

Day 1

Web Applications WG TPAC 2025 Meeting - Mon 10 November 2025 <Day 1>

Participants

(Please add your name!)

- Ananya Kittane Yogananda <a.yogananda@samsung.com>
- Andrew Sutherland <asuth@mozilla.com>
- Anna Sato <annasato@chromium.org>
- Christian Liebel <christian.liebel@thinktecture.com>
- Daichi Asakura <daichi-asakura@cybozu.co.jp>
- Dan Murphy <dmurph@chromium.org>
- Eriko Kurimoto <elkurin@chromium.org>
- Euclid Zhizhen Ye <euclid.ye@huawei.com>
- Evan Stade <evanstade@microsoft.com>
- Gabriel Brito <gabrielbrito@microsoft.com>
- Haili Bai <baihaili@huawei.com>
- Hayato Ito <hayato@chromium.org>
- Jake Archibald <jarchibald@mozilla.com>
- Jan Varga <jvarga@igalia.com>
- Jing Shen <shenjing10@huawei.com>
- Kagami Rosylight <krosylight@mozilla.com>
- Kazumasa Okbe <kokabe@lycorp.co.jp>
- Liang Zhao <lzhao@microsoft.com>
- Limin Zhu <limzh@microsoft.com>
- Lu Huang <luhua@microsoft.com>
- Marcos Caceres <marcosc@apple.com>
- Matt Reynolds <mattreynolds@chromium.org>
- Monica Chintala <monicach@microsoft.com>
- Mike Smith <mike@w3.org>
- Ming-Ying Chung <mych@chromium.org>
- Mingyu Lei <leimy@chromium.org>
- Minoru Chikamune <chikamune@chromium.org>
- Muadh Al Kalbani <m.alkalbani@samsung.com>
- Olli Pettay <smaug@mozilla.com>
- Penelope McLachlan <pjmclachlan@chromium.org>
- Philippe Le Hegaret <plh@w3.org>

- Ruoya Sheng <Ruoya.sheng@bytedance.com>
- Sam Richard <snugug@google.com>
- Samuel Maddock <smaddock@salesforce.com>
- Sangui Park <sangui.park@zozo.com>
- Shunya Shishido <sisidovski@chromium.org>
- Simon Pieters <zcorpan@mozilla.com>
- Steve Becker <stevebe@microsoft.com>
- Sun Shin <sushin@nvidia.com>
- Tab Atkins <jackalmage@gmail.com>
- Tantek Çelik <tantek@mozilla.com>
- Tatsuya HAYASHI <tathayash@digital.go.jp>
- Tin Tun Aung <tin.tun.aung1@huawei.com>
- Tsuyoshi Horo <horo@chromium.org>
- Vincent Scheib <scheib@chromium.org>
- Xiaoqian Wu <xiaoqian@w3.org>
- Yoshisato Yanagisawa <yyanagisawa@chromium.org>
- Youenn Fablet <youenn@apple.com>
- Youngmin Ji <ym.ji@keti.re.kr>
- -- add your name above alphabetically --

Chair

- Léonie Watson
- Marcos Caceres
- Diego González <luigonza@microsoft.com>
- Marijn Kruisselbrink <mek@google.com>

Scribes

- [_volunteer_](#)
- dmurph@google.com
- mek@google.com

Logistics

<<https://github.com/w3c/webappswg/wiki/TPAC-2025>>

Agenda

<<https://github.com/w3c/webappswg/wiki/TPAC-2025#-mon-10-november-2025>>

Minutes

Topic: Welcome everyone!

Topic: Working Group Status and Spec Updates

- Badging

Status report → <https://github.com/w3c/badging/issues/115#issuecomment-3268604132>

- Badge API works with push / declarative push
 - badge is often set via push
 - declarative push had to add some info for this
 - badge gets set, even if permission isn't granted, and then when the user grants the permission, it shows the badge that was invisible
 - (on Mac)
 - Android - similar model - so we can fully remove the permissions checks there too
 - (see status report - these notes are a little redundant)

-

- File API

Status report → <https://github.com/w3c/FileAPI/issues/211#issue-3315101352>

- Test results
<https://wpt.fyi/results/FileAPI?label=experimental&label=master&aligned>

- Image Resource

Status report → <https://github.com/w3c/image-resource/issues/49#issue-3315107465>

- Push API

Status report → <https://github.com/w3c/push-api/issues/404#issue-3315129337>

Difficult to test, need to set up a push server for testing

Action: Mike Smith to look into the testing of Push API

WPT testability -> <https://github.com/w3c/push-api/issues/365>

- Service Worker

Status report →

<https://github.com/w3c/ServiceWorker/issues/1787#issuecomment-3440258319>

- Web Share

Status report →

<https://github.com/w3c/web-share/issues/287#issuecomment-3413672933>

- Test results

<https://wpt.fyi/results/web-share?label=experimental&label=master&aligned>

-

- Contact Picker

Status report → <https://github.com/w3c/contact-picker/issues/80#issue-3315172049>

Implemented in webkit, but not in Safari / the OS

Chrome supports contact picker on Android.

Test results <https://wpt.fyi/results/contacts?label=experimental&label=master&aligned>

- Gamepad

Status report → <https://github.com/w3c/gamepad/issues/226#issue-3315103404>

Test results <https://wpt.fyi/results/gamepad?label=experimental&label=master&aligned>

- IndexedDB

Status report →

<https://github.com/w3c/IndexedDB/issues/469#issuecomment-3417262230>

- Screen Orientation

Status report →

<https://github.com/w3c/screen-orientation/issues/258#issuecomment-3414142431>

- UI Events

Status report → <https://github.com/w3c/uievents/issues/399#issue-3315133941>

Discussion in the room of migrating components of this specification elsewhere, and noting that there are missing components such as a processing model. Need to continue work on event-algo.bs

- Manifest File

Status report → <https://github.com/w3c/manifest/issues/1181#issue-3315145820>

- issue filed for testing: <https://github.com/w3c/manifest/issues/1194>

- TPAC2025-labelled issues

<https://github.com/w3c/manifest/issues?q=state%3Aopen%20label%3A%22TPAC2025%22>

- Device Orientation

Status report → <https://github.com/w3c/deviceorientation/issues/193#issue-3315178464>

- Geolocation
Status report → <https://github.com/w3c/geolocation/issues/189#issue-3315184259>
"updatable REC" process was terrible to go through, avoid it.
- Pointer Lock
Status report → <https://github.com/w3c/pointerlock/issues/103#issuecomment-3432348432>
Test results <https://wpt.fyi/results/pointerlock?label=experimental&label=master&aligned>
Open issues <https://github.com/w3c/pointerlock/issues?q=is%3Aissue%20state%3Aopen>
- Screen Wake Lock
Status report → <https://github.com/w3c/screen-wake-lock/issues/376#issue-3315185970>
Improved testing would help here as well.
- Web Locks
Status report → <https://github.com/w3c/web-locks/issues/119#issue-3315149294>
- Haptics Discussion

[10:30 – 11:00: ☕ Break]

(drinks and snacks) will be available from 10:30 to 11:00 and from 15:00 to 15:30 - Kobe International Convention Center, Reception Hall, 3F, 4F, 5F

Topic: Service Workers

Issues → <https://github.com/w3c/ServiceWorker/labels/TPAC2025>

Presentation: ☐ Service Worker TPAC 2025

SW static routing for App Shell

Issue: <https://github.com/w3c/ServiceWorker/issues/1796>

☐ TPAC 2025 SW static routing for App Shell

Shunya walks us through the slides, explaining how navigation works in Chrome. Service Worker adds additional steps in the critical path of navigation, but such steps can take a long time. Static Routing API was introduced to help solve this problem by letting developers

specify ahead of time how certain resources should be fetched (from network or cache storage) without having to start the service worker when not necessary.

Now the question is if we can move the navigation commit earlier as well.

App Shell architecture has the service worker return a minimal UI independent of the page content quickly, while the content is fetched via javascript later. This way navigation commit can happen when the app shell is returned, rather than waiting for network fetch of the page contents. This is dependent on the service worker though, which maybe can be solved with static routing.

Proposal to add Service Worker Synthetic response to static routing API. This way the routing can return the static HTML from cache storage while waiting for the response of the page content from the server. No formal explainer yet, but some rough ideas for API shape.

Static routing API would return app shell html, while network fetch going on in parallel can still return the body. An alternative would be for the synthetic response to just include the HTTP response headers, while the body for the response is still returned from the network, and gets appended to the response. This version would be simpler, but doesn't fully support the app shell architecture.

Problems, unclear points:

- Unclear where the response body from the network is appended.
 - [Declarative partial updates](#) proposal may be a solution.
- Should the network request have a special header?
 - Developers may want to serve a different content if the synthetic response is used.
 - Navigation preload has the dedicated header `Service-Worker-Navigation-Preload`. We may need something similar.
- CSP nonce string always change per request
 - Using `<meta>` as a workaround.
 - Another idea: browser generates the nonce string for CSP.
- What happens if the request fails?
 - Currently thinking about exposing the interface to cache the response for error content.

Chrome is currently prototyping implementing something like this.

Jake Archibald: In the case where the page could receive the response, is the preload cache good for this? It's a way of preloading resources that the page could fetch later. Can also be multiple requests.

Shunya: Preload cache as in navigation preload?

JA: No, link `rel=preload`, specific to the document. A way of pre-filling this ahead of document creation time. This could be done at same time as app-shell creation. From the page you'd still call `fetch(...)` to get the document contents, which would work even without preload cache, but with preload cache would come from cache.

Sam Richards: Have you considered a combination of dynamic content and static contents. A way of mixing dynamic with pieces of static content?

Shunya: Very interesting idea, we haven't really considered merging multiple sources into one stream yet. Combining response header with response body is the main thing thought about so far. Merging multiple sources would be an interesting idea. Also related to declarative partial updates.

JA: Declarative partial updates are an interesting separate part of the proposal to allow you get one data source and put it in multiple places.

SR: The two of them together would work for the use cases I'm thinking of.

JA: If we had a way of pre-populating the preload cache ahead of time, that would work as well for this.

Youenn: Seems there are two parts of this proposal; you might want to combine html from multiple sources. But the other part is getting the http headers soon enough to decide if we should switch process or other security http headers that affect navigation commit. Also knowing if we're getting html or some other source. Different from stitching together multiple html bits. Which part are you more interested in?

Shunya: From the browser implementation process, content type, response code and other headers are the main things we need. Navigation commit in the renderer process in chrome needs just the http response headers to be able to start committing the navigation in the renderer. But for paint we'd need html body as well.

