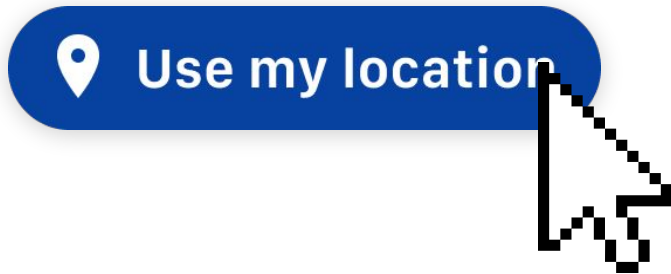
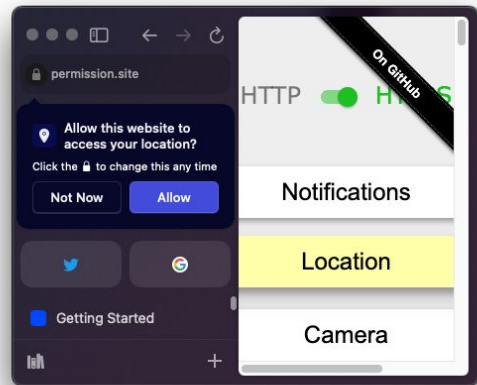
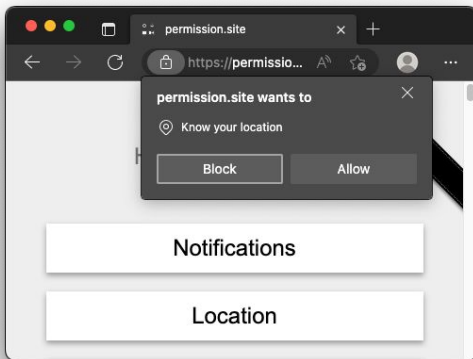
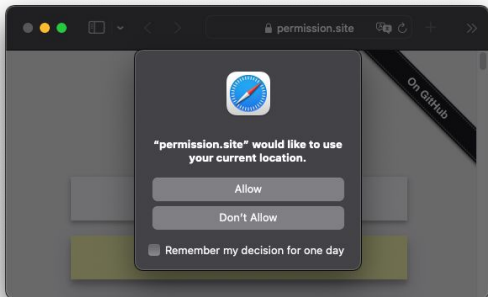
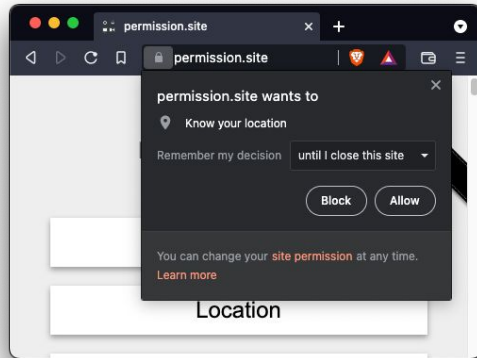
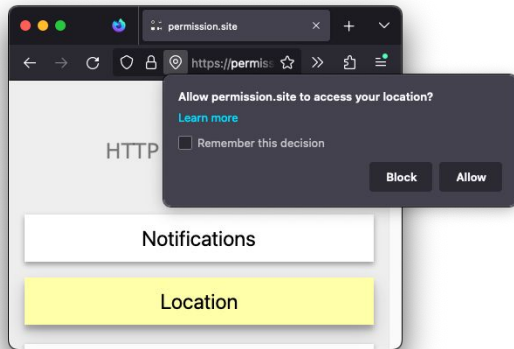
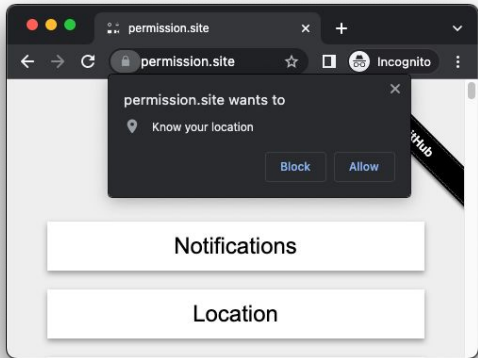


