Deprecation is hard to do and we can do it better!

Slides

Scribe: Heather!

Notes

- Ben: Welcome to the new Mozilla branding slides!
- <group> ooooooooh
- Ben: [showing slides] Deprecation basically means “don’t use this” and that’s actually easy. Taking things away, though, is harder. Classes of deprecation: unshipping them (not available), disabling them (opt-in), discouraging them (opt-out).
- Ben: Why deprecate? Accessibility (e.g., blink tags), site security, user protections, platform improvement, other…

Discussion

- Ben: so, when deprecating something, questions to ask: what are the harms if we rip it out? What are the mitigations we can do to make it less bad? What are the decision points to decide to act? One example we can work with: storage access heuristics (e.g., can’t just remove third-party cookies because authentication needs them). The heuristics were supposed to be temporary, and yet removing the heuristics brings us back to the same harms
- Domenic Denicola: This is a good example. Maybe the heuristics should have been formally defined.
- Johann: Historical context - in hindsight, it was naive to think that putting anything on the web platform is something you can do temporarily and isolated to a browser such that it would be easy to remove again. Both Safari and Chrome started to pick those up because they were trying to do the same things and running into the same problems. It doesn’t matter how big you are as a browser, you don’t want to have compatibility issues.
- Ben Kelly: Now that all the browsers have it, we would like it to be temporary, at some point we need to evaluate the priority of removing it. I’m skeptical this will be removed in the near term, given the other work on the plage. This probably comes to the “When to act”. This doesn’t (yet) reach that bar. Need input from the other browsers. It’s a collective action to remove things, but sometimes browsers act unilaterally. Is this a collective moment or a unilateral moment?

- Brian: More on the harms question - one of the challenges with understanding harms at scale is you can't understand or assess all the use cases. There is an explainer about APIs in general that I've found helpful. It states that, at scale, you're going to have a complex problem and you're not going to solve it. You're not going to mitigate all damage, but at the same time you have to try. Also sharing a second link of prior art on a large-scale deprecation. Important takeaways: discussion about self-service exceptions (no one controlling entity on the web, but if there's a place that can collect feedback where website owners and developers can go describe what's breaking and how, that can be powerful and help with harm assessment). One of the biggest challenges in these discussions is that no one browser or IdP manage this. The W3C is the closest possibility to a single point to collect self-service feedback, so that we could fill out harms into a form we could assess.
- Mike West: On the topic of self-service, one ways we've been more successful is turning it off by default but letting developers opt back in when it is necessary for their use case. Not useful for every use case. There are missed features on the platform. If you turn these off but let sites that need it to turn them back on, that helps get some of the harm of website breakage out of the way. Over time, new applications won't be requiring that extra bit, and we'll start to see these things go away.
- Lukas Weichselbaum: Would like to go further to suggest allowing developers to opt in on deprecations. By allowing developers to say "I don't need this part of the web platform" so while those parts may not be removable, they can still protect themselves
- Camille Lamy: "off by default" is not that much easier. You start reaching the web that is simply not maintained. So if you say "just do this" for those sites that are not being maintained, "this" is still too much to expect.
- Karl Dubost: it is very hard to deprecate things. It depends on what we want on the web. Might want to check webcompat.com which is a site that collects breakage reports.
- Ben: using common web compatibility tools to ensure alignment is good
- Kadir: There are new browser versions every other week, so it's really hard for developers to keep track of what's being deprecated. While there are things browser vendors disagree on, there are areas of agreement, but in all cases, they aren't great at communicating the timelines for what is being deprecated. And if it's not deprecated across all browsers at the same time, it's still that much hader.
- Ben: What channels of communication would be useful?
- Kadir: We're using devtools a lot for some of those things. Things like web features baseline, and once a year you have announcements - could also do un-announcements. There's also the interop process. "Negative tests in interop"
- Brian: Adding more color to developer communication. I spend a lot of time communicating with developers, and while most info is logged in console or other means, it's often challenging to understand what's triggering the event. Page load? User action? New policy enforcement that a browser has taken? There is an opportunity to surface more information to security and privacy controls within the browsers, and make it more clear if it's a user-generated event or something done on page load. It is a very complex space.

