Fugu Capability Process

How to move new capability through the design process

Last updated: Sept 2018

The Fugu capability effort is designed to rapidly add new capabilities to the web platform. This guide should help you figure out the order when considering a new capability and the template should serve to help you get started on the PRD.

Note this guide is **not at all a replacement for the standards process or the blink launch process**. You will still use the Blink process and file bugs as the primary tracker, as shown in Web Platform Shipping Process Diagram. This guide is just to call out some of the additional pieces such as customer organizing and permissions model creation.

This is a champion driven process. Due to the highly parallelized nature of this work, it is important that one person is on top of driving each capability forward. The champion should be pulling in the others as needed but should be actively keeping tabs on what the next step is and moving things forward.

Everything should be as public as possible and when something can be public, please remove any version of it from this doc and instead link to the public resource**.** This document should be a place for you to iterate on aspects that aren't ready for public consumption before linking out or to have any exploration that can't be public.

Suggested Reading

[Not all public, apologies]
go/standards-process
Extensible Web Manifesto
"make me an offer" security model
Control Access to Powerful Web Platform Features

Prioritization Process

Roles/Steps:

API Requester - This may be external customers, internal teams who wishes to add new API capabilities in the Web Platform. They file requests into Blink>Capabilities|Fugu>Request using crbug.

Fugu Leads - TLs for capabilities who meet regularly (bi-weekly) to go over the API request backlog. The outcome is a prioritized list of bugs with Proj-Fugu labels attached. The leads will set the bug state as “Available” - to be worked on or Assigned - ready to be worked on. The assigned bugs will be continued to be worked on by API Champions.

API Champion - owners for getting API shipped. The API champion will ensure that the PRD template has been filled out and updated on the tracking bug. Also, the champion will fill out the intended OT milestone. Finally, once ready to ship, the champion will file the launch request and add an entry into chromstatus.com.

Process

Steps

Step 1: Create bare-bones explainer & feature request bug

Everything starts with a feature request which we then turn into generic problem statements. Requests should come either from customers directly filing a crbug feature request or you filing one on their behalf. You’ll want just enough explanation so people vaguely understand the scope and you can refine it during the next steps. Do this in a private github repo; public from the start, but not broadly distributed. The Explainer will be the doc you iterate on most heavily through the entire process.

At this point you should also send your explainer to emailing chrome-security@, chrome-permissions-team@, and chrome-privacy-core@.

Apply the following template to the bug description - it can start off with "TBD" placeholders, but provides consistency and context for others (internal and external) who want to understand the details of the request:

```
Summary:
<1-2 sentence description>

Comparable chrome.* APIs:
Public Discussion: TBD
Explainer: TBD
Spec: TBD
ChromeStatus Entry: TBD
```

```
Other context (may be Googler-only, sorry!):
<link1>
<link2>
...
```

Please leave sections in even if they are TBD/empty - the bug description can be edited later to fill them in.

Step 2: Get input on your explainer and gather customers & support

Once you have enough clarity in the explainer you will want to publicize it to gather feedback.

Identify specific customers who have the problem the feature will solve and make sure to record them in this template (below); we'll want to reach out to them throughout the process. If a partner wishes to remain anonymous, you can create a restrict-view-google partner bug such as those in this list and block them against bugs of interest. Note that as a policy **we do not commit to shipping a capability unless we have a customer lined up and committed to use it once implemented.**

You will also want to gather public support in locations that can be linked to or cited. This sometimes takes place via the WICG Discourse group (example), which is also a place for shopping the explainer around and getting early feedback. Note that you should have at least one eager customer who you know for sure will adopt. If the adopting customer is internal, you should still see if they are willing to post publicly, since it will impact other browser adoption. This is also the time to file for TAG feedback

Step 3: Commit to a capability

Once there seems to be sufficient reason to do a capability, we prioritize it appropriately in the backlog. The priority will be evaluated by the champion eng and the fugu leads. When it is time to start work on that capability, the champion will send an Intent To Implement. At this point you should also have at least a POC for role, such as PM and devrel. If you have someone already for the role, just list them directly, but otherwise for each role you can ask this person who the best suited person is:

PM: Likely Thomas Nattestad
Security: ?
Privacy: Likely Martin Šrámek
UX: Likely Hwi Kyoung Lee or Namrata Kannan
BD / Partner manager: Likely Aanchal Bahadur
DevRel: Likely Pete LePage or Thomas Steiner

Step 4: Define the security and privacy model

Work with the lead eng, PM, security, privacy, and permissions team liaisons to iterate on and define the conditions under which we will expose this capability. This will follow the "make me an offer" security model. The Control Access to Powerful Web Platform Features is another guiding document to steer this phase of development.

A good start to this stage is reaching out to chrome-permissions-team@google.com team to get their input on what the permission model for the capability would be. You will eventually have to fill out a TAG security questionair and this can be a good first step.

Note that in May of 2019, we held a permission summit where we discussed many APIs. It is likely worth your while to see if your API was discussed here

Step 5: Iterate and evolve the design in public

At this point you are ready to iterate through the standards process. You'll make the API and get input from others. Make sure to explicitly run it by all your customers to make sure it works for their use case.

You'll iterate through this process extensively and spend a significant chunk of time before moving on.

As a final proof-point, an Intent to Experiment and associated Origin Trial can help you demonstrate that the proposal solves the problem while helping to demonstrate use and interest.

Step 6: Connect with DevRel to organize outreach

Connect with your DevRel to make sure they know about the feature and can plan out the creation of relevant blog posts and tutorials. As the feature advances make sure your devrel knows when the materials need to be ready. You can refer to this process document for more details.

Step 7: Get final approvals and Ship it!

Make sure you have gotten all the approvals specified by blink launch process and then ship it! Make sure to connect back to customers one last time to make sure they adopt the capability. If the customer was a Chrome App or Electron app, see if they are now interested in deprecating their previous app.

Template:

This template is only meant as one suggestion for how to organize the content necessary as you go through the process. Note that this template only has some sections as suggestions, but not everything is required and you can add any sections you think are relevant.

Fugu: <Capability Name>

Last updated:

Champion:
PM:
Security:
Privacy:
UX: namrata
BD / Partner manager: Aanchal
DevRel: Pete

Abstract

1-3 sentences on what this capability is and what problem it solves

Links

Crbug
Explainer
Intent to implement
Intent to ship
Standard / Spec
Others?

Explainer

Explainer of the API at a high level. If there is ever a public version or another doc that is the source of truth, **delete everything here and link instead**

Customers & Statement of Support

Which internal / external customers want this? Have developers publicly stated the need for this? You can also specify why this capability is important and needed for these customers

Security and Privacy Model

Under what conditions will we expose this capability? If there is ever a public version or another doc that is the source of truth, **delete everything here and link instead**

API Structure

Feel free to use this space for the API structure What does the surface of the API look like and how will developers interface with it? If there is ever a public version or another doc that is the source of truth, **delete everything here and link instead**

Eng Design

If you are the eng who will be designing and implementing this capability, feel free to do it in this area of this document or utilize a separate doc but link from here. If there is ever a public version or another doc that is the source of truth, **delete everything here and link instead**

Documentation / DevRel artifacts

If there is ever a public version or another doc that is the source of truth, **delete everything here and link instead**