Participants

- Nic Jansma, Mike Henniger, Fergal Daly, Noam Helfman, Alex Jose, Michal Mocny, Pat Meenan, Alex Christensen, Behdad Bakhshinategh, Benjamin De Kosnik, Dan Shappir, Yuzu Saijo, Andy Davies, Giacomo Zecchini, Ian Clelland, Kyle Sharp, Lucas Pardue, Marcel Duran, Nitish Mittal, Timo Tijhof, Boris Schapira

Admin

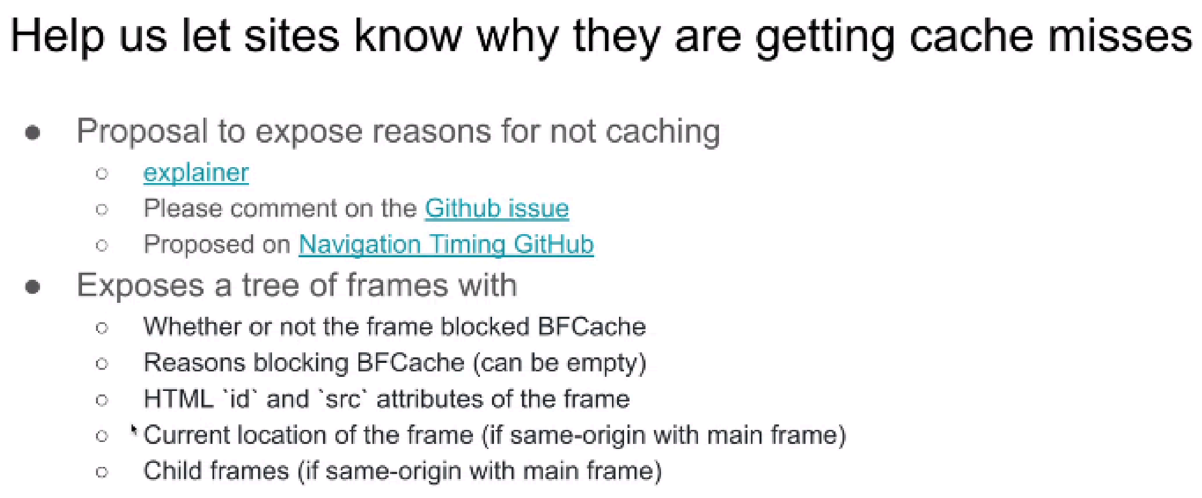
- April 14th 10am PT / 1pm ET

Minutes

NotRestoredReason API for bfcache (presentation)

Recording

- Yuzu: BFcache improves performance of history navigation, and it becomes instant
- ... Freezes the page, and on restore, we resume the page
- ... Aiming for 50%+ hit rate
- ... Can't cache all pages at the moment, need websites help to make pages BFCache-eligible
- ... Proposal helps sites know why they're not getting BFCache'd
- ... Explainer

