2025 WECG Berlin F2F, Public Notes, Mar 25

Tuesday, the first day of the four-day face-to-face meeting in Berlin (announcement: #759).
Agenda: https://github.com/w3c/webextensions/wiki/2025-Berlin-F2F-Coordination

Attendees

- Oliver Dunk (Google Chrome)
- Devlin Cronin (Google Chrome)
- Simeon Vincent (Mozilla)
- Rob Wu (Mozilla)
- Tomislav Jovanovic (Mozilla)
- Carlos Jeurissen (Jeurissen Apps)
- Mukul Purohit (Microsoft Edge)
- Timothy Hatcher (Apple)
- Maxim Topciu (AdGuard)
- David Tota (AdGuard)
- Oleksii Levzhynskyi (Grammarly)
- Nastia Mihal (Grammarly)
- Kiara Rose (Apple)
- Casey Garland (Capital One)
- Jordan Spivack (Capital One)
- Andreas Wagner (Mozilla)
- Lena Cohen (Privacy Badger)
- William Durand (Mozilla, Data Collection Consent topic only)
- David Johnson (Apple, declarative cosmetic rules topic only)
- Krzysztof Modras (Ghostery)

Notes

State of Browsers

Quick updates from each browser vendor on current priorities, recent changes, and future plans for extension APIs.

Google Chrome extensions

- Shared Extensions Platform
  - `browser` namespace
  - `permissions.addSiteAccessRequest()`
  - User Scripts permissions changes
  - `runtime.onExtensionLoaded`

- Browser Work
    - Runtime host permissions UX
        - Not changing default permissions for extensions. Trying to make it easier for users to control settings after installation. Ties into host access request API.
    - Sign-in and sync improvements
        - Always a lot of work happening in the browser: extensions team tasked with keeping up with these changes & minimizing impact on extensions.
    - Protecting users from Malware
        - Need developer mode enabled to use an unpacked extension. Was previously possible to load an unpacked extension & disable developer mode – would keep the extension installed and enabled.
    - Chrome Web Store improvements
        - Various updates
- Ecosystem Updates
    - MV2 deprecation
    - Policy requirements for extensions
        - Every 3-4 months we work on policy updates. Recently made changes to affiliate ads and default search provider settings. When we make a policy change, we also update the FAQs. Iterating on policies based on observed abuse and developer feedback.
    - Enabling AI development and experimentation (prompt API + other experimental APIs)
        - Prompt API: On device. Working towards moving it out of origin trial. A number of other very experimental APIs as extensions explore the limits of current capabilities.
- Working Together
    - “Medium effort” feature prioritization
        - Trying to identify work that makes sense to add to the platform that aren’t minor things external contributors might work on.
    - Working with other teams to bring their changes to the WECG
        - You’ve probably seen different Googlers attend WECG meetings. These tend to come up organically as someone is working on changes.

Apple Safari extensions

Newest entrant into Web Extensions.

First shipped Web Extensions in 2020, quickly followed with iOS and iPad OS support in 2021. Shipped web extensions on day 1 for vision pro in 2024.

Safari 18 - 2024: open sourced the WebExtensions implementation. Now entirely in WebKit. Aligns us closely with the open web, allows us to be open with what we’re doing, can link to our

implementation, helps resolve inconsistencies. Enjoying that this is all open source. A 2 year journey – looking forward to expanding on this work.

Done a lot since open sourcing. Added support for WebKit-based browsers, Developer ID-Signed & notarized extensions, Document Identifier APIs, getKeys API (OSS contribution), Match Origin as Fallback, main world content scripts, WPT/WebDriver support, Temporary extension installation, Icon Variants, Carlos implemented support for localization improvements that he proposed.

Developer ID-Signed & Notarized extensions:
https://developer.apple.com/documentation/safariservices/distributing-your-safari-web-extension#Distribute-your-Developer-ID-signed-and-notarized-extension-outside-the-Mac-App-Store

Mozilla Firefox extensions

In January we shipped a trial AI API for extensions based on Transformers.js, a commonly used JS library. We use the same models and API for our internal Firefox features. Model UI is not finished, which is why it's only exposed in Nightly, and should roll out in stable in ~Q3 this year. Behind an optional-only permissions – users have to opt in to this feature after installation.

Selection of changes by release version:

- 128
  - https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Releases/128#changes_for_add-on_developers
  - Many improvements because 128 is a Firefox Extended Support Release (ESR)
  - declarativeNetRequest
    - Disabling individual static rules (WECG)
    - Rule limits
    - domainType
  - Content scripts
    - MAIN world
    - sandbox
    - matchOriginAsFallback
  - optional_host_permissions
  - Event page lifetimes
    - (missing relnotes) bug 1869125 addListener at top level registers again
    - (missing relnotes) bug 1844044 permissions.request() keeps alive
    - (missing relnotes) bug 1771328 contextMenus persist across sessions
- 130
  - https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Releases/130#changes_for_add-on_developers
  - getURL() resolve inconsistencies (aligned in WECG)
- 131

    - https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Releases/131#changes_for_add-on_developers
    - storage.session.getBytesInUse() prepare for quota enforcement
- 133
    - https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Releases/133#changes_for_add-on_developers
    - DNR: bugfix: dynamic rules did not persist (backported to ESR128+ESR115)
- 134
    - trialML: https://firefox-source-docs.mozilla.org/toolkit/components/ml/extensions.html
- 135
    - https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Releases/135#changes_for_add-on_developers
    - commands.update can now assign keyboard shortcuts for commands to the F13 to F19 keys.
- 136
    - https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Releases/136#changes_for_add-on_developers
    - background.preferred_environment (WECG, thanks Carlos)
    - contextMenus.update/remove now reject with error for non-existent menus (resolved inconsistency with Chrome)
    - userScripts API available on desktop! 🎉 (will be on Android in 138)
    - (not in relnotes) webdriver-bidi contribution to load/uninstall extensions (thanks chrmod)
- 137 (upcoming release)
    - https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Releases/137#changes_for_add-on_developers
    - commands.openShortcutSettings (contribution, WECG)
    - storage.session 10 MB quota enforcement
- AMO (addons.mozilla.org)
    - Markdown support (HTML support dropped), notably from release notes and description.

Microsoft Edge extensions

- ()
    - runtime.getContexts()
    - runtime Events
    - action.onUserSettingsChanged
    - Commands API support for Cmd+Alt on Apple OS
    - sidePanel lifecycle events
    - Chromium Committer (thanks Chrome)
- Ecosystem
    - Security

    - Improved submission turnaround time (TAT)
        - (see Q&A below)
    - User experience (UX) in the store
    - Extensions on Android (20+ on Canary)
        - (see Q&A below)

Q&A: Questions & Answers

- On “Extensions on Android” from Microsoft
    - [rob] Is this a closed ecosystem?
    - [mukul] Yes.
    - [rob] Is it possible for extension developers to sign up?
    - [mukul] Yes, please ping on https://github.com/microsoft/MicrosoftEdge-Extensions with details of current Edge Store Extension Id, until a more streamlined way for signup is defined.
- On “Improved submission turnaround time (TAT)” from Microsoft
    - [devlin] What are the targets?
    - [mukul] Target is to match Chrome (based on dev feedback). We were taking ~3 days, we have publicly documented 7 days. We used to be on 4 - 5 days on average, reduced to 3 days. Top used extensions have 1 - 3 days.
    - [mukul] How is Chrome doing?
    - [oliver] The Chrome Web Store review process is documented; most reviews completed within 24 hours, 90% within 3 days; if anything takes more than a week we encourage devs to reach out.
    - [oleksii] Grammarly dev here. Is there anything that devs can do to simplify review?
    - [devlin] On Chrome’s side: make code as understandable as possible, follow policy as much as possible (e.g. do not try to get around)
    - [oliver] Generally another thing, comments can clarify but we still need to review code so this doesn't significantly change review time and isn't something we encourage.
    - [mukul] When test credentials are needed, make sure that they are correct. Make it obvious what the purpose of the extension is. Branding details in the Extension.
    - [andreas] On the Mozilla side, in addition to the above; Generally following the policy is good; we see so many developers who don’t read the policy and try to bypass policies. For example removing security-related headers. Missing data consent when extensions transmit data outside of the extensions (data collection consent topic). Another big one is failure to comply with the source code requirement - when source code is attached, the build does not reproduce or build instructions are incorrect (or missing requirement/dependencies).
    - [maxim] Why does Firefox require source code?
    - [tomislav] Manual review requires readable source code.
    - [casey] We release to Firefox once every few months, and we fail reproducibility every time. Advice on private npm packages?

    - [andreas] It comes up occasionally. Easiest is to include the dependency in node_modules.
    - [casey] Can the dependency be minified?
    - [andreas] No, has to be reviewable.
    - [tomislav] And if there are any concerns: source code is only available to reviewers, not all of Mozilla.
- On “Temporary extension installation” in Safari
    - [oleksii] Is there documentation on loading extensions as zip files?
    - [kiara] Yes, https://developer.apple.com/documentation/safariservices/running-your-safari-web-extension#Temporarily-install-a-web-extension-folder-in-macOS-Safari
    - [timothy] macOS 15.4 beta has it; new option in developer settings, enables dev to drag and drop the extension.
    - [rob] You recently mentioned this at one of the public WECG meetings.
        - https://github.com/w3c/webextensions/blob/main/_minutes/2025-02-27-wecg.md
        - Search for “Announcement: Safari Updates”
- (time check - 11:15)

State of the Extensions

Quick updates and feedback from attending extension developers.

Grammarly

- (11:15) (oleksii speaking)
- Performance, challenge: distinguish extensions from page content metrics (INP, RAM) and page crash tracking.
- Extension release and e2e test automation. Goal: fully automated release cycle on CI with gradual rollout and rollback.
- Side Panel: experimental mode.
- Speeding up review process
- New permissions. Goal: roll out new permissions with “warning” to at least new user with “Hybrid permissions” proposal.
- Beta/alpha channels: Set up beta channel for risky, permission update related or architectural changes (MV2->MV3 for example).
- Q&A:
    - On “Beta/alpha” channels
        - [andreas] Hypothetically, are you specifically looking for Alpha/Beta, or just a staged roll out? Or are you specifically looking for a dedicated user base on a separate channel
        - [oleksii] Both. Gradual roll out would be super helpful. We already use it in Chrome, and it would be very beneficial in other browsers. When I talk about Alpha/Beta channel, the question is about gradually rolling out user features.

        - [oliver] What are you looking for in the alpha/beta channels? E.g. preserving extension ID, user base, etc.
        - [oleksii] Keeping the same ID would be great. Clarity to user that things may or may not work because they are on a beta channel.
        - [oliver] Review expectations? Quickly roll out beta?
        - [oleksii] Want to push out beta quickly. From beta to stable, should be reviewed.
        - [oliver] A possible approach is to review Beta, and then push that reviewed version out to release without new review.
        - [carlos] Another motivation for multi-channel is backwards compatibility.

AdGuard

- (11:25) (maxim speaking)
- Bad experience with MV3 migration; had to start background script with our own engine to overcome limitations.
- Chrome web store rejections due to remote code policy violations. We had to disable custom rules and filters to comply. Hope that this is temporary and that we can restore the functionality later. This limitation makes the life of filter devs more difficult.
- We have improved the UI of our extension, improved mobile support.
- DNR rule-only changes; after publishing, the extension is considered corrupted.
    - [oliver] Now that we are here in person, we should sync up on that. We were not able to reproduce
- Review process difficulty, e.g. in Firefox. It seems that in the past some updates were approved automatically, and it looks like that is not the case any more.
- Gradual rollout feature in Chrome Web Store is nice, would like to see it in other stores too.
- Rollback functionality in Chrome Web Store, would also be nice to have in other stores.
- Plan for future: working on faster initialization of our engine.
- Q&A:
    - [mukul] What does rollback look like?
    - [maxim] Same extension, just the version number increased.
    - [andreas] Question to Chrome: Ever come across developer complaints where e.g. the version in the extension's own code is inconsistent with the version from the manifest?
    - [oliver] Not seen that.
    - [oliver] Potential issue with rollbacks is if extensions have a db migration that does not account for schema.
    - [oleksii] In code we treat it as a previous version, but the store treats it as a new version. The option is used as disaster recovery.

- [simeon] (11:35) Dropped WG discussion from today's agenda because of time constraints (rescheduled to tomorrow). 10 minute break.

Proposal PR Resolution

Working session to review and resolve open proposal PRs. We'll aim for clear resolutions or follow-up paths for each.

PR 370: Initial Firefox schemas, 2023Q1

- [tomislav] Oops
- [rob] Let's update schemas before merging.
- [timothy] On that note, we should also update our IDL files, as we updated our API support.
- **Action Item:** Tomislav to update PR 370.

PR 569: Proposal: i18n-system-languages

- [carlos] Waiting for Rob's review on Firefox patch.
- [rob] I have posted feedback in the PR. Once the spec discussion finishes I'll get to the Firefox patch.
- **Action Item:** Rob to review PR 569.

PR 586: Proposal: documentId in tabs.query() filter

- [simeon] I am the only assigned reviewer on this PR. I should add others.
- [timothy] When implementing documentId in Safari I found it strange that it was missing; I did not know that there was a proposal.
- [rob] I am in favor of the proposed feature; the proposal process states that proposals should not be aspirational but have a sponsoring browser that commits to its implementation.
- [simeon] Timothy, would you like to take over as the sponsoring browser?
- [timothy] Ok.
- **Action Item**: Apple, Google, Mozilla to review PR 586.
- **Action Item:** Assign Safari as sponsoring browser on PR 586.

PR 587: Proposal: downloads.getFileHash()

- [simeon] We have approvals from Apple and Mozilla. Chrome is an outlier here.
- [oliver] We wanted to go through security and privacy review, but that requires a document on Chromium's end, but that doc was never created.
- [devlin] Example of concern is abuse of the feature as an oracle in cases where the file has a relatively small number of possible values.
- [rob] The API requires host permissions, and extensions with host permissions can already fetch the URL and read the whole file content directly.
- [simeon] sponsoring browser?
- [rob] I'm fine with leaving this open until the author comes back.
- [timothy] Safari does not have downloads API, so we would not sponsor it.

- [simeon] Since there is no sponsoring browser, and our Proposal Process requires a sponsoring browser, closing this PR. We are open to re-opening the PR again if there is implementor interest.
- [carlos] If there is a community member interested in contributing a patch, could that count as sponsoring?
- [rob] Yes, if browsers are willing to accept patches. In this case it is not opposition to the feature, just lack of prioritization.
- **Resolution:** Close PR 587 for lack of sponsoring browser.

PR 598: Proposal: manifest key trial_tokens

- [simeon] Maybe missing Firefox approval.
- [oliver] This is shipping in Chrome.
- [rob] Tomislav is assigned as reviewer.
- [tomislav] I took a look before, it looks good.
- [rob] Can we merge now?
- [tomislav] Yes.
- **Resolution:** Merge PR 598.

PR 641: Proposal: Per-extension language preferences

- [rob] Status was that Oliver said that Chrome did not want to sponsor, and asked Timothy and me whether we are interested in sponsoring, but we never replied.
- [timothy] We are unlikely to prioritize this.
- [rob] Same.
- [timothy] Maybe Carlos?
- [carlos] Issue 274 seems more straightward to implement.
- [timothy] I prefer 274 over 641.
- [carlos] One does not exclude the other, but 274 seems to cover the uses cases.
- [carlos] Should I create a PR with a proposal?
- [] Same question, is there any sponsor?
- [timothy] Only if a contributor contributes.
- [carlos] I can take a look.
- **Action Item:** Look into PR 641 or issue 274 as alternative.

PR 676: Proposal: Public Suffix API

- [rob] Lots of Mozilla input to discussions, missing feedback from Google/Apple.
- [oliver] I monitor this PR from time to time, it looks like there is still ongoing (unresolved) discussion.
- [timothy] Is Firefox still the sponsoring browser?
- [rob] Yes. Contributor also submitted a patch, but before signing off I directed to the WECG for API alignment first. And here is where it is stuck now.

12:30 - break - return at 14:00

Content Script Params

- (14:00)
- [rob] Often referred to as “globalParams”.
- [oliver] Chrome implementation of an early idea: https://chromium.googlesource.com/chromium/src/+/7ec4d3b806a46c45d25ae83abc6656814d740cf7
- [rob] Some developers would like to do it per-website, some developers are ok doing it globally. Main use case is being able to access data synchronously and being able to update it later.
- [oliver] Worth noting some potential use cases. Privacy extensions might have various features that apply to a specific page. User may want to have arbitrary configuration to block specific APIs on a page. Synchronous blocking is necessary to prevent the page from accessing the APIs and saving references.
- [rob] As a baseline, makes sense to use match patterns or something to align with content scripts.
- [oliver] Initial chrome implementation was very simple. Seemed to be a global variable declaration. Is that right? Should we talk about other designs?
- [devlin] Never shipped. Wasn’t fully implemented. In general, being able to configure an object that’s exposed synchronously makes sense. More natural for the content scripts than calling a specific API.
- [oliver] What I remember from the CL is that you could mutate it from the content script. Is that right?
- [tomislav] Would be much more complicated.
- [rob] Would make sense to be able to change it, but not to propagate that change back. I’m imagining a map-like API. For example a dictionary where you can access only a specific key and the browser only has to serialize that data.
- [tomislav] Rather than debate per domain or global, what about registering per script? Site specific scripts could use a site-specific match pattern.
- [rob] Technically possible, but it is not exposed to individual scripts, it is exposed to the execution environment.
- [tomislav] Right, that’s unfortunate.
- [devlin] We did have two separate use cases covered – something site-specific and something global. Wouldn’t want to mix those concepts. Having individual separate keys and lazily loading them wouldn’t get us anything.
- [tomislav] IPC is the expensive part.
- [rob] If you send them infrequently, you have to propagate to every process. If it’s only specific keys, you only have to set one value. If you only had to set global params to a single value, but had other values you wanted to expose, say a data URL, … if a developer wants to update values infrequently, it would be nice to be able to separate those.
- [oliver] Good summary of the high level goals. Is this something we’re interested in working on?
- [rob] Would be happy to see a proposal.
- [devlin] Seems useful

- [timothy] Don't see any major blockers.
- [rob] Should we have a notification mechanism to see when a content script's params are updated?
- [timothy] Don't think it's strictly necessary. Would be fine implementing it initially without. Cost for adding this later would be trivial to add.
- [devlin] Would also need to figure out in a proposal how value would be set. If we specced it to do
- [rob] Was thinking we'd do a structured clone of the provided data in the get method.
- [devlin] I thought the original proposal was `browser.scripting.globalParams`, no explicit get method.
- [rob] Understood, but `globalParams` is a single value. Was proposing `get()` to expose more values.
- [devlin] Right, but you just expose each individual value as a property of `globalParams`.
- [rob] If you expose it as a property, it looks cheap but may be very expensive.
- [tomislav] Would also be less magical. What you get now may not match what you get later. Get makes it more clear that you have a read only copy.
- [oliver] Definitely feels weird to have a property change out from underneath you.
- [tomislav] A get() method clearly communicates that it's a snapshot.
- [devlin] Don't mean to block on this. Having a get is just substantially different than the previous proposal.
- [oliver] If we have a get, how do we handle synchronous changes to this data?
- [devlin] Wouldn't be synchronous. We're just exposing the latest value received via IPC
- [rob] API only returns data once all data has been propagated to all content processes.
- [devlin] If you're communicating with the bg script to get a value, just send the value. You can't know in the renderer if you have the latest value.
- [rob] Getting back to the use case, need to have a value synchronously. Concrete example, extensions that want to modify web API behavior before script execution: at the start of the content script you need to know if you have a value enabled or disabled. Content script needs to know the latest known value.
- [devlin] Oliver's question was is there any way from the content script side to know if you have the latest value. You can't.
- [rob] Technically true, but at the moment a script is injected you have a value and it's possible that the value changes as scripts are being injected.
- [devlin]
- [tomislav] Anything we could implement to communicate updates could be solved with messaging. I suggest we start with an MVP
- [oliver] If we start with the assumption that the value exposed will not change after first injection, changing that in the future would be a breaking change.
- [jordan] Having a watcher in the content script might be the most useful.
- [rob] Yes, extension devs can use messaging, but it would be cheaper to solve it as part of this API.
- [tomislav] Would you be making another storage area that's synchronous? Everything else has a listener for onChange.

