

The state of browser storage partitioning

W3C TPAC 2021 Breakout Session


Date: Oct 21, 2021

Time: 3:00 PM-4:00 PM UTC

Breakout MC: Mike Taylor

Scribe(s): kleber



Slides:  Tpac breakout on storage partitioning placeholder slide

Note: permissions changed from anyone can Edit to anyone can Comment after the session ended.

Agenda

1. Welcome, introductions, [CEPC](#)
2. Ground rules
 - a. What's in scope for discussion today?
 - b. 1 speaker at a time, please use Zoom hand raising to queue
3. The state of partitioning in browsers today
 - a. Chrome (5 minutes)
 - b. Edge (5 minutes)
 - c. Firefox (5 minutes)
 - d. Safari (5 minutes)
 - e. (other browsers not on the agenda?)
4. Discussion topics
 - a. BroadcastChannel partitioning ([link](#))
 - b. Blob URLs ([link](#))
 - c. Cross-site ancestor chain bit ([link](#))

- d. Web Extensions + Storage Partitioning
 - i. Chrome's current thinking
5. Ad hoc topics
 - a. Place topics below

Attendees (Sign in, please)

- **Name, Company**
- Mike Taylor, Google
- Stephen McGruer (he/him), Google
- Xiaoqian Wu, W3C
- Marijn Kruisselbrink, Google
- Brandon Maslen, Microsoft
- Sanketh Menda (he/him), N/A
- Anne Pouillard, Worldline
- Brad Lassey, Google Chrome
- Arno van der Merwe, Entersekt
- Nick Doty, CDT
- Jeffrey Yasskin, Google Chrome
- Johann Hofmann, Mozilla
- Michael Kleber, Google Chrome
- Ben Kelly, Google
- Austin Sullivan, Google
- Erik Taubeneck (he/him), Facebook
- David Baron, Google
- Wendy Seltzer (she/her), W3C
- Ayu Ishii (she/her), Google Chrome
- Sam Weiler (he/him), W3C/MIT
- Matthew Finkel (he/him), Tor
- Kyra Seevers (she/her), Google Chrome
- Andrew Williams (he/him), Google Chrome

Meeting Notes

scribenick: kleber

Mike Taylor:

- queue via "raise my hand" in the Zoom (under "participants" or "reactions" depending on UI)
- Expected behavior as outlined in the code of conduct. Will warn or remove for discrimination, harrasment, bullying.
- Please share the mic, take space & make space

- Presentations (Mike and Johann) part of the session will be recorded, but not the Q&A and discussion parts

Agenda:

- State of partitioning in browsers today, going through Chrome/Edge/Firefox/Safari
- Pre-seeded discussion items (from PrivacyCG repo and others queued up)
- Ad hoc discussions
- In scope: Client side storage partitioning, <https://privacycg.github.io/storage-partitioning/>

Chrome:

- Been working on storage partitioning for past 6ish-maybe months
- StorageWorker: we have tests where you can flip a flag and it works, more or less
- Almost working / in progress: SharedWorker, Web Storage, IndexedDB, CacheStorage, Filesystem API
- TODO: Web Locks API, Storage Buckets
- Web Platform Tests — goal is to have full coverage, but so far:
 - ServiceWorker tests (passes in Chromium behind flag and Safari)
 - localStorage (fails in Chromium now, passes in Firefox & Safari)
- Aspirational: We hope to have a partition-everything flag for testing

Edge (Brandon Maslen):

- State is basically same as Chrome, following the Chromium changes

Firefox (Johann Hofmann):

- Network partitioning by top-level site is in place
- Cookies and most storage APIs are partitioned in Private Browsing & Strict modes, working on enabling by default. Currently blocked by compatibility heuristics, trying to whittle them down.
- Storage Access API + Heuristics to "unpartition" storage
- Documented on MDN
- Goals:
 - Partition remaining APIs (service workers targeting FF'96 ~Jan22, blob URLs WIP)
 - Cross-browser compatibility, to get compatibility with e.g. Safari's Storage Access API to unlock un-partitioning; we don't want people to try to mirror our oddities of implementation
 - Longer term: further keying, like Chromium's triple-keying for cache, perhaps plus some more bits