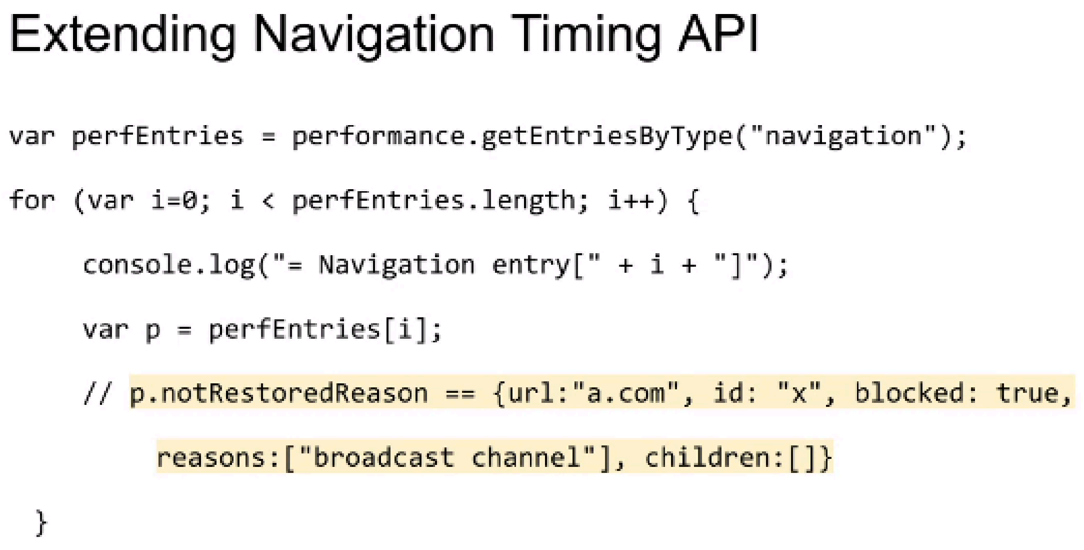


Help us let sites know why they are getting cache misses

- Proposal to expose reasons for not caching
  - explainer
  - Please comment on the Github issue
  - Proposed on Navigation Timing GitHub
- Exposes a tree of frames with
  - Whether or not the frame blocked BFCache
  - Reasons blocking BFCache (can be empty)
  - HTML `id` and `src` attributes of the frame
  - Current location of the frame (if same-origin with main frame)
  - Child frames (if same-origin with main frame)

-
- ... Eposes a tree of frames with details about why it was frozen, reasons, etc
- ... Not exposing cross-origin iframe information - mask cross-origin subtree
- ... Only report if that subtree is blocking BFCache or not, not the reason or child frames
- ... Report at point of navigating away from the page, report only the final destination

- ... Propose to extend the NavigationTiming API

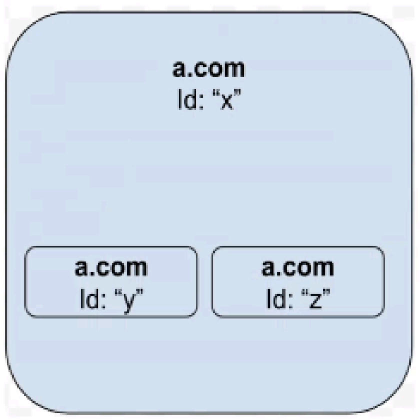


Extending Navigation Timing API

```
var perfEntries = performance.getEntriesByType("navigation");

for (var i=0; i < perfEntries.length; i++) {

     console.log("= Navigation entry[" + i + "]");

     var p = perfEntries[i];

     // p.notRestoredReason == {url:"a.com", id: "x", blocked: true,

         reasons:["broadcast channel"], children:[]}

  }
```

-
- ... For same-sites:

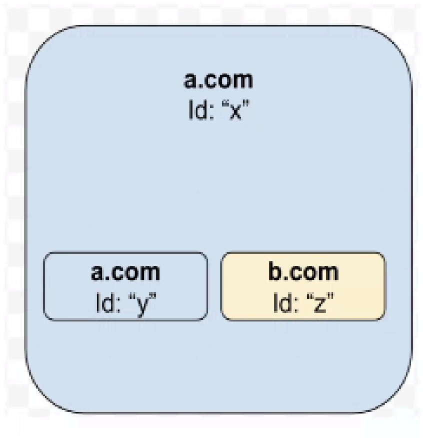
Same-site example



```
{
  url:"a.com",
  src: "a.com",
  id: "x",
  blocked: false,
  reasons:[],
  children: [
       {url:"a.com", src: "a.com", id: "y", blocked: false, reasons:[],
        children: []},
       {url:"a.com", src: "a.com", id: "z", blocked: true,
        reasons:["Broadcast channel"], children: []}]
  }
```