Youen: Still not sure which part you're trying to solve, is it both or just the navigation commit?

Shunya: Primarily the navigation commit phase being faster and not being blocked on server response.

Youenn: Maybe just get the headers so you know it's same-origin html etc.

JA: What are the cases where we don't know it is an HTML environment that we need to create?

Youen: If you get a mime type that is not HTML you might get a download instead, and not navigate the document.

JA: We know the origin, since it's a navigation - but it could be a redirect, which would result in a new fetch. Also content disposition could result in not an html document. Image or pdf does create an HTML document as well, but download doesn't.

Youen: You could be speculative and always try to create an html document?

JA: Declarative root points to the cache, is it still not fast enough?

Shunya: Static routing can get the whole response from the cache. But for some sites they don't want the whole html cached, since the content is dynamic.

JA: Can static routing point to the app shell for a large number of URLs?

Shunya: Static routing can serve the static part, but how does the browser then try to get the response, which would be dynamic?

JA: Static routing is pointing to the whole app shell, and then the page would make some fetch to get the new contents. We just need the part where that fetch can start much earlier before javascript in the app shell can run.

Shunya: Want to optimize when this network request start, make it start from the browser process to get the response much earlier.

JA: If we had a declarative route for the app shell and a way to put the dynamic content in the preload cache, would that be fast enough?

Shunya: Think that would be, yes

Static Routing: [A mechanism to detect supported conditions and sources - #1793](#)

YY: The Service Worker Static Routing API defines routing rules using a dictionary

- **Ignoring Unknown Conditions/Sources:** If a rule includes a condition/source that the UA does not support, it can be silently ignored instead of causing a syntax error.
- **Difficulty in Feature Detection:** This behavior makes it difficult to reliably implement feature detection and graceful degradation. This also makes it difficult to implement the feature incrementally.

Three options to support feature detection:

1 - API to enumerate supported features, one method call to check for everything that is supported by current browser

2 - Rule validation API. I.e. an API that lets the SW ask the browser if a particular rule or condition is supported.

3 - Required field in addRoutes(), let addRoutes reject if any of the required conditions aren't supported by the browser.

There has been some related discussion in <https://github.com/whatwg/webidl/issues/107> as well. Different APIs are implementing their own ways for feature detection. How should we solve this for static routing?

Marcos: Naive question about the API shape; is it expected that things would change after the event?

JA: You can only add static routes in the install event. That's why the API is on "event". Supported conditions is what is implemented by the UA.

Youenn: It seems that currently the API allows feature detecting, in a hacky way. But not always. In the short term the hacky thing might be fine and we could wait for the webidl version for the long term?

JA: What is it that is working now?

Youen: The getter hack, where you add a getter that throws and if it throws or not you know if something was supported.

JA: My worry about that method, but also the "getRoutingCapabilities" proposal is what if some combination of properties isn't supported together. So for that verifying a specific rule might work better.

YY: Moving on to other issues

Consult document CSP in Register algorithm - [755](#)

Monica: Clarify how CSP should apply during navigator.serviceWorker.register() and subsequent updates to ensure consistent and secure behavior.

Potential solutions:

1. Enforce CSP during `register()`
2. Leave CSP enforcement to `Fetch` only
3. Hybrid approach

Questions:

- Should CSP be enforced at registration, update, or both?
- Handling CSP changes after initial registration. Impact on documents with restrictive CSP being controlled by existing service workers.
- How CSP changes affect existing registrations and updates?
- Alignment with other worker types (e.g., `SharedWorker`).

Youen: Is there a difference in browser behavior? Do all browser do the same thing here?

Monica: I think Gecko always checks the document CSP during registration, but chrome maybe doesn't.

JA: Do any browsers involve CSP in the update process?

Monica: Not sure, chrome does somewhat

JA: Where does it get the CSP rules from? An update can happen without a page being available. So CSP must be cached/there is no CSP, since the worker was allowed to be installed originally. Seems registration time is the only time that CSP would need to be involved, since we already don't allow redirects. Providing some kind of integrity ID is not what we want to do with updates, since you could lock yourself out of updates very easily. If there are rules that we would want to enforce at update time they would be cached at registration time.

YY: If a SW is registered from page A, and page B has a different CSP that would deny the registration, it would still be controlling page B.

JA: Yeah, we wouldn't involve CSP when register doesn't cause a new registration to happen, and just returns the existing registration.

YY: Conclusion is that only checking CSP for initial registration is probably correct?

JA: Shared worker has the same issue, when you request a shared worker for a particular URL it just returns the worker that already exists, potentially bypassing CSP that would be involved with the fetching side since it doesn't end up fetching. Same model would work for service workers. The register call that creates the registration involves CSP while a call that just returns an existing worker or registration doesn't involve CSP.

Youenn: Makes sense, I would write tests to make sure browsers are doing the same. It seems logical to do the same as shared workers.

Monica: What should happen if two pages try to register the same worker, should we deny registration lookup if CSP doesn't allow?

JA: Which CSP rules are we thinking of?

Monica: When `child-src` is none?

JA: My understanding is that CSP is enforced at the fetching stage, so if we're not actually fetching than it doesn't apply. But I don't know that for sure. Does anybody here know CSP better than I do?

Andrew Sutherland: Gecko may diverge slightly here. The worker is sort of a top-level load and want it to verify its own CSP. But we agree that top-level script of the worker is more a subresource of the page and thus should verify the CSP of the document creating. On lookup vs fetching, I think it makes sense that the way things are specified and job coalescing we'd still return it.

Youen: I hope Safari already does that as well.

Monica: So next step is writing WPT test and based on test results and browser implementation update the spec.

JA: I think there are 3 phases here. The thing that causes the registration to be created; it's the pages CSP that applies there. There is the CSP of the SW itself, which governs fetches made by the SW. And then there is getting an existing registration or shared worker, and I don't think CSP should be involved there unless there is a specific CSP rule that is designed to prevent that environment to stop it from getting hold of a worker. Not sure that exists.

"no-cors" CSS SOP violation - [719](#)

JA: I think this one is from 2015, sorry

Monica: Yeah, want to make some progress on old issues.

Decide whether and how to mitigate Same-Origin Policy (SOP) leaks caused by no-cors CSS subresource requests being exposed to Service Workers and Resource Timing APIs.

Potential Solutions:

1. **Skip SW & RT for no-cors CSS** (Previous discussion supports this solution and safari implemented it)
2. Introduce a header (eg., sec-from: same-origin)
3. Require CORS for imports

Questions:

- Should SW intercept no-cors CSS subresources? (Previous discussion supports it and safari implemented it)
- Should RT expose these requests?
- What would be the impact on offline caching and developer workflows? Blocking these would break offline scenario (**~3.5% of page loads**)
- From previous discussion, it was stated that "**CSS needs to integrate with Fetch**"

Safari: skips SW for these requests.

Chrome: ~3.5% of page loads involve SW intercepting no-cors CSS requests

Firefox: ?

To some extent these are already observable.

Should we hide these from service workers? But that would cause some regressions. Although Safari already does that.

Youenn: My first thought is that Safari does that, but not sure if the spec allows that. Might be good to at least add a "May" to the spec to allow Safari's behavior. We also fixed a similar issue

with HLS. HLS manifest can be fetched when passed to a video element, and HLS document can import other html documents which can be cross origin, so same issue there. Would be good if the spec allowed that. Re resource timing, I believe the spec was fixed. I would hope that the chrome implementation is also fixed. For offline scenario, it wasn't clear what would break. The longer we wait the more breakage could be. Hope we can work with CSS to start fetching css resources with CORS to reduce breakage. Is fetch integrated with css now?

Tab Atkins: I believe it is. We could add a feature to support CORS with CSS. We have the syntax for it already, i.e. the ability to add modifiers anywhere a URL is used in css. <https://drafts.csswg.org/css-values-5/#request-url-modifiers>

JA: Interested in Tab's opinion in general. To make offline work we need to intercept everything. Security worry is no-cors subresources in CSS. Always unclear how much security/privacy is expected here. For some properties you can already detect these through get computed style. Less clear about import, since you can't get to the css source when cross origin.

TA: You can look at applied styles, so you can have some idea about what was in cross origin no-cors imports. The effects are always visible.

JA: Effects are, but query strings etc would not always be. How much do we care. If URL now supports cross-origin, there now is a way for developers to make it possible to fix this by the site developer.

Youenn: Have had no bug reports from developers asking to change their current skipping. If they do, we'd tell them to use cors now that is available in CSS. Maybe ask Noam to look at this issue.

Youenn: Is there a spec change needed to allow Safari's behavior?

Monica: Not sure where this is in the spec

YY: Might be in the fetch spec? Don't think there is

Youenn: Would be good to add a note/may statement for these specific ones.

Restrict openWindow() to http(s) schemes? - [699](#)

Monica: Determine whether clients.openWindow() should be restricted to http(s) schemes to prevent security risks and ensure consistent behavior across browsers or match window.open()

To discuss:

- Should **non-http(s)** schemes (e.g., file://, about:blank, mailto://) be blocked or should it to mirror window.open() accepted schemes?
- Test how clients.openWindow() handles custom schemes and JS URLs using WPT before adding new restrictions.
- If tests covers these cases, leave current behavior.

JA: Can you summarize what security issues for about:blank?

Marijn: about:blank is particularly useless since you can't do anything with the resulting window

client.

JA: What is the security risk of allowing mailto:? It wouldn't return a client, just like with cross origin. File: urls are bad, but are bad with window.open as well. I don't think Clients.openWindow changes anything there.

Youenn: As long as it doesn't reveal if a particular file path exists.

JA: We shouldn't have a different opinion from window.open. I feel this issue has been open long enough and we should just close it.

Smaug: And remove the about:blank special case

Youenn: Is there browser interop here today? Do all browsers allow the same things?

Monica: I don't think there is WPT coverage for this today.

JA: From vaguest of memories, I wonder if we blocked about:blank because we wondered if we could make about:blank useful in the future.

Youenn: Safari blocks all about:* URLs

Marijn: I believe Chrome does that as well.

Andrew: Our behavior is complex, we use our normal window opening flow, which also tracks origin and there might be other blocking in play. Don't think we block anything other than about:blank.

Monica: Don't think we deviate much from the spec.

JA: Looking at it more, I agree that we should just allow about:blank even if it is dumb.

AS: There have been real problems with push notifications. Clients open windows for push notifications and there is a market for abuse. I wouldn't be against maybe restricting things more. I would be worried about cross origin openWindow and attribution to the push message origin.