- [rob] Yes, would also suggest we only emit the new value.
- [oliver] If the question is between a global or an API you can call, sounds like we're leaning toward an API
- [rob] Yes, that's how I'm leaning.
- [devlin] There are tradeoffs. Making it API based makes it less web-like. May lead us down the path of reimplementing storage.onChanged.
- [rob] How would you get changes in a property based approach?
- [devlin] Content script could set a custom setter
- [tomislav] Not for unknown fields. Would need a Proxy to intercept the setter.
- [rob] Unless the extension knows the keys that they are interested in.
- [tomislav] Does any dom API work like that?
- [devlin] Feels very arbitrary to have a bundle of properties behind a namespace. I'm always interested in ways that you can avoid tying the API to a specific structure. Feels nicer to me as a developer to say "here's an object, just read from it."
- [timothy] Definitely see the appeal of a direct object access approach. Rob's proposal with get and set is making me feel like it should just be part of storage. At that point why are we reinventing that API.
- [devlin] … Also creates the weird issue where all of the other storage APIs are async, but this one is sync.
- [rob]
- [devlin]
- [rob] Another question I had for later is what do we do for values that you do/don't want to persist. The only thing we'd have to guarantee is that a value is exposed to the content process.
- [devlin] You're saying we'd have a subset of the stored keys in the storage API that would be available
- [tomislav] Would this be a new value or the same storage?
- [rob] Same storage, but cached and exposed synchronously. Making data synchronously available in the background script is also a feature request.
- [oliver] Like the direction this is heading, but it prevents devs from exposing data for specific pages
- [rob] That's fine, devs can index on keys
- [devlin] That's different – that doesn't limit where the data is exposed.
- [oliver] No way to give partial access controls based on the current keys.
- [rob] True, but also true of the current storage APIs. If we wanted to make it origin specific, we should extend the benefit to the existing APIs as well.
- [oliver] Starting to get very complicated
- [devlin] Started small, but seems to be getting unwieldy
- [rob] Is your concern about leaking data to untrusted processes?
- [devlin] Leaking, and amount of data being sent. Don't want to send down all of the data for every site. Not saying we can't start with just global data, just flagging that keying doesn't solve this.
- [rob] How much data do we want to make synchronously available?
- [devlin] Exactly. Storage API allows you to set sub-keys.

- [rob] Latest values wins. Can't set sub-keys. Can only set top level properties.
- [simeon] It's ultimately key-value pairs. You're not operating on objects, you're operating on values.
- [rob] Are we aligned on an approach?
- [devlin] We have 3 options with very different approaches.
- [rob] See the appeal of basing on storage API: persistent vs non-persistent (storage.local vs storage.session) get/clear/remove/set/onChanged for free. Also (Chrome-specific currently) incognito and non-incognito have their own storage areas (when incognito:split is used).
- [oliver] If we were to do storage, would be interested in a new storage area. Want to get to the point where we're not exposing storage arrays by default in content scripts.
- [rob] storage.getSync but use storage APIs on the back end?
- [devlin] Agree that we want to limit access, but Rob raises a good point that getting some things for free is very nice. If we decide that we're going to treat different keys different in the same storage area, that effectively achieves the same thing.
- [tomislav] That's my worry if we use the same storage area. Extensions generally don't have limits for that kind of thing. Easiest solution for a dev is to just expose everything everywhere. A new storage area helps us avoid that.
- [devlin] Don't know that that's the likely outcome.
- [tomislav] If we take an existing storage area,
- [devlin] So your concern is performance
- [tomislav] Yes
- [devlin] Okay, separating security concerns and perf concerns. We have a few options. If we reuse an existing area, we wouldn't say "make all of storage.local sync", you'd have "make these keys in storage.local sync". I'd put making the entire namespace available
- [oleksii] I feel like devs will make everything available sync. It will be much simpler to use that way.
- [tomislav] Agree, would want to make it a new area with a low amount of storage available.
- [devlin] Browser vendors wouldn't want to make everything available all the time. We'd want a way to make a subset of the data available, documented as such.
- [oleksii] From the developer perspective, if there's a clean interface and a way to protect it with types, it would be easier to work with. Would prefer that over returning some kind of promise.
- [devlin] Wouldn't want to return a promise. Sync retrieval wouldn't be done through a synchronous promise.
- [rob] Compressing the discussion. Desirable features: Persistent vs. non-persistant, content scripts and background should be separate, updating a value in the bg would be async, retrieving a value is sync,
- [oliver] Would be interested in seeing a brief section on how we can expose data per origin and how different approaches might make this easier.
- (post-meeting edit: Rob created PR 793 with a proposal of an API; this PR was mentioned briefly in the regular WECG meeting on Thursday, and revisited in the issue triage on Friday)

Data Collection Consent

- (14:50)
- [will] At Mozilla we have review-related challenges around our data collection consent requirement. Our policies require developers to implement UI to explicitly request consent from users to collect data. Reviewers and developers don't always agree on what these terms mean or the implementation requirements. The idea is that we may be able to solve some of these issues directly in the browser itself. That's where the proposal comes in. We remove the need for developers to ship their own UI. We get a more unified experience for users (currently very inconsistent).
    - Introducing data collection permissions
- [will] We call this concept "data collection permission", because it is similar to permissions (API permissions, host permissions …), but different enough that we don't want to confuse the concepts. If the data collection isn't required by the extension, it can be declared as optional and requested later with `permissions.request()`.
- [carlos] This would not change what APIs return?
- [rob] Correct
- [casey] How can I test this?
- [william] Prototyping in next Nightly, at which point it can be tested by any extension developer.
-
- [devlin] Have a number of concerns. This appears to be incorporating a number of policies from a given store into the manifest itself. May be challenging if individual stores have different (possibly conflicting) policies. We have some overlap in that we display disclosures in the store (timothy: as do we) and the set of information collected or displayed may not have a great way of aligning across our use cases. I'm hesitant to have this as part of the top level manifest because we're likely not going to have a lot of overlap. See a lot of value for having something like this in Firefox. Is this something you might be okay having in browser specific settings?
- [rob] We've considered this. In some earlier iterations we've considered arrays of `data_permissions` and `optional_data_permission`, but changes this to a dictionary to allow more flexibility in the data structure. Open question about what browsers would do when they encounter values they don't recognize.
- [devlin] A lot of the requirements we have stem from regulation/law. We aren't regulators, don't think this group could effectively define the right set of data
- [timothy] Similarly we're likely to keep this as app store specific data.
- [mukul] Data in the manifest has to live there forever.
- [rob] AMO has a place to put the privacy policy. In practice how many users are actually reading that?
- [mukul] True for developers as well. Most people just click okay. Still, most requirements are coming from law.
- [oliver] Could you speak more to the motivation for putting this in the manifest? Seems like it would be easiest to have a UI to set some values. Could be an artifact that's generated and included in the file.

- [will] We treat XPIs as self-contained binaries. We decided to put it in the manifest as it's a developer authored value. Not all add-ons are listed on AMO. Thought it made the most sense to have the manifest be the source of truth.
- [oliver] Would object somewhat to the classification of the manifest as the source of truth for this type of data.
- [tomislav] Do you not support use cases where the extension is installed without an internet connection?
- [devlin] If you don't have an internet connection, then it can't collect data. Also wouldn't see this information in a privacy policy if you're offline.
- [andreas] We have the concept of "unlisted" extensions (which is different from CWS's concept). Not all extensions are listed on AMO and we still need to display this information.
- [andreas] Add-on packages are immutable because they're signed. When an extension changes, it gets difficult to handle that in the store. If you package it in the version, you ensure that it's accurate (assuming the developer didn't lie).
- [timothy] Worry this is getting out of scope for this group. Distribution and stores are out of scope. Trying to avoid server requests by shoving data in the manifest doesn't seem like the right solution. Don't see benefit from including it in the manifest.
- [rob] Impression so far sounds like there's no interest in including this data.
- [devlin] Actively opposed for Chrome.
- [rob] Sounds like we should move this to a separate file, not in manifest.json.
- [timothy] Not being in the manifest is better for cross-browser purposes as well.
- [david] Not sure I follow this example: collecting bookmarks. Is it possible to describe specific usage?
- [rob] Will, are developer-provided explanation strings included in the proposal?
- [will] No, but we intend to include this on AMO.
- [oleksii] How does this work as classifications of data change over time? For example, a new regulation classifies an existing piece of data as PII. In the current experience the screen is essentially a short version of the privacy policy.
- [will] Same issue exists for existing extensions. The solution is that the extension has to submit a new version.
- [rob] If the extension implements its own UI, it's more likely to implement its exact requirements for the current legal landscape.
- [timothy] Would like to see reason strings for permissions to give developers a chance to explain what they're requesting and why.
- [devlin] We discussed that; conceptually we'd be onboard. Might require significant changes in the manifest. Would Firefox disable the extension when a developer adds a new data collection value?
- [tomislav] New permissions prevent the new version from being installed. New data collection consent values would also block installation.
- [devlin] Developers are constantly afraid of introducing a new permissions and getting disabled in Chrome. Today, collecting more data doesn't block installation.

- [rob] Data collection permissions is a new feature where we can implement changes from the start. Independently of that, permissions also have the same issue and we should ideally improve that. There is a WECG issue tracking this (issue 711).
- (15:30)

DNR: What's missing

- (15:50)
- [timothy] We have a lot of issues filed about DNR, labeled topic:dnr. What are good next steps to focus on?
- [maxim] We prepared many topics

Issue 468: [DNR] Redirect rule does not allow applying other redirect rules to the resulting request

- [simeon] All I recall about this one is concern about infinite redirects.
- [devlin] Can be addressed with a reasonable max number of redirects.
- [devlin] The feature request here is that you should be able to chain redirect rules to remove query parameters. The generic request is that you can chain redirect rules.
- [maxim] Our use case is we have several rules and we want to apply each of them to the same URL. Currently only one rule is applied, the others are ignored.
- [rob] In all browser?
- [maxim] I believe so.
- [rob] Surprising, expect that additional rules would apply in Firefox. Maybe something actionable for this issue is to create a minimal test case to help browser vendors investigate and understand the problem.
- [timothy] We have an internal issue filed but not investigated yet.

Issue 731: Incorrect application of "domainType: thirdParty" in DNR

- [maxim] Currently only Chrome is supportive. Firefox and Safari are needs-triage.
- [rob] Looked into this this morning. Firefox's implementation was modeled on Chrome for compatibility. Some main frame requests are considered 3rd party in Chrome. Intentional?
- [devlin] No. Right now, firstParty if request matches initiator. But if initiator is missing, then it should be first-party.
- [rob] If the initiator is an opaque origin, would it be first or third party?
- [devlin] We probably wouldn't always consider it a first party. If we had to pick one, third party. If the initiator origin is opaque, it potentially makes sense to consider it a first or third party request based on the precursor origin.
- [rob] Are you in favor of treating this case as first or third party?
- [devlin] If it matches, it could be treated as first party. Don't feel strongly, though. That doesn't match how the web generally treats it. I can see arguments either way.
- [rob] Agreed. Mostly want it to be consistent.
- [timothy] Not sure how WebKit handles this.
- [rob] Did you have the same concept of first and third party before DNR?

- [timothy] Yes, we that to our DNR implementation.
- [rob] Oh, but that is a bit different.
- [timothy] I don't know the rules offhand. Trying to look through code now. I can find an answer and follow up.
- [rob] Would this be useful to have a test for in WPT?
- [kiara] Confirmed that Safari behaves as expected.
- [timothy] Looks like WebKit checks if the top level domain matches the request, that's it. If they match, it is `firstParty` otherwise it is `thirdParty`. All main frame navigation are `firstParty`.
- [alexi] Wanted to raise `mailto:` links. Short version: users set up gmail to be their mailto handler. When a new tab is opened, PrivacyBadger accidentally modifies some links.
    - [alexei] https://github.com/EFForg/privacybadger/issues/3066
    - [rob] This issue was also raised in a recent meeting (minutes: 2025-03-13-wecg.md)
    - [alexi] Seems to be past the protocol handler. A tab is opened and we can't associate it with the link.
    - [rob] Need to investigate this more. DNR doesn't currently support matching missing or opaque initiators.
    - [devlin] Would be curious to know where these requests are coming from. Alexi, you mentioned that this may be coming from a service worker?
    - [alexi] Correct.
    - [rob] I'll investigate this more.
    - [devlin] For XHRs being initiated by clicking mailto links, those are being labeled as 3rd party, but should be considered first party (if they're coming from the gmail SW as posited by Alexei).
    - [rob] Is this Chrome-specific?
    - [alexi] Don't know. Have only received bug reports for Chrome.
    - [timothy] Safari does not redirect `mailto:`
    - [rob] If you can try this out in Firefox and verify it's a cross-browser issue. If not, follow up in a Chrome bug.
- **Resolution:** top-level navigation without initiator should be classified as domainType: firstParty
- **Action Item:** Firefox to implement resolution of issue 731.
- **Action Item:** Chrome to implement resolution of issue 731.

DNR: Cosmetic Rules

- (16:16)
- Related issue:
    - Issue 362: Proposal: Declarative Cosmetic Rules
- [timothy] We discussed this for a while, also in San Diego. Action item was for Apple to come up with a proposal. David created a proposal for what we think may make sense.

- [david] (screen-shares early draft of proposal; the proposal will shared at a future WECG meeting)
- [rob] Is the "selector" condition the only proposed condition, or one of multiple?
- [david] This is the only currently proposed condition. I think we can do others like URL matching in the future.
- [devlin] Do we assume that most DNR conditions will be applicable, or will we only support DNR conditions we see strong use cases for?
- [david] We can have namespace discussions and may want to use this as an opportunity to rename DNR.
- [devlin] We can have the namespace discussion separately. Something like `initiator` may not make sense for a CSS rule *[so we should discuss if we support all conditions]*.
- [rob] We could do this based on if the condition matches the main frame request.
- [devlin] Would be confusing for developers, and hard to implement as a lot of required information is not kept. Also, what happens if it used to match but no longer does because DNR stripped headers?
- [devlin] Do we need network request data, or say rules only ever require things we can determine from the document?
- [devlin] Would also be good to discuss other actions.
- [timothy] Adding styles would always need host permissions.
- [devlin] Would be good to find other actions that don't need adding styles.
- [tomislav] For example, `visibility: none`.
- [devlin] Maybe there are other cases where we can safely allow setting specific properties. For others, we require a host permission.
- [rob] Sounds like we need to include requiring and not requiring host permissions in the design. E.g. `{ action: “css-display-none”, fallbackActionCSS: “display:none”}`. Browsers not supporting “css-display-none” would apply the fallbackAction (and require host permissions). Browsers supporting css-display-none would hide the matching elements, without requiring host permissions.
- [devlin] Can see that working.
- [casey] Can you see a world where we avoid host permissions?
- [devlin] Yes. We want this API to also include additive parts. Adding things without requiring host permissions gets tricky fast. Things like the tabs permission, where we selectively leak URLs…
- [timothy] Doesn’t work for Safari (because the “tabs” permission does not reveal the URL of a tab, only host permissions does).
- [oliver] Might be out of scope for this, but declarative badging
- [devlin] Would require host permission.
- [oliver] What about badging images with a static image?
- [devlin] Early feedback is that extensions want to be able to do more than show a static image.
- [oliver] 1Password does this. It just injects a static image on input fields.
- [devlin] Other folks wanted richer interactions. Ability to react to hover, display custom content, etc. You can address that with an iframe, but you also know when you’re being

included. Going back, we can start with it being subtractive and in the future consider additional capabilities.

- [rob] The idea mentioned before of having a `fallbackActionCSS` gives extension devs a way to safely fall back to an expected behavior, even if a browser does not recognize/support a feature. Compare with plugins, legacy `<object>`/`<embed>` elements that fall back to showing fallback content if the plugin is not available (and show the plugin's result if it is).
- [casey]
- [david] Extensions can block requests, but doing so leaves holes in the document. We want to give them a way to clean up those issues.
- [devlin] Would likely want to also put some limitation on the complexity of the selector. We've seen that watching complex selectors for changes can be quite costly.
- [timothy] Don't have a similar concern. Recently opened up content filtering to allow the `:has()` selector once we made it performant. Defining a common set of performant selectors sounds difficult.
- [rob] Alternative for extensions is to use content scripts, which would be considerably more expensive.
- [devlin] Right, depending on how they do it. "Selector" is overloaded, but I assume this includes ancestors and children. But don't think you should be able to chain a 50 has selectors.
- [rob] Going back to the proposal, is there more we wanted to cover?
- [david] I would like to figure out the MVP for this, use cases etc. Don't want to let perfect be the enemy of good.
- [devlin] Concrete use cases include dark mode and high contrast extension. I like the idea of having a way for a high contrast extension to change colors without needing <all_urls>
- []
- [timothy] Safari can support any DNR match cases for this.
- [rob] Also curious how many match cases you'd need to cover.
- [maxim] 24,000 rules in AdGuard Base filter are hiding rules.
- [devlin] All of those would need to be available in the renderer.
- [timothy] We support this, and I know a lot of content blockers do today as well.
- [oliver] It's not like you need to look for those on every page, right?
- [maxim] Right.
- [devlin] Depends on when you need them. Content scripts currently do this at document_start.
- [oliver] I assume that when you call insertCSS you're also solving this.
- [devlin] That's shared with all processes.
- [rob] David, do we need to return to this topic?
- [david] I have enough to think about for now, but if time allows and folks have more feedback, happy to revisit.
- [oliver] Different browsers have different limits. Would be nice to try to align on a common set of limits across browsers.
- [david] (AdGuard) Easylist has 14,000 generic CSS rules that apply to every website.