PEPC *Page Embedded Permission Control*



Safely embedding permission entry points
in web content

Permissions are for
verifying user *intent*

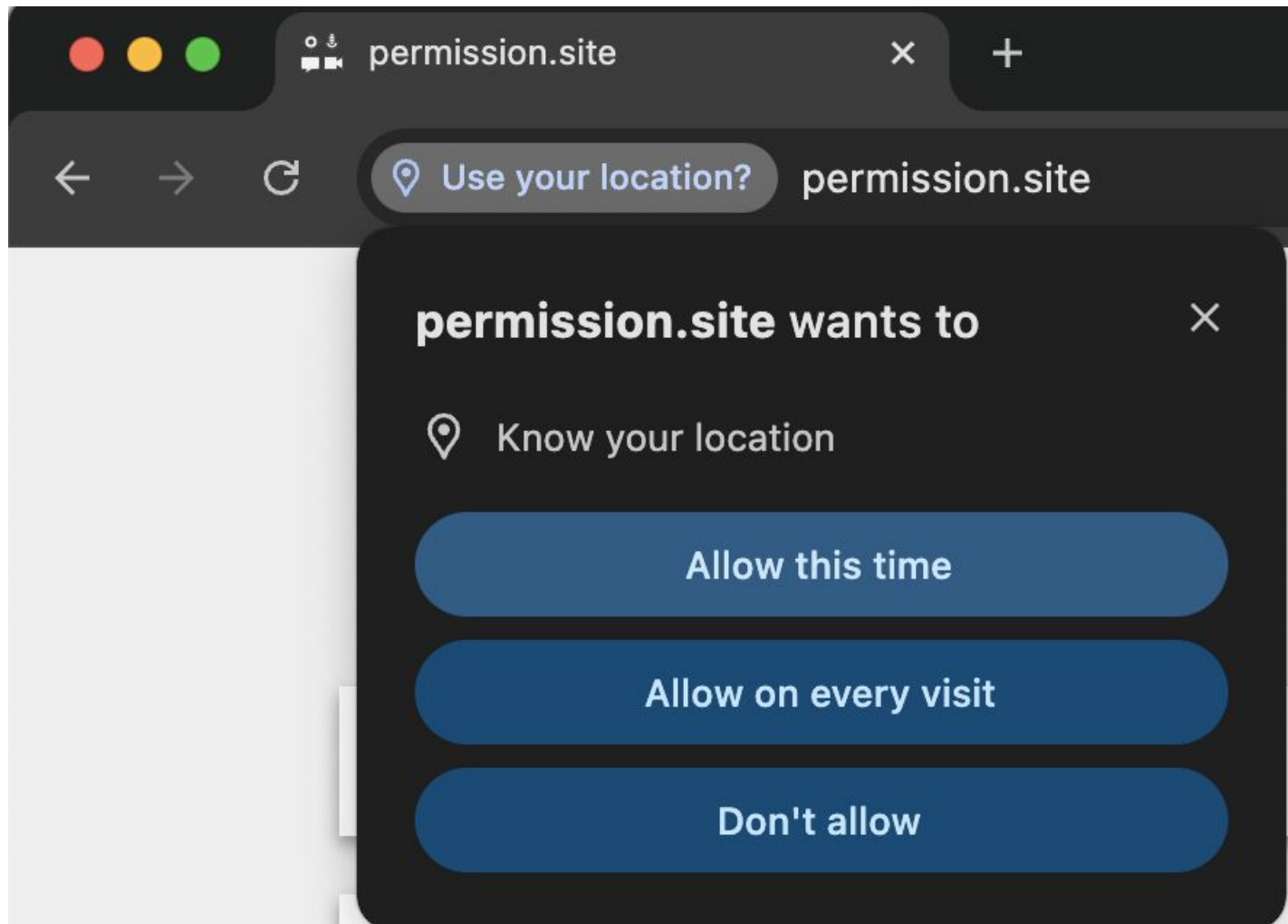


User problems

- * Developer-push model → Lack of user and *browser* context
 - Browser cannot differentiate user initiated and developer gamed
- * Interruption

User problems

- * Developer-push model → Lack of context
- * Interruption



permission.site wants to



Know your location

Allow this time

Allow on every visit

Don't allow

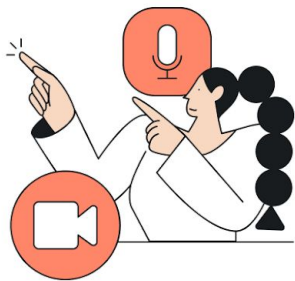
meet.google.com wants to

Use your microphone

Use your camera

Block

Allow



Click **Allow**

You can turn off your microphone and camera anytime you want

Display Audio

Display Audio

FaceTime HD ...