--recording ends here--

Safari (John Wilander):

- Implemented cache partitioning in 2012, shipped in 2013
- Shipped partitioned cookies in 2017, but encountered compatibility issues — too much interdependence with how servers expect things to work — so un-shipped in 2018, block 3p cookies instead
- Also partition Service Workers, did from the get-go, and also doing Broadcast Channel
- Have been partitioning on eTLD+1, but have started moving to origin instead (PSL is bad)

- When we partition storage, we also make it *ephemeral* — in-memory only, not persisted
- Q (Ben Kelly): When are you doing by-origin partitioning?
 - John: Started with registerable domain in 2012, more recent partitioning work uses origin, IndexedDB or localStorage uses origin instead, don't remember details — would need to go to the engineers working on it to be sure
 - Ben: Design goal?
 - John: Super interested in maintaining performance, so if there is a performance win from going to origin, we'd discuss. Otherwise we're most interested in aligning with other browsers — very happy to no longer be shipping this alone. More reliance on PSL is bad, so prefer Origin on basic principles

Johann: Brave has an implementation on top of Chromium — partitioning + ephemeral, clear it out 30sec after written. Pete had conflict and couldn't make this part of this session.

Discussion Items:

Unpartitioning non-cookie storage? — Storage Access API or via automated heuristics

Johann:

- Safari originally chose to not un-partition after Storage Access, Firefox did un-partition.
- We want to get consistency, and we're seeing how it's hard not to with Service Workers
- Saw Safari wanted to align with Firefox re. Broadcast Channel un-partitioning, so wanted to talk about this all in the same room

John:

- We have permanent partitioning, and Storage Access API gets you access to your 1p cookies.
- Opportunity to get aligned
- It's not only Service Workers where un-partitioning is hard. IndexedDB has been the main problem — could have an active DB connection, how can you handle them when you change partitioning state? No good solution. This drives us to permanent partitioning other than cookies, where you may need to un-partition in order to log into a cross-site resource

Brandon:

- We see room for alignment, doesn't make a huge difference for Edge.
- Main use for API is for sites to unlock and get back to default state. Adding extra protection or restrict storage would maybe be unnatural.

Johann:

- Sounds like alignment, nail down details in email or threads

John:

- We've had some feedback that switching between partitioned and non-partitioned makes headaches for developers. They would prefer un-partitioned but all the time, of course! But if they're going to have some partitioned, they would prefer consistency rather than having different parts of a page in different states.

Ben:

- We have no plan to do un-partitioning, but of course we might learn stuff about compatibility after we ship

- Plan in our back pocket: We'd rather expose it on a separate bucket via storage buckets approach, so avoids the mixed-state problem John refers to
- But best would be to avoid the complexity and not do un-partitioning at all

Johann:

- Can we make it possible to opt into un-partitioning in the future? Buckets might be the solution

Broadcast Channel:

Looks like we don't need additional discussion, FF/Safari just said it all.

Blob URLs:

Also solved based on what FF is doing? What do we partition on ?

AnneVK:

- We ran an experiment with blob partitioning, "open in a new tab" didn't work, and all the breakage we saw was due to that.
- Making it work requires blob URL store to differentiate agent cluster key vs. normal url for new-tab lookup. Could have an agent cluster and a global map for top-level navs.
- Need to get back to this and haven't yet.
- Issue sounded like Chrome had some breakage numbers? 0.1%? Sounds like a problem

Marijn:

- We added some metrics but I haven't looked into which sites break. 0.1% breakage seems high-ish
- Top-level navigation initiated by script seem minimal, haven't added usage metrics for other top-level navs

AnneVK:

- copy/paste URL + open in new tab, download URL + open — those are the places we're worried about very user-visible breakage

Consider including a "cross-site ancestor chain" bit in the storage key?

Ben Kelly:

- Normal algorithm for determining whether a request includes Same-Site Cookies depends on whether there are intermediate cross-domain iframes in the ancestor chain.
- This isn't relevant for partitioning, but is relevant for workers — that says to just use origin as site for cookies (per spec).
- That's bad because nested iframes can hit workers and send cookies when otherwise they wouldn't have.
- Can change to construct site-for-cookies just from partition key, if only we add the "was there a cross-site intermediary in iframe chain?" bit to the key
- Downside: more partitioning of workers, so e.g. would need separate SW registration that uses two different sites-for-cookies
- I think we're going to pursue it in Chromium, want to hear about interest in alignment or concerns

- Similar to HTTP Cache: instead of cross-site-ancestor bit, they have been talking about an "is this resource inside an iframe?" bit, but for similar ideas; maybe we should switch to that same A>B>A meaning there too.