- [tomislav] Is that injected on every page?
- [maxim] It depends. Some are complex, some are simple. Also wanted to ask if this proposal would cover scriptlets or if that's out of scope.
- [tomislav] Starting with CSS because that seems more manageable. I expect we'll learn a lot in that process and may
- (16:45) closing

2025 WECG Berlin F2F, Public Notes, Mar 26

Wednesday, the second day of the four-day face-to-face meeting in Berlin (announcement: #759). Agenda: https://github.com/w3c/webextensions/wiki/2025-Berlin-F2F-Coordination

Attendees

- Oliver Dunk (Google Chrome)
- Devlin Cronin (Google Chrome)
- Simeon Vincent (Mozilla)
- Rob Wu (Mozilla)
- Tomislav Jovanovic (Mozilla)
- Carlos Jeurissen (Jeurissen Apps)
- Mukul Purohit (Microsoft Edge)
- Timothy Hatcher (Apple)
- Maxim Topciu (AdGuard)
- David Tota (AdGuard)
- Kiara Rose (Apple)
- Casey Garland (Capital One)
- Jordan Spivack (Capital One)
- Kees Kluskens (Tango)
- Elijah Sawyers (Apple) - Testing Sync Only.

Notes

Issue triage (Part 1)

- (10:00)

runtime.onInvalidated event and related

- [carlos] Group of related topics
    - Issue 138: Proposal: let content scripts listen for event to detect when the extension is unloaded (for various reasons)
    - Issue 353: Support onEnabled event when the extension from disabled state to enabled state

    - Issue 789: Proposal: runtime.onInvalidated.addListener
    - https://issues.chromium.org/issues/40384477
- Discussion
    - [devlin] Discussed last year too; not much have changed on our positions - still supportive, no bandwidth to prioritize.
    - [carlos] Willing to create a proposal.
    - [oliver] We need a sponsoring browser.
    - [devlin] If Carlos is willing to do the hard work of the proposal, the actual implementation would be straightforward.
    - [timothy] Going forward we should try a more inclusive language than onEnabled/onDisabled. For example we use turned on/turned off at Apple.
    - [devlin] I renamed Issue 138 for clarity. From: “Proposal: let content scripts listen for event chrome.runtime.onUninstalled”, To: “Proposal: let content scripts listen for event to detect when the extension is unloaded (for various reasons)”
    - **Action Item:** Carlos to create PR for issue 138 with Chrome as sponsoring browser.
- Additional context added post meeting:
    - Carlos created PR https://github.com/w3c/webextensions/pull/792
    - Issue 138 and PR 792 are revisited on Thursday (Lifecycle & Matching topic).

Issue 700: Declaring compatible websites (without prompting the user)

- [oliver] I don’t think that more manifest keys would help here.
- [devlin] Agreed.
- [timothy] Happy with the current approach. Not sold on a new key being the solution.
- [devlin] I think the goal is more consistent UI across browsers. That’s a browser problem, not a platform one. Safari currently withholds hosts and Chrome is working towards that.
- [rob] Isn’t addHostAccessRequest the general solution?
- [devlin] We think so, yes.
- [timothy] Agree. Unless we’re renaming things in a manifest bump, the current names are fine.
- [rob] addHostAccessRequest is a general way to do it in Chrome, could be added to Firefox, but doesn’t seem like a solution in Safari.
- [timothy] That’s why we were kind of against that API; it’s not possible to do in Safari.
- [oliver] We settled on that URL taking URL patterns. That could work in Safari, right?
- [timothy] I recall that now, could be useful.
- [simeon] Next steps on this?
- [devlin] Seems like Chrome and Safari are opposed. Firefox?
- [rob] I see the intent. Should ask whether addHostAccessRequest would address this use case.
- [rob] Another related issue is what to do with new extension permissions on update (issue 711). A general solution could cover both issues.
- [devlin] I’ll work on a reply.
- **Resolution:** Close issue 700, Chrome and Safari are opposed.

Issue 756: Proposal: Exclude Matching

- [carlos] Basically the proposal is for negative match patterns. Either an explicit exclude match list or some kind of negative pattern (prepending exclamation mark ! before match pattern).
- [timothy] Clever approach. Not sure we'd go out of our way to add it if exclude isn't already supported by a given API.
- [carlos] Only cover it in existing APIs that have exclude matches?
- [timothy] If we support it at all, it could be included anywhere. Could potentially replace existing exclude_matches. Don't see a clear demand to introduce this syntax to match patterns. Currently neutral. Could keep in our pocket in case we need it
- [rob] Seems like it could be useful. I largely agree with Timothy. In cases where we don't have exclude, we could introduce this syntax. If we want to introduce this, we could stop erroring now and eventually, once we have enough support in historical versions, we could actually adopt it.
- [oliver] Okay, we're talking about new syntax, not a new key.
- [rob] Correct.
- [timothy] I was gravitating to syntax, yes.
- [devlin] I see two questions. Do we want to allow this as syntax vs. a key, and what the syntax would be.
- [rob] If we adopt a syntax, we can apply this logic across all APIs that take a set of match patterns, without requiring separate exclude keys in each API.
- [devlin] Sounds like an implementation detail. In Chrome we have UrlPatternSet. That code would be invalid if we introduce the concept of negation.
- [timothy] We have a match pattern class too, but we don't have a set that covers the whole set.
- [devlin] The idea that this will magically work in all cases is heavily browser specific.
- [rob] Right, would need to be audited and tested for support.
- [devlin] If we're auditing every API, it's orthogonal to changing this one piece. Whether we change the syntax to allow negation or introduce exclude matches, those don't have to be directly linked. Is there a reason they have to be linked or can we discuss separately?
- [rob] If we prefer one over the other, we can skip implementing the other. Less total work.
- [devlin] That's only a point for the syntax, not the negation.
- [rob] Current title is "exclude matching", you're suggesting we split negation out into a separate issue?
- [devlin] Not necessarily. Do all browsers agree that it's a good idea to include exclude matches in more places?
- [carlos] Syntax and excludes are potentially different in what they can cover.
- [oliver] Agree we should support exclusion in more places.
- Agreement from all browsers.
- [devlin] Sounds like we're generally supportive of this use case, but how we do it still needs discussion. Any browsers that would immediately leap on this (next 3 months)?
- [rob] Don't see a big need demanding that speed.
- [timothy] Same.

- [oliver] contentSettings API also uses match patterns where a negation syntax would be strange, especially in the UI associated with it.
- [rob] Neither Firefox nor Safari have this API.
- [devlin] In Chrome this wouldn't work. That wouldn't support exclude matches today and it would be hard to support.
- **Resolution:** Chrome, Firefox, Safari in favor of supporting exclude of match patterns across APIs; the exact mechanism is still up for discussion, in issue 756.

Issue 701: Block extension from specific hosts

- [simeon] Seems like this is related to the previous discussion.
- [oliver] … Would be difficult to communicate this setting to users.
- [timothy] The request is removal after the fact, your description makes it sounds like an install time feature.
- [oliver] Yes, but there are still concerns around communicating that access revocation to users.
- [rob] Letting the extension drive this could cause confusion. Proposal states that the browser can do this, which is true, for example with enterprise policies. Concerned about exposing this data to extensions as it may leak information the enterprise doesn't want to expose to extensions.
- [timothy] We have a tristate for host permissions, can ask, not granted, granted. Allow grants host permission, Deny is "not granted" (what this issue calls "blocked").
- [rob] Would you be supportive of the extension changing "never ask again" to "ask again"? That is what the second part of the issue is asking for.
- [timothy] Don't necessarily agree with that. Don't want the extension to see what's blocked outside of the extension's own requests to block. But I'm okay with the idea of the extension putting something into the deny list and allowing users to change that later.
- [devlin] here we're talking about a block/deny list on a per-extension basis. (timothy: yes) Would there be any reason for a user to modify that list? Don't see a reason to allow a user to force an extension to have access to a site. Agree with concerns around leaking data for blocked sites to the extension. I see this request as affecting the extension's requested sites, not the user's choices. Allowed sites are the intersection between permissions requested by the extension and granted by the user. I think I'd rephrase this request in a couple ways. Start with Carlos' previous proposal for disallowed permissions in the manifest. Follow on that to allow the Permissions API to dynamically add entries to that list.
    - Issue 123: Proposal: add disallow_host_permissions
- [timothy] That description matches what I had in mind when I added the supportive label.
- [rob] This is a case where negation may be useful, right?
- [devlin]I think in MV4 we make permissions a dictionary. I don't think having a "not" would be very helpful. Adding and removing entries in the blocked set isn't too useful.
- [rob] How would this help a user?
- [devin] User doesn't have anything to do with this IMO. As I see it, my tweaked version of this proposal only affects what the extension asks to run on.

- [timothy] It's really useful for an extension that asks for everything and wants to restrict down to places it knows it doesn't want to run. Example: shopping extensions.
- [casey] We work hard to not track things we're not supposed to. Ideally a user could inspect things to verify.
- [rob] So this is more about dropping permissions (good practice). Because there is no good way to surface this information to users, because an extension update can always remove items from this list.
- [devlin] I see this as more of a CSP-style principle.
- [oliver] I like that CSP is entirely hidden from users. Not sure if we should have something like this shown in extension UI. Don't want a race of longer exclusion lists.
- [devin] Don't think we'd include this at first. Want to avoid situations where removing an exclusion creates a permission request. If we show the user this info, then we'd have to do that.
- [timothy] We don't currently show denials. Internally, we'd approach this as a denial that we don't show.
- [casey] Would like a way for power users to verify.
- [devin] Yes, see the utility for an admin or a regulator
- [devlin] Wrapping up
- [timothy] I'd be in favor of closing this in favor of issue 123.
- [devin] See value in making sure we cover this case. As long as we update 123 to include this case, I'm good with closing.
- [maxim] See potential in having this API.
- **Resolution**: Close issue in favor of issue 123, the "do I have access" part of the request will be covered by an issue Carlos is going to file.

(11:00 - break)

Platform Documentation

- (13:30)
- [timothy] Can we align on MDN as the browser-neutral, authoritative documentation source? Let's talk about browser-specific API docs, how we cover cross-browser topics, and leveraging extensions (e.g., Rob's) to improve BCD data.
- [simeon] Which extension?
- [rob] I created a tool in the past to auto-generate the set of supported APIs in Chrome: https://github.com/mozilla/webextension-polyfill/pull/122
- [brian] I'm Brian, a technical writer on the MDN team. We're looking at the extensions docs and we're currently doing a big reorganization. Good time to look at WebExt docs.
- [vadim] Also tech writer on MDN, joined half a year ago. See that current documentation is Firefox-focused. Especially with support from different vendors.
- [simeon] On the content operating side, Richard Bloor on our side handles writing WebExtensions content in MDN. I'm hoping to spend some more time this year on approaching the documentation process.

- [simeon] How much interest is there on aligning on MDN as the browser-neutral authoritative source for WebExtensions documentation?
- [timothy] We are interested. We are at least updating BCD data for Safari, and getting content into the articles.
- [rob] With regard to content being Firefox focused, we try to be browser neutral. When there are cross-browser differences that I know of, we try to integrate them into the content updates. I believe Safari is primarily relying on MDN and BCD documentation. My impression is that Chrome would like to keep the c
- [devlin] We continue to need a place for Chrome-specific documentation that MDN does not want/need to host, e.g. ChromeOS-specific platform apps. For API-specific bits we are open to either MDN or developer.chrome.com or some combination. Currently we auto-generate lots of documentation from IDL files and schemas from Chrome, that pipeline is very useful.
- [oliver] Wondering if there are things, e.g. labels or things we need to monitor. A vision/roadmap on what you want would also be useful.
- [oliver] When I spoke internally earlier, a concern was about a lot of work and changes overnight; we are willing to discuss and consider changes.
- [simeon] Can the MDN folks speak to the current process of getting contributions from browser vendors.
- [vadim] Neutrality, e.g. calling it WebExtensions (not “Firefox extensions” or “Chrome extensions”). We also have code owners, and automation to assign topics to specific reviewers (e.g. svg experts, JavaScript experts).
- [vadim] One challenge is, on the web we have a common spec and slight deviations; in WebExtensions there are more differences.
- [devlin] There are also a number of APIs that are not Chrome OS-specific, but also supported on other platforms, but not available in other browsers. For these APIs, do you envision documentation on MDN?
- [vadim] We have a policy that any API available in stable browsers can be documented on MDN. We already have bunch of Chrome-specific web platform APIs on MDN.
- [simeon] In the WECG we have it as a part in our founding principles (charter) that browser vendors can deviate.
- [simeon] In Firefox, MDN is our only site for extension API references. At least the way I think about it, we need a way with Firefox-specific content. I find it reasonable to extend that to browser-specific APIs.
- [rob] Because of the existence of cross-browser differences, it may be difficult for extension developers to figure out what they can use for the browsers they target. BDC may cover that, but it requires the reader to consciously jump to the table. If they are not aware, at worst they may ship something that unexpectedly breaks in a browser. Is there any plan to improve the visibility of WebExtensions compatibility info in the API articles?
- [vadim] Project in mind to make compatibility more visible, mostly part of Baseline. We already show baseline on many pages.
- [rob] Extension APIs have methods and individual properties whose availability varies by browsers. It would be very useful if developers can see at a glance whether specific

methods/properties are supported, without requiring the reader to consciously scroll to the Browser Compatibility table data.
- [vadim] We don't have that yet, but are open to supporting it.
- [tomislav] Not sure I understand the reference to Chromium-specific APIs that already exist on MDN. Are they on standards track?
- [vadim] We don't document things that aren't on standards track. These wouldn't be APIs that are created for a specific browser with no intent for cross-browser implementation.
- [devlin] For the purpose of this discussion, there is no spec.
- [vadim] Usually every documented web platform feature has a spec. Does not need to be an official W3C spec, and authoritative documentation would also work.
- [simeon] On guides, me and Richard would focus primarily on Firefox-specific documentation. Other browsers would have to contribute their part.
- [timothy] Makes sense. I'm willing to contribute from the Apple side. Jon Davis is also willing.

Finally Write a Spec?

- (14:00)
- [timothy] Should we take a WHATWG-style approach — write down what browsers do, use that to identify inconsistencies, and iteratively evolve it into a formal spec? What would it take to get there?
- [simeon] What is the WHATWG-style approach?
- [rob] I have only seen the output (the spec), not the process behind.
- [simeon] In abstract I like it, but I don't know how it went there.
- [timothy] I don't know either, my understanding is writing down what everyone did, of the current behavior, not necessarily the desired behavior, and then hashed it out with notes, inconsistencies, all these things. More a documentation effort than a "what it should be" effort.
- [oliver] When we discussed this before, I thought that we decided to focus on what most browsers do, not literally everything that any browser does.
- [timothy] I think a big driver of our spec work will probably be our tests. As we write tests, build out the spec, identify inconsistencies, clarify intended behavior, etc.
- [tomislav] I think that's part of the history of how WHATWG got started. I think that would be very useful. Going forward we can ask not just documentation of corner cases, but also tests for them.
- [devlin] One of the open questions from my point of view is how explicit do we want to be when defining the behavior. How specific do we want to get with browser behavior and features? Depth- or breadth-first approach? If we go for breadth, that will look very different from what other specs look and feel like.
- [rob] Laying down a skeleton seems appealing, but also limits the utility of the specs.
- [devlin] If we try to produce W3C style specs, that's a lot of work.

- [tomislav] What you see in the specs today isn't where they started. They ended up here after a lot of iteration and discussion to resolve inconsistencies. They all grew naturally as an increased level of detail came to light.
- [devlin] Worried about our collective velocity. If we try to go deep, that will take a lot of energy. If we go broad, will we be providing the level of detail developers expect? Will it have the level of validity expected from W3C specs?
- [timothy] I am fine with a breadth-first approach. The example of "create a tab" leaves open a lot for interpretation, but "tabs.create" also has multiple properties that can meaningfully be described.
- [devlin] I expect there will always be that level of interpretation, but say parse the URL and if there's an error throw.
- [rob] Having that is easier to clarify than to write something from scratch.
- [oliver] Does not have to be completely breadth-first; specific cases can be elaborated during specification.
- [devlin] My main concern is still velocity.
- [oliver] Velocity is a good concern, but I'm more worried about usefulness than trying to go too quickly and ending up with a copy of MDN.
- [devlin] In that case, how do we ensure that there is progress? (all: not great) The parts that we do have are well-specified (e.g. content scripts). How do we ensure progress?
- [timothy] Smaller PRs, faster turn-arounds would enable faster iterations.
- [oliver] What you said seems contrasting to depth-first, but they can be complementary.
- [timothy] Be detailed, but do not let it block the completion of a PR.
- [devlin] Not concerned about the ability to write something useful for anything. I'm concerned about the extension platform producing something useful in the next decade.
- [timothy] We need something substantial for a WG.
- [oliver] Does anyone feel that the detail stops them from getting started?
- [devlin] I'm not put off, but think that it takes times.
- [tomislav] Content script example, is that a right level of detail?
- [devlin] There are tradeoffs. There are things we've glossed over, but overall it was pretty good. And it took a year. Question is which is more valuable: covering more or hitting this level of thoroughness?
- [tomislav] Timothy, you were the newest implementation. What was the most helpful and what were the most challenging missing pieces?
- [timothy] We relied on Chrome's docs and MDN, and testing in various browsers to figure out what to do.
- [rob] And there are so many layers and indirections in Chromium. I don't think the level of detail is the primary roadblock. If we have limited time, the question is where we spend it. … is a very good first step. If we put in investment up front, that sets us up to be more productive in the end.
- [oliver] Devlin mentioned that changing the level of detail may 4-5x the number of submissions. That's true, but that might only take us from one to four or five PRs. We really need to focus more on how we find more time to write more.
- [rob] Not one size fits all. Where it matters we should spend more energy on level of detail.