Planning



sica, Ethan and Lani are in this call

Join now

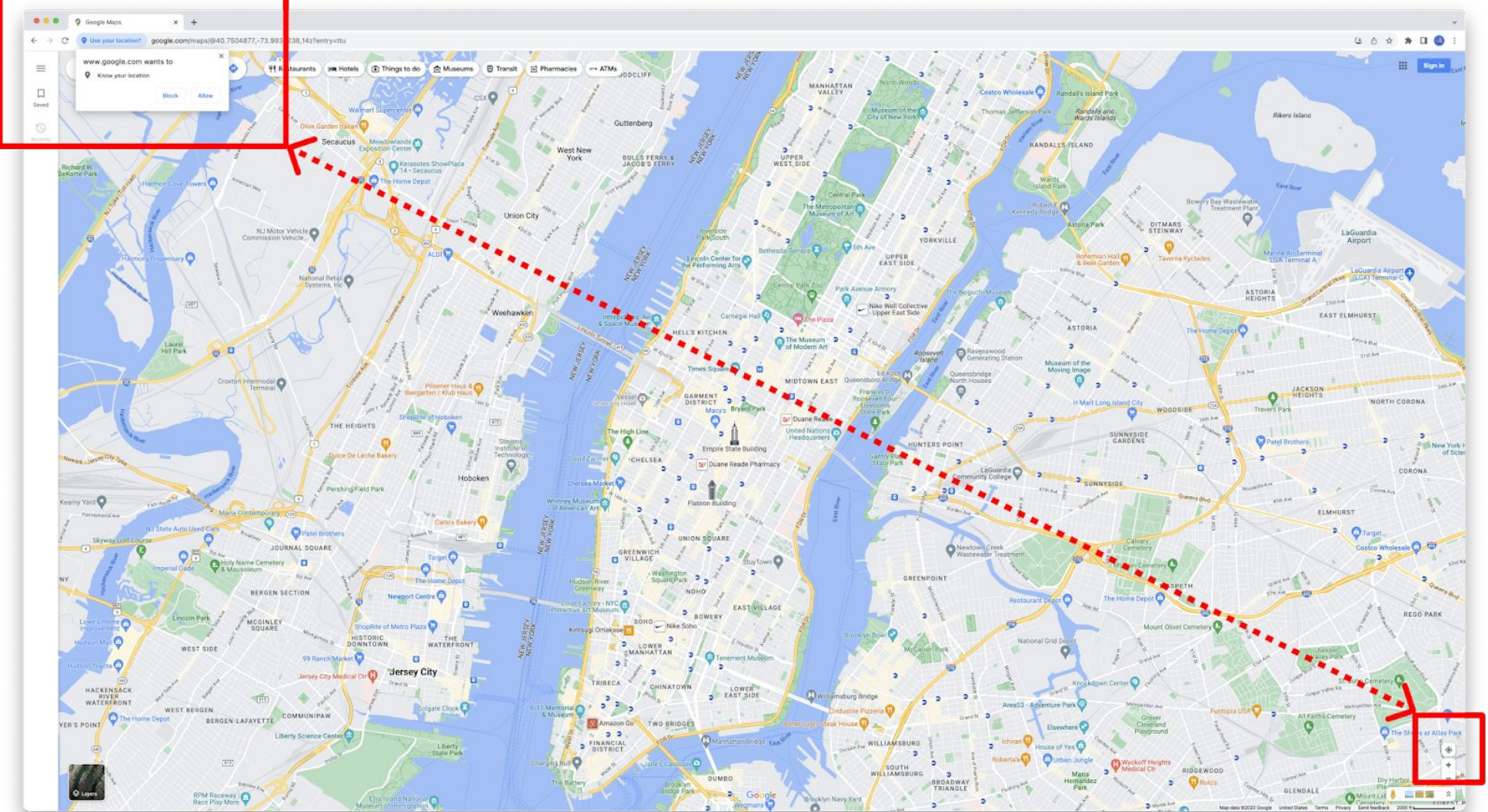
Present

Other joining options

Meeting link

User problems

- * Developer-push model → Lack of context
- * Interruption
- * Visual discontinuity



www.google.com wants to know your location

Block Allow



User problems

- * Developer-push model → Lack of context
- * Interruption
- * Visual discontinuity
- * Annoyance & prompt blindness