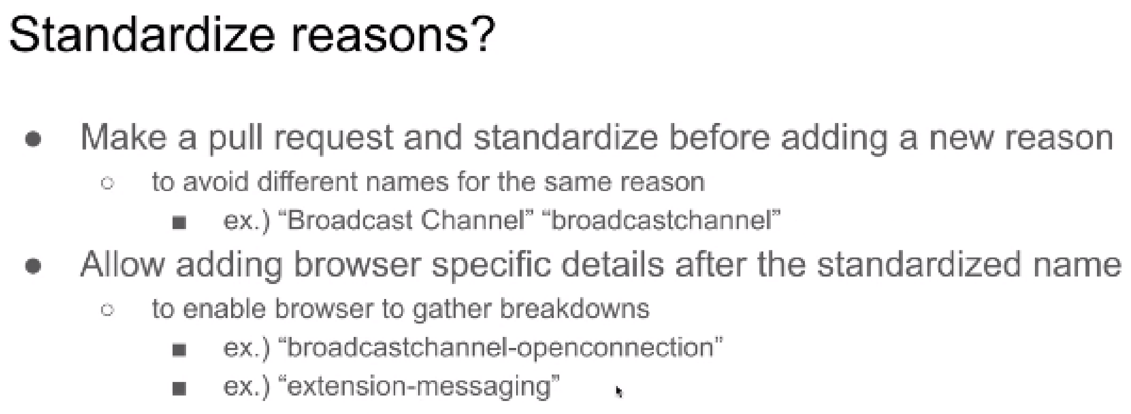
-
- ... Cross-site example:

Cross-site example



```
{
  url:"a.com",
    src: "a.com",
  id: "x",
  blocked: false,
  reasons:[],
  children: [
    {url:"a.com", src: "a.com", id: "y", blocked: false, reasons:[],
    children: []},
    {url:"", src: "b.com", id: "z", blocked: true, reasons:[] (or null),
     children: []}
    ]
}
```

-
- ... URL, reasons, children are empty
- ... A point to discuss is whether or not we should standardize "reasons"
- ... If we don't, it could confuse developers
- ... Allow adding browser specific details after the standardized name



Standardize reasons?

- Make a pull request and standardize before adding a new reason
  - to avoid different names for the same reason
    - ex.) "Broadcast Channel" "broadcastchannel"
- Allow adding browser specific details after the standardized name
  - to enable browser to gather breakdowns
    - ex.) "broadcastchannel-openconnection"
    - ex.) "extension-messaging"

-
- [end of the presentation]
- Benjamin: Do you have a list of reasons that you've found so far?
- Yuzu: Yes, we expose the blocking reasons on the Dev Tools, and I think we're going to expose the same set of reasons
- ... Chromium reasons
- Yoav: From HTML spec perspective, are those reasons all in the spec as things that should prevent browsers from putting things in BFCache, or are some things that we're still trying to get over but are just an implementation limitation in Chromium?
- Fergal: At the moment, it's a mix of both. HTML spec doesn't have many cases that it says are not cacheable.
- ... WebLocks, for example, if you're holding a lock, then no one else can take the lock

- ... I don't think the spec mentions that right now
- ... And then there's a bunch of things, where, in order to get out the door, we've blocked BFCache for a number of reasons but we haven't yet gotten around to fixing them
- Noam H: Understand that you propose to add a property to PerformanceNavigationTiming entry right, and this would be specific to BFCache entry types?
- ... Are we concerned we're expanding structure where it's only appropriate to one type, or we are expanding it to other types too?
- Yuzu: I think that concern is valid, we're thinking it's fine, but I don't know
- Noam: Not first one that's exposed just for specific types of entries. Is there another structure we can expose that's more generic?
- Fergal: No one on our team in BFCache world is involved in this structure, so if anyone has opinions on it, we can talk about it
- Yoav: Third item on agenda is for pre-render cases where we're talking about adding an activationStart attribute to NavTiming entries. Might be worthwhile to look at API shape more holistically, shape it differently?
- ... I think NavigationTiming entries is the right place for this, we don't need a separate entry
- ... But maybe we can compartmentalize for data that's specific to only specific navigation types
- Noam: Maybe a property that holds navigation-type contexts
- Yoav: Or having type-specific information in an attribute below the top type
- Timo: Main thing that came to mind is potential size, for RUM collectors who serialize objects blindly on server-side. Is that the concern?
- Noam: I did not think about that concern, but it's valid. It was mainly about API structure and ergonomics.
- Dan: My question is when will the list be known or known to be complete? Things can be added or removed from preventing BFCache dynamically, so when should this property be queried?
- Yuzu: When clicking back or forward button we know, the state should be final. We should have a complete list of reasons.
- Dan: I need to wait until I am *not* restored from BFCache and then report it?
- Yuzu: Yes
- Dan: I think that should be emphasized. Wondering if there was a way to query for reasons why I wouldn't be BFCache-able.
- Michal: I think you could do that locally with lab testing, audit button. Not sure if there would be differences in the field
- ... Another, I think it's convenient to list all trees, but could it just be a summary for all things in the tree? Or is it necessary to have the full tree with perfect annotation? Can it be reduced to a core of most-necessary information?
- Yuzu: In Dev Tools we reported only the flattened list of reasons, hard to know which frame was causing it.
- Michal: I know for local testing that can be frustrating, but in RUM data are you just looking for patterns.

