Origin Trials - Proposed Updates to Blink Launch Process

Status: Draft
Authors: chasej@chromium.org
Last Updated: 2016-12-15
Public Document

Objective

Implement changes to the Blink launch process to ensure origin trials are used appropriately to ensure we standardize and ship the optimal version of web platform features.

Note: this document will be deprecated when the proposed changes to the Blink launch process land. At that point, the associated reasoning and background here will move to the "Running an origin trial for a feature" guide.

Background

As the use of origin trials is increasing, it is important to be very clear about when and why a feature should run a trial.

We have also heard negative feedback about:

- Origin trials being run without a good reason, or for the wrong reason
    - E.g. seen as a shortcut to shipping
- Feature authors expressing a desire to avoid breaking changes when progressing from origin trial to enabled-by-default
    - e.g. feature authors want to avoid inconvenience to web developers that are participating in a trial, so do not rename/alter/remove sub-optimal API surface

For deeper background, previous discussion/documentation on the topic:

- Public explainer document
- Google internal presentation
- blink-api-owners-discuss thread Oct 2016
- Origin Trials: What, Why, How Discussion 2016

Updates to the Blink Launch Process

To address this feedback we propose to replace Step 7 of the Blink Launch Process with the following:

7. Optionally, run an origin trial for your feature to collect developer feedback/data, as input to the standardization process.

- If you answer “no” to all of the following questions, an origin trial is unnecessary.
    - Is there disagreement about how well this API satisfies its intended use case?
    - Are you unsure about what API shape will be the most ergonomic in real world scenarios?
    - Is it hard to quantify performance gains without testing on real world sites?
    - Is there a reason that this API needs to be deployed to real users, rather than behind a flag, for data to be meaningful?
- If you decide to run a trial, or are unsure, please first consult with the Origin Trials core team.
    - Email experimentation-dev@chromium.org. Google employees can alternatively schedule a meeting directly with origin-trials-core@google.com.
- If you've decided to run an origin trial, do the following steps. If not then skip to step 8 of the launch process.
    - Follow the instructions on how to run an origin trial. Google employees should see go/running-an-origin-trial.
    - Email blink-dev using the "Intent to Experiment" template. Wait for LGTMs from at least 1 API Owner.
    - At the start of every subsequent release, post an update on usage of the feature on the intent to experiment thread in blink-dev.
    - There is an automatic and mandatory 1 week period when your feature is disabled at the end of the origin trial, before it potentially graduates to Stable. Tokens will expire 1 week before the earliest stable channel launch date, but note

that a stable launch takes many days to deploy to users. This exists to encourage feature authors to make breaking changes, if appropriate, before the feature lands in stable, and to make clear to clients of the origin trial feature that in all circumstances the feature will be disabled (hopefully only briefly) at some point.

Rationale

The process updates are motivated by answering some fundamental questions about origin trials. The relevant questions are:

1. What is the purpose of origin trials?
2. Why run an origin trial?
3. How does a feature graduate to shipping at the end of an origin trial?

The questions have been discussed before, but the answers are summarized in this document for convenience.

The purpose of origin trials

The purpose of origin trials is to help deliver the right web platform features at a faster rate. Prior to origin trials, teams would spend years designing a feature and then enable a feature by default only to learn it wasn't designed correctly. Those mistakes then resulted in a new multi-year cycle (e.g. AppCache to service worker). With origin trials, teams delay shipping their (possibly poor) first version of a feature in exchange for learning fast, iterating, and shipping the *right* version of the feature.

It is not sufficient to just ship some version of the feature in Chrome, as soon as possible and consider it done. Origin trials support shipping the right version of a feature through science: iterating on designs (i.e. theories), collecting data, and making evidence-based decisions.

Why not to run an origin trial

In other words, origin trials are not a shortcut to shipping a feature in Chrome. Origin trials are not meant as an early access program, i.e. not for web developers to start using new features before they are standardized.

Why run an origin trial

Origin trials should be used to collect data on features, as input into the standardization process. That is, to allow feature authors to design and ship features based on data, not opinions. There are two categories of data to be collected:

1. Demonstration of utility to web developers and users that justifies the feature
2. Feedback to address uncertainty in the feature design

In both cases, usage behind a flag might not be sufficient. Web developers are more likely to engage with a feature if they can try it out on a significant number of their users. Relying on users to enable a feature via a flag reduces the number of engaged users. An origin trial allows the feature to enabled by the web developer on behalf of their users. Therefore, an origin trial can ensure a sufficient number of users, where runtime flags cannot.

An origin trial should be run to answer specific question(s), not simply increase the number of users. The questions should be stated at the outset of the trial, and feedback designed to collect appropriate data to answer the questions.

Run an origin trial for any of the following:

- Demonstrate a feature is able to satisfy a missing capability or specific use case
- Test the ergonomics of a feature design
- Measure real world impact of a feature intended to improve performance

How does a feature graduate to shipping at the end of an origin trial?

Origin trials are about experimentation. This means feature authors and web developers should actively expect APIs to undergo breaking changes or disappear entirely.

Why will there be breakage?

Feature authors should be learning from developer feedback and these lessons may well result in API changes that break existing code (in form or semantics).

Although all parties were made aware the feature is experimental, some web developers may be in fact be using the API in production websites so feature authors may be tempted to avoid breaking changes, even those that improve the API.

We believe feature authors *must* prioritize the long-term, over short-term breakage for trial participants.

To achieve that, we believe intentionally breaking all experimental features in live origin trials will cause all involved will become accustomed to breakage, and plan accordingly.

How will breakage be implemented in origin trials?

Origin trials ensure that experimental features are automatically disabled at the end of the trial.

For features that do not graduate to shipping, that is essentially the end of the story and all must live with the breakage.

For features that do graduate to shipping, the feature may become available on stable with timing such that there was never an interruption from the start of the origin trial, through to shipping. This is the case where we fear feature authors may be inclined to avoid breaking changes, rather than taking the opportunity to use the feedback to improve the API.

To prevent that case, we propose to enforce a gap of at least one week between the end of the origin trial and the stable release date of the shipping feature. To achieve this, tokens will be not be issued with expiration dates beyond one week prior to when the feature would be enabled on stable.

Features are disabled for a minimum of one week

Features will be disabled for a minimum of one week, leading up to the stable release date. The rollout of a release can take several days to reach all users. This means that features will likely be disabled for closer to two weeks.

If the goal is to set expectations by enforcing actual breakage, it could be argued to have features disabled for a longer period. While the goal is not to eliminate disruption, it should be minimized where possible. That is, use the most narrow mitigation possible to address the concern that developers don't embrace the experimental nature of origin trials. Without a technical/security/privacy reason to disable the feature, it should be made available and provide a useful transition through all development stages.