User problems

- * Developer-push model → Lack of context
- * Interruption
- * Visual discontinuity
- * Annoyance & prompt blindness
- * Difficult to revert past decisions

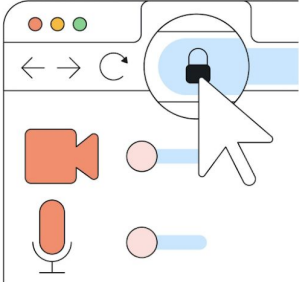
New Tab x New Tab x New Tab x New Tab x +

mail.google.com/mail/u/0/#inbox


Meet

ericahill@ink-42.com
[Switch account](#)

×



Allow Meet to use your microphone and camera

1. Click the  lock icon in your browser's address bar
2. Turn on microphone and camera

an and Lani are in this call

[Present](#)

Joining options

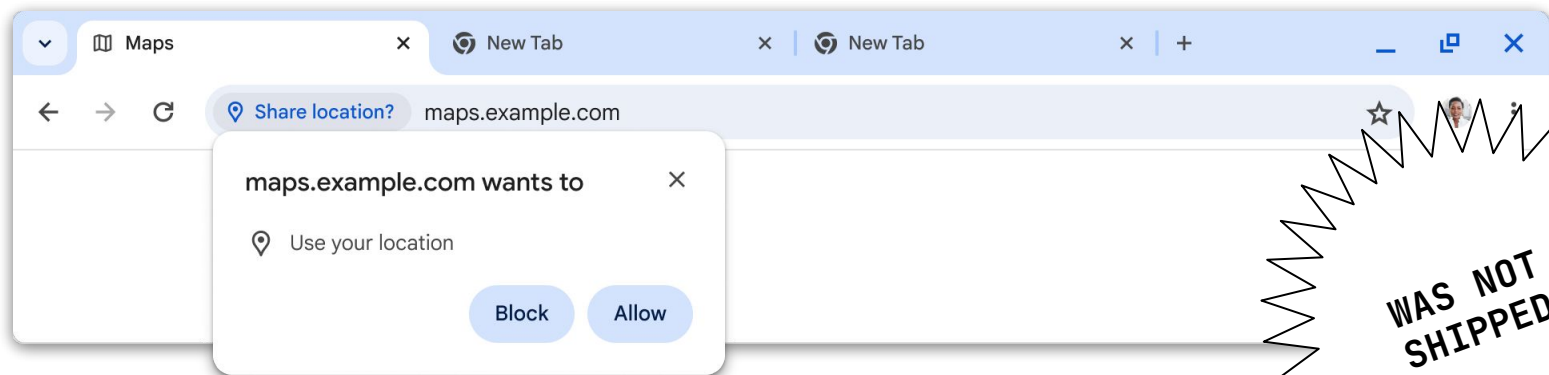
Meeting link

Display Audio Display Audio FaceTime HD ...

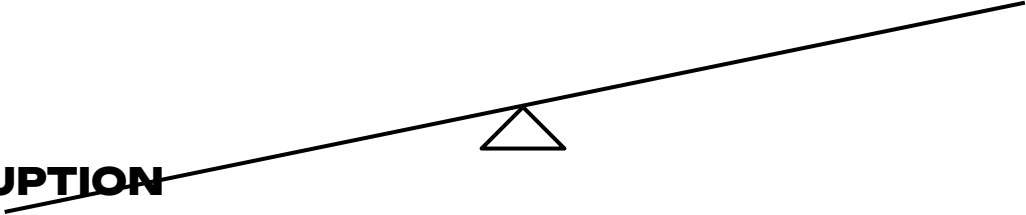
Things we tried...

What if we made prompts less interruptive?