JA: You can only open windows on notification click now, right?

AS: Yes, and in theory notification has an origin associated with it, and there are UI affordances in the notification.

Youenn: about:blank blocking is interoperable, so why should we spend time changing it

Smaug/JA: Fair enough, just leave it

AS: Looks like Chrome recently added metrics around this?

YY: Yeah, but not enough data from it yet to draw any conclusions.

AS: It would be interesting to see what the data shows, I'd definitely be on board with potentially constraining more, even if it is only to avoid people from filing bugs that aren't really bugs.

Expose environment ids? - [643](#)

Monica: Allow environments (e.g., documents, workers) to access their own unique IDs.

Potential solutions:

1. Adding `clients.thisId` or similar property? If so UUIDs should be mandated or if less strict uniqueness is sufficient.
2. Constructor-based access to a Client object for the current global (e.g., `new Client(window)`).
3. Instead of passing the global explicitly, could use a constructor from the correct global context (e.g., `new Client()` / `new self[0].Client()`).

To discuss:

- Constructor approach vs global constructor
- Synchronous (for current) vs. asynchronous access (different context)

Andrew: Using the realm and just exposing the constructor makes sense to me. Seems consistent with how snapshotting state is specified and how Gecko implements it.

Youenn: A new client means that you can message it, so you can `postMessage` to yourself.

AS: I agree that is a weird edge case

Youenn: I would prefer “`thisId`”, seems simpler than exposing the whole Client object to a window. Don’t know if there were other requests for exposing Client/Clients more widely to windows.

Marijn: There have been various requests over the years for something like that.

Youenn: Still would prefer just the id, since that seems safer/simpler and probably good enough.

AS: That also works. I think it makes sense to expose the clients API to windows, but it would be fine to just expose the ID for now.

registration.ready promise - [770](#)

Monica: Provide a way to wait for the current Service Worker registration to have an active worker.

Proposed solution:

Add `registration.ready` promise

- Resolves when specific SW is active
- Useful for versioned SW scenarios

JA: I still think this is a good idea.

AS: Agreed, this still seems like something that makes sense. `ServiceWorker.ready` is a major footgun, `registration.ready` would be better.

Youenn: Good for tests, so probably also good for web developers.

JA: Vague memories of workbox having a polyfill for this. I know some developers create a separate SW for push messages, where page wouldn't be in scope. Having this would make it easier to wait for that worker to be ready.

Add name param to `Clients.openWindow()` method - [711](#)

Monica: Add a name parameter to `Clients.openWindow()` so that:

- Windows opened by a Service Worker (SW) can be referenced by name, similar to `window.open()`.
- This avoids complex `postMessage` chains between page ↔ SW ↔ new window.

Potential solutions:

1. Add a **name** parameter to `Clients.openWindow()`
2. `windowClient.Close()`

To discuss:

- Is adding name parameter enough, or do we also need a standardized way to query by name?
- [#1475](#) - Instead of adding a name, attaching data to clients and use web locks to elect single window as the handler (using client state should define constraints on size/shape but name parameter is easier (string))

AS: More general thing of letting a page describe some of its characteristics so that client match API can use it. But always the problem that if we specify a generic data mechanism, we also need to define quota etc. Name seems very specific and something I don't like. This proposal seems very specific and a more generic solution seems more useful. Wouldn't want to let a client set a GB of data on the client information.

JA: This would only be in memory, not have to be stored to disk?

AS: Yeah, I don't think it would need to be. Would be reasonable to limit to a small amount of data.

JA: Tricky case would be browser restart, should the information remain there. That's why navigation API has its serializing. Agree that general API seems like a better way, but has big open questions.

Monica: So something broader/more generic seems better than just name.

Youenn: What is next step for the more general solution?

AS: Somebody needs to do the work to figure out what happens with session restore, if data gets stored, etc. Easy way would be to limit to string.

JA: Navigation API already has done some of this, could do the same.

[12:30 – 13:45: 🍽️ Lunch Break]

Lunch will be available from 12:30 to 13:45 - Portopia Hotel, South wing, Ohwada 2/3, 1F

Topic: Web App Testability

General "web app" / "PWA" - <https://github.com/w3c/manifest/issues/1194>

- Test when a web app is installed. side-effect it enables
 - badging
 - push api
 - clicking links, what happens
 - go back into browser?
 - stay in app
 - Chromium - navigation capturing tests
 - "what we give the OS for things like name, icon, etc"
 - be able to return the parsed manifest.
 - e.g. test that the manifest id doesn't change.
 - maybe something that would work as a general web api instead of webdriver
 - "does stuff work"
 - do my videos play
 - does my sound work
- how much would webdriver fake, vs how much can happen for real
 - does the application get installed fully?
 - webdriver doesn't normally have the ability to install or uninstall applications
 - virtualized environment - install and uninstall. but how valuable is that?
 - chromium - we have integration test. We put them under the home directory, the bundles
 - but that's because chrome - they connect back to chrome, were as safari it would be a totally separate application
 - can you have two webdriver things and maybe communicate back and forth with each other?
 - maybe do a proper investigation of what WebDriver can support here for Safari
 - If not doing that, can we fake it enough
 - full fake? pretend you're installed.
 - launch webdriver in app mode?
 - how to enable various APIs
 - Application is separate from safari - they are compile time flag to enable APIs - faking it on WebDriver with safari wouldn't help, you have to install the app with the extra flags

- There is also tests that you would run in the app that is installed, vs testing the installation of the app.
 - first - you only have to run the webdriver test in this other thing.
- 3 things?
 - **be in the app**, check display mode stuff, etc. orientation / orientation lock
 - **be in a browser tab**, make sure the install works
 - can this just be the parsed manifest?
 - you wan
 - **browser <-> app communication** Most complicated is going to be the navigation capturing logic of click on A.com link in safari, have it launch / be handled by A.com installed web app.
- investigation needed

cloud gaming

- apis - keyboard lock, pointer lock, gamepad, fullscreen
- for gamepad, no way to detect if gamepad is connected for webdriver by wpt tests. Just doing API interface call testing. so working with webdriver bidi
- pointer lock - it's interface testing. there are no actual pointer events. Just returns an exception if not initiated by user gesture.
- pointer lock
 - unclear, but chrome and edge just start blocking the event
 - pointer lock test in the web test - no way to evaluate pointer lock API
 - Who owns pointer lock? We do.
 - FF has the same issue as Chrome.
 - pointer lock promise - implementation has been added 1st year. There is an api. But the test API also added on the web platform test - but we don't have a way to evaluate through web platform test. When you go to [WPT - dashboard - pointerlock api](#) - the tests.... it fails all the time. There is no way..
 - to enable the pointer lock, you have to get into the pointer lock mode. when we start pointer lock, it rejects because there's no user gesture. we can't use webdriver event for it - it's a different kind of user gesture.
 - os level user gesture
 - the test SHOULD work for pointer lock API, but we just don't mint an appropriate user gesture "thing" from the OS in FF / Chromium WebDriver implementation.
 - Failing:
 - https://source.chromium.org/chromium/chromium/src/+/main:third_party/blink/web_tests/external/wpt/pointerlock/pointerlock_promise.html
 - <https://wpt.fyi/results/pointerlock?label=master&label=experimental&aligned&q=pointerlock>

webapps mobile interop?


- just testing touch events is a lot
- gamepad automation?

- Chrome-side - this is owned by someone else, so we can't commit his time.
- We need to define requirements of what we need.
- PWA testing interop proposal [1139](#) "Installed Web App Webdriver Testing (Investigation Area)"
- **blocked!** we don't have enough webdriver testing expertise here...
 - how are browsers....
 - how are people checking installability
 - can we abstract that to something handled by webdriver - this is installed, extract appropriate apis for that
 - "can I use the push api" - you can only do that from isntalled web application - same with badgin api
- Push:
 - <https://github.com/w3c/push-api/issues/365>
- Badging
- Gamepad
 - <https://github.com/w3c/gamepad/pull/224>
- Fullscreen

Topic: Future of UI Events

- UI events [spec](#). It is a fundamental spec, has been around for 20 year. Was supposed to define how user interaction events worked on the web
- other specs came along, and started defining eventing model as well - html, dom spec, ... mainly in html
- This one - given its age - is difficult to maintain. How it does not have an editor.
- Duplicates many things in other specs. but important spec.
- What do we do with this specification?
- Should we delete things here? How much is defined in DOM, how much is defined in HTML.
- Ideas
 - deletion rampage
 - rewrite - start from scratch
 - pointer events - take all the mouse events from here, and specify in pointer events spec
 - work - investigate this some more
 - then take out all of the mouse events from UIEvents spec
 - Joint deliverable?
 - it would be in pointer events. These are legacy events.

- Wheel should be tied to scrolling? No - it triggers scrolling but different
- focus events belong in HTML spec
- Load event - why is that in uievent spec? load, unload, select, input. But input is in editing wg.
 - input should be in input events level X specification. editing wg
- UIEvents should keep keyboard events and composition events.
- <https://w3c.github.io/uievents/event-algo.html>
- Touch events are separate - was abstracted to pointer events.
 - There is still legacy simulation - you touch, then after a second, it will trigger a synthesized click. layout
 - If it's a touch event, then there are complications with where that specification is happening.
- how much would this remove if ____ would move to HTML
- What is special about UIEvents and what it's defining that's not already in HTML?
 - In the models?
 - There isn't a model. That's part of the problem. event happens, mutate dom, something something, unclear what is supposed to happen.
- Mike - I have a PR that looks right, but there is no editor, no reviewer at all. I came across this independently from implementing - and issue tracker shows that there are issues opened on this from 2018
 - example: <https://github.com/w3c/uievents/pull/392>
- Question
 - how much work do we do before move this stuff to html
 - or do we find a real editor to this spec
- Even if we move spec to html, we should move the issues if they're real issues.
 - so we need to go through them anyways.
- anyone interested in spending time every couple of weeks on this spec, triage, and figure out what is left to do
 - AI: Mike Smith <mike@w3.org> will volunteer to get something going to do some triage with folks. Happy to move something somewhere for better maintenance....
 - load events should be easy
 - move some stuff to pointer events
 - and then we remain with composition events and keyboard events
- HTML spec already has a model defined for focus events, that is reusable
 - html might be understaffed
 - this is an opportunity for us to do it right (in the html spec)
- we need an editor. no volunteers.
-

[15:00 – 15:30:  Break]

(drinks and snacks) will be available from 10:30 to 11:00 and from 15:00 to 15:30 - Kobe International Convention Center, Reception Hall, 3F, 4F, 5F

Topic: Badging API

Issues → <https://github.com/w3c/badging/labels/TPAC2025>

[Notification permission requirement #110](#)

[BREAKING CHANGE: remove permissions check #121](#)

Marcos: As said earlier, for the badging API you don't really need the permission. Operating systems let you call the API which will set the number but just not display until there is permission granted. We had tied badging to the notifications permission, but it's kind of nice for this to be under control of the end user.

