Participants

Nic Jansma, Yoav Weiss, Alex Christensen, Giacomo Zecchini, Annie Sullivan, Noam Rosenthal, Benjamin De Kosnik, Philip Walton, Mike Henniger, Lucas Pardue,

Minutes

Admin

- Next call is on April 28th 8am PT
- TPAC
  - TPAC is officially going to be in-person/hybrid
  - September 12-16 in vancouver
  - Details on hotels are coming
  - Alex: Hoping to be there, but unsure regarding company policy
  - Big corps may wait till last minute for that
  - Nic: Same. If not, we'd go virtual
- Specs
  - Yoav: LCP and EventTiming don't say they're adopted, because they're in editor draft and not working draft yet. Would like to adopt as a First Public Working Draft (FPWD) so they get the right status.
  - … Thinking of sending out a Call for Consensus, but want to make sure there aren't any objections on the call first

Prerender and timeOrigin - Yoav

Summary:

- We have multiple options: leave timeOrigin as is and let developers rely on activationStart explicitly, or move the timeOrigin to be activationStart (either with or without negative timestamps)
- Unanimous consensus on the call was that it's better to have developers and RUM providers look at activationStart (even if that requires updates to existing code) rather than try to retrofit the timeOrigin, as it can have many undesirable side-effects. (what happens with PerformanceObservers pre-activation, etc)

Minutes:

- Yoav: When talking about prerendering, there are navigation speculation rules
- … Jeremy discussed these here a few times
- … Prerendering is coming back to Chrome after a while of turning it off
- … As far as timeOrigin goes, thinking of including an activation timestamp for when the user actually clicks on that page and sees it
- … RUM may think first paint happened after an arbitrarily long time

- … Another approach is to modify the timeOrigin be the activation time, but that leaves other issues – what do we do with timestamps that happened before them?
- … Change them all to “0” or make them negative timestamps? Compat risk with both
- … Wondering what the RUM providers would think?
- Noam: One thing about negative times - it’s not actually possible, because you might have performance observers running before the activation. You might have a script running with PO while prerendering.
- Yoav: Good point
- … If I understand correctly, we don’t have a choice other than go with activationStart and hope RUM providers adapt
- Philip: From RUM perspective, I get that they’ll have to change something to make that work. I think changing the timeOrigin to be activation would be a one-off case.
- … doesn’t seem hard to get RUM providers to move
- Nic: From a RUM provider perspective, we’d have to adapt to that and see how it affects metrics. Doesn’t seem too challenging.
- … Rather deal with activations rather than change the logic around time origin
- Phil: I’m certain I’ve written code that throws away negative values and their entries. Negative values can have a very bad impact on such code as they’d be seen as invalid
- Nic: We do that as well
- Noam: One of the effects is that we kept saying how the time origin is user experience based, and in this case it’s going to be activation start. But 0 will cease to be when the user clicked
- Alex: Let’s assume that we have the time origin be the point in which the prerender started. Can analytics providers see the point in time that the page activated?
- Noam: activation start
- Alex: So seems like giving that as accurate information as the time the user started thinking about this page as open, RUM providers would be able to process that data and provide it
- Nic: RUM providers will need to adapt regardless of the choice we make. Better to have the timestamp available and have RUM providers deal with it
- … awareness would be critical for the websites, libraries, backend services, etc.
- … So need to evangelize this
- Noam: Some metrics will make more sense from 0 and some from activation start. A lot of the evangelism would be around that
- Nic: Network timestamps would be from time origin vs. user visual ones that would be from activation time
- Noam: Except for things that are deferred to activation
- Nic: As a RUM provider I would expect us to have to evaluate all of our metrics and see how they would be affected by prerender/activationTime
- Benjamin: What would be deferred until activation time?
- Noam: A lot of things: starting a new SW, broadcast channel, device APIs. Everything about the user being active in the page
- Benjamin: In this current scheme it would be possible to have LCPs of 0?
- Noam: LCP would always be post activation because there are no paints before that

- Nic: But could be zero-ish
- Noam: LCP-activationStart could be close to zero
- Benjamin: So you'd expect the page to have sub-100ms LCPs?
- Noam: Depends on the page
- Benjamin: So it can create a bi-modal distribution of the data. Sounds like it is.
- Yoav: Consensus on keeping things as they are.
- Nic: Also evangelize about it, would be useful to talk about it. Could use some additional voices
- … Would be great to put this is a newsletter that the WebPerfWG can put out

Redirect opt-in - Noam

Summary:

- 3 options for how to allow an opt-in for timing information with redirects
- General sentiment towards option 3 in https://github.com/w3c/resource-timing/issues/220#issuecomment-1094966082
- Skepticism about this helping with the navigation-start dilemma

Minutes:

- Noam: 3 topics are somewhat related. Something that happens and something that happens before it.
- … Discussed a while ago - TAO does not allow redirect timing, it's about the current resource and allowing it to expose its own timing
- … relevant for Resource Timing but also for Navigation Timing as you can tell how many redirects happened
- … Created TAO but wanted it to mean more things as we went along
- … it's a problem to load new meanings to the header that exists in the wild
- … For redirects, it was important to move it due to NavTiming/160 - with cross origin redirects
- … I thought that have we have an opt-in for redirects, that would ease that pain
- … Created 3 proposals for how to let a redirect say it's OK to share its timing
- … Can use it as a way to avoid overriding TAO
- … The first is Redirect-Timing-Allow-Origin
- … Could be a precedent for adding new meanings to TAO
- The second is similar to Access-Control: Timing-Allow-Scope
- … The use case I have in mind: amazon.com=>amazon.co.uk
- … The third one - similar to CSP, allowing to share things with the origin. It's more complex but very extensible
- … Can add more parameters to each origin
- … If servers start adding that, UA would fail in processing it, considering it an invalid TAO
- … Dunno if people have thoughts about this
- Nic: So the first and third one allow you to do per-origin permissions and the second one does not
- … First and second are slightly more backwards compatible, where the last one…
- … maybe it depends how it interacts with TAO