- * We tried this with the Chrome permission request chip
 - Move permission requests out of the way into the omnibox
 - ... **Grant rates dropped from 20% to less than 1%**

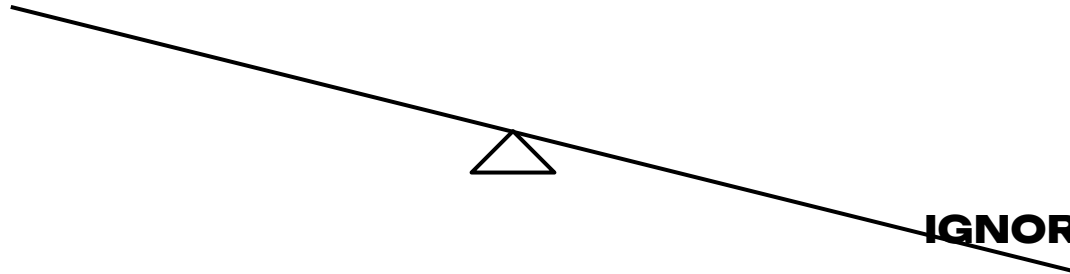


INTERRUPTION



IGNORANCE

INTERRUPTION



IGNORANCE

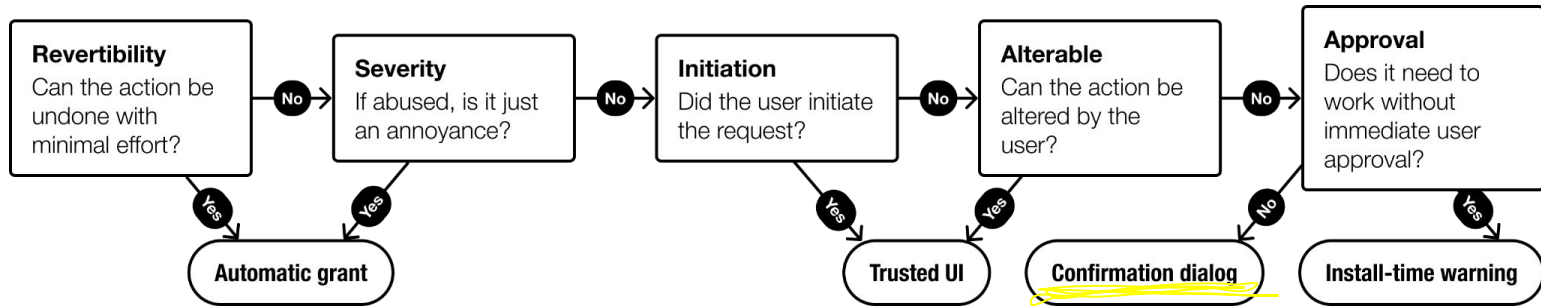
Too many prompts?

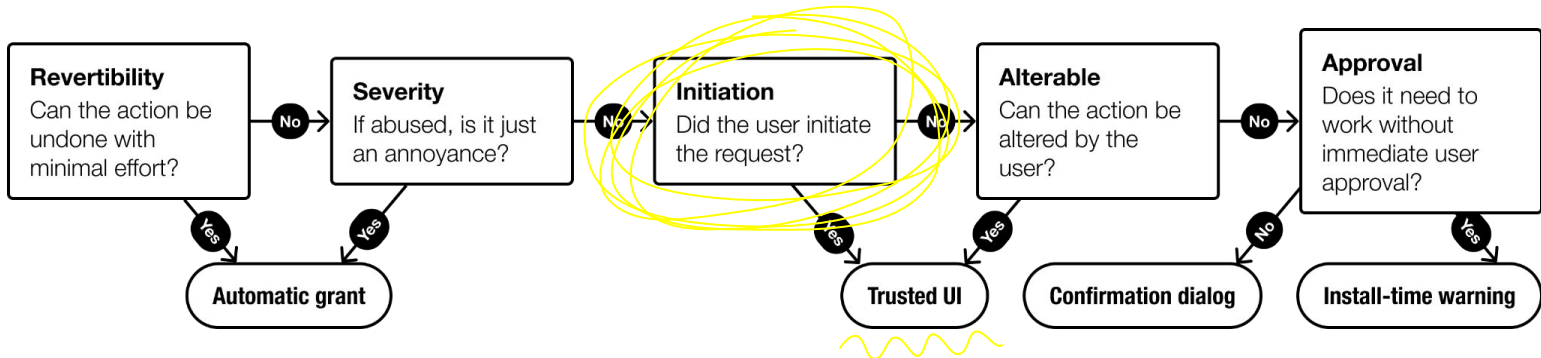
* Require a user gesture for all prompts?