Marijn: What we found is that the badge on macOS shows up if you never deal with the notification system, but hides it if you deal with notifications and don't have permissions. We do probably need some way for sites to know to request notifications permission.

Marcos: I think Safari just automatically asks for notifications permissions if the site tries to set the badge.

Marijn: We don't want sites to request notification permissions anytime they are setting the badge.

Marcos: Sites shouldn't do that, as long as browsers make sure to request notifications permission when the badge is set. On iOS it won't prompt when setting the badge, so we'd need to fix it in WebKit to request the notifications permission.

Spec used to optionally reject with NotAllowedError, but would be better for the browser to just request permission for the site and have setAppBadge always succeed (and just not show the badge if permission is not granted).

Marijn: If we expect browser to request OS level permission "magically" from the set app badge call to be called, we should call out in the spec that browsers should make sure that this permission is requested.

Marcos: That seems like a reasonable addition.

Vince: If you call setBadge in safari in an installed web app, will it just work?

Marcos: It will prompt the user, and the user can decline

Vince: Is there some way for the site to see if it was denied?

Marijn: Maybe they can see if the notification permission was denied?

Marcos: Maybe you can observe this by seeing if the notifications permission state changes after trying to show the badge.

Vince: What behavior would we want? Do we want sites to be able to know that they declined or not?

Marcos: It's not so much a privacy issue, but more an elevation of privilege issue. If you can set the app badge, you can now also show notifications.

Marijn: To the user it looks the same as if you just requested notifications permission

Marcos: It might be that badging for the installed app only grants you badging and not things like play sounds for notifications etc.

Marijn: I don't think the OS lets you "upgrade" your notifications permission request. You can only request permission once, and can't later request more permissions. So requesting just badging initially might not work.

Diego: On macOS there is no way to just request badging permissions?

Marcos: We'll need to figure out desired permission behavior, and can then decide what to do about it. Either way we probably want to get rid of the NotAllowed error.

Marijn: We could reject with NotAllowed if permission isn't there/is denied.

Marcos: Currently setAppBadge just resolves with undefined.

[Add 'set the application badge for installed web application' algo #116](#)

Marcos: Context is that declarative push needed a way to set the app badge for a specific application. So need some kind of identifier and a hook to then set the app badge. Just needs anne's approval to make sure this meets the push APIs requirements.

Kagami: Planning to implement declarative push, but hasn't gotten to it yet.

Marcos: Does Mozilla have any add-to-homescreen/app installation that would allow for badging?

Smaug: Not currently

[Change color of the badge #64](#)

Marcos: Is there interest for changing the color of the badge?

Marijn: Does this need to be different from the theme color of an app?

Diego: Could be, at least on Windows it depends on sometimes the theme system color.

Windows allows setting different colors. Use cases could be showing the status in instant messaging applications, or just going with the brand coloration. Theme color might not work since it could have to contrast with icon. So proposal is to add a hint to the badging API.

Windows and maybe some versions of android support this.

Marcos: On Apple's platform it makes no difference, since the OS doesn't support it.

No objections to this being added, it's just a hint.

SR: One thing to add might be to specify system color?

Marijn: Wouldn't that be not setting the color?

SR: Not specifying might be different from the system theme.

Marcos: Worth looking at.

Topic: Gamepad API

Issues → <https://github.com/w3c/gamepad/labels/TPAC2025>

[Interop 2025 investigation effort - Gamepad API testing #219](#)

[Integrate with WebDriver and WebDriver BiDi for automation #224](#)

Matt: Last year at TPAC we discussed having gaming as a focus area for interop. Turned into improving the testability of gamepad API. So maybe in the future we can work on improving testing rate as a full interop project.

Main problem is there isn't really any way to test gamepad behaviors. There is a way to test some parts, but not anything that actually involves pressing buttons etc.

Spent some time looking at existing gamepad testing APIs in the various browser. They cover many of the same capabilities, but all don't really test low level capabilities about how

gamepads are connected and how buttons/axes are mapped. Wanted to test something a little lower level than what browsers currently to.

Initially tried specing low level HID etc behavior, but the spec doesn't really define that either. So instead going from a report descriptor to a map describing buttons and axes might be good enough. Want to make sure that values are propagated correctly. Usually gamepads expose things as an integer range that needs to be mapped to the 0-1 range the API expects. Also lets you specify output capabilities, i.e. vibration.

Other things that we're not quite sure about is things like vendor id and product id. They are usually exposed in some way, but the spec doesn't say you're supposed to do that, so can't really test it. Trying to connect everything in the spec to the testing API, and not go deeper than that yet.

To hook it up looking at existing testing API; there are existing APIs for mouse and keyboard input, gamepad seemed similar to that so want to represent them as input actions as well.

Why bidi? For just input could do regular web driver, but there is output (vibration state), which can change based on action of website, so we need some sort of event to signal when vibration starts/stops.

WebDriver API is not currently implemented anywhere, but started some work on the backend in Chromium.

For actual tests, probably makes sense to start with the new testing API, rather than trying to clean up existing tests.

Also came up with a manual test suite, and ran through manual tests in various different browsers. Identified some issues, and filed those against browsers (mostly chromium).

Scored our interop outcome as 54% because of all that was achieved.

Implementation in Safari is blocked on lack of WebDriver BiDi support.

[Add a means to test the API #175](#)

See above

[Make gamepads accessible by web worker #37](#)

Matt: I believe from last years discussion there wasn't really any reason why we shouldn't expose this to workers.

Steve: I tagged it because we're interested in working on this. No concrete proposals now, but will probably publish an explainer in the future. A little bit more detail is that this is interesting for game streaming applications. Last missing piece to allow game streaming apps to move most of their logic into a dedicated worker to improve latency and performance.

Marcos: Question that came up on the WebKit site, use cases and scenarios were very clear but what about mouse keyboard touch pointer etc which are also used by games. Why don't they need the same support? We should be mindful of that, since we're changing one input type but none of the others.

Matt: No good answers for what to do about those either.

Jake: Difference is that pointer events support hit testing. Maybe when pointerlock is there mouse events could also go to a worker? You could do a similar keyboard capture.

Marcos: There is keyboard lock

Smaug: But it is not a proper spec yet, I think there is a counter proposal for that too somewhere

Marcos: But pointer lock and keyboard lock could be some kind of answer for what to do about mouse/keyboard events and how those could be exposed to workers.

Simon Pieters: It is only dedicated workers though? Since it might not make sense in shared workers?

Jake: What happens with RTC, if the RTC channels are transferable? Can they be sent to a shared worker?

Simon: I think so

Marcos: Good question, current proposal is just for dedicated working, can expand in the future if there are more use cases.

Jake: Locking APIs would only allow one thing to have a lock, so can only go to one page at a time. While gamepad is different, multiple iframes etc can access the same gamepad at the same time. So workers would be just another environment. For mouse it's a bit different, you can feed it deltas but what would you do with it.

Matt: gamepad is different from mouse, since it also has a vibration output

Jake: Trackpads also can have vibration

Matt: Multiple apps can send vibration to the same gamepad

Vince: Could make up a pretty plausible use case for why you'd want mouse events on a worker as well.

Matt: You might want to check keyboard and gamepad state in the same event loop, so can't just message between contexts.

Vince: Not sure how much more complicated it would be to bring over the other input events. Not even sure how scary it is.

Jake: If it is for gaming, mouse/keyboard are popular for gaming as well. Use case for doing this in a worker, is that for offscreen canvas?

Marcos: Yes, it gets too slow when having to post input over to worker and back.

Smaug: What is left on the main thread if you push everything to the worker?

Vince: Aren't there dom operations etc on the main thread that could be blocking?

Jake: Once rendering is done in the worker, the definition of what blocking is changes. If all rendering is in a offscreen canvas, now the worker is the main thread. If for gaming purposes the gamepad API should go directly to a worker it seems you should be able to do the same for mouse and keyboard for the same reasons.

Matt: I think I agree, although there are some differences where mouse and keyboard should follow system focus, so it's a bit harder.

Jake: I agree, you'd be requesting a lock to be transferred to the worker, but that lock could be lost on focus or whatever.

Steve: Will investigate and come back with answers

[Should fire events instead of using passive model #4](#)

Matt: Very old issue, had a previous attempt at this that didn't go anywhere. Trying again, this time only focusing on raw inputs/change events. Not trying to synchronize on requestAnimationFrame, just firing any changes as events. That should allow applications to stop polling and just listen to events.

One issue that came up prototyping, when do you fire events when the game loses visibility/lost access to gamepad inputs. Do you fire anything? Fire what was before it was lost or as it

regained visibility? If you're holding down the button when focussing the page again that should not be considered a new action, as there wasn't a button press.

Jake: So the question is what happens if button is pushed, then user switches away from your app. Does it get a "release" event, and how should it treat it?

Matt: In the polling model it is up to the site to do whatever they want.

Jake: Isn't it weird if change events get out of sync with what happens?

Matt: Agreed, but you have to be careful with not firing events when there wasn't really a button press, and somehow signal discontinuities.

Simon: Is the general idea that the page doesn't get updates when not visible?

Matt: Yes

Jake: But that means underlying model changes without change events.

Simon: So either release all the buttons when becoming invisible, or hold previous state until become visible again.

Matt: There is always going to be some discontinuity, sites could listen for visibility etc events, but easier if it can just be exposed via the gamepad API.

Jake: Is the proposal that you get snapshotted state in these events?

Matt: Not exactly sure, but I think event will be the current state and a list of changes.

Jake: The encouraged way to do it is for the event to contain the actions, but you still go to the gamepad for the current state. While the page is invisible we're not dispatching events, but the model does change.

Matt: Currently if you try to access gamepad while the page isn't visible you don't see any changes.

Jake: Is the rule that you dispatch events when the model changes, i.e. changes become visible to the page?

Matt: I think there should be some way for the app to know that a change event happened while the page wasn't shown, and that it shouldn't treat a button down as an actual button down?

Jake: Should be a field on the actual event object then

Matt: Some flag to say that you shouldn't look at this event.