- [oliver] We discussed `background.preferred_environment` in the leads meeting, we should write that up sooner than later.
- [rob] Also okay to sometimes leave a notice stating that there are subtleties involved to watch out for, to be specified later.
- [timothy] I can take on `preferred_environment`. Should be able to make some time in the next month.
- [rob] Are we aligned on where to start?
- [devlin] Not quite yet. I don't have a strong opinion about which strategy to pursue. We've been saying for years we should spec more. I'm wondering if there's something concrete that we should change or try to produce more.
- [rob] IMO prerequisite is to have WPT. Allows someone to create tests, see how it behaves everywhere, write that down.
- [devlin] Agree that WPT is helpful, but worried about trying to solve the need to write more spec by adding a dependency on writing tests.
- [kiara] Would be helpful to list out what we want to have written. Even now it's unclear what the goal is by the end of the year. Maybe just list out 5 areas we want to attack, have a sponsored browser to write those areas.
- [simeon] If it is not someone's deliverable, it is not going to be delivered.
- [timothy] I'll take on the background area. `preferred_environment` is directly related to the rest of the `background` key. Focusing on the manifest is probably best. Can't have an extension without a manifest. Suggest others take on other manifest keys.
- [rob] Have we written down our intent to not produce hard errors unless absolutely necessary?
- [simeon] It's not in our spec.
- [devlin] That's a bit hard to write down. Doesn't reflect the reality of Chrome.
- [rob] We expressed this aspiration on error handling before, and universally have allow unrecognized keys at the top level.
- [timothy] That's the WHATWG approach, document the reality, not the aspiration.
- [mukul] We can follow what Rob is suggesting - use tests to drive spec.
- [devlin] Don't want to block on tests.
- [mukul] Agree there, but they can augment and assist.
- [tomislav] Going back to Kiara's proposal, let's try to agree on a few items that we should document and spread out. `host_permissions` seem like another major important area to cover. If we could come up with a few more items before the end of the year.
- [oliver] How do we feel about user gestures?
- [rob] We have some (common) baseline of the concept of user gesture requirement, but there are also aspects that differ (e.g. duration of validity).
- [oliver] Agree we should document reality, but if we have simple things we can address as we spec, then we should do that.
- [devlin] That's documenting reality – we're making those changes true. With regard to user gesture, my concern is that right now if we exclude expensive code changes, we'd be limited to saying "a user did a thing."
- [timothy] I think we'd have less disagreement there than you expect.

- [rob] See value in writing down what we have, then we can iterate on that to introduce improvements.
- [devlin] I'm fully supportive of someone working on this as long as it's not me. Have we done anything on extension origins?
- [oliver] We have a header and no content below it.
- [tomislav] Permission concepts is also a good candidate. I'll take that.
- [timothy] name, version, description
- [rob] To get something, we need to commit.
- **Action Item:** Timothy to add background to the spec
- **Action Item:** Tomislav to add permissions to the spec

Extensions & AI

- (14:35)
- [simeon] "Discussion on experimental AI-related extension APIs that Chrome and Firefox have been trialing."
- [devlin] This category is broad, what do we want to discuss?
- [timothy] Added this as both Firefox and Chrome are experimenting here. Are we in a alignment on naming? Are they still trials? Are we potentially moving AI stuff into a new surface/namespace in extensions?
- [rob] To answer that, the feature set needs to be somewhat stable. My impression is this is iterating quickly right now. Correct?
- [tomislav] Me and Tarek (from the machine learning (ML) team at Mozilla) have had some chats with Chromium folks. There is no consensus within browsers, and extension teams are not the owners of the features. Not sure we're in a position to mediate discussions. They're already in direct contact. Don't think we can mediate alignment in the near future.
- [rob] So what can we do?
- [tomislav] We can try to make it not hard to be compatible with both or either.
- [simeon] I think we should have a recommendation on how APIs are exposed in extensions.
- [devlin] What APIs? In trials? APIs intended for the web platform? I don't think we necessarily need or want to have an opinion about how these are exposed.
- []
- [devlin] Thinking that trial APIs should be under the regular namespace to avoid prefix issues.
- [tomislav] For us, `trial.ml` API is behind `trial` because we're not sure this is the final API we want to ship.
- [devlin] There are APIs we've built for AI data collection that we have no intent to ship. Having those separated out by namespace makes sense. Question is is this intended to be a temporary or permanent restriction.
- [tomislav] Not entirely comfortable with that. If someone came to you and said 'we intend to launch this on standards track for the web, but we want to prototype in extensions' would you recommend that be namespaced or not? How would you approach that?

- [devlin] In that case, I'd say build it out like you would for the web and restrict it to extensions.
- [tomislav] Okay, that makes sense to me. I'd have similar guidance.
- [devlin] The way extensions APIs are built and the way web APIs are built are very different.
- [rob] To make sure that we are about the same thing: by "build like the web" you mean following the web conventions, such as `xxx.onmessage =` instead of `xxx.onMessage.addListener()` .
- [devlin] Yes. If you end up bringing it to the web, then you end up introducing massive changes. If you end up not bringing it to the web, that's okay. Extensions use web-like things even if they're originally intended explicitly for extensions.
- [rob] And not exposing them on the `browser` namespace, right?
- [devlin] If you want it on the web, `browser` does not exist. But that's my perspective. And to Tomisalv's point, we don't own these APIs.
- [rob] Any extension developer participants here that have questions/comments on the AI topic?
- [oliver] Note that the next topic is about browser automation, which looks like it covers potential AI/agent use cases.
- [tomislav] … with the trial.ml namespace, one of the intentions was that this won't exist for the next decade.
- [devlin] The point of origin trials is that we'll experiment and things *will* break. If it is something that you have no intention of shipping, that's when I'd say a separate namespace is best.
- [oliver] To Devlin's point, we have `chrome.aiOriginTrial`
- [devlin] Developer has to opt into it. It's restricted but limitless.
- [oliver] We should spec the `trial_tokens` key.
- [rob] Did I hear a volunteer?
- [oliver] I can do that.
- [timothy] Should be relatively easy to specify.
- [oliver] We can say it's an array of strings. Beyond that, not much to say.
- (discussion of things that could be specified)
- [tomislav] I'll take an action item for what we can do in Mozilla, because unlike Google, Mozilla does not have a self service trial model.
- [devlin] Oliver, can you test as if you had an origin trial?
- [oliver] You can generate your own token.
- [devlin] Waiting to get a live origin trial that you can use in an extension is less useful than having a way to start experimenting immediately.
- [oliver] You can mint your own token using the steps here: https://chromium.googlesource.com/chromium/src/+/main/docs/origin_trials_integration.md#manual-testing
- [devlin] What's Safari's support for origin trials?
- [kiara] I don't think that we support origin trials.
- [timothy] We have locally turned-on feature flags only.

- [devlin] Would be good if we have somewhat consistent support for origin trial-like things. I have a preference for origin trials, as it was built for this exact use case.
- [timothy] Not against it, we just don't have anything that would use it right now.
- [devlin] Might be good to have a lighter weight way to have the same kind of thing as origin trials for extensions.
- [timothy] I think if we wanted to do it, we could have origin trials for extension purposes. We haven't intended to ship any experiments like that yet.
- **Action Item:** Tomislav to check how trial_tokens (issue 454) can be supported in Firefox.
- **Action Item:** Oliver to add trial_tokens to the spec.

Browser Automation

- (15:15)
- [simeon] In a recent conversation with Brian Shultz (CTO of Tango), he suggested that we consider discussing browser automation and agentic client use cases. In short, tools like Browser Use have taken off recently. From that tool's announcement Show HN post:

  > [Browser Use] allows you to build agents that interact with web elements using natural language prompts. We created a layer that simplifies website interaction for LLMs by extracting xPaths and interactive elements like buttons and input fields (and other fancy things). This enables you to design custom web automation and scraping functions without manual inspection through DevTools.
- [simeon] Potential overlap with accessibility (exposing accessibility tree to extensions)
- [kees] (Tango) Some of the things we want to do in Chrome extensions are painful; in browser use, the DOM needs to be processed (e.g. exclude hidden / non-interactive elements) to extract as much context as possible. We call it "simplified DOM". We use `document.elementsFromPoint(x,y)` , which does not reveal elements outside the viewport. It would be useful to more accurately detect relevant elements on the page.
- [rob] How would the browser know what is relevant to the extension?
- [kees] For example, you don't want to expose elements that aren't visible and interactive to the user. Every automation tool will need to build something like this.
- [kees] Another area is simulating clicks and keypresses; we currently use the `chrome.debugger` API for it, and don't like the prominent notification bar (infobar) shown to the user.
- [rob] For this, you're asking if it would be possible to simulate trusted clicks or user interactions, correct?
- [kees] Yeah. Essentially every automation extension will need to do something like this. `chrome.debugger` is very powerful and overkill for this kind of use case. You'd just want to simulate clicks and keypresses.
- [rob] I've also built a number of extensions that simulate these events with DOM APIs (`.click()`, `.value`, dispatchEvent, etc.). Why is dispatching simulated events insufficient?

- [kees] There are some very nuanced events you need to simulate if you want to do that work across all web pages.
- [timothy] A lot of this overlaps with WebDriver.
- [kees] It's not available in an extension, right?
- [rob] The debugger API exposes a subset of WebDriver functionality. Firefox doesn't support it. I don't think Safari does either.
- [timothy] WebDriver has all of that.
- [kees] That is separate software; we want to run within the browser.
- [timothy] I can run a WebDriver session with Safari; it is not an extension but does not have to be an extension.
- [simeon] It may help to show how Tango has evolved into desiring this functionality.
- [kees] Tango makes it easy for people to develop workflows; users can share workflows with others, and the extension can execute these workflows to automate certain actions. Currently it requires full access to the debugger API.
- [simeon] Tango product originally targeted capturing workflows; later they took the captured steps and replays the workflows to automate most of the workflows where needed.
- [timothy] I get it. This might be a situation where we want to expose the extension functionality through the extension APIs. But that capability is super sensitive. We currently expose webdriver behind bio-auth.
- [rob] I see both the power and danger, and the desire to only use a small subset of that API. If we want to get anywhere with this, I think the most concrete step is to identify the common subset of webdriver that we'd want to expose, then work with our internal teams to safely expose those capabilities.
- [timothy] WebDriver is a RPC-style API. It's completely different from the general usage pattern of existing WebExtensions APIs.
- [rob] If the capabilities we expose are a subset of the protocol and a subset of what extensions can already do, it may be enough.
- [timothy] Not sure if we'd have much interest in that.
- [oliver] Regardless of what we do, having that background information on the subset of capabilities would be useful for possible future work. Also complicated because we don't have a BiDi implementation directly within Chrome. We have ChromeDriver commands that invoke Chrome DevTools Protocol capabilities.
- [rob] Would you prefer dedicated extension APIs that build upon the same internals over exposing WebDriver-bidi?
- [timothy] Yes. I can see us reimplementing a set of APIs that are designed for this. Don't want to even entertain the idea of using the developer tools APIs.
- [kees] To be clear, I do not necessarily want the debugger API, it is just the only available method available currently. It has too many capabilities that I do NOT want (along with a warning bar).
- [timothy] What we'd need is a better picture of the API surface you'd need. Clicking, scrolling, etc.
- [devlin] Concern about the possibility of feature creep. Just adding a click API may sound easy; looking through the set of different parameters, there is a ton.

- [timothy] We'd be reinventing WebDriver with a different API surface. Why can't this case be served with WebDriver?
- [rob] Barrier to entry is much higher for a native application.
- [timothy] This would be introducing a parallel version of webdriver with all of it's complexities and subtleties.
- [devlin] We use clicks for more than just the web page thinking there's a user gesture. It's a signal of trust. There are APIs, permissions, browser capabilities, etc. behind user gestures. If an extension can simulate that perfectly…
- [rob] Are you interested in true clicks, or basically "enough" for web pages to think that a click was simulated?
- [kees] Fully understand the concerns about completely simulating the user gestures. We do not need event.isTrusted.
- [devlin] Content script injection requires a user gesture. But for some of your flows, if a user clicks something you may need to replicate the set of user gestures to identically replay the interactions.
- [timothy] Sounds like we may need to put this behind bio-auth in Safari.
- [rob] With all these concerns, how would accessibility tools work?
- [tomislav] They use natively installed software.
- [timothy] And there are a lot of scary prompts to install them.
- [devlin] And some OSs have canonically supported tools.
- [oliver] Would also like to talk about accessibility trees.
- [devlin] Automating user interaction has a lot of value, but it does not necessarily have to be an extension that solves this. An extension with this level of power would be very scary.
- [simeon] I don't think that we necessarily have to implement all of this, it may be enough to set `event.isTrusted=true`.
- [rob] I don't even think that it is necessary to have `isTrusted=true`; from experience of writing these kinds of extensions, the difficulty is that many websites have different expectations on the results of user interaction.
- [simeon] `event.isTrusted` came to mind because it is a property that scripts can filter on. As I see it, the
- [rob] I don't think isTrusted is the key (which can already be done by extensions, e.g. by overriding prototypes), I think easily simulated the entire set of events is.
- [kees] And we could do it today using the debugger APIs, but that's too much power. We don't want that.
- [devlin] `window.open` requires a user gesture.
- [timothy] Another example is video playback with audio.

Issue 693: Add API to switch focus from sidePanel to main document to enable accessibility features

- [kees] One more question; moving focus from the sidepanel to the web page. How to get that further along?

    - [rob] We have discussed this topic at last year's TPAC (issue 693): https://github.com/w3c/webextensions/blob/main/_minutes/2024-09-24-wecg-tpac.md#discussed-issues
    - Issue 693: Add API to switch focus from sidePanel to main document to enable accessibility features
- [devlin] From the conceptual point of view I am amenable to that. It's mainly a matter of prioritization and resources.
- [oliver] This is in our potential list of issues for Google Summer of Code.

Accessibility trees

- [oliver] Accessibility trees! Currently half a dozen or so ways of getting it: debugger API, behind a flag to enable the Accessibility API (private API), experimental AI data API may also expose it. There are lots of ways of doing this. With folks working on AI projects, this seems like there might be some more general use here.
- [tomislav] I don't think there's much interest in exposing these generally as it's not a public API. The tree itself is not well specified and is subject to change. The Browser Testing and Tools group (which works on WebDriver) is also responsible for that. Is there a shared shape of the tree?
- [tomislav] Don't think so.
- [simeon] Every browser has their own implementation.
- [rob] Shared concept, but different details per browser. Similar to debugger API discussion, could expose a shared wire format.
- [oliver or devlin] I would like a more API-shaped.
- [rob] Chrome already has `accessibilityPrivate`; if the functionality has general utility, why is it then not exposed?
- [devlin] It is an internal implementation detail of the browser.
- [oliver] From what I have seen on what people are building, they may be comfortable with not-a-real-API and updating the extension to keep up with changes.
- [devlin] I'm concerned that if we expose this as a higher-level thing, that developers expect consistency.
- [timothy] Agreed.

Testing Sync

- (16:00)
- [simeon] How do we get WPT testing across the finish line? Writing tests, getting them running on WPT.fyi, and aligning toward 2026 interop.
- [timothy] We have made a lot of progress; I'll hand it over to Kiara and Elijah who work on this.
- [kiara] PR in WPT to load and unload extensions through WebDriver (and RFC). Currently waiting for review.
    - WPT PR: https://github.com/web-platform-tests/wpt/pull/50648
    - RFC PR: https://github.com/web-platform-tests/rfcs/pull/219

    - ([rob] ^ is this the main PR you're referring to?)
- [kiara] A goal on our end is to be involved in interop 2026. Ideally extensions would be a focus area.
- [simeon] I'm interested in digging more into interop.
- [kiara] I think they discuss topics near the end of the year.
- [timothy] If we're not ready for interop 2026, we could go for an "active investigation area." A lower stakes, but good to make progress on approach.
- [tomislav] Quick status update on the Firefox side: I've successfully implemented this on top of the Safari patch that executes the current tests. One issue on top of the current design is that in the current PR, the browser.test event listener is only added after the promise for loading an extension is resolved. Currently relies on the parent page closing – not sure that's a strong enough guarantee on the origin of the events.
- [kiara] What we do on our end is queuing messages until loading has happened.
- [tomislav] I implemented queuing as well; the question is what we should queue etc. E.g. if we run a second test, should the queued message count.
- [rob] … Let's table these details for PR discussion and the regular testing sync that we have, and focus today's meeting on the higher-level strategy?
- [oliver] Curious if I missed it. Is there a reason you can't register the test before?
- [tomislav] You don't know the extension ID before you load the extension, so you don't know what to listen for before you start.
- [rob] What are the other questions we should focus on today?
- [tomislav] Do we have someone on Chromium team that can look at the current PR and point out potential blockers or the viability of this approach? Can we get someone to take a look?
- [devlin] I can try to find the time to take a look. Don't know if anything here is a hard blocker, so this may further delay work.
- [oliver] Alex Rudenko (automation team / WebDriver) may be a relevant expert.
- [tomislav] Any way to assess viability without fully implementing?
- [devlin] At some point, yes.
- [tomislav] How should we proceed?
- [devlin] I can try to take a look soon. Would it be helpful to review now? Would be valuable to have input from someone on Chromium before proceeding, otherwise we may only have 2 implementations. I think the main outcome will be determining what can be done in a given amount of time. My biggest reticence was around not doing more, but doing less.
- [timothy] We've implemented test.runTests in WebKit.
- [kiara] For testing, may be as simple as adding a `browser==chrome` check.
- [devlin] We've had discussions before that it's not just modifying browser.runTests, but doing that under a certain set of conditions.
- **Action Item:** Devlin to review the WPT PR.

wpt.fyi and interop

- [rob] Another point of the agenda was getting tests running on WPT.fyi. Is that an open point of discussion?
- [kiara] Is that site just the result display for the interop project?
- [rob] wpt.fyi predates interop.
- [timothy] No, interop is a subset of the tests on that site.
- [kiara] Having a list of … This could be expanding the test for the extensions runtime namespace. Could add tests for the permissions API. Maybe (browser) action. Other APIs that would be good to get the ball rolling.
- [tomislav] Not sure how much of the permissions can be tested without implementing granting permissions. That might be a separate blocker.
    - [timothy] I think we were planning on auto-accepting any dialog that got presented, which would include permissions requests.
    - [kiara] Yeah. Also considering modifying the load command to adjust that behavior in certain cases.
    - [tomislav] Ah, didn't realize. I was just assuming permissions from the manifest would be granted on initial install.
    - [kiara] Correct.
    - [tomislav] Missed the second part about `permissions.request` being auto-granted. We also do that in some of our tests. Not sure if it's better as default or non-default behavior.
    - [timothy] That was the initial plan, then possibly adding a toggle later to test deny requests.
    - [simeon] Would this only apply at the extension level? Would you not be able to mix approving some requests, denying others?
    - [timothy] Worth discussing. Could be designed a number of different ways.
- [oliver] Are there specific benefits we hope to get out of interop? Or is it just a public signal of intent?
- [tomislav or timothy] Isn't the intent to publicly signal alignment?
- [rob] Isn't it premature to schedule being part of interop while we don't have cross-browser WPT? It would be too aspirational.
- [tomislav] Interop could also be "having cross-browser WPT".