AnneVK:

- This should probably be in the spec, if we can make it work
- Good to align with cookie model for CSRF across the board, seems like the right security perspective
- Are you also willing to try it for script access purposes? Like A1>B>A2, can A2 get script access to A1 via window.parent.parent? Kind of an extension of the same problem?
- Ben: This is an extension of the ancestor problem, seems tangential to how workers handle cookies, which was the motivation here. I think the web compat problem would be pretty serious here.
- Anne: From the CSRF POV, A2 could get A1 to do stuff for it, so maybe a flaw
- Ben: This is indeed the case, where A1/A2 see different storage
- Anne: We do know there are libraries that create frames that proxy functions inside them. But maybe no website relies on the storage partitioning details here.
- Ben: Not sure how we could answer that Q without trying it and seeing what breaks. Not enthusiastic about attacking this if there are compat issues
- Johann: Want to coordinate across browsers, want to hear what Safari thinks about it. Could we sever sync scriptability? Across browsers though, don't want to cause cross-browser compat issues.
- Wilander: Always interested in these things, please ping, I'll make sure right WebKit engineers take a look
- AnneVK: You mentioned HTTP Cache. Have you looked at this for network state in general?
- Ben: Matt Menke mentioned something similar for HTTP Cache; I don't know enough about network state partitioning to speak to it. Good to drive everything to same partitioning key on general principles.
- Brad Lassey: I think they are using the same key right now, but want to hear about it if Anne knows of a difference
- Anne: I think network uses top-level site + document site, but cache also uses is-navigation as a bit. That's different from the cross-site-ancestor-in-chain bit discussed.
- Brad: I'll need to dig in.
- Ben: If we could sever sync scripting in cross-site-ancestor-chain scenario? I feel like there are probably web compat issues, would have done this long since if it were possible.
- Anne: I don't think we've pushed on this much, don't recall it being in the discussion. We were looking at document.domain and the like.
- Mike: In the interest of time, should move on. Is there a place to discuss sync scripting issue? Place to file a new issue, or keep it in #25 in Storage Partitioning repo?
- Anne: I'll split it if needed.

Web Extensions + Storage Partitioning?

Mike:

- If you have an extension that frames something and do something interesting with site storage, it won't work. If an extension has host permissions for a site, we could consider treating the extension's chrome://... URI as if it's same site.

Johann:

- We have someone else working on this, Rob Vu (?), I can forward.

John:

- We do support Web Extensions. We've done special things around cookie APIs, but I don't know if we have a position or opinion on partitioning, would need to forward to the right people.

Josh Karlin, re networking key difference:

- For HTTP cache partitioning, we have key include top-frame site and request site, plus a bit "is this a subframe navigation request?", to prevent one origin from trying to determine whether another origin has loaded something via timing attack on iframe loading.
- Anne: That wasn't actually the question :-). Q was whether you're going to use the cross-site-ancestor bit for network state as well.
- Ben: This is for A1>B>A2 case, where A2 sets the cross-site-initiator bit, so this is related but different.
- Josh: That was for Same-Site Cookies protection. This isn't in code or in our plans. Happy to discuss. Could just use the whole ancestor chain. Right now this is just HTTP cache, not other network partitioning, which is just triple-keyed.
- Anne: The navigation bit we cannot really reuse for storage. If the site adopts Service Workers, then it would be vulnerable to that cache attack.
- Ben: We are partitioning by 3p iframes... don't understand what the bit is protecting against. Doesn't normal triple-keying do the same thing?
- Anne: If you have a SW for the entire origin, then you can do the same probing via timing attack
- Josh: Not sure.
- Ben: Partitioning solves that, right? Unless the intermediary is doing an attack, rather than a top site doing it.

Mike: We're out of time, jump back to Issue 25 to spend more time on it.