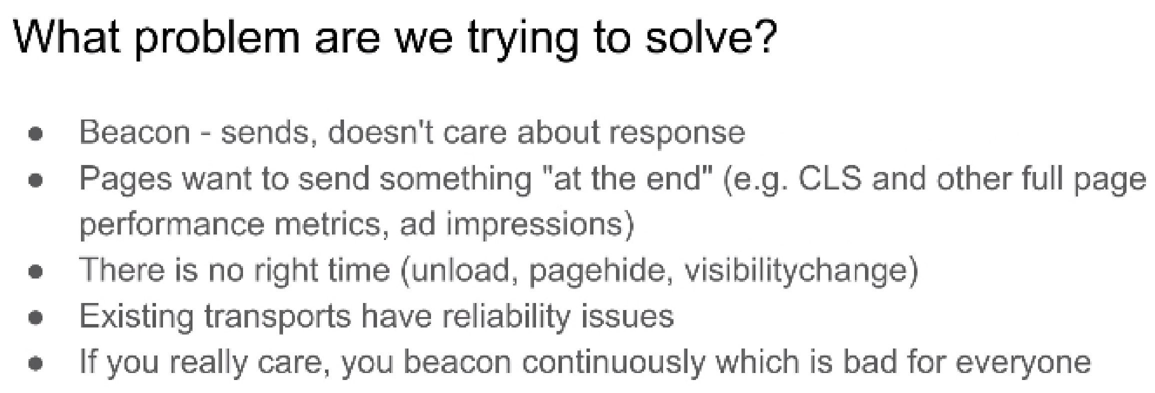
- Yoav: Advantage from getting that data in the field is you can know which 3P to point fingers at.
- ... Agreed that tree structure is a bit unusual in previous things we've done.,
- Fergal: Is the concern around the size of the tree?
- Michal: You want it to be simple, but no simpler. Maybe just saying collectively these things are happening so take care of it. Or some shortened list.
- ... Makes total sense to me in dev tools when you click that button you want perfect information
- Pat: Maybe it... (example that was too long to log, sorry Pat!)
- Fergal: Idea of reaching into frames to get more information from them, that's problematic when going cross-origin. We'd have to be careful if we tried.
- ... A bigger blocker: You might be blocked by an iframe that's not there when you come back, dynamic IFRAME may not be there, so you can't reach into it.
- Alex: With regards to standardizing a list of reasons, over the years we've done a number of pushes to reduce the number of reasons, and occasionally we've added reasons. So in my view that list should be somewhat dynamic and be available to add things before we release.
- ... But I am onboard with standardizing things
- Fergal: In the Github issue where it was raised, he suggests a PR that should always be accepted.
- Yoav: Even if there's only one implementer with this specific reason, that's OK
- Dan: Reference something that Michal mentioned before about the importance of getting data from the field, often this is about 3P scripts added via tag manager.
- ... Developer may not have info about what's added
- ... E.g. fbevents.js has an unload handler, and I assume that affects a lot of websites
- ... Value getting from field and not just lab conditions
- Yoav: 3P scripts are added to the top-level frame. Current API will surface that "unload" is added, but not which script added it
- Dan: Knowing is great, being able to know *which* 3P script is better
- ... Especially if it allows cross-origin, assuming it's doable
- Fergal: We've reached out to a lot of 3P libraries recently, should be improvements soon
- ... Not sure what stage of rollout is, but I'm told it's gone
- Andy: From a RUM point of view, we'd like to start with the simplest implementation possible. Whether a page is restricted from BFCache, and get into reasons later down the road.
- Nic: To Michal's point, with LongTasks we have a vague reason RE the reasoning, and it's not as useful - it hints at the problem but not more. Leaning towards providing as much info as possible. As a RUM provider I can trim that list of reasons down and like having an option for that. Would be more flexible. Our customers want to know what to do next when we point out a problem.
- Timo: Was there new information exposed by cross-origin things? I think all the info is already inferable today.
- Yuzu: We've had a security review, and we don't believe we're exposing something new.