- Johan: Two points - responding to “need what other platforms have”, as someone who has done deprecations for half my professional career, I appreciate this property of the web platform that we support websites essentially forever. It would make my job easier if we didn’t, but I think it’s the right thing to do. With all the efforts we have to how can we deprecate things faster, we should also remind ourselves of this property. Is there a safe subset of the web that we could recommend to websites? Maintaining that property explicitly is something I’d be interested in. The other thing is that we should segment developers into different categories: those paying a lot of attention to web processes to those that have abandoned their sites. There is another category that touches on how much they’re able to make changes (e.g., they may not be able to take action because we’re taking something away that’s essential to their website).
- James: Adding stuff to specs and heuristics - it is true that if you need heuristics to unship a feature, that should be in a spec. But every browser has the capability of site-specific work arounds. There are web compat mechanisms. But overall, this is information is unshared.
- Ben: yes, we need site-specific mitigations and coordination of them would be good.
- Ben Kelly: There is a difference between unilateral lists in browsers and site self-service lists. When we use unilateral lists, we’re optimizing for users at the expense of developers. What I like about the Origin Trial system is that it is an opt-in; sites have to register for it. All the browsers have these unilateral lists, but we also let sites opt out by using a well-known file. That’s a good balance between the user and the site having control they need. There is a well-known file spec in the IETF that might be useful to explore. It’s specific to third-party cookies but might be interesting.
- Ben: Something I’ve heard as a concern in the federated identity stuff is the vague threat of potential deprecation in the future makes decisions harder. This touches on Johan’s thought about a safe web space. This is informing design choices in FedCM.
- Brian Campbell: I feel like there has been the depreciation or threat thereof is used as an underhanded way to force change not necessarily related to the original problem. There are knock on effect to the consumers for this stuff that aren’t captured in the metrics of what a particular deprecation breaks.
- Johan: Responding to that, it’s a pretty direct reference to the privacy space. We’re in a very difficult situation as browsers to make privacy improvements for users. Some of the most used features on the web have properties that are problematic for user privacy. If we ambitiously want to create a world where the user agent has a grasp of the privacy and control about how much info about the user can be transferred across sites, that does mean some difficult implications for the future. We shouldn’t announce plans too early because it has massive implications for the ecosystem. It has to be so ominous and veiled that I want to acknowledge it’s a very difficult space.
- Greg: we brought this up about two years ago, and there wasn’t much appetite for a comms location, but it feels like an easy option. Second point: how many websites leverate the reporting API to report back to browser vendors? So browsers can understand the impact of deprecations?
- Johan: is this a new proposed feature to the reporting API?

- Greg: browsers don't know what they don't know. Developers also have responsibility here to, so while a centralized comms location is good, a reporting API that comes back to me can democratize in a private way sharing information.
- Camille Lamy: Replying to that, websites already have communication channels with browsers. When we were doing private network access, they just filed a bug with us so we could engage in discussion directly. Most of the time the developers were not looking at reporting or dev tools until things hit stable.
- Greg: not what I was referring to. I have to care about 6 different browsers, they all report in different ways at different levels of fidelity. It's not as simple as watching one browser; having a central communication place would make life easier.
- Ben: There are also different definitions and timelines
- ??: I interviewed several developers about this issue, and they all talked about how painful it was. The follow up question was "what was the next deprecation you're preparing for?" None of them had an answer, and they didn't have an answer as to how they were learning about the deprecations (websites, social media). Having one point to subscribe to would be so helpful.
- Ben: the challenge there might be getting runway and communicating
- Etienne: It is hard for developers to know deprecations are coming. In the app world, what if we started aligning on a date to get them to start to expect change on a specific timeline.
- Johan: that's not what people are used to on the web.
- Philippe: if we have one place where browser vendors inform their intent to deprecate something and we can have dialogs, a GitHub repository would be easy. We could have the conversation there and then triage it to the group that's talking about it.
- Ben: let's pull out of communications to developers; we have good ideas there.
- Mike Taylor: Want to acknowledge the idea of centralized communications are good but also difficult to get corporations to align on branding. It's a multi-year problem, so maybe technology is the way to go like an API that someone could just deploy. We have different goals and deprecate things on different schedules, but old TLS, app cache, those have deprecated well. Browser vendors all blogging on the same day was very powerful.
- Brian: majority of websites aren't actively maintained, so while great to inform developers and getting them to take action, we need to recognize there are no developers maintaining those. Want to challenge this group, especially the browser vendors, to create a common framework for the users to understand the the risks. Notification of what risks might be is very different across browsers. It would be good to have a common model when letting users into a danger zone.
- Ben: TLS and click throughs and warning papers…
- Johan: it's about how composable the web is. Who are you warning? Do put a prompt on all pages on the web?
- James: A lot of this is case-by-case. People are coming from privacy and security, but sometimes its a feature or mutation events ("this feature sucks").
- Ben: 10 minutes left, so need to start to wrap up.