Smaug: This isn't any different from keyboard events? Websites just need to handle this.

Simon: Other case is holding down button while active, release and re-press while not active, and then go back to the page. That seems like the button having been down the whole time, and should not fire a change event.

Jake: If model stays the same this doesn't feel like a change.

Jake: How do keyboard events handle this?

Smaug: No keyup event, but there is a blur event that sites can handle to reset their state.

Steve: We're actively prototyping this, any feedback is welcome.

Jake: Need to make sure that change events happen after focus.

Smaug: Feels a bit weird to clone the gamepad state into the event. Feels like it should be a different kind of object/expose some static values. Snapshot shouldn't be an EventTarget, so not a pure Gamepad object.

Jake: Do we need a snapshot in the event at all?

Matt: It seems useful, but maybe not necessary. You could use the polling interface to get the gamepad state.

Jake: Event can just give you the actual "live" Gamepad object rather than an index and/or snapshot of it.

Matt: Maybe snapshot was from an earlier idea, since it wasn't clear if we were following the live object model or snapshot model? Gamepad is not live, it is a snapshot.

Steve: Also difference between firefox and chrome, where in firefox it is a little bit live.

Jake: Can always add a to-json method if you want an easy way to get a snapshot, but doesn't seem a problem here.

Matt: The event target is window, so that doesn't tell you the gamepad.

Gabriel: But if gamepad is not live object, can it be the event target?

Smaug: Explainer lists gamepad as event target

Steve: That is out of date, we originally thought we were going with a live object.

○

Day 2

Web Applications WG TPAC 2025 Meeting - Tue 11 November 2025 <Day 2>

Participants

(Please add your name!)

- Ananya Kittane Yogananda <a.yogananda@samsung.com>
- Andrew Sutherland <asuth@mozilla.com>
- Bryan Ellis <erisu@apache.org>
- Carlos Jeurissen <carlos@jeurissen.co>
- Christian Liebel <christian.liebel@thinktecture.com>
- Dan Murphy <dmurph@chromium.org>
- Daichi Asakura <daichi-asakura@cybozu.co.jp>
- Euclid Zhizhen Ye <euclid.ye@huawei.com>
- François Daoust <fd@w3.org>
- Haruki Kinoshita <kinoshita.h-es@nhk.or.jp>
- Hisayuki Ohmata <oomata.h-ik@nhk.or.jp>
- Kagami Rosylight <krosylight@mozilla.com>
- Kristin Lee <kristinlee@microsoft.com>
- Lu Huang <luhua@microsoft.com>
- Matt Reynolds <mattreynolds@chromium.org>
- Niklas Merz <niklasmerz@apache.org>
- Samuel Maddock <smaddock@salesforce.com>
- Steve Becker <stevebe@microsoft.com>
- Tantek Çelik <tantek@mozilla.com>
- Vincent Scheib <scheib@chromium.org>
- Marcos Caceres
- Xiaoqian Wu <xiaoqian@w3.org>
- -- add your name above alphabetically --

Chair

- Léonie Watson
- Marcos Caceres
- Diego González <luigonza@microsoft.com>
- Marijn Kruisselbrink <mek@google.com>

Scribes

- [_volunteer_](#)
- Dan

Logistics

<<https://github.com/w3c/webappswg/wiki/TPAC-2025>>

Agenda

<<https://github.com/w3c/webappswg/wiki/TPAC-2025#-tue-11-november-2025>>

Minutes

Topic: Web Share API ([issues](#))

[Web Share API images have optional title, but seem to be missing alt. #285](#)

- Does mac have the ability to receive alt text from share intent thing?
- Windows
 - [DataPackage Class \(Windows.ApplicationModel.DataTransfer\) - Windows apps | Microsoft Learn](#) (no support)
- Android MIGHT have a way to provide this via [extra title](#).
 - Unclear how this is rendered - seems like often a subtitle.
 - There seems to be a "android:contentDescription" that can be specified, so that is a possibility, unclear.
- MacOS - you can provide a title for each individual imageDataPackage Class
- It seems like the right approach to have the alt text / accessibility is to make it part of the file object

[Provide a means to check if the API is supported by the platform or OS #286](#)

- navigator.share requires user activation so it's expensive for devs to call it
- navigator.canShare could be called instead without a parameter and that could fail if the API is not supported (and success if supported), but is that backwards compatible?
 - Currently it always fails: ["If none of data's members title, text, or url or files are present, return false."](#)
- Investigation needed
- maybe remove the default value, but that's impossible per the webidl spec... can't make it nullable either

- maybe check the members and see the object is empty, but that wouldn't be forward compatible if we add more members (`{ foo: true }` would be an "empty dict" for the current impl but not necessarily for the future impls)

[Web Share API fails to share a fully loaded data object/partial data object with file #279](#)

- Depends on each app implementation detail
- Not really actionable

Aside

- macOS downloads content from the shared URL, but we want to guard it so that it happens only after user interaction
- We could add something in consideration section so that impls disable the content preview (safari did)

[11:00 – 11:30: ☕ Break]

Topic: Web App Manifest ([issues](#))

[Web Install \(#1178\)](#) & [Add navigator.install\(\) \(#1175\)](#)

- Frustration with UI for add-to-homescreen. Nice if there was some imperative means of doing the install
- previous - beforeinstallprompt from Chrome
 - basis - it would show a banner (kinda mobile-y) - prompt
 - you don't want that from showing while user is doing something, so could trigger it via button click etc
 - Safari doesn't support it, and supports a 'smart banner', and you use it by not adding the manifest link until you want it to show
- wouldn't it be cool for navigator.install?
- idea: Now we have PEPC - what if we jsut had a button, html button you could just press?
 - took the API shape from install, to an element
 - no formal proposal - but alternative approach
- beforeinstallprompt - a lot of usage - so webcompat issue potentially
- navigator.install gives us a clean slate, similar to install button as well
- but doesn't get rid of legacy code that relies on beforeinstallprompt - if they decide to do a banner /prompt using that API we would need to support that being shown.
- other things
 - wow the beforeinstallprompt api is really confusing and hard to use
 - in the past it wasn't reliable, but now it is
 - partners that are testing the install API
 - feedback about how much smoother the experience is (for background page install)

- TAG-side
 - feedback to keep it super simple
 - API has gone through several shapes. One had a two-way handshake between manifest file and install source
 - 2 or 3 tag reviews
- imperative API now pretty simple
- Christian
 - I would not like to wait for PEPC button - I would like to use API - didn't seem done
 - seems fairly new
 - Marcos - it's not PEPC, it's just the model
 - navigator.install, 0-params
 - fallback plan - reuse beforeinstallprompt - the API is not nice, confusing. But only have to write it once.
 - worry here is that , if it was an element, it wouldn't be available soon
- Diego - unless we have buy-in from other browser engines on new model, it would be an API on another experimental API
 - fallback version would be needed if not all user agents supported
 - does this mean that if we had more swes that could implement on all user agents it would reduce risk?
 - marcos
 - the imperative api thing might be red herring
 - if it's always going to be a button, we should just make it a button - there is no change for the api surface from navigator.install vs just being a button.
 - any reason why it should be attached to navigator.install vs just a button
 - if you had an install button that uses beforeinstallprompt, you can just wrap this with this button itself.
- Rob
 - There is one area we have been trying to explore - 'different document' install
 - Button for navigator install for 'different document' install is hard
 - install_url, manifest_id, hard to show UX for install button in a trusted way
 - load install_url in background
 - find manifest link
 - load icon in manifest
 - (lots of steps)
 - don't want to fetch the install_url background fetch until user intervention
 - ideas
 - extra manifest_url on install element
 - moving to something crazier like one ap per origin
 - having two clicks - one click fetches the info, the next installs
 - or - it would open the document in a popup or something
- Diego & Rob - parameterless version is what is in scope for this convo

- Christian - we can extend the install element later on, we can add whatever we want to add there.
 - Element design seems risky to rely on if our design isn't final.
 - we should give ability to devs to decide what the install button/add-to-homescreen button looks like
- Kagami
 - is there risk for spoofing, with API?
 - Marcos - the API allows you to lie much more than the element does
 - Diego - we can take HTML and CSS and and trick user to click on install button... to ask for something else.
 - so imperative, dev can create button for not installation, and then clicking on it would trigger install UX
 - for element, because devs can mimic that look and feel
 - e.g. we shouldn't have trusted UI at all in an untrusted viewport, as if the user gets used to that, they malicious websites can mimic that and be able to get a 'click' on something (to do something else, like get microphone)
- Marcos
 - Let's give ourselves a month to come up with an element proposal, run it internally to see if people are interested.
 - If it flies, then we should seriously consider the API. We already have a pull request, it should be easy to implement.
 - (added from memory after meeting, need to confirm) Kagami: I think I like declarative
- What prevents merging?
 - ID or no ID required in the manifest
 - problem - the start_url changing causes multiple apps.
 - so we want to require an id in the manifest file so that we help encourage the ecosystem to not have this huge problem.
 - <active discussion>
 - We can solve to say user agents will have different requirements.

[Way to detect if a web app is installed \(#1092\)](#)

- proposals?
 - current document is in installed mode...
 - installed: yes
 - navigator.standalone might bring web compat issues for Chrome
 - Sam - I would probably prefer a media query
 - installed: yes
 - so this should NOT be true if you are installed but in a browser tab
 - but what about open in browser tab?
 - so - we can get away with this by having installed: yes, display: browser
 - yeah that's nice
- agree?

- only standardize `installed` media query, `installed: yes or no`
- `getinstalledrelatedapps`
 - not today
- Kagami - the installed media query means - people could consider it as "no back button" while Firefox is going to show one in browser chrome
 - Sam
 - the install media query is to know if you're in a app or browser tab
 - e.g. the install button showing or not (rather than back button)
 - the display mode media query is to adjust UI
- AGREEMENT EVERYONE GETS A STICKER

[Web Manifest Overrides \(#1045\)](#) &

[Add support for defining a theme color for both light & dark modes \(prefers color scheme\) \(#975\)](#)

- extensions
 - each image has properties...
 - When a browser wants to get an image that satisfies X constraints, then:
 - the browser starts at the top - does it support these conditions? If it does, then use. If not, use the next one
 - Go until it finds one with matching conditions
 - allows you to easily stack additional conditions
- mek@ it's very confusing when there are 6 different constrains, but you only have some icons for each
 - if you want 512x512 monochrome, but there is only a 256x256 monochrome, what do you do? If you have a full color 512x512....
- Christian - we cannot introduce another scheme now.
 - so if we do `"color_schemes": ["dark", "light"]`
 - if user agents don't support this, then it might pick the 'dark' one?
 - The spec says to pick the first
 - so spec says you would have to put the dark mode icons at the bottom.
- Proposal:
 - 'color schemes' property
 - spec text around putting at end
 - we know that this can be compatible with browsers / old implementations that don't support it because browsers consider earlier icons first
 - change does into.... image resource? or manifest?
 - Yes - image resource - Diego will spec
 - how to do theme color with color schemes?
 - `"colors": [`
 - `{ "background_color": "darkred", theme_color:`
 - `"darkblue", "color-scheme": ["dark"]}`
 - `{ "background_color": "red", theme_color: "blue" },`
 - `]`
 - "Colors" might be better than "themes", since we're only putting colors in there.