- Fergal: Last thing, is we're going to start implementing soon, so might be looking for an Origin Trial partner
- Yoav: That would also include 3P origin trials (RUM vendors)

Unload beacon proposal - Fergal (presentation)

recording

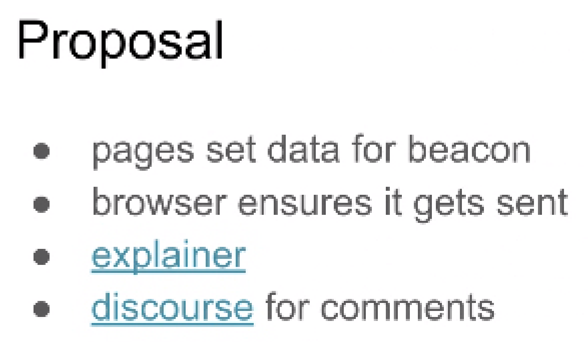
- Fergal: Work with Yuzu as well and in BFCache
- ... Unload is a big enemy of BFCache, so if we can provide APIs that allow people to stop using Unload that could help
- ... Proposal, that we're starting to implement



What problem are we trying to solve?

- Beacon - sends, doesn't care about response
- Pages want to send something "at the end" (e.g. CLS and other full page performance metrics, ad impressions)
- There is no right time (unload, pagehide, visibilitychange)
- Existing transports have reliability issues
- If you really care, you beacon continuously which is bad for everyone

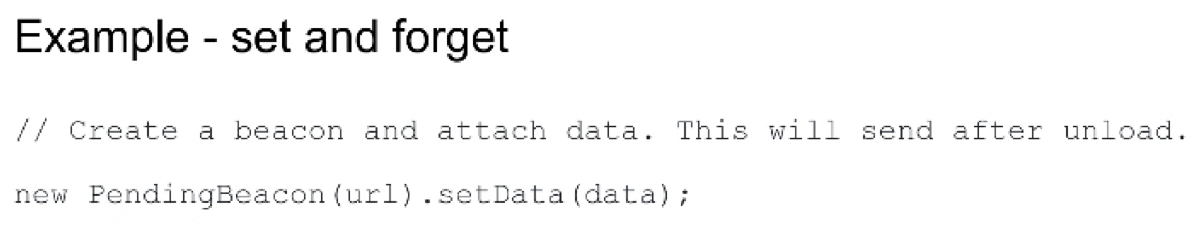
-
- ... Problem of beacons that people want to send reliably at the end of the page.
- ... For example, CLS or other full-page performance metrics with data that's better to accumulate over the lifetime rather than sending throughout
- ... Not a good time to send, events not reliable
- ... Existing transports have reliability issues, navigator.sendBeacon() not always guaranteeing to deliver something
- ... If you really care about data but not user/battery life, you can beacon continuously, but sort it out on the back-end. Bad for everyone.
- .... What someone could do already if they wanted to, thinking if we're adding new risks to platform
- ... Proposal is to allow pages set data for a beacon, and browser ensure it gets sent



Proposal

- pages set data for beacon
- browser ensures it gets sent
- explainer
- discourse for comments