- Camille Lamy: sometimes its shooting yourself in the foot, but sometimes it's a security issue because having it out there makes other website vulnerable. It needs to go away or any website can be attack. That's harder to deal with than TLS.
- Mike Taylor: It's important to consider the property of the web of backward compatibility. In the blink project, we have a concept of API owners who are individuals who reason about the implications of what's going to happen, what's the severity of the breakage, is it worth the pain to developers and humans to make this change.
- Ben: What are our next steps? What do we do in the W3C? In WHATWG?
- Domenic Denicola: the concrete ideas about using interop or baseline, there are different categories about how urgent browsers find deprecation. Sometimes they align. Is there anything this year we all agree needs to deprecate?
- Ben: that's a good point of what we should do in next steps. We need to identify concrete actions to take.
- Ben Kelly: It would be useful to have a metric to help us decide if we're improving. The reporting API could go a long way.
- Domenic: the Interop project has a process where several people submit things, they suggest metrics, and browsers get together to decide.
- Ben Kelly: I don't know we have a metric or survey that says "how many times did your site break this past year because browsers removed something?"
- Camille: from security, we're more interested in deprecation options. Anything where we have more coordinated efforts is something we're very interested in.
- Ben: is there a good venue to continue this conversation?
- Mike Taylor: there is appetite to continue conversations. We should identify people who want to be involved and identify a venue later.
- Ben: WHATWG/webcompat is a great rallying point.
- Ben Kelly: File an issue on webcompat
- Ben: Yay! Don't email me! respond to the issue in webcompat.
    - https://github.com/whatwg/compat/issues/270

Harms	Mitigations	Whether or not to Act
Unknown use cases cannot reach zero but must try (Hyrums Law)	Storage access heuristics • weren't standardized because we wanted to not make them permanent	Mitigations are not impermanent
You break things, and in theory websites should work forever	Off by default instead of hard removal (depends on threat model justifying deprecation) • initially opt-in to removal • ObD is not much	Prioritization: harms

	easier than hard removal, unmaintained websites	
Outsized influence on smaller dev ops	Common webcompat tools and alignment in mitigation	Alignment
Don't want to use threat of deprecation to leverate new web spaces that would not be - comes across like a bait-and-switch ● sometimes browser have to be vague	Communications to developers ● coherence of action by browsers	Presence of an alternative
Ecosystem implications, economics and incentives	Deprecation safe subset	user benefit
	Reporting API! Thanks Devs!	Platform benefit (perf, arch, UX) ● v0 of a spec ○ who is likely to be using it? will they hear? ○ how ossified is it?
	Deprecation Holidays	How does the feature affect other sites?
	Probably Multiyear	What is backwards compatibility?
		Will the action create a new problem down the road?
		Need human feedback

Attendees

- We have people here and have no consensus on anything. (42 people in room)