... it's really easy to get users to click on things! :/

* *Doesn't mean we shouldn't do it, but:*
→ *it doesn't solve the problem entirely*
→ *... and does not solve user recovery*

Whenever there is a seemingly
unwinnable dichotomy, we should
ask: can we get more *specific*
about the problem?





By letting people initiate the permission moment, we can be ***confident in the user's intent*** to enable the capability, and design more opinionated, easier grant flows.

PEPC evolved from trying
to solve *all* of these
challenges



SOLUTION



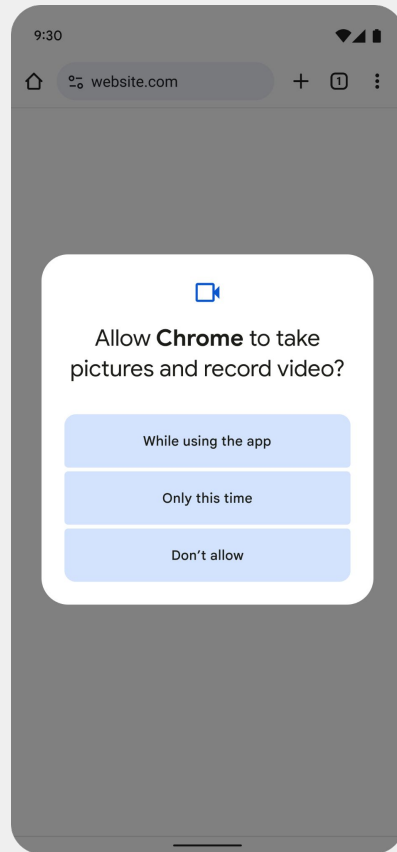
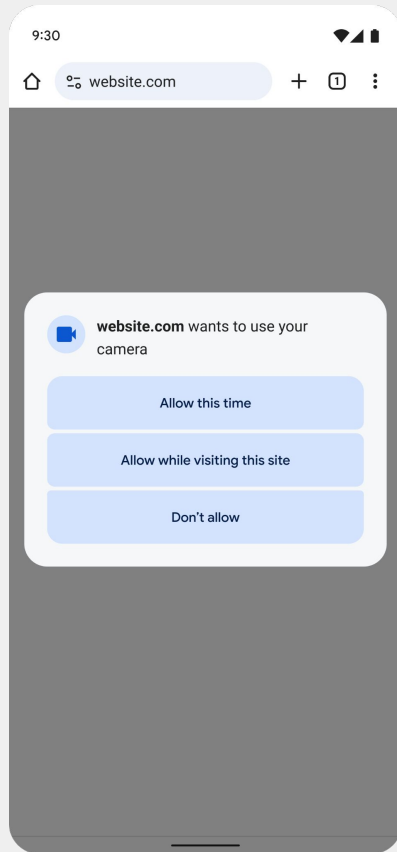
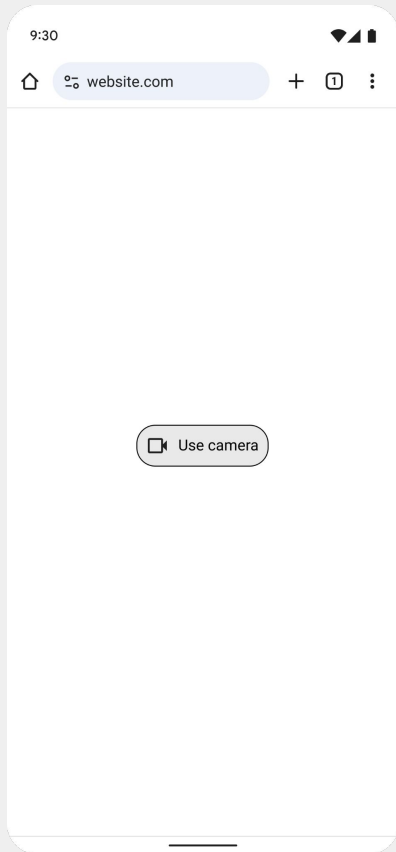
The Page Embedded Permission Control (PEPC) is a *new HTML element* embedded into web content that allows users to initiate a permission request flow.