- similar format to icons
 - global "themes" - array with members that have the color theme, other properties - and in this case, background color
 - color-scheme, background_color, theme_color
 - same as icons algorithm - browser finds the first one that is bestly appropriate
- Sam - this is not necessarily matching for web, this is for the platform. So - 'material theming' - that's not in css
 - it should match semantics of css, maybe not the name?
 - how do we represent things beyond just light and dark mode
- aside "if there is a key in here the user agent doesn't recognize, skip the entry"
- Carlos
 - this is implemented in Safari
 - Chrome is implementing?
 - unmerged patch on firefox
 - https://bugzilla.mozilla.org/show_bug.cgi?id=1809958
 - marcos - what is the monochrome solution?
 - it would be a condition, it would fit on a condition
- Limin
 - can see a flicker, rendering background then turning to black
 - certain apps.... sometimes when the user selects a theme... the site needs to be able to run javascript before determining the color to render
 - how to solve?
 - SW register something?
 - How do we know the color before rendering?
 - Dan - you can do this by using a dynamic manifest
 - serve a different manifest based on cookies or filename
 - you can add a query param to the manifest url, and the server can use that to change the contents?
 - But this might be bad - other ideas
 - js api to save color
- AGREEMENT EVERYONE GETS A STICKER
 - just for icons
 - we need a hero for proposal for the theme color / background color thing
 - christian.liebel@thinktecture.com wants to take a pass
 - will work with Carlos.

Did not get to

- [Add API to retrieve document's computed manifest \(#1179\)](#)
- [Discuss PWA Widgets at TPAC 2025 \(#1195\)](#)
- [Scope: Allow developer to have more fine-grained control of app scope. \(#996\)](#)
- [PWA Migration \(#1189\)](#)
- [Navigation Capturing / Management \(#1196\)](#)

- [WPT/dev testability: Create way to test features related to / currently dependent on install \(#1194\)](#)
- [Manifest-url-only install. Perhaps standard .well-known/manifest.webmanifest \(#1193\)](#)

[13:00 - 14:15: Lunch]

Topic: IndexedDB ([issues](#))

[Slides](#)

<https://github.com/w3c/IndexedDB/labels/TPAC2025>

Steven Becker:

Progress since TPAC 2024:

- Indexed Database API 3.0 remains a working draft.
- Added new getAllRecords() API.
- Introduced more specific errors for disk read failures.
- Addressed ~20 spec bugs with updates.

Also improved test coverage. Several areas that are hard to test with WPTs though, around corruption and other disk usage scenarios. Should we have tests for those?

Jan Varga: Yeah, I think it would be good to have tests in that area. There are many cases in Gecko where users report problems. Gecko has special tests for some of these cases. If possible, it seems like a good idea to test edge cases for this.

Evan Stade: I believe we'd have to add something to test harness to simulate these situations.

Andrew Sutherland: Yeah, seems feasible/a good idea

Steven: Okay, we'll think about this more. Distinguishing between bad database and bad values, a lot to discuss and spec out in this area.

Fergal: For large database, would it make sense to have some kind of pre-populated "big database" that many tests can run against without having to spend the time to set it up.

Steven: Yeah, that's the idea, but different browsers might have to do it differently.

AS: In Gecko we lower quota for these kinds of tests, so we don't have to create multi-gigabyte test files.

Steven: Agree, running with an actual full disk would have other problems.

Fergal: So should we agree on a common quota?

Marijn: Or a webdriver API to "set" quota?

Steven: Yeah, something like that.

Jan: Buckets also let you set quota, so that might be easy. In Gecko we set a preference which lowers quota, which works in a unit test but is harder in a full browser for WPT, since this pref would effect everything, not just the test. Gecko also has tests where the entire profile is part of the test setup, to test things like corrupted databases. Since creating a corrupted database is not easy. Also for upgrading database on disk.

Steven: For next steps, we can file a github issue for how we think this might work, and go from there.

Evan: Not just a question of how to test it, but also we should specify how specific errors are handled.

Andrew: The storage spec is very clear that data clearing is per bucket; we can't rely on sites to reliably handle partial data errors in general. This would need to be something sites need to opt in to. The storage standard pov is that if something bad happens the entire bucket gets wiped.

Steven: The IDB spec only mentions corruption once. There are some apps out there that might be interested in trying to recover from corruption. Testing is a good first step, and determine how implementations handle this today.

Evan: The spec says that whenever a bucket gets cleaned it should be cleaned in its entirety. Does any browser do that?

Andrew: In gecko, for clear site data yes we do that. We have a service worker mitigation. We have a number of bugs on file to wipe the origin whenever any storage api encounters corruption. Previous proposals by Ben Kelly about notifying sites when storage might be about to be cleared etc. Gecko currently doesn't proactively wipe all data when corruption is detected. Also just start throwing errors if corruption is detected after open, rather than clearing. Having WPT tests would be good to ensure interop.

Evan: There is a push to delete all of DOM Storage whenever IDB has corruption (for an origin).

Andrew: That is what the spec leans to. There are things like workbox. It makes sense to assume interdependencies between IDB and Cache Storage, where cleaning one but not the other can cause problems where if a site doesn't explicitly opts in to being able to handle this clearing all would be safer. Cookies seem more separate from this, and even local storage might be different and safer to keep alive.

Evan: I think it would be surprising to developers if all storage gets cleared because one storage API got corrupted. But do agree that sites are likely not handling these edge cases where half the data would be deleted. Localstorage doesn't really have a way of reporting errors, there we just silently pretend the data isn't there. We've observed where site owners think that localstorage is super reliable and never has issues, which might be because there is no error reporting. No errors so it must be working.

Andrew: We've broken websites by throwing on every localstorage access.

Steven: Yep, seems like there is work to be done, will follow up on github.

Autocommit Transaction w/o Handlers [#482](#)

Steven: A potential optimization to be able to complete transactions sooner.

Sketched out next steps in the bug, need to figure out what breakage may occur.

Fergal: Other weird case is people are doing transactions in page unload handlers, which can't succeed without this, since the page would be unloaded before they get committed. This would let those transactions start working, which could also be a breaking change.

Andrew: Definitely interested for Gecko. Even more interested if other people get the data. Conceptually seems fine.

Speedometer

Steven: Microsoft has a proposal ([TodoMVC indexedDB workload for Speedometer](#)) to add IDB workload to Speedometer.

Adds IndexedDB workload to todoMVC.

Introduces pagination with 10 items per page.

Reads chunks of 10 items from IDB.

3 Measured Steps: Write, Update, Read & Delete (see slides/doc for details)

Fergal: Not familiar with speedometer. Is there a reason the goal isn't to measure IDB background operations time?

Steven: The proposal does discuss that option. The goal of speedometer is to measure responsiveness of real world application, so background work doesn't directly impact that. But yes, does seem it would be nice to measure background work as well.

Fergal: Some argument that it does interfere with responsiveness. While writes are pending you should stop navigating away to prevent data loss, so this would be a visible slow/measurable user interaction.

Steven: Good callout, also related to previous issue. Agreed.

Steven explains implementation details from slides.

Steven: Test feels pretty trivial and small. Maybe could add variable for db size and maybe have multiple dbs and more complex interactions.

Evan: If there are concerns would be good to engage with the speedometer folks working on this. If there is something that storage folk would find useful, speedometer people would like that input.

Andrew: Seems primarily test for lower level storage overhead, i.e. things like opening a database. Seems useful. Previously google had created a performance test for IDB that also seemed interesting. For things to optimize in IDB we found cursor prefetching to be a big win, it was nice to have something to show that that actually had real world impact. Based on what is described here it seems this would be useful to help optimize db opening latency, but not much more.

Evan: Agreed, since this is a small database, the actual reading/writing time might be minimal. IndexedDB is slow. Since you mentioned cursor prefetching, chromium does that too. If we do commit this speedometer test, that gives an incentive to go and change parameters to optimize for this specific test, which could cause implementations to overfit on this test rather than real world workloads. So we should be careful of what we actually test for.

Andrew: I think we prefetch much more than 10 records, maybe even based on size of the data. But agree that that is a concern for any benchmark. People optimize too tightly. But this doesn't sound like something we'd optimize too much for.

Explicit Control over Transaction Lifetimes [#34](#)

Steven: Discussed this last year. Interaction of IDB with promises is hard to make sure the transaction doesn't commit with outstanding promises. Would need to opt in to manual commit, probably combined with a timeout for stuck commits.

Andrew: This still makes sense from Gecko's perspective

Steven: Didn't have the chance to work on it last year, maybe this year

Jan: Time out is important. One of the reasons for autocommit was to not let sites break themselves, since it is easy to forget to call commit.

Andrew: Need more discussion on [#425](#), thought that was explicitly for stuck renderer processes. Which could apply to this too, but good to get a little bit more specific what that would mean for this.

Jan: Something like after processing last request set a timer with some timeout. Also when there was talk of a sync API for workers, there was also some discussion on a timeout or something.

Evan: Agree that originally commits were automatic because of avoiding footguns. But IDB also predates promises. Primary motivation here is to make IDB work better with promises. IDB isn't in the shape it would be if we would design it today.

Jan: Aside are there considerations for some kind of simple async key value store, there were some thoughts within Gecko.

Evan: Yes, definitely. Although the presence of many wrapper APIs that already solve this kind of lowers the priority of doing this in the browser. But those wrappers tend to use promises, so figuring out transaction lifetime with promises is relevant here.

Jan: Long term perhaps provide a promisified API, since that is what people expect today.

Evan: Definitely on folks radar.

Steven: If you have an internal proposal, definitely interested to see that.

Andrew: I think Jan is referring to Dominic's KV storage proposal as a wrapper around IDB. I don't think there was a Gecko proposal. Really like the "convenient storage" name for that, since it seemed nice.

