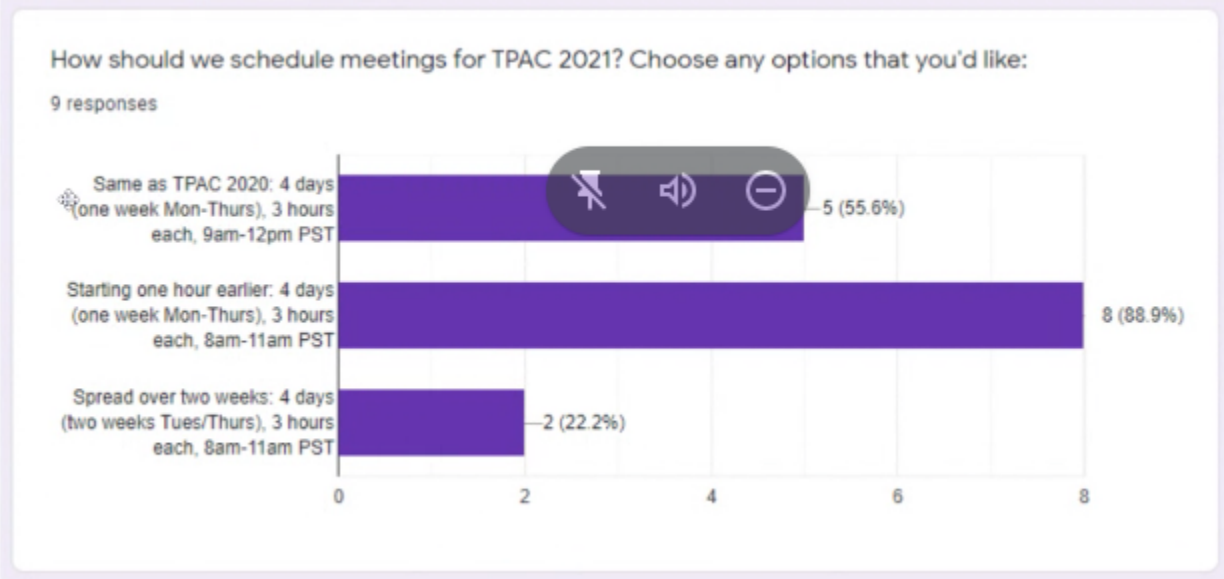
Participants

Nic Jansma, Yoav Weiss, Giacomo Zecchini, Alex Christensen, Benjamin De Kosnik, Tom Mckee, Noam Helfman, Ian Marge, Sean Feng, Annie Sullivan, Nicolas Pena Moreno, Ian Clelland, Michal Mocny, Boris Schapira

Admin

- Will record the presentation
- Next call is September 2nd same time
- TPAC survey
    - 

    - Winner is 4 days in a row 8am-11am PST
    - We'll follow up with an agenda and invites

Issue triage

https://github.com/heycam/webidl/issues/2

- Yoav: WebIDL defines a DOMTimestamp that is not widely used but used by some specs
- ... May have been inspiration behind DOMHighResTimestamp
- ... Proposal to fold into High Resolution Time spec
- ... Fine from spec perspective, but maybe issues from process perspective?
- ... Augment scope of High Resolution Time to cover non-high-res definitions
- ... Maybe we'd want to change the spec name to reflect that
- ... Not sure if we'd need to change the charter
- Benjamin: The move to HR time seems fine
- ... Lean against renaming HR time as it could affect charter
- ... Path of least resistance is to move and not rename

- ... I don't think that'll be confusing
- Yoav: We could also take the approach of moving it, not renaming/touching charter, and if confusion arises we can solve it rather than ahead of time
- Benjamin: The impactful thing here is the change to double, would that affect this?
- Yoav: Seems orthogonal, not sure it needs to happen here
- ... Geolocation depends on it?
- Benjamin: Option to move it, change it to double, and deprecate the old DOMTimestamp?
- Yoav: Alternatively we could change GeoLocation and remove this altogether
- ... See from WebIDL folks if we can move into HR Time without renaming
- Yoav: AI to sum on issue

https://github.com/w3c/navigation-timing/pull/152

- Nic: PR by Sergio to update the diagram to make it clear that the unload event is not serial to the other events in navigation timing, as it's not always happens before them
- … Not too controversial, any feedback?
- <crickets>

https://github.com/w3c/navigation-timing/issues/151

- Nic: unloadEventEnd and unloadEventStart shared the same text
- … Already addressed in the editors draft
- … So this is already covered
- Nicolas: Do we need to check if any published version has that issue?
- Nic: Looks like it's fixed in L2

