Chromium Design Doc Template (public version)

This Document is Public

Set doc permissions to: "Anyone on the internet can find and view", Add contributors@chromium.org as either editors or commenters, but uncheck "notify" when doing so).

Authors: username@chromium.org
$MONTH $YEAR

One-page overview

Early design docs sent to chromium-design-docs@chromium.org need only this one-page section filled out (of course, include any additional information if you know it). If you post just a one-pager to the list, please follow up on the thread when the full doc is complete. Do so early enough to be able to incorporate significant feedback into your implementation.

Summary

1-3 sentence description of what is changing and why.

Platforms

Mac, Windows, Linux, Chrome OS, Fuchsia, Android, Android WebView, WebLayer, iOS.

Team

Best way to contact you. Could be a team email list, the email addresses of the TL and PM, or whatever makes the most sense.

Bug

Pending.

Code affected

Examples: "Network stack", "Android frontend", "Sync", "Extensions", "History page". Many features will have more than one. This is to give readers a general idea of what's changing so you don't need to be precise about directories.

Design

Discuss any background and motivation not already in the summary above.

Link to mocks, specs, and related features.

Give implementation details. Especially important is data flow between threads, processes, storage, and the network. Discuss the responsibilities of major components.

Metrics

Success metrics

You should list what metrics you will be tracking to measure the success of your feature or change. This could be a mix of existing and new metrics. If they are new metrics, explain how they will work. If you aim to improve performance with your feature or change, you should measure your impact on one of the speed launch metrics.

Regression metrics

You should define what metrics you will be tracking to look for potential regressions associated with your feature or change. This could be a mix of existing and new metrics. The speed launch metrics are good candidates for use as performance regression metrics. If you're using new metrics, explain how they will work.

Experiments

If you are using Finch to run experiments (see go/newChromeFeature for advice), describe what experiments you intend to run and what you are looking for in the results. It's important that you know in advance what the acceptance criteria are. List the experiment names so people can look them up (links to the dashboard are even better).

Rollout plan

If you're just checking in the code and doing a dev-beta-stable progression, just write "Waterfall" here. If you're doing a standard experiment-controlled rollout, write that with the experiment name. If you're not doing the standard rollout, describe what you're doing and why it needs to be different.

If there are external deadlines, call them out (if you don't want these to be public, use the internal template). Otherwise, releases should be quality driven and you shouldn't be targeting a milestone.

Core principle considerations

Everything we do should be aligned with and consider Chrome's core principles. If there are any specific stability concerns, be sure to address them with appropriate experiments.

Speed

Describe the considerations you're making with respect to how this work impacts Chrome's performance (speed, memory usage, power, etc.). Note that no change should regress the speed launch metrics.

It can be very hard to predict the end-user impact of a change on performance due to the wide variety of web content, device types, network connections and other factors in the field. Therefore, speed releasing team strongly recommends that finch is used for any launch that could plausibly affect speed. Early indicators of performance can be seen by running benchmarks on the perf trybots or cluster telemetry, but ideally performance impact would be measured by the speed launch metrics on end users.

Simplicity

Discuss the user-visible effects of your change. Especially important are new top-level UI controls or additional decisions users must make. If users need to understand new concepts, have information available to make decisions, or have certain mental models, explain the background your feature assumes. Mention switching costs of existing users adapting to any new workflow and how those users will perceive the change to be in their best interests. Acknowledge use cases you are intentionally making more complex or difficult. Note that this section is not about implementation simplicity; indeed, sometimes a simpler UX requires a more complex implementation.

Simplicity effects are not always obvious. Googlers should consider involving Chrome UX Research throughout the concept, design, and implementation stages of their feature to help guide these decisions. Chrome desktop UI features should also go through the Desktop UI Process to involve relevant engineering consultants early on.

Security

Sketch your threat model and describe the system's security mechanisms, especially around the handling of untrusted data. Be sure to describe any attack surfaces, any known vulnerabilities or points of failures, as well as any potentially insecure dependencies. If your feature doesn't have security considerations, explicitly state so (and why).

Please be aware of guidelines such as the need for the browser process to assume a compromised renderer process, the Rule of Two, URL display guidelines, Use origin rather than URL for security decisions..

Privacy considerations

Features with privacy implications should use the internal template for privacy review.

Testing plan

It goes without saying that all code should have good tests run on the waterfall. You don't need to write about that.

Here is where you should describe what the test team may need to consider before approving your launch. Some features won't need special testing considerations. If so, say this and why. Otherwise, give any directions needed to exercise your feature. Call out any special platforms conditions that may require extra attention.

Followup work

How will you assess the success of this work? What needs to be followed-up on? What (experimental code, for example) needs to be cleaned up after the code has reached the stable channel?