Jan: IDB kind of has a bad reputation because of the complex API. But I guess there are no plans to replace IDB with a different API. Should do some thinking long term to make it more appealing. But this is all long term.

Evan: I think everyone is open to that, just a matter of prioritization.

Batch Requests [#69](#) [#376](#)

Steven: Discussed previously, polyfills exist so apparently there is some usage for these kind of APIs. Can have batched reads and batched writes. Some edge cases depending on how your object store is configured. What types of requests should support batching? Everything? Just get?

Andrew: One interesting thing (besides that chrome experimented with batch puts and seemed to not be a win), other things might give the same performance benefit. Issue [#477](#) about the spec not handling constraints errors, might need some spec cleanup for a batch put to correctly handle auto-increment keys. One thing we have seen in firefox is that structured serialize overhead can be significant. Doing more serializing in one task could cause jank. WebIDL Async iterators could allow chunking up data more in a way that is friendlier to the event loop.

Steven: Yes, this is about cramming more into a single task. Could get you data sooner if your event loop is busy, not having to wait for multiple events.

Andrew: Definitely something to be said for getting data at once for reasonably sized requests. Also fine to let sites do bad things but work.

Steven: I think next steps are to prototype and measure possible performance gains.

Andrew: Back to the speedometer thing, it would be interesting if those tests could show the benefits and overhead of having requests smeared out. Would be more like A/B testing with speedometer.

Steven: Could use getAllRecords to get data more efficiently in speedometer. Is an interesting idea. Hopefully build of initial speedometer proposal, although slower moving with changes only once a year.

Evan: Will also say that as far as chrome investigating, there were some promising results, but then got deprioritized. Mostly didn't go forward because priorities being shifted around.

Andrew: Good to know.

Daniel Murphy: Yeah, implementation never got to the point of really doing batching.

Inactive Transactions During Key Serialization [#476](#)

Steven: Edge case with different implementations. Spec PR says we should resolve this by making transaction inactive during serialization.

Andrew: Many thanks to Nolan Lawson for bringing this up.

Request Priority/Preload? [#428](#)

Steven: One reason sites use localStorage is that it is faster to get initial data during app boot. What could we do to help here? Needs lots of analysis, maybe could even be addressed in browser implementations.

Andrew: Talking with engineers on our team, we really didn't like request priority. But having something on the storage bucket that something is important/needed fast sounds good. Setting priorities on requests is problematic with things getting reordered and still be well defined. Transaction might maybe be okay, otherwise seems messy.

Steven: Yeah, priority on storage bucket could be incentive to adopt storage buckets.

Daniel: Is there a way to predict that a bucket is used on a given page, and preload for it?

Steven: I don't know

Andrew: Something we thought of, if we ever have enough time the browser could try to learn what kinds of data a site/url tends to load early and auto optimize for that. SW static routing is interesting here in being the first kind of example of explicitly specifying things. Haven't worked too much with preload cache and adding things there. Could have hints on storage buckets too.

Jan: Real speed of localStorage is preloading the data. In gecko data is preloaded from localStorage as soon as a request for a url is started, so data is ready when page asks for it.

Since many sites try to access localStorage very early on, so performance tests regressed without this preloading. If the question is can we bring IDB performance closer to this?

Preloading would be a lot of work and consume a lot of memory, not sure that is the right direction. Is the goal to get rid of localStorage long term?

Steven: I don't think we'll ever get there.

Jan: Seems good to do something about this, but not clear what can be done. Should spec be explicit about this or should browsers just do whatever optimizations make sense for them? For example using statistics about what data is used a lot, and preload that somehow. From an implementation POV this doesn't seem easy. If you want to bring IDB speed closer to localStorage we'd need something.

Temporal and Boolean Keys #429 #76

Steven: Temporal implementations are becoming available, while adding that could also think about adding Boolean, which was requested longer ago. How to compare these keys with existing key types?

Andrew: For Gecko both make sense. Last year for Boolean was said “wait for Temporal”, so now both exist it makes sense to do them. Want to double check on proposal for ordering. And should specify how it works.

Query Engine Support #403

Steven: Recent activity, which asked about potential DocumentDB collaboration. DocumentDB kind of looks like IDB, but definitely not 100% compatible. Any query engine support would need to be able to deserialize record values, which would need JS engine, so this would be more about convenience than performance.

Andrew: Gecko is a hard know about this. WASM and sqlite make it very feasible to have a query engine if a site needs that. The way Gecko is architected query engine still wouldn't run close to the data, so any of the benefits of a query engine wouldn't really be there.

Mike: Are there any implementers that are actually interested in this?

Steven: Don't think so

Living Standard #402

Steven: Should we adopt the Living Standard model instead of IDB 3.0 CR.

Marcos: All the other specs kind of have the ongoing thing. What does the community expect when they see a new version number? Do they expect breaking changes? The versioning mostly serves as a marketing purpose, or does it not have a purpose at all here? For HTML we dropped the 5, and for all other things we also dropped the version because the web is unversioned. Why does IDB need to be versioned?

Steven: That's what I'm wondering too. IDB has done a good job marking the differences between versions.

Jan: I think it is a good idea, because I think the version was never important for anyone.

Andrew: Yeah, living standards are good, I only ever look at the ED for anything. There are a few things Gecko is behind on, it's good to be able to feature detect a certain set of functionality, but don't think a versioned spec is essential for that.

Jan: It's always about does this work in other browser, or do we support this new feature. It was never about do we support this version of something.

Daniel: It was way easier to sell “ship IDB 2.0” to higher management as something we need to work on, rather than arguing for individual features.

Andrew: I think interop scores are very motivating now for us.

Evan: For my education, when does “.tentative” get dropped from a WPT test?

Marcos: When there is agreement that something is part of the standard by multiple implementers. The model we're supposed to be following is that it shouldn't be in an editors draft unless there is enough buy in, so if it is in the editors draft the test should no longer be tentative.

Mike: Just in general, this is just one aspect of that, we shouldn't be merging things into spec that we don't have at least two implementers that have expressed interest.

Evan: When we have a spec PR and tentative tests, when you merge the spec PR should also remove the .tentative from WPTs. Asked because I agree with what Dan said. If you're trying to get a group of people to dedicate resources, it is harder to argue for 5 small things rather than one big thing. Using WPT/interop there is kind of a chicken and egg problem where there needs to be buyin before things are counted in stats.

Mike: Happy to help figure this out too.

Topic: W3C Process feedback - AB Visitor

W3C Process Document: <https://www.w3.org/policies/process/>

[DanAppelquist introduces the AB, the TAG and the Process Refactor project]

DanMurphy: I don't think I've ever taken things to CR

Marcos: chairs and the team usually take care of these kinds of administrative tasks
... I'm hoping the editors don't need to worry about the process at all
... why do you think it will be a problem for editors?

DanAppelquist: we have a big Process Document that is hard to engage with
... having a simpler document might be beneficial for the groups

Marcos: Editors might be interested in testing, decision making for PRs, objections

Marijn: what do editors need to know about getting to CRs?

MikeSmith: wide reviews from different groups, how to respond to feedback

DanAppelquist: in WebAppSec, MikeWest made a number of comments, for people who not only care about the Editor's Draft, e.g., lawyers, care about IP; also governments, some entities, they care about W3C having a Process for shipping documents to a snapshot, it has to have some records, for other entities

MikeSmith: for people developing specs or implementing the specs, why do they care about CRs?

Marijn: agree, managers seldom care about making CRs

MikeSmith: working in a CG might make things way easier

DanAppelquist: we are hoping to improve the Process to make things easier or more efficient for everyone, e.g., to get IP commitments

... if there is WG consensus
... there can be Process changes that fit your WG working mode

DanMurphy: after someone commits a change, what matters to me is getting review

Marijn: right, ideally we should get horizontal review for every PR

MikeSmith: I've an idea to set up an editors' team
... if you submit a PR that relates to Privacy, you should ask the PING team in the PR one-on-one instead of waiting for 6 months
... it doesn't need to be W3C wide, we can have some minor experiments

DanAppelquist: perhaps you can move to APA to see if they are willing to give it a try

DanMurphy: it's important to understand what will be blocking feedback
... if there would be a disagreement
... perhaps adding an FYI flag

MikeSmith: for everybody involved in the Process, we need to make sure DO NO HARM
... doing wide review, it should be a feature, not a bug
... PR, don't merge things, it should be up to editors to decide

DanAppelquist: I like the DO NO HARM suggestion, we want to make it simple, make it clear, it should be up to editors, up to WGs, to decide how they want to do things

FrancoisD: we are improving the tooling as well, to make things more automatic
... e.g., the Web Features project, WebDX

Marijn: every web feature should be a WebDX feature?

FrancoisD: that should definitely help

MikeSmith: it's hard to explain to people about terminologies
... CR, can we come up with a better term?
... I offer some alternative terms, some other people did as well
... in the Process Document repo

DanAppelquist: I agree with you, they are difficult terms

MikeSmith: thank you for doing it, and doing the outreach

DanAppelquist: we are doing this under the [ProcessCG](#)
... please come to talk to us

DanMurphy: explainers, it'd be great to have a tool to help understand the document

Marijn: an AI tool?

DanAppelquist: there was a breakout session yesterday about explainers

... some people pushed back the idea

... why do we need an explainer?

DanMurphy: you should think about use cases

Marcos: the purple boxes in the Updatable REC are problematic

... in the Process, REC with amendments, forces editors to write the spec in a way that gets too complicated, become a burden for the editors

... we hope this can change

... if the wide review groups need the purple boxes to identify the changes

... Geolocation is an example, we can't modify the spec for 6 months because of the review

... we want to move the CR immediately after REC with Proposed Amendments

MikeSmith: not many people know we can publish CRs right after moving to RECs

DanAppelquist: yes that's exactly the frustration we want to avoid

Marcos: we did it for the Payment Request API

MikeSmith: if it's something people are doing in reality, we should remove the restrictions in the Process

[16:00 – 16:30: ☕ Break]

Topic: Screen Orientation ([issues](#))

Related to [Add can-lock-orientation media feature to detect if locking is possible #206](#), there is PR [Add can-lock-orientation CSS media feature #273](#)

- Marcos: screen lock should require user activation
- Marcos: There was an argument that this API is bad - we should merge this into fullscreen as it's already very deeply connected with fullscreen api
- Vincent: user activation should be able to be consumed specifically by screen lock, i don't want sticky activation
- Marcos: when fullscreen is exited the screen lock should be unlocked, which means it's tied, same for keyboard/pointer locks
- <https://github.com/w3c/pointerlock/issues/85>
- Marcos: Implementation is done but not specced properly

- Marcos: If you exit fullscreen you also want to trigger the orientation change animation to fit with that
- (back to can-lock-orientation)
- Kagami: I'll ping Makoto Kato for can-lock-orientation
- Resolution: Web apps should get reviewers and then send this to a CSS specialist.