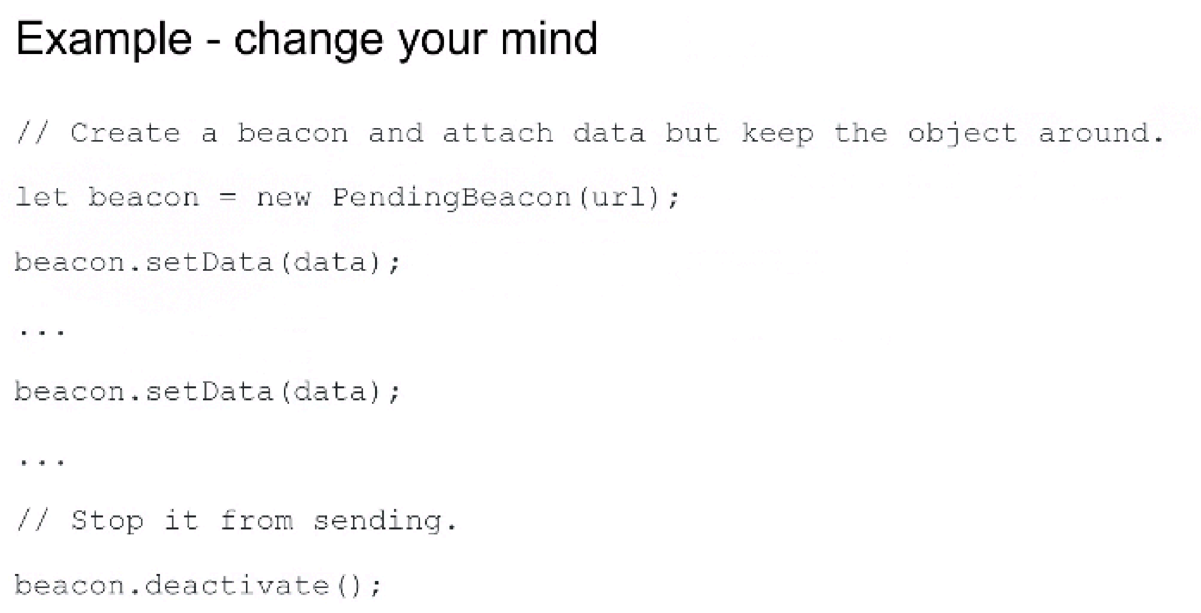
-
- ... Timing is after page is gone
- ... Example:



Example - set and forget

```
// Create a beacon and attach data. This will send after unload.

new PendingBeacon(url).setData(data);
```

-
- ... Script loses reference to the object it just creates, it can't do anything about it, after the page is being unloaded. If the page goes into BFCache and times out, we'll still send it.
- ... If page crashes, we plan to still send data after crash
- ... Example 2

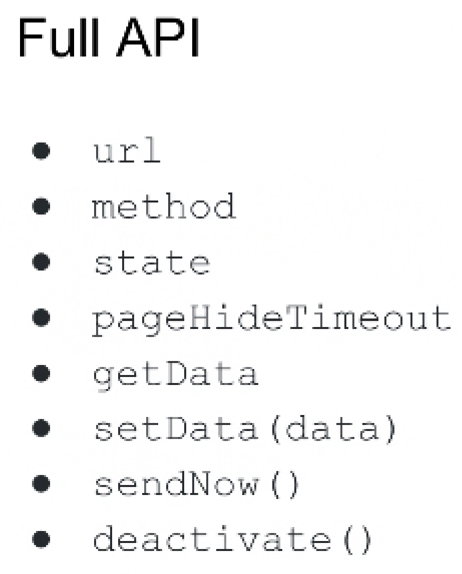


Example - change your mind

```
// Create a beacon and attach data but keep the object around.

let beacon = new PendingBeacon(url);

beacon.setData(data);

...

beacon.setData(data);

...

// Stop it from sending.

beacon.deactivate();
```

-
- ... always keeping the beacon up to date (e.g. CLS), if you do nothing else, when the page goes away, it'll send your data.