2025 WECG Berlin F2F, Public Notes, Mar 27

Thursday, the third day of the four-day face-to-face meeting in Berlin (announcement: #759).
Agenda: https://github.com/w3c/webextensions/wiki/2025-Berlin-F2F-Coordination

Attendees

- Oliver Dunk (Google Chrome)
- Devlin Cronin (Google Chrome)

- Simeon Vincent (Mozilla)
- Rob Wu (Mozilla)
- Tomislav Jovanovic (Mozilla)
- Carlos Jeurissen (Jeurissen Apps)
- Mukul Purohit (Microsoft Edge)
- Timothy Hatcher (Apple)
- Maxim Topciu (AdGuard) – starting from dNR: What's Missing?
- David Tota (AdGuard) – starting from dNR: What's Missing?
- Kiara Rose (Apple)
- Casey Garland (Capital One)
- Jordan Spivack (Capital One)
- Krzysztof Modras (Ghostery)
- Jan Biniok (Tampermonkey)
- Oleksii Levzhynskyi (Grammarly)

Notes

WG Discussion

- [simeon] Advance the conversation around creating a formal Working Group. What process do we need to follow? Who would chair? How would the Community Group and Working Group coexist—could the CG act as an incubator? Let's establish a plan for governance, scope, and collaboration going forward.
- [simeon] Desire is to have a Community Group like we have today, and a Working Group.
- [timothy] Community Group is more open to external contributors, whereas a Working Group is more limited to invited experts.
- [devlin] Are there other examples where a WG and CG co-exist?
- [simeon] Privacy groups.
- [simeon] In formal working groups, there are three levels with varying levels of openness in meetings: public, group, team. What would change here is that the chair would have to dig in the W3C policies so we can follow the documented procedures. A WG has more formality, which is partly why I don't like to convert this CG, since our current practice encourages broad participation.
- [timothy] Agree, I'd like to keep that.
- [oliver] Our CG has multiple mailing lists. public-webextensions and internal-webextensions
- [simeon] public is public, internal for all W3C members who have signed up. We also have teams-webextensions.
- [oliver] What is team-webextensions?
- [simeon] W3C staff and invited experts. We asked a mailing list for a leads list, and this is what we got.

- [oliver] One of the questions in the WG charter template is to choose the venue for discussion. I recommend going with Github issues over mailing lists given our good experience with that format.
- [simeon] Agreed.
- [simeon] The most substantive change is the way we publish technical reports, which have more formal requirements.
- [oliver] What is the point in time where we publish something? Are we basing it on the manifest version, for example?
- [simeon] To some extent for the editor to decide. Also something we would need to decide, the frequency of publication.
- [oliver] Is a living standard possible? I'd like to lean towards that if there is an option.
- [devlin] I'd like to not tie to manifest version - we'd like to publish more often than updating the manifest version.
- [simeon] Expect that the biggest change for publishing updates will be the need to call a W3C API to formally publish drafts as we iterate on them.
- [oliver] Sounds like we'd like to retain the flexibility of updating content as needed, without too much hurdles to publish outdated content.
- [krzysztof] How can the community group support the WG? Does it preclude CG members from contributing to the specs from the WG? Or can we establish a process where external contributors can submit PRs to the spec and be accepted?
- [simeon] The way I see it - CG is focused on discussing substantial changes, and the WG on the execution on what has been discussed. When we have a proposal in the CG right now, that it would support specification in the WG.
- [carlos] What are the objectives of creating a WG? What are you trying to achieve with a WG?
- [simeon] A limitation of CG is that it cannot produce formal standard specs, WG can.
- [rob] Is the assumption of community members being unable to contribute true? Why would contributors not be able to propose a change and then be accepted by an editor.
- [simeon] Patent obligations, copyright, licensing requirements, etc.
- [oliver] We see opportunities of collaborating with other working groups. There is no hard requirement for the group to become a WG, but WG has a higher status and credibility that can help with involving other relevant specs.
- [oleksii] Can contributors add a disclaimer to give up the rights necessary to allow the content to be merged in the spec?
- [krzysztof] That is the purpose of the invited expert role.
- [simeon] Another in-between is an Interest Group, which is a more formal version of CG.
- [rob] Any advantages in terms of spec writing?
- [simeon] Interest groups are still not empowered to produce standards.
-
- (Simeon going round the table) Any concerns about WG?
- [devlin] (Google) I cannot speak to the legal aspects; Personally open to exploring WG.
- [mukul] Fine.
- [simeon] Capital One folks?

- [casey] (Capital One) How would it impact the WECG? E.g. would it impact the frequency?
- [simeon] The existing biweekly public meeting cadence will continue. I would like the change on the WECG side to be null.
- [timothy] Not expecting changes. We already have Leads meetings in the alternative weeks.
- [] Excited about potential positive impact on developers.
- [carlos] Is this mostly about getting a spec down?
- [simeon] A WG has specific deliverables; more progress on specification is needed for creating a WG.
- [] Do you see the WG as something with an end date?
- [simeon] A WG has a required end date; however it is extremely common for a WG to be renewed. A WG term is 6 months to 2 years, and can be extended.
- [carlos] How would the creation of a WG result in more spec work, since the CG has currently not made much progress on spec work? Does the creation of a WG result in more time allocation?
- [simeon] No guarantees, but this can help with organizational buy-in and is making more of a commitment towards deliverables.
- [oliver] Similar.
- [rob] We (Firefox) are favorable towards the direction of WG.
- [oleksii] If a WG is established for 6 months to 2 years. What is the expected outcome? E.g. a new manifest version?
- [simeon] I don't think that a new manifest version would be desired outcome. A specification of a common platform that underpins WebExtensions across browsers, and a test suite to automatically verify that.
- [oleksii] Is it not about introducing breaking changes?
- [simeon] On the contrary, reducing inconsistencies.
- [timothy] Documenting what we already have, and integrating community proposals in the spec.
- [krzysztof] Wondering how subtle differences will be accounted for in the standard.
- [simeon] It is unusual yes. I think that we should spend time and energy on approaching this.
- [krzysztof] Overlap with WebDriver spec?
- [simeon] The way I see it, the Browser Testing and Tools group allows for other groups to extend their spec.
- [krysztof] An extension standard would also help them; currently overlapping parts are explicitly not specified. Having an extension standard can facilitate cross-referencing.
- [rob] You mentioned overlapping specs and I know that you contributed the extension loading part to the WebDriver spec. Are you referring to what held you back of being more detailed, or is this about something you observed in that group?
- [krysztof] For example, evaluating scripts in a web context. They could not include information about web extensions contexts because that doesn't exist in existing specs. That in turn creates implementation differences in the platform. Chrome integrated

extension into their WebDriver support, but Firefox passed on it. Having those concepts in a standard would enable WebDriver editors to reference and include those ideas.
- [simeon] Apple folks?
- [timothy] (Apple) We have been pursuing a working group internally and are near approval.
- [simeon] Looks like we have clear consensus on this being useful.
- [casey] What happens if some core members do not get approval to proceed with a WG?
- [simeon] Depends on the concerns. We would likely try to work through the objections, which may affect the scope of work.
- [oliver] To add to what Devlin said before, we are having initial conversations internally, and final approval can only happen when the charter is finalized. That having said, I don't expect major issues.
- Next steps: finish draft of charter.
- **Action Item:** Oliver to drive finishing draft of charter for WG

Upgrade Permissions

- [simeon] Explore proposals to improve the extension permission model, especially for onboarding and updates.
- Relevant issues/PR:
    - Issue 711: Proposal: Enable changing the permissions for only new user
    - PR 788: Proposal for "Hybrid Permissions"
- [simeon] Issue 711 - issue generally is that extension updates with new permissions prevent users from continuing to use the extension. This topic discusses a way to introduce permissions that are granted at install time but optional on updates.
- [oleksii] Big challenge for extensions with an existing user base is introducing features dependent on new permissions. Would like new users to have the complete experience, and migrate existing users over. PR 788 proposed a way to request permissions for new users, but allow existing users to update without requiring the permission to be granted. The exact name does not matter. The desire is for the permission to be granted without it being removable. There are also permissions that can currently not be optional, only in (required) permissions such as declarativeNetRequest and devtools.
- [simeon] Immediate question that comes to mind is - how is this different from the optional-only concept?
- [rob] They are opposite.
- [oleksii] From the user perspective, a post-install permission request, after the user has just approved an install prompt is not ideal. Users may also miss the notification and fail to grant the necessary permissions.
- [rob] Do we need to discuss use cases any further? If we all agree that this is a problem, can we proceed to solutioning?
- [devlin] I think it's an issue we should address, but I have a different solution in mind. This is trying to solve a challenge with how browsers present UI. Doesn't necessarily need a manifest change. We want to move to a world where all permissions can be toggled by a user. Chrome allows users to toggle all host permissions. Want users to be

able to control all permissions. What if, rather than introducing a new key, what if we just adopted the optional behavior for permissions in the `permissions` field? On install, you get the declared permissions. On update, new permissions entries are treated as optional (not granted by default)

- [rob] Developers would need to opt in, or else extensions designed without this model in mind can break.
- [devlin] A new permission in the permissions key would also mean new code to interact with that permission. For developers, would anyone prefer having your extension disabled when a new permission is added? (all: laughs) Okay, about what I expected.
- [jan] I like the way Firefox handles things today. If a user doesn't want the new permission, they can stay on the old version.
- [devlin] Discussed a bit on the first day. Don't think that's something we'd do in Chrome.
- [jan] Hard for users to determine what permissions are necessary. Having to always check what declared (required) permissions are available is unnecessary overhead.
- [devlin] Don't have to solve every permission being toggleable as part of this discussion. There's power in defaults. Toggling is likely a power user feature. TamperMonkey handles some of these cases well, e.g. prompt notifying the user that host permissions are required. Technically possible to make any permission optional today via enterprise policy.
- [jan] It's fine to make all permissions optional as long as there's a way to give it everything.
- [devlin] Extension should message the user to make it clear when the user doesn't give them the permissions needed. For example, if you install a camera app and deny the camera permission
- [oliver] Concern: extension devs would have to keep track of the full history of changes to determine which permissions may potentially be missing.
- [devlin] Concern no matter how we tackle this. If you introduce a new manifest key, that would still be an issue. I'm curious if anyone wants their extension to be disabled.
- [rob] Starting from the wrong premise. Of course devs want a better experience. Firefox's approach is already better than Chrome's. The bar for improvements is higher than "disabling extensions completely" vs something slightly better.
- [oliver] Better question might be are there any devs that would prefer Chrome's behavior over Firefox's
- [krzysztof] All approaches to adding permissions are scary today. Idea of making all permissions optional is interesting but very complicated. Take the tabs permissions – there are hundreds of places where this capability is assumed. How would you meaningfully make that optional?
- [carlos] Extensions have different types of permissions: core permissions (really required), helpful permissions (that can be disabled), and very optional permissions (requested by the extension in some cases).
- [rob] By permissions that can be disabled, do you mean what would be described here as "hybrid"?
- [carlos] Yes. Browsers could handled these differently, but this would give a clear way to communicate intentions with the browser.

- [devlin] Say we introduce this new field & fast forward 5 years, is any extension going to add a new permission to the ultra required ones?
- [oleksii] Yes, I expect so. E.g. permissions without warning.
- [rob] And if a permission had a warning, the extension would be disabled. I think that developers aware of the nuances would not add new permissions with warnings.
- [devlin] Agree that Firefox has a different consideration. Given that we don't want to do that on Chrome and that most devs try to target multiple browsers…
- [rob] Why not in Chrome?
- [devlin] We want extensions to be up to date. Same reason we want users to update the browser. Should be as easy as possible to have the latest security improvements, minimize the number of users lingering on older versions.
- [tomislav] You don't disable Chrome if people don't update it.
- [devlin] We're closer to that than you may expect. We get very noisy and have considered forcing restarts.
- [carlos] Can't customize behavior for every different browser/store. New key keeps backwards compatibility.
- [rob] Would suggest pursuing this with a new key, but using a boolean rather than an array. Flip the default value of the boolean in a future manifest version. More backwards compatible.
- [timothy] Like that approach. Gives a clear signal that they're opting in.
- [devlin] Open to having a boolean. Biggest thing I want to avoid is more sets of permissions. To Carlos' point, I understand that there are multiple different sets of permissions, but don't think we need discrete fields for each.
- [carlos]
- [rob] That's separate from making everything optional
- [carlos] Starting with a boolean would mean you miss this information. It's all about better expressing intent as an extension developer.
- [rob] I understand there are permissions that the extension really wants. For example, with activeTab the browser does not have enough information to tell what the extension's intent is. There's a difference between an extension that really needs to run everywhere or if it just can run everywhere, when the user chooses to click on the extension button for a specific task.
- [carlos] I can try to write something up to be more clear.
- [devlin] Concern is that browser UI could become too overwhelming. Don't think users should see 3 sets of permissions on the extension's settings page. Want to move towards a world where users can toggle permissions even if the extension doesn't want that because user choice comes first.
- [carlos] Extension may not be able to do anything in that case. Better to not have it installed at all.
- [devlin] Extension can uninstall itself if it truly feels that way.
- [rob] Discussion of everything optional should be tabled for now. We should not block on that to resolve the current problem.
- [oleksii] .

- [devlin] That's the argument that says we shouldn't make it the default, leave `permissions` as fundamentally required. But extensions that have an established user base will almost never touch this field after getting a notable user base. If that's true, then we've effectively done that by changing the behavior of the key.
- [simeon] Expect that the big challenge with reusing the existing field is a change in developer expectations.
- [devlin] Right, need documentation, messaging, etc. Could also be addressed with a boolean key.
- [rob] What about host permission?
- [devlin] Same approach.
- [rob] In Firefox we tried not granting host permissions by default in MV3, but that resulted in lots of friction, so we grant host permissions by default at install. New host permissions do not prevent updates, but they are currently not granted on update. The new key can be used by extension devs to opt in to preventing updating until the user opts in.
- [devlin] Then you can have the unspecified case be the current behavior.
- [krzysztof] What kind of support timeline would this have? Say ESR releases.
- [devin] Don't think that's any different from how you'd approach supporting old versions. Set a minimum version.
- [rob] Sounds like we are all aligned on a new boolean.
- [timothy] Yes.
- [devlin] Maybe an enum in case we want to have more states.
- [rob] In Firefox we would already have three states, with host permissions (not) granted on update as third state, as discussed before.
- [oleksii] Would it make sense to simplify permissions in MV4 to reduce the number of fields?
- [devlin] I think we'd change the schema for permissions in MV4. One thing we've generally aligned on is having permissions justifications in the manifest. But that's some time off.
- [rob] To recap, we aligned on a new manifest key with an enum value.
- [oleksii] Ask that if we go this direction, permissions without warnings are auto-granted.
- [devlin] That makes sense.
- [rob] I'm leaning towards only applying the "not granted" behavior to permissions with warning.
- [oleksii] I'll create a PR with a new proposal.
- [carlos] Do we have a sponsoring browser for this proposal?
- [rob] I'm willing to sponsor; I think that extension developers would see most value for this in Chrome given their current behavior of disabling extensions.
- [devlin] Accepting patches, but cannot prioritize its implementation ourselves.
- [timothy] We will eventually implement it.
- **Resolution:** Chrome, Firefox, Safari supportive of a new enum field to customize permission granting behavior on update to address issue 711
- **Action Item:** Oleksii to create proposal PR for issue 711.

DNR: What's missing?

- (13:30)
- [simeon] Discussion on limitations and missing features in `declarativeNetRequest`, with a goal of identifying browser support and potential sponsors for improvements.
- Relevant issues:
    - Issue 439: Proposal: Add filter for DNR `modifyHeaders`?
    - Issue
    - Issue 493: Support for redirect-only rules.
    - Issue 783: Conditional enabling/disabling of DNR rules.
    - Issue 694: Unlike webRequestBlocking, DNR rules are not applied to redirected requests — should this behavior be reconsidered?
    - (the last two issues were not discussed due to time constraints; topic may be scheduled tomorrow)

Issue 439: Proposal: Add filter for DNR `modifyHeaders`?

- [carlos] Issue 439 (modifyHeaders filter) - no progress. What's next?
- [rob] Status is: patches are welcome? Implementation-wise probably feasible.
- [timothy] Proposal would be welcome.
- [devlin] Someone on the Chrome extensions team is slowly working on this.
- [timothy] One gripe, the proposal proposes `match_all` whose snake_case syntax is inconsistent with the other properties that use camelCase.

Issue 468: [DNR] Redirect rule does not allow applying other redirect rules to the resulting request

- [maxim] I showed example to Rob; both rules have broad conditions and redirect actions that have removeParams.
- [rob] DNR is based on conditions. If there's a redirect rule, that will be the final rule applied. In this example, all URL filters could match and cause parameters to be removed. The current result is that all-except-one of the redirect doesn't get applied. There is an understandable interpretation of the rules that would have all of them applied. Does Safari have `removeParams`?
- [timothy] Not sure offhand. (Edit: We do.)
- [oliver] Would it potentially work if we treated removed params as a condition?
- [rob] Before approaching solutions, is this a case we'd want to solve? I think yes.
- [oliver] What is the impact of not having this on AdGuard?
- [maxim] Multiple rules are generic, should work together with more specific rules to ensure multiple websites aren't affected. It's hard to fit everything into a single rule as previously suggested in this issue.
- [devlin] Why doesn't the second redirect rule apply after the redirect is taken?
- [rob] After the first redirect is applied, the conditions for the second rule no longer match.
- [krysztof] We have coincidentally encountered the same issue in Firefox. We solved this in our case to merge all removeParams options together in one rule.

- [rob] That would work if conditions are overlapping, but wouldn't if not.
- [krysztof] For webRequest, not DNR.
- [maxim] We have multiple conditions, e.g. resourceTypes and exclusions that prevents us from merging multiple in one.
- [devlin] I think that this makes sense to solve.
- [timothy] We do support removeParams in Safari.
- [rob] Your implementation predates DNR though, right?
- [timothy] We added support to make it DNR compatible, but don't know the specific details.
- [rob] Can we create a test case?
- [timothy] I can try that, but don't block on me.
    - [timothy] We also apply only the first rule.
- [devlin] There are a couple different types of conceptual redirect actions. On one hand: Make an example.com request go to a different domain. On another: tweak this existing request. We could look for combinable actions and apply all of them.
- [krysztof] Not all are combinable. For example, block.
- [oliver] Could we have a new action type, like safe redirect?
- [devlin] Was thinking about that, but what about other cases like modifyHeaders?
- [rob] modifyHeaders can apply multiple times. Normally if you redirect elsewhere it could match other rules. … Right now, problem is that when you match the first request it executes the action. You're suggesting adding a field to make them combinable?
- [devlin] Or, following oliver's suggestion, introduce a new type of action that is inherently combinable.
- [rob] How would extensions like to do this? Any preferences?
- [maxim] Both approaches could work.
- [krzysztof] Neither will be reliable during a transition period. It will always generate some breakage. If we make this a separate action, what if there's another redirect rule that should apply to the new URL?
- [rob] It's a new request, so all rules will apply again. An alternate option is after an action has been executed for a specific request, opt out of the redirect chain. Not a perfect solution: could create an issue where redirecting to remove an affiliate goes to a new URL that introduces a new affiliate.
- [oliver] Is it enough to say it opts out of the next redirect in the chain?
- [devlin] Real weird, hard to predict, necessitates more rules.
- [rob] Another option: If the redirect action didn't change the URL, go to the next matching action.
- [devlin] Not backwards compatible.
- [rob] Could add a flag to control application of the rule in a backwards compatible way. Can you think of a case where an extension wouldn't want this behavior?
- [devlin] Only tautological cases. Currently highest priority rule wins, if we did this and a not-highest priority rule wins, that'd be a tautological violation. Having a new action type seems explicable and clean. Also okay with having a property on the rule. Are there times where cases with, say, remove param, where you don't want it to be combined with other actions?

