not two UHDs. This may be due to the amount of available RAM, or memory bandwidth
- Using multiple media decoders can improve client-side advertising use-cases
- Previous answer: Multiple stream is out of scope because it would be hard to give a reliable answer. Related: issue #102 (transition API)

Transitions

Transitions

- Media Source Extensions v2 includes SourceBuffer `changeType()`
- Allows codec/container transitions within an MSE SourceBuffer
- Use case: advertising may use a different codec or encryption configuration from the primary content
- Some implementations may or may not support transitioning between configurations, e.g., where the transition would require a different decoding pipeline (e.g., HW vs SW)
- Current status:
  - No implementations, discussion around three proposed API shapes
  - Unresolved discussion on EME (#251)
- Is this important for Media WG to prioritise?

#102 Discuss transition() ergonomics

Proposal 1: Add `MediaCapabilitiesDecodingInfo.transition()` method

```
// Query an initial decoding configuration
const info = await navigator.mediaCapabilities.decodingInfo(...)

if (info.supported) {
  // Query a second decoding config, result shows
  // if the combination is supported, smooth, power efficient
  const transition = await info.transition(...);

  console.log(transition.supported, transition.smooth, transition.powerEfficient);
}
```

#102 Discuss transition() ergonomics

Proposal 2: Add `decodingTransitionInfo()` method

```
const config1 = { ... };
const config2 = { ... };

// Query an initial decoding configuration
const info = await navigator.mediaCapabilities.decodingInfo(config1);

// Query a second decoding config, result shows
// if the combination is supported, smooth, power efficient
const transition = await.navigator.decodingTransitionInfo(config1, config2);

console.log(transition.supported, transition.smooth, transition.powerEfficient);
```

#102 Discuss transition() ergonomics

Proposal 3 (PR #165): Add `codec_transitions_supported` flag to `MediaCapabilitiesDecodingInfo`

```
const config1 = { ... };
const config2 = { ... };

// Query decoding configurations
const info1 = await navigator.mediaCapabilities.decodingInfo(config1);
const info2 = await navigator.mediaCapabilities.decodingInfo(config2);

// Check both are supported, including transitions
return (info1.supported && info2.supported) &&
       (info1.codecSwitchingSupported && info2.codecSwitchingSupported);
```

Rendering capabilities

Display capabilities

- See HDR explainer
- CSS Media Queries Level 5 includes video-* prefixed features for devices with separate graphics and video planes: video-color-gamut, video-dynamic-range
- Proposed but not specified yet: HDR headroom (#9306)
- For video plane width and height, preferred approach is to expose a new deviceVideoPixelRatio property (see #4678, #5044, #6891)
- To do:
  - deviceVideoPixelRatio needs support from CSS WG
  - Implementations needed

Audio rendering capabilities

- Media Capabilities includes both audio decoding and rendering capabilities
- `spatialRendering` “indicates that the audio should be rendered spatially. The details of spatial rendering should be inferred from the contentType. When true, the user agent should only report this configuration as supported if it can support spatial rendering for the current audio output device without failing back to a non-spatial mix of the stream”
- Recognised as not an ideal design

```
dictionary AudioConfiguration {
  required DOMString contentType;
  DOMString channels;
  unsigned long long bitrate;
  unsigned long samplerate;
  boolean spatialRendering;
};
```

Audio rendering capabilities

- `channels` is currently hand-wavy and inconsistent with Web Audio. We should decide whether we want to keep the definition as including the sub (ie. .1) or if we want to stay fully consistent with Web Audio
- Inline spec issue: The channels needs to be defined as a double (2.1, 4.1, 5.1, ...), an unsigned short (number of channels) or as an enum value. The current definition (DOMString) is a placeholder
- Needs to be clear about whether multi-channel audio will be downmixed to stereo. Some content providers may wish to provide their stereo audio track rather than rely on an unknown downmix
- Web Audio: unsigned long numberOfChannels
- Proposal document: Improving Audio Capability Signalling (see #160)

Questions and discussion

- Are there additional use cases that should be covered?
- Any other feedback or questions?

Thank you!