W3C Media WG Meeting

12 December 2023

Agenda

Date and time
12 December 2023, 22:00-23:00 UTC

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

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

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

Agenda

- Welcome
- Media Session #273
- Media Capabilities #209, #203
- Media Capabilities prioritisation
- WebCodecs #558
- Media Source Extensions #329
- AOB

w3c/mediasession#273 Extend MediaMetadata to capture video chapter information

```
[Exposed=Window]
interface MediaMetadata {
  constructor(optional MediaMetadataInit init = {});
  attribute DOMString title;
  attribute DOMString artist;
  attribute DOMString album;
  attribute FrozenArray<MediaImage> artwork;
  attribute FrozenArray<ChapterInformation> chapterInfo;
};

dictionary MediaMetadataInit {
  DOMString title = "";
  DOMString artist = "";
  DOMString album = "";
  sequence<MediaImage> artwork = [];
  sequence<ChapterInformation> chapterInfo = [];
};

dictionary ChapterInformation {
  DOMString title = ""; // The chapter's title
  double startTimeSec = ""; // The start time of the
                            // chapter in seconds
};
```

- Add artwork per ChapterInformation, so the UA can display an image that represents each chapter if the website provides it?
- Is startTimeSec is sufficient, or should we have a new MediaSessionAction for seeking to a chapter?
- Integration with position state?
- Is the WG interested in adding this?

w3c/media-capabilties#209 Align exposing scalabilityMode with the WebRTC "hardware capabilities" check

- Problem #1: WebRTC-SVC uses Media Capabilities API for discovery
  - Indicates if a configuration is “supported” “powerEfficient” or “smooth”
  - Media Capabilities API not limited to capture context
- Problem #2: SVC rarely supported in hardware
  - Today, few devices support “powerEfficient” SVC
  - Simulcast often preferred to SVC to save power
  - Result: `scalabilityMode` support of little value for hw fingerprinting
- Problem #3: WebRTC-SVC exposes less information than Media Capabilities
  - Calling `RTCRtpSender.setParameters()` or `addTransceiver()` with `RTCRtpEncodingParameters.codec` exposes whether configuration is “supported”, but not “powerEfficient” or “smooth”
- Proposal: (From November 21 WebRTC WG meeting) Limit exposure of power efficient / smooth for `scalabilityMode` to capture context only?
- See also issue #176 General approach to capability negotiation

w3c/media-capabilities#176 General approach to capability negotiation

- PING question: Why expose device capabilities to the app for purposes of negotiation? Couldn't we instead have sites expose available media formats and have browsers (perhaps in a way not exposed the application) pick the one they like best?
- pes10k: Spec needs normative protections against fingerprinting risk
- Security and Privacy Questionnaire
- How do privacy characteristics compare for both approaches?
  - Website provides a list of available formats, browser selects
  - Browser allows website to query supported formats

w3c/media-capabilities#203 Browser interop issues

- MediaCapabilities.encodingInfo() type “webrtc” vs “transmission”. Chrome and Safari use “webrtc”, Firefox uses “transmission”
- Safari has special behaviour to show `supported: true` and adds a `supportedConfiguration` object to the result. `scalabilityMode` parameter is ignored, see webrtc/samples#1596. Should we spec `supportedConfiguration`?
- Chrome >= 101 reports `supported: true` for type “webrtc” and `scalabilityMode` parameter. But SVC is only supported in Chrome >= 111, see webrtc/samples#1597. This is a browser bug where the MediaCapabilities would report that the encoders are technically able to do SVC but the WebRTC encoders are not able to be configured for SVC.

Media Capabilities API prioritisation

- We have many open issues and PRs
- Example: issue #102 and PR #165 API for configuration transition capabilities
- How should we prioritise, compared to other specs in the WG?
- Proposal: Organise a WG meeting to triage and prioritise issues - draft slides

w3c/webcodecs#558 Opus packet loss concealment

- libopus says: “Lost packets can be replaced with loss concealment by calling the decoder with a null pointer and zero length for the missing packet.”
- Should WebCodecs expose packet loss concealment, and if so is an empty buffer the right API?
  - `audioDecoder.decode(null)` / `audioDecoder.decode()` ?
- Implementation for other codecs?
- Per @sandersdan comment: detect PTS discontinuity to apply loss concealment?
- Add detail to Opus registry entry? But registry only covers: Recognized codec strings, `Encoded{Audio|Video}Chunk` or internal data, `{Audio|Video}DecoderConfig` description bytes, and expectations for `Encoded{Audio|Video}Chunk [[type]]`

w3c/media-source#329 Add Managed Media Source Extensions API

- Current status: all comments addressed in the PR. Propose to merge in 1 week if no objections
- Next:
  - #325 Redundancies, duplications, and general spec health (see PR #327, #328, #340)
  - #322 Add quality attribute to ManagedMediaSource
  - #341 Describe eviction policy