Modify headers will always apply matching rules in a defined order. With something like remove param, would we always want those to be combined? If so, having a property would mean devs have to just always include the param.

- [rob] Should only be combined if it's the only modification in a redirect. Don't necessarily think we need a new property if the expectation is to just drop it. Argument against a new action is that it's not unique to params. Upgrade scheme could also apply: that case is resolved by having the upgrade be a lower priority. Do we want to introduce a new action when we encounter a redirect?
- [devlin] Would probably be a broader category of combinable redirect.
- [oliver] Eventually has to be implemented as a redirect, but individual rule matching doesn't have to apply each separately
- [krzysztof] Could be cases where there are multiple rule authors that aren't actively coordinating.
- [oliver] Should learn from past like case sensitive checks, where we had to change the default.
- [krzysztof] Not feasible for adblockers to try to determine which rules can be combined.
- [rob] Inclined to change the behavior to something developers can expect.
- [devlin] Would it break anything to make any redirect action – Not sure how to avoid breaking things. Scheme, port, path, target, query params
- [rob] Could automatically include removeParams in the condition.
- [devlin] That would force a redirect every time. If you had 7 redirect rules that remove params, apply each separately exhausts the redirect limit and is more expensive to execute those requests. Two questions. With something like remove param, could we just make that change? Say next version always applies remove params.
- [rob] could do it in Canary-only for a few cycles?
- [devlin] Too hard to see the effect, too many moving parts. Would prefer to send out an email and ask "if we change this, does it break you?" If no one replies, we're good.
- [oliver] Could we do an origin trial?
- [tomislav] Doesn't tell you if it breaks people that don't opt in.
- [oliver] If AdGuard tries, it doesn't tell us about others, but it does tell us about AdGuard.
- [devlin] Could do that to gauge changing the default, but need outreach here.
- [krzysztof] Do you guarantee the order of the rules? Say we have multiple remove params, can I be sure that one rule will always apply first?
- [oliver] Order of rules with the same action and priority is undefined.
- [krzysztof] Having all of them apply best matches expectations, can't do it the other way around
- [rob] Are we converging on trying to remove them all at once?
- [devlin] Going that direction, but we should do outreach. Should also consider where else this could apply.
- [rob] Don't think we should apply this to append, could go on forever.
- [devlin] Why?
- [rob] Say you have a rule with abc and you drop b, you then match c. Removing all params at once makes sense because

- [oliver] I think you're saying: do we match on the initial request with an initial set of query params…?
- [devlin] Common case: take the original URl, take all matching removeparams, that's the output url. After that it goes back into DNR and looks for more matching rules. If you have one that happens to match, it becomes part of the redirect chain. That's fine, you matched the new URL, not the original URL. If you had a match on abc and not "a not b c"
- [rob] Could limit number of redirects to things that are likely to stack together.
- [rob] If removing a param could match other conditions, then we may fail to apply rules that should have applied.
- [devlin] This isn't a new problem, though. Don't we already have this with headers?
- [krzysztof] Once you've finished removing the params, you trigger a redirect and then re-check matches.
- [rob] Say you remove Y if the param contains X, but not Z. Then there's another rule that drops Z. Point is removing a param can change conditions. Not safe to assume you can remove params together.
- [devlin] Also applies to headers, right?
- [rob] How do you solve that? We haven't implemented matching headers yet.
- [devlin] I think we look at the initial state and assume everything else matches.
- [krzysztof] Doing multiple passes may not be an issue if you're removing params. There aren't that many remove params rules. Separate index may not be such a big problem.
- [rob] Comment about scanning makes me wonder if we should allow rules to reference other rules that should be considered.
- [devlin] Don't think that will scale.
- [oliver] Seems like both issues we're discussing are subtle changes. Don't think doing one prevents changing to the other in the future.
- [rob] To address the issue of conditions not really matching, we could introduce a property to allow extensions to specify whether an initially matched condition should still match after other transformations.
- [oliver] If there's minimal risk to having the behavior, let's not add a property?
- [rob] Let's create a proposal and discuss it there.
- [oliver] Would like to get some real word data on how this is working.
- [rob] Are you not worried about mis-applying rules?
- [oliver] I think we have the same issue with header rules and so far it hasn't been an issue.
- [devlin] Think we should post to the group to raise visibility and gather feedback. Ask the community what issues this might cause.
- [krzysztof] Just checked EasyList, they have a total of 240 remove param rules. They seem to be on the simple end.
- **Resolution:** Explore feasibility of solving issue 468 by changing DNR to apply all removeParams rules at once.
- **Action Item:** Oliver to solicit feedback from ext devs to determine feasibility of applying multiple removeParams by default.

Issue 493: Support for redirect-only rules.

- [maxim] What should be done next to progress this issue?
- []
- [krzysztof] We could come up with a solution, but you may not like it.
- [devlin] We discussed a solution last year, just haven't had bandwidth to implement.
- [krzysztof] Kindly request you prioritize this as it's about website breakage.
- [rob] Can we generalize this?
- [devlin] I think this is the generalized issue.
- [oliver] Why don't allow rules work here?

Action & Sidebar UI

- (14:30)

Issue 198: Proposal: action.default_area=navbar and action.default_area=hidden

- [carlos] Allow extensions to express their preference for the location of the extension button. Firefox already puts the button on a different location if requested. Even if browsers do not use this information right now, it could also be used to inform the browser on the request to be added to pin an extension button. Chrome used to support pageAction to inform where the button was shown (next to the address bar), but it was dropped at some point.
- [oliver] Proposal only makes sense if the browser is likely to change their UI. Would we show "this extension wants to be pinned"?
- [devlin] No.
- [carlos] Edge displays open sidebar extension. An icon like that could also be used for a popup. Currently for the extension menu it's not clear that there is or isn't a popup attached to it.
- [devlin] Seems orthogonal to this key. Agree that it's a pain point that it's unclear what clicking the action would do, how does that relate to default_area?
- [carlos] Allows you to signal intent.
- [rob] So you want to express that the extension button is an important part of the extension, and not e.g. auto-generated by the browser.
- [timothy] We can mostly infer that.
- [devlin] If you're showing indication, we wouldn't want to conditionally show that it will open a popup based on something that's not a popup.
- [krzysztof] What about the argument of giving user's choice?
- [devlin] User already has the choice to click the pinned button.
- [krzysztof] They have to know about that.
- [devlin] Yes.
- [krzysztof] User testing has shown this is difficult for users to find and use.
- [devlin] Concerned about preserving user choice.
- [carlos] Consider installing an app. You have to add multiple screens to see all the apps you have installed.

- [devlin] Conflating multiple things. One is an API. manifest key. (...) defaults. With respect to the API, we are not going to allow extensions to take over user control. Agree that it's harder to use, but does not fundamentally break.
- [devlin] Do you want any of your extensions to be hidden?
- [carlos] Yes, some are useful in the background, some are important in the foreground.
- [devlin] Would you actively tell users to hide the extension button? Or if you had an unpin, would you call a method to unpin it?
- [carlos] That would advocate for the extension to be pinned by default.
- [rob] Are there cases where you would actively not choose to put yourself in front of users? The thought behind Devlin's question is that most extensions would probably consider themselves to be important.
- [carlos] For some extensions, yes. Being part of the install flow would be an improvement there.
- [devlin] If the default was pinned, is that a good state? If I made it so all extensions are pinned by default, would that be okay?
- [carlos]
- [krystzof] How about Safari's default? Show the first few, and then hide additional one. What's the average number of installed extensions?
- [devlin] Relatively few.
- [krzysztof] Maybe that's fine then.
- [devlin] Hesitant with key because it includes browser UI-specific parts in the extension UI. navbar may be browser-specific.
- [carlos] Goal is to communicate intent and give options. Ultimately its the browser's choice what to do with the data.
- [oliver] I get the idea of offering to pin some but not all. Don't know that it makes sense to add a key that doesn't do anything.
- [devlin] I think it's valuable to have consistent behavior for extensions. Don't want some extensions to appear in once place and some in another.
- [rob] Get that the current default is too hidden. But if a user installs multiple extensions, you still hit the situation where you have to tell the user what to do. Might solve for the majority, but doesn't solve for everything. May not be an API issue, it's a browser UI issue.
- [krzysztof] Never a good moment to do this. Onboarding flow, permissions flow, etc. are already occupied.
- [jan] Would it make sense to show this in the installation dialog?
- [krzysztof] How many choices do you want to expose during the installation flow?
- [devlin] Exactly. Agree that there's a discoverability issue with the current flow.
- [krzysztof] What about allowing the extension to pin itself with an API?
- [devlin] User gesture is a low bar.
- [rob] Are there other ways to get the desired outcome? Say "yes pin, no, remove extension"
- [devlin] oof
- [rob] Trying to minimize abuse.
- [devlin] Personally okay with prompt, not sure if Chrome UX would be.

- [krzysztof] Would it be possible to pop the extension out of the menu to remind the user it's there?
- [rob] action.openPopup()
- [simeon] That's essentially how the installation flow works right now – the extension emerges from the menu, hides inside it when the notification is dismissed.
- [krzysztof] Pop it out once a week to remind users it exists.
- [carlos] Devlin, could you ask the UX team?
- [devlin] Yes. I'm surprised that the people here did not want to be pinned by default.
- [krzysztof] To be clear, this is a very big need for us.
- [oliver] We have heard this feature request before.
- **Action Item:** Devlin to ask Chrome UX whether it is acceptable to allow extensions to prompt for the button to be pinned to address issue 198.

Issue 128: Unified sidebar proposal

- [timothy] We discussed this before; we agreed that `sidebar` would be the ideal name and that perhaps we should alias at some point.
- [carlos] What needs to be done to get there?
- [timothy] Safari supports introducing a new name / alias.
- [oliver] Are we all aligned on a common namespace?
- [devlin] In a perfect world, we would have a common namespace. Would like to see a mapping of what the properties should do. That could unblock us. If there is no way to map, that would be an issue.
- [rob] If someone were to come up with a mapping, would Chrome be willing to add `sidebar`.
- [timothy] Since we looked at implementing `sidebar_action` / `sidePanel`, we are willing to drive a proposal.
- [simeon] Would Firefox be open to introducing the concept of a contextual side panel? Want to make sure there's consensus on the direction first.
- [rob] Open to it. I'm looking forward to what Timothy comes up with, and align in Firefox to have cross-browser compatibility.
- **Resolution:** Chrome, Firefox, Safari in favor of a unified `sidebar` namespace if feasible.
- **Action Item:** Timothy to create proposal on unified sidebar API to supersede sidebarAction and sidePanel.

Lifecycle & Matching

Discuss support for top-frame matching/exclusion and lifecycle event proposals.

- Two topics
    - Lifecycle events (#138, #353): Proposals like `runtime.onInvalidated` to detect when an extension is unloaded.

    - Top-frame matching (#763): Proposal for `topFrameMatches` / `excludeTopFrameMatches` to give extensions better control on the top-level document content script injection.

Issue 138: Proposal: let content scripts listen for event to detect when the extension is unloaded (for various reasons)

- [carlos] Created PR at https://github.com/w3c/webextensions/pull/792.
- [rob] Typically when an extension is loaded or unloaded, an extension breaks. For `runtime.onInvalidated`, when would it fire? Before invalidation, after, during?
- [devlin] In Chrome, you'd get the event but we can't make any guarantees about extensions APIs being available when you receive. After the event is dispatched from the browser process, we'd unload the extension from the browser process. Even though the renderer would allow it to be called, it would then fail with unknown extension.
- [rob] If we specify this, what would be the order? Can we say anything will work?
- [devlin] Would say you can't rely on any extension capabilities once this is fired. Things aren't guaranteed to be removed, but they also aren't guaranteed to work.
- [timothy] We don't unload content scripts.
- [rob] Wonder what we should specify for this event.
-
- [rob] Would you be okay with keeping the extension context alive as extension work is done?
- [devlin] We already do in Chrome.
- [krzysztof] can we get one tick? Reminds me of the Firefox bootstrap extensions days.
- [rob] There's a hack available: extensions can execute code in the MAIN world. If extension devs are okay with one tick, I'm fine with that.
- [devlin] You could have async cleanup to do.
- [rob] For example could have a Promise that need to resolve before you can reach a stable state.
- [devlin] If we want to give developers time to clean up, it seems reasonable that some of that could be async.
- [krzysztof] So you could return a promise from your event handler?
- [devlin] Moot for us, but yes.
- [rob] I can see sync or async resolution. But don't need to spend too much time on that now.
- [oleksii] What would the behavior be for updates?
- [rob] If we had async support, this could allow extensions to do some kind of handover from the old to the new scripts.
- [tomislav] Do you do that now? Handoff between old and new content scripts?
- [oleksii] We try to. Another use case – if the background is suspended, ports are disconnected. No way to know why the port was closed.
- [rob] In the interest of time, suggest we move on.

Issue 763: Proposal: Add topFrameMatches / excludeTopFrameMatches

- [carlos] Trying to gauge interest in browser support for a proposal.
- [timothy] I changed our position to supportive.
- [devlin] We'd go with matches for permissions; topFrameMatches would only be a condition, not part of permissions.
- [timothy] I'd like to be consistent, top vs main.
- [rob] main_frame is only in webRequest/declarativeNetRequest.
- [carlos] We don't currently have top beyond web APIs.
- [rob] My preference is top. We extend the web platform and would prefer to stay with those conventions.
- [devlin] Also prefer top, but because we have had cases where there are multiple main frames: portals. They're gone, but they existed.
- [rob] Can you support origins vs. full paths?
- [tomislav] Why not full paths?
- [rob] Leaks information. And full URL may not be available in the process in cross-origin situations.
- [devlin] Origins make sense. Both process availability and matching on paths is weird. Trying to think of cases where you might want to match on top frame path.
- [rob] Maybe situations where a username is part of the path, but don't think we can support them.
- [devlin] Do we have a concern with matching by origins?
- [rob] Chrome already supports location.ancestorOrigins, which already offers the information. Firefox does not offer that API out of information leakage concerns (privacy).
- [devlin] Top frame condition where the top frame injects, content script knows the parent is the top frame, then we leak information about what that top frame is.

  If you have a content script that runs on foo.com, and you have the condition that you want to match on foo.com when it's on example.com, injecting the script on foo.com leaks information about the fact that you're on example.com. Didn't realize that ancestor origins already leaks this.
- [krzysztof] Is this a realistic concern?
- [rob] On the web leaking origins has privacy implications, with extensions one can do worse.
- [krzysztof] That's more meaningful on the web than in extensions.
- [devlin] Attack that might be more concerning is around intranet resources. An extension could extract information about the internal network topology. Acknowledge that domains are not super secret.
- [carlos] Stick with origins, should we use a different name?
- [timothy] Would prefer to stick to match patterns and document/enforce that the path is not used.
- [rob] Technically we could support paths with same-origin contexts.
- [devlin] Should stick with matches. An origin has a scheme, host and port.
- [carlos] Any browser willing to implement and sponsor?

- [timothy] I'm supportive, not a priority to implement.
- [rob] Implementation is probably easy, unit tests are the more time-consuming part.
- [carlos] If I create a proposal doc, would –
- [devlin] Don't think we're ready for a proposal as we don't have a sponsoring browser.
- [rob] What do we do with cases where we're willing to accept patches, but we don't have time to implement?
- [simeon] Use a label
- [tomislav] "patches welcome"
- [oleksii] Welcome by whom?
- [rob] That's indicated by the supportive label, right? There are a number of cases where we'd need this.
- [devlin] Would be useful to enumerate the sources where this would need to be supported. Should ignore paths in match patterns. Should document that this leaks origins and we're okay with it.
- [rob] (looking at Carlos) If you want to do the implementation I am willing to mentor.
- [timothy] Same.
- [carlos] ralph_im_in_danger.gif
- **Resolution:** Supportive of matching by origins (issue 763). Firefox/Safari willing to mentor external contributions and be the sponsoring browser in that case

2025 WECG Berlin F2F, Public Notes, Mar 28

Friday, the fourth day of the four-day face-to-face meeting in Berlin (announcement: #759).
Agenda: https://github.com/w3c/webextensions/wiki/2025-Berlin-F2F-Coordination

Attendees

- Oliver Dunk (Google Chrome)
- Devlin Cronin (Google Chrome)
- Simeon Vincent (Mozilla)
- Rob Wu (Mozilla)
- Tomislav Jovanovic (Mozilla)
- Carlos Jeurissen (Jeurissen Apps)
- Mukul Purohit (Microsoft Edge)
- Maxim Topciu (AdGuard)
- David Tota (AdGuard)
- Kiara Rose (Apple)
- Casey Garland (Capital One)
- Jordan Spivack (Capital One)
- Krzysztof Modras (Ghostery)
- Jan Biniok (Tampermonkey)
- Timothy Hatcher (Apple) – afternoon

Notes

Issue triage (Part 2)

Issue 713: Should tabs.Tab.lastAccessed be optional?