This reframes the current permission model from developer-push to *user-pull*, the site can chain behavior using the PermissionStatus change event.

 Use camera & microphone

[Explainer](#)





PEPC Advantages

- ✱ Embedded directly into the web page, users have the context of interaction with PEPC
 - ▣ Anti-clickjacking measures prevent intrusive/annoying use of the PEPC
 - ▣ PEPC can be overlaid on top of a map, or in the green room of a video chat, *users know the context because they clicked on it!*
- ✱ No prompt unless the user chooses to click on the PEPC, therefore no prompt fatigue, users can use the PEPC at the appropriate time on their own terms
- ✱ Users can easily revisit past decisions (*including recovering from accidentally blocking access*)
 - ▣ Interesting use cases like a “settings” page where permissions can be toggled (and past decisions can be easily revisited)
- ✱ Users are extra safe because the PEPC confirmation UI (contextually triggered) is equivalent to control UI, but easier to understand because the user has context

PEPC Annoyance Reduction

- ✦ PEPC looks and feels consistent between websites, limited customization supported
- ✦ PEPC is disabled with a brief cooldown if:
 - ➡ Other elements draw on top of it
 - ➡ Page scroll or layout shift
 - ➡ Many PEPC elements of the same permission type are placed in the page
 - ➡ PEPC is taking up too much screen area

PEPC camera/mic
is in Origin Trial on
Chrome

Origin Trial early results

- * PEPC has been in Origin Trial since August, used in production by multiple VoIP vendors
- * Findings so far:
 - ➔ ~250% increase in recovery success rate!
 - ➔ evidence for latent demand for users to more easily revisit past permission decisions
 - ➔ Zero regression on grant rates, even when control rates were extremely high, *more users prefer to “one time” grant which we consider a success*
 - ➔ The average time to action for PEPC permission decision is reduced by about ~25–30%
 - ➔ $\frac{2}{3}$ of users with OS blocked permission click into OS UI to resolve the issue

Generalizing PEPC: a safe entry into browser UI

There are a variety of contextually relevant reasons users might want to engage with browser UI

- * App settings (for an installed PWA)
- * Page content settings
- * ... and more, probably!

PEPC ensures we have captured user *intent* to access a browser setting.

Open problems

Problem: PEPC duplicates existing functionality

PEPC proposes another way to do something that can already be done such as via Permission API or direct capability access.

Discussion: We can't incrementally improve our way to a PEPC.

- Annoyance safeguards depend on safe visual representation
- Intrusive confirmation UI depends on securing user intent, or risks amplifying user annoyance & disruption

Problem: Defensive styling restrictions

Can we actually keep users safe while shipping something developers will actually want to use?

Discussion: styling restrictions on PEPC are for *spam prevention* not *security*. (Security is in the browser generated confirmation UI.) We opted for maximal styling restrictions in the initial proposal, and think they can be relaxed in the future. Clever hacks around styling restrictions can only *annoy* users.

We've engaged multiple 3rd parties in videoconferencing and retail verticals, some of whom have active OTs running with the PEPC. The restrictions have not been a blocker (*caveat: lack of icon support has been an issue, however, this is an implementation shortcoming and we plan to add icon support in the future*).

Problem : Technical complexity of styling restrictions

Most of the complexity in the styling restrictions of PEPC is actually in IntersectionObserverV2. IntersectionObserverV1 is already **baseline widely available**. IntersectionObserverV2 is used in ~9% of page loads, and is an interop 2025 request.

For CSS properties specifically we use a combination of 2 methods:

- 1) Set bounds on some properties (via building min/max/clamp expressions or by directly correcting the value for keyword-based properties). This automatically corrects the style set by the site. For some properties we rely on the internal implementation of [calc-size](#) which is currently in the csswg.
- 2) Mark the element as invalid in the cases where it's not possible to correct the style (currently applies to **font-size** and **color/background-color**)

Problem: Can the PEPC be used without the confirmation UI?

We think so, but haven't explicitly explored this yet.

Possibilities:

- for low risk permissions
- restricted circumstances, such as when the user has previously granted the permission