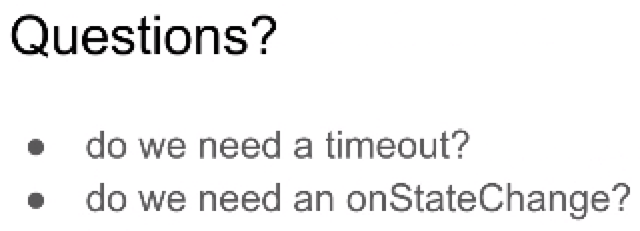
- ... or you can stop it.
- ... full API:



Full API

- `url`
- `method`
- `state`
- `pageHideTimeout`
- `getData`
- `setData(data)`
- `sendNow()`
- `deactivate()`

-
- ... pageHideTimeout specifies that browser should send beacon after page goes into hide, and people may not want beacons coming in 45m after page has gone into BFCache
- ... state gets updated after being sent, failures, after sendNow(), etc
- ... Extensions should be able to block these just like any other network request. How we're going to do that is a bit tricky, as extensions might be surprised in some states of the page
- ... If an extension blocks a beacon, should the page know and how?
- ... Surviving crashes and restarts, if the render process crashes, the browser process is still alive and can send the data.
- ... But if Chrome is closed, it's not possible to send all beacons, so we'd send on restart
- ... If we survive across restarts, it might be a different network on restart, same issue as with Reporting API
- ... Only thing that's different from a page beaconing all the time, for example
- ... Couple questions:



Questions?

- do we need a timeout?
- do we need an onStateChange?

-

- Nic: Excited about this proposal - would solve a lot of problems
- … question about the pageHideTimeout, would it apply after restarts RE how long it took for chrome to restart
- Fergal: If the page came out of BFCache before the timeout, I'm not sure if that results in a send. Needs to be clarified
- … Do we need a timeout "send this when the page is done or in a few minutes?"
- … Basically how long you wait before sending it. Data that you want to send in a certain time frame
- Nic: I'm not looking for a timeout but a max age - "don't send it because it's been 10 minutes and we don't care"
- Michal: Is this a timeout for when the page is still in the foreground?
- Fergal: yeah, "just send this in 2 minutes"
- Pat: Feels like the API surface allowing the sending allows pages to set timeouts on their own. Max age feels like a better one, once the page is no longer in control
- Nic: Either a max age or a timestamp when the data is queued that would allow server filtering
- Fergal: Added a parameter at some point, but now it's a blob, so people can add that themselves
- Dan: What I often see with people using unload, is a flush operation
- ... Going back to fbevents.js, they flush all events that they've collected
- ... API should be designed with that in mind
- ... If the approach is don't send anything and we'll send it once the page is closed
- ... Or if there's a mode where you can periodically flush the sending and you can also flush at the end
- ... Second point is that every time you're setting data, you're accumulating data, is there a concern around the amount
- Dan: We're just replacing the data, not accumulating.
- ... Let's say there are two scripts with an unload event, they would each contain their own data and control it
- Dan: If it's replaced, consider using API called replaceData() -- wasn't clear that setData() replaces it instead of accumulates it
- Marcel: My question is with regards to retry, are we considering a retry when a user is offline and returns online. Not necessarily crashing. For mobile.
- Fergal: I think the network stack would know it's offline and not try in the first place
- ... We haven't talked much about replies, if we don't get a 200 from the server when we deliver it, should we try again?
- ... Maybe make that customizable
- Patrick: Feels like it should behave like Network Error Logging
- Marcel: Retry is something I usually control by myself with backoff etc, but it would be nice if it handled it on its own, with some controls, etc
- Yoav: In the cases you care about, you can't implement your own retry here
- Patrick: For CSP, NEL reports, anything browser sends on behalf of page but not in context of page, the same retry should be used. Automatic way that the site shouldn't care about