- Noam: It’s backwards compatible where TAO just means policy for a resource
- Nic: But if you wanted to provide a policy, you probably also need TAO:* (for backwards compat), then have TAP take precedence if given?
- Noam: Better implementation fallback.
- Nic: Small gotchas with each of these. Minute differences
- Benjamin: Is the plan to deprecate the existing TAO headers?
- Noam: Would be good to deprecate them. Also good opportunity to add a same-site keyword. But it would be progressive enhancement
- Nic: Isn’t same site implied?
- Noam: same origin is.
- … Not sure if we’d have a lot of conversation around it here.
- … Would also give us progress towards reducing nav start as a redirect leak
- … Also connect to the previous discussion where the meaning of timeOrigin is moving
- Nic: Reminds me of a conversation about CORS/TAO
- Yoav: Design of including multiple policies into TAO, we should reverse course on extending TAO to mean more things.
- … I don’t have strong opinions on which API shapes or header shapes is better, and I’ll have to take a deeper look
- … Model of CSP where we can say the meaning of TAO right now is “this” keyword, and as we add things it’s “that” keyword
- … Less optimistic that this makes the cross-origin redirect cases easier to migrate off of
- … I don’t expect a ton of providers will add those headers
- … Solution in longer-term is to figure out some form of privacy-preserving aggregation of that data
- … A lot efforts in Attribution Preserving Reporting
- … Maybe we can use that for things that are a cross-origin leak, and can provide that data in aggregate
- … Can tell sites for X % of your visitors, these are the cross-origin redirect delay that they’re seeing
- … I like this design and it could be an interesting path forward for TAO, not sure it’ll be viable for the other one
- Noam: I’m thinking activationStart can play a role in the migration
- … As one of the first steps, we can have “0” be the safe post-cross-origin time
- … If you didn’t have a pre-render, activationStart would be zero.
- Benjamin: Can you clarify which?
- Yoav: I think the most extensible one is the CSP-like one, otherwise we’ll have to keep adding more headers
- Benjamin: Agreed
- … Seemed initially like a big change, but may be easier to understand. Moving towards option 3 the CSP-like policy
- Noam: So slight preference towards the TAP suggestion, and some skepticism towards this helping to fix the cross-origin leak
- Benjamin: Would you try to stage this around activation time
- Noam: Moving the redirects issue is not done yet, so will be staggered

- Nic: Personally think that the TAP style header would give us the most flexibility. CSP is perceived as complex, so not sure if we should take that feedback into account. Could be a consideration here.
- Yoav: would be interesting to understand where the complexity is coming from.
- Noam: Hopefully people already ran into this with CSP and this would be familiar.
- Nic: seeing permission policy taking a similar syntax

Preload consumption - Noam

Summary:

- Seems useful to have this information (e.g. for CDNs)
- Adding a new entry would require some adjustment for RUM providers
- Using the same entry is a bit clunky because it needs to change attributes after it's created which might affect how/when it's queued to performance observers.

Minutes:

- Noam: This is about a missing timestamp. Preload kicks off a fetch and someone uses it
- … But usage isn't monitored other than a console warning if it's unused
- … Should add that timestamp. A few options on how to go about it
- … One is about preload consumption having its own resource timing entry
- … Every RT entry corresponds to a fetch(). Consumption is a fetch(). E.g. we have to re-check CSP
- … So consuming a preload is a bit like getting something from the HTTP cache, but implementation pretend it's like the "list of available images"
- … but the memory cache is not specified.
- … Right now those timestamps are not handled at all
- … One option is to provide a separate entry, and add an attribute on "delivery": coming from http cache,memory cache, preload, etc
- … disadvantage is that you can't connect it with an actual preload
- … Another option is to add another attribute in resource timing to say when the resource is consumed
- .. The issue is that the entry would have to be updated after it's delivered.
- … We already do that in Navigation Timing - not queued to the observers until it's activated
- … So wanted to get opinions on those two options
- Nic: On the proposal to add the timestamp - we would not add that to an observer until the resource is actually consumed?
- Noam: Yes, because it would be a breaking change. Otherwise, we'd be sending a second entry after it's changed.
- … Feels more like user timing where you have a start and an end
- Nic: My main concern about that - in the case that you really want to know that something wasn't used, you won't see that in observers.
- Noam: You would choose the time when you say "I want to know which preloads weren't used" and you'd get that in the performance timeline

- Nic: Is it feasible from a browser perspective to update the reference to the perf entry? You’d still be seeing all the entries, and depending on when you beacon it, it may be there
- Benjamin: What does consume mean?
- Noam: “consume a preload” in Fetch. When the image start loading after it’s preloaded
- Yoav: Answering Nic’s question, PerformanceObserver entries can be updated, but that doesn’t fit the observer pattern very well.
- Nic: Don’t want to box us out of that approach
- Yoav: We could also go with a separate entry for consumption. What do folks think?
- Nic: RUM that doesn’t adapt to it would over report. If we take it into account, it’s not more data to the beacon.
- … An important case for this: some CDNs automatically try to preload things, and getting active feedback on that would be helpful
- … So this kind of information would be extremely useful for us
- .. We’d probably consume it regardless
- … If it was a separate entry type, it may help with compatibility: NotConsumedPreloadEntry
- … May be a separate API..
- Lucas: +1 for this being useful
- Noam: Good to know that this would be useful
- Nic: healthy discussion on the issue so let’s try to converge there and go from there?
- Noam: Sounds good!