Case study:
Popup with user-activation across browsers

mustaq@chromium.org, May 4 2017

Test details

Popups in different contexts
- Popups without user activation (control)
- Immediate calls from within event handlers
- Calls from janky event handlers
- Calls scheduled from within event handlers
- Calls executed through a Promise
- Calls from a subframe

Time-span of user activation state

Other observations
- Input events allowing popups
- Consumption of user activation state
  - With disabled popup blocker
- Origin of the popups
- Popups from simulated clicks

Open questions

Browsers utilize user gestures (or “user activation” as in the HTML spec) to filter out unwanted use of abusable APIs. Opening popups (using `window.open`) is one such API for which perhaps all browsers have some user activation based filtering. A list of all APIs in Chrome that are user activation dependent is available here. Activation consumption differences among browsers is covered in this explainer.

This document catalogs the similarities and differences between Chrome, Edge, Firefox and Safari for popups with user activation. It seems that there are many inconsistencies between these browsers even for this widely implemented user activation behavior. This observation provides another justification for a simple user activation model that could converge the Web to a predictable gesture behavior spanning all browsers in all platforms.

Test details

- We tested the following page in 5 browsers: Chrome (Android & Linux), Firefox (Android & Linux), Edge (Windows 10) and Safari (iOS 10.2.1). The next section omits OSes because no browser behaves differently in a different OS.
- The test page subframe has generous sandbox parameters (`allow-scripts allow-popups`) to rule out additional filtering in the test.
- Test page: http://mustaqahmed.github.io/web/popup-with-user-activation.html

Popups in different contexts

We tested the six different "contexts" to trigger `window.open` calls. The following subsections describe the similarities & differences we observed for the browsers in each context.

Popups without user activation (control)

We tested the following two control cases where the popup calls are triggered from outside input-event handlers:

Control 1: Direct calls after the document is loaded.

Control 2: Calls made from a subframe in response to messages posted from main frame after the document is loaded.

None of the browsers allows popups in these cases.

Immediate calls from within event handlers

All browsers allow popups directly from within input-event handlers provided that there is no long delay, see next subsection.

(Note, however, that the type of allowed events vary between browsers, see the section on event types below.)

Calls from janky event handlers

When input-event handlers are janky, we observed two different behaviors:

- Chrome stops allowing popups after approximately a second of jank.
- Edge, Firefox and Safari always allow popups, no matter how long the jank is.

Calls scheduled from within event handlers

When an input-event handler schedules a `window.open` call for later execution (through `window.setTimeout`), Edge is different from the rest:

- Chrome, Firefox and Safari allow scheduling delayed calls, each upto a maximum delay of one second.
- Edge doesn't even allow such calls, even with a 0ms delay.

Chrome allows a “call” depth of 1, so a nested `setTimeout` call doesn’t allow popups. Firefox seems to behave similarly. We didn’t test any other browsers.

Calls executed through a Promise

Popup calls executed through a Promise behave differently in all three browsers:

- Chrome allows such calls as long as the Promise is resolved within one second.
- Firefox never allows such calls, even when the Promise is resolved immediately.
- Edge and Safari always allow such calls, no matter how long it takes to resolve the Promise.

Calls from a subframe

For `window.open` calls made from a subframe while responding to a post-message dispatched from a main frame input-event handler:

- Chrome and Safari allow subframe popups initiated through main frame event handlers.
- Neither Firefox nor Edge support it.

Time-span of user activation state

There is no consistency among the browsers in how long the user activation state seem to survive after a user action:

- For a slow event handlers, Chrome withstands up to one second of jank, but none of Edge, Firefox & Safari have any time limit.
  [Update 2017-08-29: Firefox now expires activation after 1 sec, as noted here.]
- For `window.setTimeout`, the limits seems to be exactly one second in Chrome, Firefox & Safari. Edge doesn’t allow delayed calls.
- For Promise, Chrome again have a one-second limit. Edge & Safari have no limit. Firefox doesn’t even allow such calls.

Other observations

Input events allowing popups

[Update 2018-06-20: here is a nearly exhaustive list covering all major browsers.]

We tested five input-events: `mousedown`, `mouseup`, `touchstart`, `touchend` and `click`.

- Edge allows from all 5 events.
- Chrome & Safari allow popups from `mousedown`, `mouseup`, `touchend` and `click` events.
- Firefox allows from `mouseup`, `touchend` and `click`.

The events we tested is not exhaustive. For example, Chrome allows KeyboardEvents to trigger popups, and has some accessibility related triggers too. We didn't examine other browsers beyond those 5 events, but it's very unlikely that the set of all other triggers would be consistent among them.

Also note that for any browser, a different set of events could be used to filter out another API (say, fullscreen). So we can't really say that a browser defines user activation as "the list of events that can trigger popups".

Consumption of user activation state

Here we tested whether a popup call consumes the user activation state to prevent repeated calls from within the same context.

- Both Chrome & Edge consume the state. Safari started consuming from Safari 11 in 2017 [Update on 2017-11-27].
- Firefox doesn't consume.

With disabled popup blocker

On a related note, when popup blocking is disabled through browser settings:

- Both Chrome & Edge allow many popups with/without activation.
- Safari 11.0.1 allow one popup with/without activation.
- Firefox allow many with/without activation.

Origin of the popups

To confirm if browsers filter popups differently depending on the link for the popup window, we tried links with same and different origins (vs the caller).

- No browser differentiate between same- vs cross-origin popups.

Popups from simulated clicks

When a JS code simulates click, browsers are different in interpreting it as an user activation.

- Chrome & Firefox: simulated clicks can't open popups.
- Edge: Can.
- Safari: ???

Open questions

We studied only the popup behavior, which is perhaps the oldest use of user activation. Yet the major browsers don't show a consistent behavior, making it hard for web developers to predict the filtering outcomes. If we want to make at least the popup filtering behavior consistent across browsers, we will need precise definitions for:

- User activation: which low level events define an user activation?
- Consuming: should popups consume an user activation at all?

- Time-span: should a browser pass the user activation state to delayed popups/subframes/promises? If yes, what's the time limit?

To solve the general problem of unpredictable user activation behavior, Chrome is considering a simplified model that other browsers would possibly agree to implement (hence standardizable). Details of the plan appears here.