- Fergal: Being consistent with them would make sense
- ... Particularly for the unload replacement, they're using sendBeacon() or a keepalive fetch(), and you're not going to be able to control the retry behavior on those, so we're not making anything worse in that case
- Michal: I know from Chrome's own implementation for when we decide to beacon, when there's an unexpected crash and you retry when coming back. But another thing is we try to beacon early, where we share the sentiment that a single beacon is valuable, and we don't want to wait for the next time it's available.
- ... Wonder if retry beacon on crash, is there a per-policy beacon where I can request it gets sent on hide
- Yoav: It's possible that the onstatechange proposal, if the browser decided to send before it was destroyed, script could understand that and it could send more data in the future
- Michal: But this is when the browser is destroyed
- Yoav: In that case you did the right thing. If you sent early and it wasn't destroyed, you want the developer to know that.
- Michal: Specific case: You hit Home on Chrome, the renderer is not closed, the app might be back into foreground, it's not unloaded or in bfcache. That is the perfect scenario.
- ... Alternatively after Home, Android cleans up the tab, now you have to set that in persistence state, but that could be a while where beacons are lost. Ask to eagerly send beacons before unload.
- Fergal: You can do it a bit in the page itself, visibility change, where you're the page on top. Other cases where e.g. Android pauses, and it has all these beacons it sends.
- Michal: Worried on just visibility changes, too quick. Is there another event where you get an event that indicates you're very likely to get unloaded, e.g. tab switcher.
- Fergal: If browser is getting killed, it's unlikely it's going to be able to send all beacons
- ... If we offer policy choice from page?
- Michal: I prefer the API to have control over when to send. And the simple default could be when to beacon to have as much coverage as possible. But does the other use-case need treatment?
- Katie: One of the really common use-cases for unload sending beacons, we want to track the link the user clicked on. If the setData() is a full-replace, if I'm a RUM provider I'll keep an object in memory and periodically call setData() to replace it.
- ... Is there a race condition where I want to grab this data from page context memory and move it over to the beacon, will that get cancelled?
- ... We might end up in situations where we want to update it at the right of the end of the user session, but we can't because the page is unloading
- Fergal: You could also create a new beacon where you put that information into
- ... Hopefully network stack will send in one connection
- ... Don't think there's a concern there
- Ian: Wanted to point out there used to be a lot of complexity in the Reporting API around max-age, etc. We pulled a lot of that out around concerns around implementability, etc. Wondering if other browser vendors are willing.

- Nic: only concern is using setdata for updating would result in 2 copies of the data. Maybe we can have a reference and dedup the data?
- Timo: The idea that the browser stores the data offline makes me offline. That seems contrary to user expectation, so a max expiration would make me more comfortable, basically to have an upper limit on that.
- Fergal: At the worst case a site can continuously beacon, so sending it eventually doesn't seem worse, but we probably don't want to keep them for weeks

Chat Log

Michal Mocny11:02 AM
Recording?

Yuzu Saijo11:11 AM
https://docs.google.com/spreadsheets/d/1li0po_ETJAIybpaSX5rW_lUN62upQhY0tH4pR5UPt60/edit#gid=0

Benjamin De Kosnik11:13 AM
thanks Yuzu

Yuzu Saijo11:13 AM
slides:
https://docs.google.com/presentation/d/1VWl1YElzEkB4vQq3NgtqFc1_zez7e5dMo5bDT7vLGa0/edit#slide=id.g1209aebb9ab_0_33

Michal Mocny11:16 AM
"options" bucket like measures v3?

Timo Tijhof11:22 AM
privacy/security; not unlike what we have with event timing and input delay attribution etc.

element timing*

Katie Sylor-Miller11:24 AM
+1 to Fergal's point, it's important to note that a lot of times iframes are cross-origin and inserted via things like GTM and not under direct control

Katie Sylor-Miller11:27 AM
+1

Michal Mocny11:45 AM
setTimeout(beacon.sendNow, 2000)

Michal Mocny11:48 AM
I like set, same as Map

Timo Tijhof11:48 AM
yeah, setData would be clearer than addData.

Michal Mocny11:49 AM
beacon.data = ... vs .setData(...) maybe

Noam Helfman11:49 AM
how about setDate(key, value)? This will allow simpler control of the state instead of managing it outside the beacon object.

Michal Mocny11:58 AM
(Or Noams idea, to just add one key.. though I'm not sure if beacons are a map like that)

Kyle Sharp11:59 AM
Have to drop. Super interesting discussions, thanks everybody!

Katie Sylor-Miller11:59 AM
we do usually send beacons as a key value store yes

weakmap?

Michal Mocny12:01 PM
expiry will be ergonomically useful even for the beacon receiver