https://github.com/w3c/resource-timing/issues/276

- Nic: Definition of initiatorType removed from the spec with Fetch integration, and we no longer have a list of that
- Yoav: this is now defined in Fetch's callers, so we could add a note and include links to places where they are defined (although not all callers are properly defined right now)
- Nicolas: Should we also have an explanation in the README? Right now the spec is hard to follow, but maybe we need an explainer that's more detailed.
- … Would be good to add an explainer
- Nic: I like that. We'd also need a link from the spec to the explainer
- Yoav: We could also add a non-normative section with all that info.
- … And now the question is - who's signing up to do that work?
- Nic: Is it just a copy paste from the previous spec?
- Nicolas: It was a mix between explanation and spec. We want something that's just explaining. We can look at the history and reuse some text, but I suspect we'd need to add some additional context
- Tom: I can tackle this

https://github.com/w3c/performance-timeline/issues/182

- Ian: stronger support for adding new types of events. Haven't heard explicit support to try to use existing navigation entries.
- Nic: I think this is the path being followed by App History. Did you get feedback about being able to observe all of them at once?
- Ian: nothing specific about the API shape.

- Nic: do you mind adding the feedback to the issue?
- Ian: Sure, will do that
- Michal: will we not have a common identifier for these different types of navigations? The idea is that other performance entries should be able to link to the navigation being used.
- Yoav: whether we have a single place or multiple observers, I don't think this impacts navigationId, and we can still do this across different navigation types.
- Michal: current proposal doesn't seem to expose that information. It seems like this might be more brittle. You'll need additional listeners in different places
- Yoav: What do you mean by multiple listeners? The question is about having the same entryType vs not, and I don't think this changes the code that much.
- Michal: I interpreted the AppHistory proposal as not exposing performance observer
- Yoav: I hope you're wrong

Presentation: Rebooting the Network Information API

(* Thomas: Current API exposes

- API can say 4G when you're on WiFi which can be confusing
- On mobile the type exposes "cellular" which can result in privacy issues
- CanIuse shows Chromiums, Firefox for Android and KaiOS
- In a little more detail, Chromiums contain the full information
    - "Type" only exposed on mobile
- Firefox exposes connection type, but not e.g. `effectiveConnectionType`, which can cause breakage IRL
- Very popular API: 40% of all page loads
    - Used in a lot of popular widgets: FB, Youtube, performance plugins and analytics tools (e.g. Wix)
- Some overlap with use cases with SaveData API, which started as part of netinfo
    - Now SaveData is defined in a separate spec
- Has privacy issues: Mozila considers it harmful, Apple objects to fingerprinting opportunities it exposes
- TAG was generally positive
- Maybe time for a reboot
- Objectives for the reboot
    - Limit the shared data while enabling the same use cases

    - Triggered by issue #91
    - Driven by issues #85 and #84
- Proposal
    - The NetworkInformation interface

```
[Exposed=(Window,Worker)]
interface NetworkInformation : EventTarget {
  readonly attribute boolean metered;
  readonly attribute unrestricted double sustainedSpeed;
  attribute EventHandler onchange;
};
```

- *shows code samples*
    - Download essential content but not more
    - Limit bandwidth usage - e.g. WebTorrent
- Client Hints to accompany the JS interface that reflect the same values
- Speed would be in coarse buckets growing exponentially, based on sliding window
- Speed can be defined as infinity if UA considers it sensitive
- Welcomes feedback
- https://docs.google.com/document/d/1RDA23zSNdDuIcxZTX9Xo3zlD2fxf8dZg9-e0YaJQv0g/edit#heading=h.g6vu1yi4k1s7
- Alex: Question - used on 40% of page loads. Any data if data was actually reduced? Bytes used with the API indicating high bandwidth vs. low? Bytes used with savedata?
- Thomas: No data. Instant.page uses it for prefetch decision. Wix uses it for analytics. I don't know if we measured bytes saved based on that.
- Nic: Awesome attempt to get this information in a more standardized state. In mpulse we use ECT to bucket data. With sustained speed there would be tension between the observed throughput and the theoretical throughput. E.g. if the user navigated to slow websites and then went to a fast one, that would be different than a speedtest where they tried to maximize their connection. Dunno what the solution, but you could get different results depending on the background activity.
- Thomas: Vendors could play with the sliding window and get more accurate info. You could also run your own "speed test" and download files actively. I was browsing the web and watched the speed meter, and it was pretty reflective of what I was feeling in terms of speed.
- … Need to talk to Chrome eng to figure out what the implementation would look like. I kept it vague on purpose so that conservative browsers from a privacy perspective can still implement this.

- Nic: Let’s continue discussion on the documents