- [oliver] We implemented in Chrome; in MDN marked as optional; In Firefox set to tab creation, but there is a code comment stating that it may be set at startup in some cases. On MDN set to optional. Chrome always set to when tab is created.
- [oliver] I think that it should be when the tab is created, and not optional.
- [devlin] Most compelling reason to show nothing is that browser start is inaccurate. Current handling in Chrome is that lastAccessed is set to tab creation time.
- [krzysztof] Would the value be synced?
- [simeon] Would be nice, but don't expect it to be.
- [devlin] It won't be. Don't think we should conflate creation time with last accessed.
- [rob] Use cases?
- [simeon] My use case is to review my oldest tabs
- [rob] Based on the name I would expect it to match whenever the user has touched (activated) the tab for the last time, potentially from a previous browser session.
- [devlin] Looking at code, in Chrome we persist the lastAccessed value across sessions.
- [oliver] Sounds like we're aligned that matching expectations and having this be optional, and unset until the tab is first accessed, is the direction we want to head. This is as long as it is a safe enough change given current usage.
- [rob] I'd have to scan extensions for current API usage to determine whether it is a safe change to make, but from the API perspective it sounds sensible.
- **Action Item:** Rob to scan for usage of lastAccessed (issue 713)

Issue 717: Feature Request: [ContextMenus] Include <canvas>s in the "page" ContextType (or it's own ContextType)

- [rob] Is Chrome supportive to adding a "canvas" context type?
- [devlin] See the utility of not having to use "all". Hesitant about "canvas". Could this be covered by image? Do we put image context menus on SVGs?
- [rob] Not sure.
- [devlin] Gut feeling is that we should do "image"
- [krzysztof] What about cases where the page is drawn on one large canvas, e.g. Figma?
- [simeon] Extensions cannot meaningfully distinguish contents of a canvas – you'd have to parse the image data or get access to underlying (JS) renderer.
- [rob] If extensions want to do something with canvas, they'd also need host permissions. Knowing that there's a canvas element doesn't necessarily help the extension provide functionality.
- [simeon] If you invoke a context menu on an image, is the image data transferred to the background?
- [devlin] Image data is not, but the source URL is.

- [rob] Is the use case broad enough that we should add support for this?
- [devlin] Don't think we should have a canvas type. We currently have conceptual types. Arguably we could put it under "page" as a safe fallback if "image" isn't a good fit. It's ultimately strange that you currently have to use "all".
- [simeon] Agreed.
- [rob] Can you review Chromium source to see what would make the most sense?
- [devlin] Can do that async.
- [rob] What are your thoughts on CSS selector?
- [devlin] Can see utility, but that introduces new privacy and security considerations.
- [rob] Wouldn't have to watch. It's directly driven by user selection.
- [devlin] Right, but then you have selector counts, etc. Don't want to pursue that solution for this request.
- [rob] So not supportive, but could be convinced?
- [devlin] Neutral to negative.
- **Action Item:** Devlin to check why "all" has to be used instead of "page" for canvas in Chrome.

Issue 783: Conditional enabling/disabling of DNR rules.

- [maxim] We have generic rules; we'd like to disable some dynamic rules based on some conditions.
- [rob] Dynamic or static rule?
- [maxim] Currently dynamic, but could be useful for both. Can't fit all conditions in a single generic rule.
- [rob] Trying to understand usage potential. Say you have 1000 exceptions, that may work, 10000 may be a stretch. E.g. if extensions use 10000 very broad conditions, each with their own "except this condition", that is harder to optimize.
- [krzysztof] We have a use case for this.
- [maxim] I've only calculated refer rules, ~590. When users report issues with stealth feature is breaking their site, filter developers use this to disable the feature on a specific site.
- [krzysztof] At Ghostery we also have a need, and try to use "allow", it cannot always be used. We'd like to toggle a fragment of a ruleset (more than one, possibly 1000).
- [devlin] Overall seems fairly reasonable. Valid use case, can't be trivially done with the current API. Introducing a new action type here doesn't seem terribly complex or like API bloat. Not opposed.
- [rob] Let's table and revisit later today if there's time.
    - (this issue was revisited later today)

Issue 694: Unlike webRequestBlocking, DNR rules are not applied to redirected requests — should this behavior be reconsidered?

- [jan] Title may be a bit misleading; TamperMonkey doesn't have complete control over the network request. Now use DNR to modify headers, but this does not work after

redirects. I expect headers to be added post redirects since this is also specified in https://www.rfc-editor.org/rfc/rfc9110.html#name-redirection-3xx. Only automatically added headers are supposed to be removed, and I don't think that DNR-added rules should count here.

- [rob] I can see that viewpoint, I don't think that it should always be viewed as automatic.
- [jan] If a user script executes a long pull, that effectively block the extension for 30 seconds.
- [rob] Assumption is that the extension doesn't have enough granularity to match the specific request, so it has to match all requests. That isn't necessarily true. Create a new tab in the background, use that new context to execute the request. Not the only way to do it, but demonstrates that there are alternate solutions. Probably better to have an iframe.
- [jan] Doesn't the specification apply to DNR rules?
- [rob] DNR is a generic API; dangerous to automatically add it to all requests across redirects.
- []
- [rob] Potential way to address this is to add a rule to make header modification sticky.
- [jan] Works around the fact that the browser doesn't follow the spec.
- [rob] It's subjective whether this is "automatic". You aren't creating the request – maybe you are – but it could be coming from the page.
- [krzysztof] What about adding a custom header? I've seen this done elsewhere. Can be a footgun
- [rob] for clarity, I'm in favor of the use case, but trying to find an API that's a good fit for this and is harder to accidentally misuse. There's an existing issue tracking _ headers. If that were added, would you still want this functionality in dnr?
- [jan] Still think the current behavior is buggy, but that could be used.
- [devlin] It's an inconsistency between browsers. That should be addressed. I think it makes sense to preserve headers that would normally be preserved across redirects.
- [rob] They are if the web page sets them.
- [devlin] I think it makes sense to preserve them if they're set by a DNR header.
- [rob] If it set a cookie header, would you keep that?
- [devlin] If it would be preserved in a normal redirect scenario, yes, if not, no.
- [jan] Treat them like other headers.
- [rob] Are we aligning on if possible, try to follow the spec if it makes sense?
- [devlin] Yes. If there's a behavior change, we should do outreach. At a high level, aligning on web behavior makes sense.
- [rob] Still concerned that modifying a header for example.com, whether that carries over.
- [devlin] Only if it would normally apply to example.com. Basically the request would follow the normal flow of redirects. In cases where it would be stripped, it's stripped. If it would persist, it persists. What are the action items?
- [rob] We don't do anything special currently. Would like to avoid creating custom behavior in extensions for this.
- [devlin] I should behave the same way it would normally request. The extension said "I want to apply this header", it should apply.

- [rob] I'm comfortable doing this on the extension origin. I'm not comfortable with the general web.
- [devlin] Extension specific header handling seems weird.
- [rob] Seems like it's more about the intention of the extension developer.
- [jan] Behavior already exists for normal web requests against a web server. … Just a question of when the rule are applied in the request chain.
- [rob] Only fetch can set headers. If this applied to main frame requests, would you be okay with handling that differently?
- [devlin] Would want to see what the impact was. If we do outreach and see if developers are supportive.
- [rob] Neutral.
- [devlin] Do you agree this is something we should fix because it's an inconsistency.
- [rob] Agree fixing inconsistencies is good. Don't want to do things differently
- [devlin] As Jan said, it probably depends on what step in the request chain you're applying the rule. Seems we all agree something should change. Devs in the room who use this API all the time seem to think it's useful. Think we should follow up with developer outreach.
- [rob] I understand that this issue is driven by a desire for functionality that cannot be achieved in other ways right now. Concerned that the focus on this specific implementation approach can result in negative consequences. Supportive of the capability, not immediately convinced that the requested alignment is the best way to do this.
- [simeon] Break time. Devlin, please add a comment to the issue to capture the current status.

Issue 198 Action.default_area

- [carlos] Briefly discussed this yesterday; Chrome expressed a personal position with the caveat that this is ultimately decided by Chrome UX. Wondering about other browsers.
- [carlos] Is Edge favorable towards the ability for extensions to specify that the extension button should be pinned?
- [mukul] We generally try to follow Chromium's implementation.
- [krzysztof] This is a UI/UX question, and the question is whether individual browsers can ask their UX teams to allow to be pinned.
- [simeon] They can ask, but it's out of scope for this group to decide.
- [carlos] Would Edge be able to customize the logic in their fork of Chromium?
- [mukul] Would like to add it to Chromium code if we did.
- [devlin] If it wouldn't be exposed by default in Chrome, it wouldn't be in Chromium.
- [mukul] We could consider that, but it would be in Edge, not Chromium.
- [carlos] Why wouldn't Chromium allow something not used in Chrome?
- [devlin] We don't want code that we don't use.
- [rob] Non issue in Firefox, right? We support default_area.
- [carlos] I'll test to double check.

- [kiara] We pin icons by default. Only way for it to not be pinned is if the user customizes the toolbar to remove the extension.

Issue 12: request: allow to retrieve a frameID from an <iframe> element

- [oliver] I've periodically checked on this in the past. The conversation that it always comes back to is the desire to not expose frameID in the renderer process. Would be more willing if the information is less useful (powerful/dangerous).
- [rob] How about documentId?
- [devlin] Haven't had that discussion. In theory it's opaque, so there shouldn't be a concern. Not currently exposed in the renderer.
- [rob] Is it a must-have to keep it out of the renderer, or are you open to keep it out?
- [devlin] Don't expect so.
- [rob] Feedback from Giorgio based on yesterday's storage discussion is that its utility is limited without documentId. Would you be open to something like getDocumentId for the current frame?
- [devlin] I think so, yes.
- **Resolution:** Chrome, Firefox, Safari in favor of runtime.getDocumentId()

Issue 617: manifest key to enable automatic injection of content scripts after installation/update

- [carlos] In Safari and Firefox, content scripts are injected on installation; in Chrome they are not. Proposal to allow extension to specify whether to inject on install/update. In my extensions I inject on installation and have to do so manually by parsing content_scripts.
- [simeon] This is a common, but not frequent request. I'd ballpark that I see this every ~6 months.
- [devlin] Do you (Firefox) ignore run_at…?
- [rob] run_at is the earliest point of injection, not the latest.
- [devlin] Concern is that ignoring run_at causes problems for scripts injected late.
- [rob] If there was an option to control this, developers would have to consciously opt in.
- [devlin] No concerns from a performance point of view. Some concerns around prioritization and cost. Seems non-trivial for us to implement. Understand that it's a pain to work around, but it's nonetheless possible. Ideal solutions are not uniform. Some extensions will want to inject late, some will want to reload the page, maybe others. Not saying it's infeasible to implement, just not sure it's the right issue to prioritize. Suspect that this will not eliminate special handling. Still need to add custom logic to handle cases where you're injected late.
- [jan] Expect that document_start is the document's start.
- [simeon] But it's not always, as just discussed.
- [krzysztof] Surprised – wonder how other extensions handle this. If you have to shim APIs, the only thing you can do is run before the page.
- [rob] Haven't had issues filed against this.
- [krzysztof] They get filed against the extensions.

- [rob] Devlin - are patches welcome or are you opposed?
- [devlin] Currently mildly opposed. Violates guarantees provided by the API, confuses the injection model.
- [krzysztof] Would you be open to a cross-browser approach to this request?
- [devlin] Not against it, but would rather explore other solutions.
- [krzysztof] Alternative is to have 2 APIs to handle document_start injection.
- [rob] How would that solve?
- [krzysztof] Gives a way to recover.
- [rob] Content script can set a flag at injection time.
- [devlin] Happy to continue discussing in the future.
- [rob] are you open to having some way to signal intent?
- [devlin] Open to discussing, e.g. a way to signal that a script can run soon.
- [rob] And if that is used, would you run after installation.
- [jan] document_idle scripts are already prepared to run late
- [devlin] Yes, in those cases they can be dynamically injected in response to a browser action click.
- [rob] Scripting APIs have injectImmediately. Would you be open to that in content script declarations?
- [devlin] Too much bikeshedding. Don't want both injectImmediately and run_at. Combining those concepts is contradictory. Would want a flag that indicates its okay to do this.

PR 793: Add proposal: Synchronous data at startup

- [rob] Devlin, thoughts?
- [devlin] Don't like the names.
- [rob] Names may change, we can bikeshed on that later.
- [devlin] We'd like to do something in Shared memory for performance reasons, not excited about that. In theory it helps with some cases desired by user/content scripts, but the lack of separation by site is going to be a major limitation.
- [rob] Open to `matches` or something. Extensions may have their own idea for what should match (e.g. etld+1), so this may not always work.
- [devlin] Don't know how we'd handle that from an implementation point of view.
- [rob] Could offer extension authors a map primitive to store the data, and extensions can use keys composed of the key name plus site/document origin/etc.
- [devlin] 2 limitations. Size limit become much more challenging for devs. Currently proposal doesn't have site specific data. Not having that concept, expect that devs will hit this limit much more often. Without it, there's no a security guarantee for limiting data exposed to a renderer.
- [rob] Because the data has to be synchronously available, there should be a global limit.
- [oliver] Hesitant about the fact that we have many use cases that are site specific and this proposal requires workarounds to address that.
- [tomislav] Also tab specific, but that's out of scope of what we've covered so far.
- [rob] Could we mark those issues as future work?

- [devlin] Don't know that this is the right API to solve that problem. It can help with it and solve other problems, but don't know that this solution would apply to site specific etc. data in the future.
- [rob] I'm okay with not covering those case in the future with this API.
-
- [rob] Different case, … any concern about the chattiness of this? Current onChanged is very chatty.
- [devlin] Not as much of a concern in the larger scheme of things. Storage API has never been performant. We guide devs away from it for perf sensitive cases.
- [rob] For this proposal, since it's a new API I'd want to remove the perf issues in the implementation. If the devs really need the data, they can already fetch the value for a key. If they want the old data they could fetch it from the script.
- [devlin] Don't like having those kinds of inconsistencies across storage areas. The closer it can emulate the existing areas the better. Don't think onChanged would be a major perf hit.
- [rob] Perf may be fine, but it's currently not possible to listen for specific keys. A listener would have to subscribe to all events.
- [devlin] Storage API is bad. We can improve it. What's the goal here?
- [rob] Besides reading synchronously, extension authors may be interested in subscribing to changes. I tried designing the API with minimal requirements. Including oldValue / newValue in the onChanged event would require the values to be read from shared memory, even if the specific content script does not care about it.
- [devlin] Agree that's an issue, but would prefer to align as closely as possible to the existing surface.
- [rob] That means we'd have to dispatch old and new value.
- [devlin] Seems orthogonal. Don't think we should have big differences if we call this a StorageArea. But those are issues we could address in the future.
- [rob] I'll update the proposal to mention that oldValue and newValue are included as usual. This may have more overhead at runtime, but is at least consistent with other StorageArea.
- [rob] What are quota sizes that would be acceptable in Chrome.
- [devlin] Between 1 and 10 MB. Assuming its in shared memory, its not that costly.
- [rob] Assumed a lower starting value (100kb) in the proposal, willing to consider 1 - 10 MB if you're fine with it too.
- [devlin] This isn't just content scripts, also in background. Same limits in content and background?
- [tomislav] If the areas are different, then they can be different.
- [rob] We can resolve the remaining issues async.
- **Action Item:** Rob to update PR 793 based on feedback, to more closely match StorageArea semantics.

Issue 719: Adding a temporary listener just for the current run of the service worker or event page so it doesn't wake up next time for this event

- [simeon] Short version; toph is looking for a way to register a listener that does not trigger startup of a background service worker.
- [rob] (…) Right now it is impossible for a browser to tell what an extension wanted. If extensions want to control what wakes them, we could have a manifest key that takes events that would to prewarm the background, and stop trying to auto-detect.
- [timothy] Interesting, I'd be interested in an idea like that.
- [krzysztof] We did something like this in Safari. We had synchronously registered event listeners in the background to ensure that we'd start up. This hack is no longer necessary.
- [maxim] Wouldn't it be better to have an option on the listener?
- [rob] That's an option. By putting it somewhere statically, you give a strong signal that doesn't have to worry about timing.
- [krzysztof] … Based on feature flags we could make a choice on whether or not to enable the listener.
- [oliver] You mean declarative, not just in the manifest, but having both a manifest entry and an API for it.
- [devlin] Generally supportive of doing things to help the browser guess less.
- [rob] There was a proposal before about changing addListener completely.
- [devlin] I think the manifest key is the direction we were leaning before. Being able to know beforehand from the extension is very useful. In the past there have also been discussions about having a queue of events that could be drained after you asynchronously register a listener.
- [krzysztof] May be too late to process those events.
- [devlin] Right, if you need a synchronous listener.
- [krzysztof] Devs can also manually create their own queues
- [rob] Manifest entry would help, but having a way to register a non-waking listener would still be useful. Are there any APIs that take a non-object as a listener's second parameter?
- [devlin] Not that I know of. You could potentially change filters to configuration options.
- [oliver] Just looked through Chrome APIs and couldn't find anything where we take anything but an object as the second argument of a listener.
- [krzysztof] MV3 design and docs state that only listeners assigned synchronously on first parse in order to receive the listener.
- [devlin] One of the issues with async registration is that if it's async but fast, it will still work. Not saying that's a good thing, but it will work most of the time if you're fast enough.
- [timothy] Safari is permissive here. We wait half a second or so after the page load event (or SW equivalent) for async registrations.
- [krzysztof] Would argue that's harmful as it gives a false impression of reliability.
- [oliver] Would be hard to define the right conditions. Maybe we could show a warning saying that we've detected an unreliable pattern.

- [rob] To conclude this issue, sounds like we're aligned on using the second parameter as an object with a property to specify whether or not to persist across restarts.
- [timothy] I'd be supportive of that.
- [rob] And from the discussion before, both Firefox and Chrome are also supportive of this.
    - Updated labels and added comment: https://github.com/w3c/webextensions/issues/719#issuecomment-2761477404
- **Resolution:** Accept second parameter in addListener to take options to control persistent listeners (issue 719)

Issue 783: Conditional skipping of DNR rules.

- (continuing from the morning where we briefly touched on this issue)
- [rob] Added feedback to use "skipping" instead of "disable". Updated the issue title to reflect.
- [rob] Are we aligned at a high level?
- [devlin] I think so, yes. I think we're close enough for someone to create a proposal, assuming browser interest in implementation.
- [timothy] We don't currently support disabling rules.
- [rob] Not disabling. This is skipping rules during the current evaluation.
- [timothy] Only issue is we don't currently preserve IDs after translating rules into the internal representation.
- [rob] How does it work with header rules? Multiple rules can stack.
- [timothy] We don't need IDs for that. Anyway, I'll add the supportive Safari label.
- **Resolution:** Chrome, Firefox, Safari in favor of supporting a way to conditionally skip DNR rules.

Issue 77: Proposal: Provide a secure drop-in replacement for window.postMessage()

- Objective: is there support to send an extension message directly to a specific frame element?
- [carlos] This was proposed by a security researcher to address a security issue where it is difficult to send a message to a specific frame.
- [devlin] This is saying, from a content script context, to sending a message to a content script in a sub-frame? And it doesn't need to be synchronous?
- [carlos] Correct.
- [devlin] If it's async, it's possible today by proxying through the background context.
- [carlos] Not in all cases. Popups don't have tabs.
- [rob] And to forward a message, a frameId is needed. Not all browsers support the `runtime.getFrameId()` method.
- [devlin] If this were going to be asynchronous, just tweaking the APIs to let you proxy through a background context would be ideal.
- [carlos] How would you link everything?
- [devlin] Taking the case of a tab, parent frame takes runtime.sendMessage and posts a message to the background, background relays it to a frame.