PR [Standardize window.orientation API #271](#)

- window.orientation is in the [WHATWG compatibility](#) spec
- (multiple people: why is it in compat.spec.whatwg?!)
 - The compat spec is catch-all
 - Euclid (Zhizhen): (question about compat spec and standardization)
- Marcos: We want to describe the behavior better
- Vincent: The new screen orientation API is similar but can't be used interchangeably; the values are different
- Resolution: Good idea, get reviewers on PR and submit it.

PR [Refactor fullscreen rejection handling to use WebIDL react pattern #269](#)

- Marcos: needs review from mostly Mozilla
- Resolution: Kagami will review PR.

Issue [Require a gesture? #210](#)

- Discussed requiring a user gesture, and decided it introduced too many difficulties. E.g. entering fullscreen requires a user gesture, and then orientation would need to wait for another gesture.
- Vincent proposed having a gesture that can be consumed per feature, e.g. for orientation it can be used only once, but the same gesture could be used for other features still.
- Resolution: Closed PR and issue.

[Promise for unlock\(\)? #104](#)

- Marcos: The idea is being able to lock orientation and then unlock later. However, complications include race conditions.
- Dan: Android doesn't have an asynchronous thing to wait on.
- Marijn: Throwing a promise could mask exceptions that web apps may rely on
- Resolution: Close issue, depend on exitFullscreen to also unlock orientation.

Topic: Navigation capturing

Live demo of <https://developer.chrome.com/docs/capabilities/pwa-navigation-management>

- Demo app: <https://googlechrome.github.io/samples/pwa-testing/chat-app-deep-linking/>
- Showing previous behavior of clicking links to web apps and having a new tab opens each time. However for some apps that isn't desired.
- With navigation capturing, can click many links and refocus the app and then navigate within that existing window.
- E.g. slack, we don't want to open multiple windows of slack

- Marcos: Trying Safari app added to dock. If in the web browser, if you are in scope of an installed web app then safari & Chrome offers the option to open the current page in the installed web app.
- **Aside from Day 1- Navigation Capturing chat**
with Jake Archibald <jarchibald@mozilla.com>, Simon Pieters <zcorpan@mozilla.com>, Dan Murphy <dmurph@chromium.org>, Marijn Kruisselbrink <mek@chromium.org>
 - Navigation capturing happens **before** something like navigation API happens.
 - the LaunchConsumer is a kinda worse version of the navigate event.
 - Extra reqs - we need our information to be visible - happening on startup of app
 - idea
 - add file handles to navigation entry
 - we can define this as an observable or something that always queues up the events for the dev, and when they observe, they get all of them
 - And the one happening on page load can just be accessed on the current navigation entry.
- After deciding 'focus existing' or 'navigate existing' then Navigation api is about same-document navigation.
- Launch handling (chromium-only) should be polyfillable on top of the newer navigation api
- Specify the navigation capturing?
 - There are options for 'navigate new' and 'focus existing'
 - Specify which link clicks are capturable?
 - See last TPAC presentation
 - <https://www.w3.org/2024/09/WebAppsWG-TPAC2024-Meetings-Minutes.pdf>
 - Presentation <https://bit.ly/pwa-navigation-capturing-pres>
- Marijn: Unsure this should be specified and consistent across agents. Browsers may want to make their own decisions about what e.g. shift clicks mean.
- Dan & Marijn: What might be spec-able:
 - Is it capturable
 - What app controls this URL
 - Marcos
 - when coming from the web, it should probably stay in the web. only links from native apps should go to apps.
 - Dan & Marijn
 - Great, so by that reasoning, what about coming from Notes or Mail - PWAs installed from safari (At least used to) handle these link clicks. Links from apps in the OS open in the installed PWA
 - **So** - this launch handler feature allows the app to say whether a new app window should be opened, or the current one is used (if one is open)
 - <no resolution - time ran out before understanding>
- Resolution: not current interest in specifying the capturing moment. The navigation API would be good after launch turns into a url

Topic: Wrap up

co-chairs: Thank you all!

[adjourned]

Install API Lunch Chat

Lunch convo about install API

Goal: We seem to agree that same-document install is good & possible. **Can we achieve consensus on the approach to ship same-document install on all user agents?**

Context

<install> element

- UI says "install"
- prototype in chromium is working for same-origin
 - Likely can go through I2S in 2025
- 'what happens to element when already installed'?
 - it would say 'launch', matching all other app store functionality

pros

- Universally seen as a good way to do it. just says 'install'
- It is not possible for developers to trigger an interruptive user prompt via javascript with just a click
- Allows browser to control alternative UX when already installed, button can still launch etc (matching other app stores & user expectations)
- No javascript required / declarative model means simpler to add for devs, more flexibility for user agents.
- Discoverable for ecosystem (screen readers, ai agents)

cons

- The 'pepc' part seems unclear - scared that discussions about 'how this looks' is going to stall
 - (can we agree on sizing asap to unblock this?)
- Devs don't have complete control over style.

cross-origin

- UX needs more work here to investigate what we need to expose to users - what is shown after 'install'
- needs work to make this efficient if loading name & icons from authority.

navigator.install javascript API

- Already implemented and mostly through process / experimenting with on Chromium.

pros

- work has been done to implement, discussions complete
- provides total button UX flexibility for devs
- Easier to integrate with cross-origin install (future thinking)
- Provides 'launch' functionality if installed.

cons

- API *can* be called before manifest is on page, leading to potential misuse or mistakes
 - Chromium likely will require the manifest file attached with a specified ID, to prevent footguns:
 - dev didn't attach manifest in time
 - dev forgot to specify manifest when changing the start_url
- Allows a developer to trigger a interruptive user prompt via javascript, only needing a click.
- "what happens when already installed" not clear

cross-origin

- api arguments already defined
- are we ok with full dev control of UI to then trigger cross-origin install?

"beforeinstallprompt" event

- shipped in Chromium. sends event when manifest that is attached is
 - not installed
 - has start_url
 - display != 'browser'
- dev can call method on event to show install dialog to user.

pros

- Usage already exists today
- API makes it hard to trigger w/o a manifest (if a user agent wants that)
- provides total button UX flexibility for devs
- impossible to call API when already installed, as the event is not dispatched.

cons

- API is weird / cludgy, consistent feedback from devs about it being bad & hard to use.
 - Also doesn't require 'id' to prevent that footgun
- Allows a developer to trigger an interruptive user prompt via javascript, only needing a click.
- Does not provide an ability to 'launch' unless the user agent supports PWA navigation capturing.

cross-origin

- unexplored how this might expand to cross-origin case - might need to be dispatched from a specific html link element? No obvious way here.

Notes / Agenda

Goal: We seem to agree that same-document install is good & possible. **Can we achieve consensus on the approach to ship same-document install on all user agents?**

TAG approval for same-origin installs: <https://github.com/w3ctag/design-reviews/issues/888>

- Can a browser tab know that the app is installed on WebKit
 - you cannot know if the app was installed if it was already installed
 - a little developer & user hostile - as a user, I want to know if I installed it or not.
 - also, if dev has to save that state on server, webkit won't reveal if app is uninstalled, so the website won't promote install, user isn't able to install
 - suggested from Marcos: save a cookie you were installed, then don't promote install.
 - if user uninstalls, then site still can't advertise install
 - so - still bad - but Marcos: let's solve that later, when we see the problem.
- solved with element case, as it can say 'launch'
 - pepc will make it not possible to detect the pepc element is changed between install and launch
- webkit, always have install button, and cannot go away
- If in installed app, what should the button say?
 - what would you like?
 - now show?
 - Say 'uninstall'?
 - say "congrats you are installed"
 - lazy developer...
 - on Chromium - if you redock in browser, then it would say 'install' again
 - Mike - No matter what, no matter what solution, the dev ALWAYS has to choose to hide this after install. So - we can make a choice here, but it's up to the developer for what to render.
 - Yes - we need to define this - assert this is generally immaterial.
- Mike Jackson - app.spotify.com, install button, click it - they hide it
 - browser tab, app.spotify.com. I want to show launch.open
 - in some browsers, you might recognise you are already installed, you would make the choice as a developer
 - in any case, you are going to be in a strange situation if UA don't expose access to that stae.
 - if you have element, site doesn't need to know, as the element can know.
 - yes - in this case - as long as the element behavior is defined to be something reasonable like launch
 - then after launch, media queries can hide it.
- Christian - install would be in a modal, I would need to know how to close that modal

- element has event on it - it tells you the user interacted with the prompt. Then you can figure out what to do. Also - you can use media queries.
- Mike Jackson - if it's the same origin case, I kinda don't care whether I tell you or not.
 - cross-origin use-cases push the design in the direction of install element, as we probably don't want to expose installed information
 - not crazy either API or element would give information for same-origin install case, that might not be available in cross-origin install cases
 - but not thinking about that right now
- Mike MSFT - finished event?
 - mkwst - user interacted with dialog, or dismissed dialog - that's the info we give. We can consider enriching that if it's a good idea - can change
 - Store app - list PWAs on it
 - even if I have the whole backend situation, and you click install, you still need to 'stop the spinning donut'
 - mkwst - you get a hook for 'user is done with whatever browser is asking them to do'
- Christian
 - Declarative install - only for same document installs
 - id constraints - that is the case for all approaches here
 - talk about it later
 - webkit - remember it was clicked, store in database
 - element - opens, user cancels or whatever
 - cancel or install - you get 'promptdismissed' and 'promptabovrted' - we can do whatever makes sense
 - we can change to be 'installed' 'notinstalled', 'dismissed'
 - chrome / maybe ff
 - you use display mode media query
 - also on webkit
 - installed app - you get something - depends on user agent for button UX if you are in app context
 - Maybe "open in Chrome" to move back to browser
 - TAG side says same document installs good and we should do it
- Christian - conclusion - install element it is
 - WebKit happy with this
 - Chrome happy with this
 - FF (from previous convo) happy with declarative element

clarification

- install element is NOT pepc, UA can decide how to display it in a way it feels
- permission element is not the way to think about it - For chrome - we don't want the element occluded, and we don't want the element to have invisible text. Other UAs can decide what they want.