- [rob] If extension developers are content with using postMessage without the response behavior, could we have an extension API like `window.postMessage` that offers one-way messaging mechanism with structured cloning support to send a message from content script to a content script in a specific frame?
- [devlin] A bit worried about practicality. Similar to the storage API, extension messaging isn't great. It's slow, it broadcasts, etc. If we try to solve everything, we're not going to do it.
- [rob] Could we ask the web platform if they'd be willing to accept a targeting parameter, an extension could then use that to send a targeted message
- [devlin] We've been working with a Chrome person (Dominic) on WHATWG to explore appetite for integration of extension concepts into web platform APIs. This could be something we could explore after the initial proposals.
- [devlin] Personally supportive of leveraging web messaging. Not certain whether it is something we could get to in the next 12 months. Opposed to `runtime.sendFrameMessage()` as a method. If we use extension messaging to solve this (in the nearer term, since leveraging web messaging is further out), would prefer to fix any gaps (e.g. getFrameId()) to enable it in existing methods rather than introducing a new one.

Open discussion

- [simeon] Opportunity to ask questions that don't make sense as questions.
- [krzysztof] documentId should be added to all webRequest implementations.
    - [timothy] Have supported this in Safari Tech Preview for a few releases. Will be in the next release.
    - [krzysztof] Okay, that covers what I was after.
- [krzysztof] Storing UInt8Array in storage.local. Firefox already support this. Safari supports this too (editorial note: this is wrong, see below). Chrome should ideally support this.
    - [devlin] I'd like to support more values in storage.
    - [timothy] Surprised it works in Safari, but cool. We store them internally as JSON.
    - [rob] Did you get a true Uint8Array back, or a regular array of integers.
    - [krzysztof] Upon double-checking, it is an object with indexed map of bytes, e.g. `{0: 123, 1: 123, 3: 123}`.
    - [timothy] That's something we've wanted to do. Don't know how feasible it is in the short term.
    - [tomislav] Now that we discovered this serialization issue, do you use some form of structured cloning?
    - [timothy] We use JSON. Surprised it's doing something different.
- [krzysztof] Firefox supports native Map and Set in storage.local
    - [oliver] We were just discussing moving to structured clone to address this case.
    - [rob] That is why it works in Firefox - we use structured cloning.
    - [tomislav] structuredClone is a well established tool for this kind of use case. Would prefer to adopt that here.

    - [oliver] If we can identify enough types that need to be stored differently, it would make sense to just move to structured clone.
    - [timothy] I think that structured clone makes more sense than just solving it once.
- [krzysztof] Observed that our complicated UIs take a while to load. We don't bundle JS, we have well structured individual files. Would browser vendors be open to some kind of cache warmup? We use JavaScript modules.
    - [rob] Sounds like something browsers could/should do without API changes. Makes sense to do this for background scripts, especially.
    - [krzysztof] Hack solution for an extension is to load these files in the background (e.g. a background tab) to warm the files before injection.
    - [timothy] This is what we do in Safari already; frequently used code is compiled and cached.
    - [krzysztof] This happens when code runs once. Could we warm up the cache?
    - [timothy] Probably something we could do. Default behavior has been to let it happen when the user interacts with the resource.
    - [oliver] Is there a reason why this is more of an issue here than on the web?
    - [krzysztof] Large production websites don't use ESM directly. They bundle scripts.
    - [simeon] Chrome uses a completely different loading path for extensions resources.
    - [rob] If perceived cold startup is a concern, could you add a small script or style that shows progress.
    - [devlin] In Chrome we don't cache extension resources. Artifact of current implementation. Something we'd be open to. I think the extension service worker is cached, but nothing else.
    - [krzysztof] Wasn't sure if it was worth having an issue.
    - [devlin] We've had a few threads about this in the extension mailing list.
    - [oliver] Yeah, but most have been higher level requests. For example, we've had requests to freeze the background service worker instead of terminating.
    - [krzysztof] Would recommend against that. Violates expectations of web script behaviors. Devs don't expect that their context will live forever.
    - [rob] Going back, Firefox has been working on sharing compiled code across pages.
    - [krzysztof] I haven't noticed performance issues in Firefox. Specifically observed in Safari in Chrome.
    - [tomislav] We also preload the action panel content in Firefox when it seems likely for the user to open the panel, when the user hovers over the extension action button.
- [krzysztof] Can we create new surfaces to communicate with the user? Say a space on the new tab page. Looking for more predictable and dismissable outlets.
    - [timothy] This is an area we'd be interested in and would like to see proposals for.
    - [krzysztof] New tab page is also of interest, but we haven't explored it because we didn't know what kind of interest there would be here.
    - [timothy] This is one of the surfaces we're interested in exploring.

    - [krzysztof] Want to let new users know there's a new feature or resurface a feature a user may have missed. Would be open to a single surface that rotates between extensions. Don't want to be intrusive. Would prefer a place where users are expecting this kind of message.
    - [timothy] We were more interested in it being a user-customizable feature where a user can choose to surface a component from a given extension.
    - [krzysztof] Like some kind of extension widget. Got it. Maybe an iframe.
    - [tomislav] Maybe not iframe; new tab page are heavily optimized.
    - [oliver] Not something we're specifically looking to implement, but would be happy to look at and consider a proposal.
    - [mukul] Doesn't have to be a single thing implemented by every browser
    - [oliver] Would be nice to have a common surface with established expectations.
    - [timothy] Whatever the proposal is; it would need to be very flexible for browser interpretation.
    - [krzysztof] having a common denominator would be useful. Ultimately something is better than nothing.
- [krzysztof] Does anyone remember "Firefox Collusion"? It was a project to showcase which scripts load other scripts, and other scripts. At the Ad Filtering Dev Summit, there are often presentations that show this kind of concept. Tracking requests and stack traces and showing it somehow. We'd like to have this functionality to effectively block more trackers. A script loaded by a tracker is a tracker as well.
    - [timothy] We have something similar in WebKit. Don't know if it's the same shape as Firefox.
    - [simeon] Do you mean like a HAR file for a page load?
    - [timothy] We have our own resource load tracking data for our tracking prevention feature. If it's useful for extensions, we could consider exposing that data. https://webkit.org/tracking-prevention/
    - [krzysztof] Devlin, in a previous conversation you mentioned something about "resource tainting"?
    - [devlin] We don't have that information.
    - [krzysztof] Not the script that initiated the request? Devtools shows it.
    - [devlin] We have initiator.
    - [rob] That is the document's origin, not the script that had the code that triggered the request.
    - [krzysztof] If performance is a concern, could do this on specific page loads rather than all page loads.
    - [tomislav] We are unlikely to bring back Collusion, but we have a database similar to what Timothy mentioned that tracks resource loading paths. Might be possible to expose that, but will need to look into it.
    - [krzysztof] Are all parties open to exploring?
    - [timothy] Yes, but no guarantees
    - [tomislav] Same.
    - [devlin] We don't surface anything to the extension today, so would want to know more about what you're interested in accessing.

    - [krzysztof] If I can determine that A triggers B, or B is triggered by A, then we can build that map.
    - [timothy] We at least have the data of what triggers what.
    - [tomislav] We have categories of requests, like trackers, social, etc.
    - [krzysztof] but that's metadata classifications applied
    - [tomislav] Yes, but the point is there's some precedent to exposing more data.
- [david] The MV3 service worker's dev tools are unable to open IndexedDB, whereas the MV2 version was able to do so. Actually, we have an inversion: MV2 devtools did not allow opening storage.local but allowed opening IndexedDB. MV3 devtools allow opening storage.local but do not allow opening IndexedDB. Note: devtools for frontend shows both, but devtools for service worker only shows storage.local.
    - [devlin] Sounds like a bug.
    - [oliver] Storage one surprises me. I tried to make that show in as many places as possible.
    - [mukul] Should someone open a bug?
    - [oliver] Yes please.
- [david] Chrome-specific: TextDecoder / TextEncoder in Chrome are relatively slow; Faster in WebKit and Firefox. Turns out to be related to small strings. After some threshold the native TextDecoder/TextEncoder is faster in Chrome, but before that it is much slower.
    - [devlin] Sounds like a bug. I don't know about its implementation is to file a bug.
    - [timothy] Under the hood WebKit uses ICU.
    - [oliver] Having a bug with sample data and code would enable us to investigate further.
    - [david] Ran into this issue last year. Also tried before this meeting and wasn't able to reproduce it. With some experimentation, found that with a deep enough call stack and memory used, it could be 4x slower in service workers. Was this known and fixed?
    - [devlin] Don't know. There have been performance improvements around SWs, that may or may not be related.
- [david] Chrome-specific: Profiling. In MV2 it was easy to run the profiler for the background page. When I open the devtools for the extension service worker, it throws an error.
    - [simeon] Sounds like a question for the devtools team. Speculate that it may be because there is no page.
    - [oliver] Please file a bug.
- [kiara] `browser` namespace?
    - [devlin] We prototyped it, and tried running, but noticed that extensions relying on webextension-polyfill broke.
        - https://github.com/mozilla/webextension-polyfill/issues/721
    - [rob] The webextension-polyfill introduces Promise support without changing API behavior, except for runtime.onMessage to support Promise return values, because it is heavily relied upon by extensions.

    - [devlin] We're supportive of having a promise version of onMessage. There's a dependency on us implementing that. Once we do, we can turn on browser because the polyfill will be unnecessary.
    - [oliver] Currently enabled behind a flag.
    - [timothy] Do you know how you're going to fix it beyond modifying the polyfill?
    - [devlin] On our side once we add promise support to onMessage, the polyfill will be a noop.
    - [timothy] Would also be ideal if the polyfill can be updated to make it a noop. We made a change in Safari to make it a noop. Forget exactly what the change was.
    - [oliver]
    - [rob] I'd be open to a minor release if it helps.
    - [timothy] The change I made: https://github.com/mozilla/webextension-polyfill/commit/871b49d98968e940011ebf4506028befe9859b0e
    - [oliver] Most of the Chrome Web Store uses webextension-polyfill 0.10.0.
    - [rob] 0.10.0 was published 3 years ago (2022).

Long Term Direction

- [simeon] Strategic discussion on where the platform is headed. Topics include event targets, API design principles, declarative vs. imperative trade-offs, and how to align on a forward-looking vision for extensions architecture.

Events and live objects

- [simeon] Fundamentally different event model between extensions and the web. That we do not have event targets, for example.
- [timothy] I'm interested in getting to a more web-based extension model. We had it before. Clear hierarchy of event targets – browser, window, tab, page. Interested in exploring ways to make our event APIs more hierarchical and event-based.
- [rob] Are you specifically interested in the event bubbling aspect of this?
- [timothy] I've been thinking about something where at the lowest level there's a resource type that maps to request events, bubbles up to a live object with a documentId, that has a frame object, that bubbles up to tab, window, global browser object. Fundamentally a different paradigm than what we have today. Organically lends itself to live objects, which we've discussed before, where you could do something like `browser.currentWindow.activeTab.frames[1]`.
- [simeon] I may have misrepresented your view: I thought you were hesitant about live objects.
- [timothy] That was Chrome. I am in favor of live objects.
- [devlin] I am hesitant about live objects. If events are bubbling through events and targets, what does that look like when you've got an event that's propagating and the background context gets torn down?
- [timothy] You have to retrieve it again; but because events bubble you can listen for it at the top level.

- [devlin] That eliminates some of the benefits, right? Figuring out how this works with ephemeral contexts is one of the things we'd need to think about and work through.
- [rob] Question about the previous example. (`browser.currentWindow.activeTab.frames[1]`) This information isn't synchronously available in the process.
- [timothy] Basically listens for everything and keeps live objects alive. Tries to keep everything up to date.
- [tomislav] If we promise an API like this, that would mean that we would have to propagate all potential changes down even if extensions do not need this. If at any point you can do `browser.currentWindow.activeTab`, you haven't captured that object. I don't know a non-naive way to approach this.
- [simeon] I can imagine something where you have to use an accessor in an outer context, but that doesn't align with the kind of fluent API expressed in the example being discussed.
- [alexei] I am wondering, what inspires this future idea?
- [timothy] It is more a desire of matching closer how the web works. Not driven by what extensions do today.
- [rob] Noting that an aspect of live objects is that they are mutable, whereas current extension event parameters are immutable.
- [timothy] How is this different from service workers?
- [devlin] Service workers do not listen on live objects; they listen on `navigator.serviceWorker` and do not have state associated with them.

Named globals

- [simeon] Rather than one global, importing specific namespaces.
- [devlin] `imports {tabs} from 'browser';`?
- [simeon] Yes.
- [tomislav] We already lazily expose namespaces.
- [krzysztof] Can easily be polyfilled.
- [carlos] potential reduced API surface for abuse.
- [tomislav] and forbid dynamic import
- [carlos] Yes.
- [rob] As an extension dev, I'd rather use a browser global.
- [timothy] I tend to agree. We also populate namespaces on first access.

browser.addEventListener

- [tomislav] Going back to the idea of `addEventListener` on the existing browser namespace, that seems like a pattern that could be adopted.
- [timothy] Could easily introduce EventTargets for all the browser namespaces. That avoids the potential problems discussed with the more detailed set of live objects.
- [rob] Would we have an event object with event.details where that property exposes the current value? Or would we want to merge the event properties with the current Event object?

- [timothy] We'd have to have an `Event` object. Could be a custom event with properties that are exposed directly on the object, not just embedded in the details property.
- [rob] If there are no conflicts, that would be nice. Seems possible. How would we handle filters
- [tomislav] `addEventListener` has an `options` object. Could be added to that.
- [rob] And if we provided chatty events, we could expose filters where developers could narrow down their interest.
- [simeon] The lack of filters in current extension APIs is one of the things that's bothered me about our current systems.
- [tomislav] Architecturally, filtering is less important in DOM because everything is happening in the same process.

API design principles

- [simeon] As I think about APIs and potential approaches. Are there ways to forward-declare to allow the browser to guess less often to get the desired outcome? Imperative APIs are more flexible (e.g. webRequest).
- [devlin] Decided on a case-by-case basis.
- [simeon] Are there principles we could gather into some "design principals" resource doc we'd have. What would we want to include in such a doc?
- [devlin] General shape of APIs, thinking of future compatibility, objects instead of single params (to allow future additions), success/failure indication, async methods should return Promise and not promise. Alignment on language (`onclick` vs onClicked).
- [timothy] Naming consistency. Consistent use of case. Style guide on dos/donts for APIs.
- [devlin] We have largely been aligning on new API namespaces should always require a permission, even if there is no visible warning.
- [oliver] May be too much detail, but some conventions around verbiage in a specification.
- [simeon] For clarity, I was thinking about including this in a design_principals.md in the repo.
- [oliver] Some of this could also be included in the spec.
- [rob] Consistency would be my biggest item to focus on. We regularly say "we'll do better" and then do something unique again.
- [timothy] Design principals are that guidance.
- [timothy] With so many (new) contributors it is imperative to be consistent.
- [carlos] Improving the contract between extension developers and the platform. Allowing extensions to signal preferences, that browsers can make decisions independently of other browsers. Allow extension developers to express their desires across that diverse set of implementations.
- [rob] Can be summarized as "offering information so that the browser has to guess less"
- [carlos] Intentions from developers
- [tomislav] Preferences and intentions from developers rather than requests / demand. Give browsers more leeway to decide how to handle that. "I would like access to google.com" vs. "Give me access to google.com this moment."
- [timothy] I like that framing.

Wrap-up discussions

- [timothy] Final session to review action items, check for consensus on key decisions, and reflect on outcomes from the week. We'll also discuss post-meeting follow-ups, including minutes, and timeline for the next steps.
- [timothy] 4 better than 3?
- [devlin] Yes.
- [tomislav] Yes.
- [timothy] Thought that we would have issues with filling the schedule, but it filled up pretty well.
- [tomislav] More relaxed schedule, started relatively later.
- [simeon] Relax schedule also offers room to schedule breakouts later, follow up on topics after some offline work.
- [simeon] C.
- [tomislav] Dev feedback
- [carlos] More relax, fluent and productive. I like Simeon's 5-minute time checks.
- [oliver] Think the approach to breaks were nice.
- [carlos] Clear objectives for issues was helpful.
- [rob] In some discussion I tried to cut the conversation short to make sure we were interested in solving the problem before proceeding, and clarifying action items. Any concerns with that approach?
- [simeon] No concerns on my end; moved on before spending time on things we didn't need.
- [carlos] Would be helpful to have more detail beforehand. Many sections on the wiki had "TBD".
- [timothy] I updated the TBDs once I realized they needed to be filled out.
- [rob] Half way through the week I realized our main github issue did not link to the wiki page that we created later. I've added it, but it may be too late for people who do not regularly participate in our biweekly meetings.

Next meetup and goals

- [timothy] Is Google hosting next year?
- [oliver] I'm supportive.
- [tomislav] It might be good to spread locations. Maybe host in Canada?
- [timothy] Slightly later in the Spring may be better.
- [devlin] Also consider when TPAC is; this year it is later, so host our next meeting later so that sufficient time has passed since TPAC.
- [timothy] Next year might be later as TPAC will also be later next year.
- [mukul] Should also keep in mind other conferences happening around the same time. Maybe conflicts with I/O, WWDC, etc.
- [timothy] April may be the sweet spot.
- [devlin] Don't like doing it the last week of a quarter.

- [devlin] Should we talk about what we want to have done by TPAC? (simeon:yes) We should write more spec text? How much do we need to have done?
- [simeon] I can follow up after I sync with my standards folks.
- [oliver] Would like to have submitted something and have started the process by TPAC
- [devlin] Agreed
- [simeon] +1
- [rob] On testing, the ability to run extension tests in WPT, so that tests can be shared and run. WPT bots takes care of synchronizing tests across repositories, having the ability to run them (even if manually) would be useful.
- [devlin] Good call.
- [krzysztof] I'd like to get script evaluation tests created.
- [tomislav] We're actively working on this with Chrome. We have a meeting scheduled with the Browser Testing and Tools group for their next session. Kiara and I will join. Would like to have tests running across multiple browsers with a design that's aligned across all browsers by TPAC.
- [krzysztof] Can we align on getting DNR in tests?
- [timothy] Supportive of getting some DNR tests built out.
- [devlin] Sounds reasonable.
- [tomislav] I know WPT has a dozen different domains that can be used for testing.
- [krzysztof] Ad blocker community should be able to put together a decent set of tests.

- [simeon] We can talk a bit more about building out our spec. What areas are we interested in building out?
- [timothy] I'm planning on working on the background manifest property
- [devlin] Specifying the behavior of the extensions origin, permissions field.

- [rob] Planning to review the meeting notes and extract action items. How would you like to be notified of action items?
- [tomislav] Tag us in the doc.
- [rob] Nice part of our approach to notes this week is that we're already ready for publication.