Proposal for delegating user activation to child frames

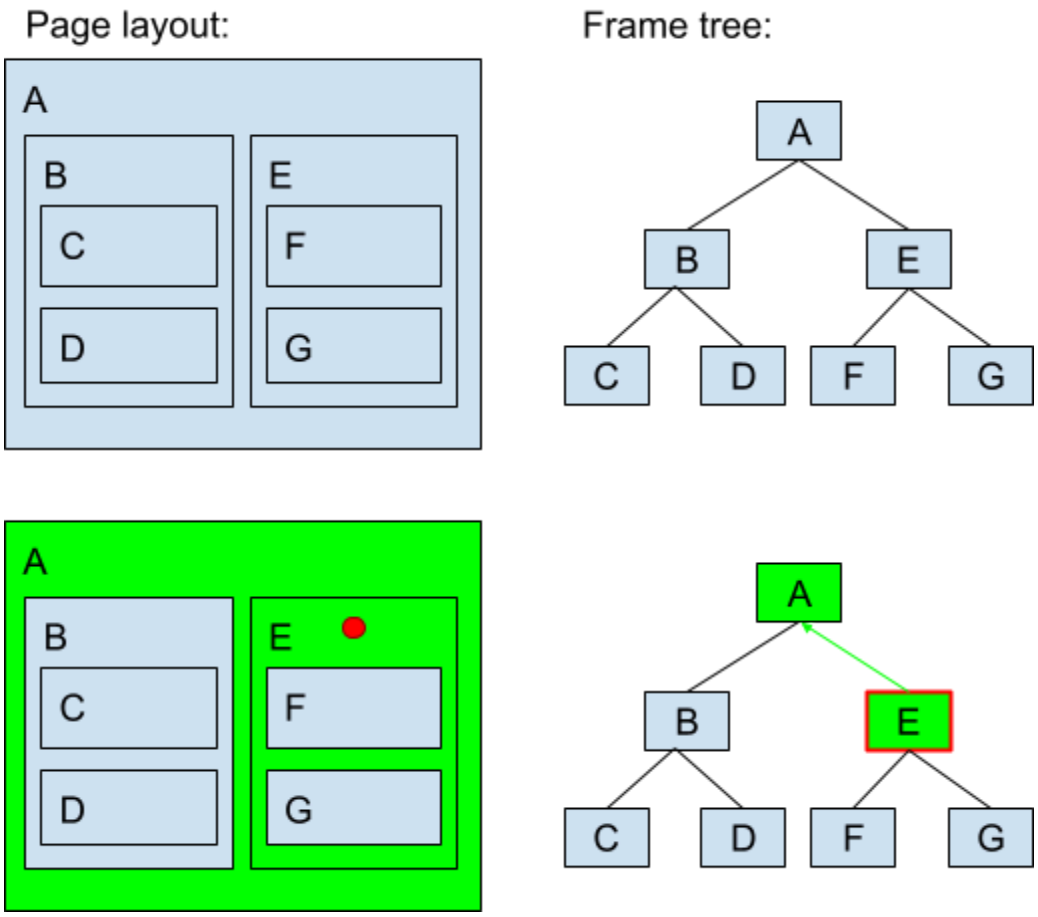
Status: Proposed
Author(s): Ian Clelland <iclelland@chromium.org>
Created: 2017-03-22
Last modified: 2018-10-24
Tracking bug: crbug.com/728334
This document is public

Problem Statement

Today, there are features on the web which are only available in response to a user gesture on the page. This is to prevent malicious action, and works to help preserve the principle of least surprise for the users. Features such as popup windows, vibration and fullscreen all require a gesture to either be currently active, or to have happened on the page in the past, in order to be activated. (Chrome APIs using user gestures)

When pages embed content from other sites, or applications, or ad networks, we need to be able to determine when a gesture "counts", for the purposes of allowing a feature to be used in a given frame.

The current model for deciding this is that any gesture in a frame counts as a gesture in that frame, and in any of its parents, all the way to the top-level frame in the page. In other words, if you click anywhere in the page, then you've interacted with the top-level document, but in order for an ad to vibrate, you need to click on that ad specifically (or a frame contained within that ad).



Example: Clicking any frame (Frame "E" in this case) creates a gesture which can flow up the frame tree to the root, but never flows down to any embedded children.

This is reasonable in many cases, and generally stops the use of powerful features in malicious advertising, but may be too strict for all cases. Authors often want to frame content from third party service providers, and have the user interact with that content as if it were a part of the top-level page. From the end user's perspective, it shouldn't matter what origin the content was served from: if it's supposed to be a part of the main page, then it should act like it is.

There has also been at least one case where an advertising provider has wanted to use these features in a non-malicious way, but the restrictions built into Chrome have made it impossible to do so seamlessly. We would like to be able to make a cooperating page and advertiser able to allow such features.

Proposal: <iframe delegate-activation> attribute

I'm proposing a "`delegate-activation`" boolean attribute for the HTML iframe element. It defaults to false if omitted, but if present, allows gestures in the containing frame to be used in the contained document.

If activation is required in order to activate a feature in a frame, then that requirement is satisfied if:

- it is satisfied by a gesture in that frame, or
- it is satisfied by a gesture in a descendent frame, or
- it's frame owner has the "gesture" attribute set, and it is satisfied in the parent frame.

(What exactly it means for a gesture to satisfy the requirement currently depends on the feature -- some features require a gesture to be in progress, others simply require that a gesture have occurred in the past. This proposal makes no changes to this, but just to which frames are allowed to "use" that gesture. In some cases, no gesture at all is required for the main frame -- in that case, that "implicit" gesture would also be granted to any content embedded by a frame with the "gesture" attribute set)

If a frame does not have the "gesture" attribute set, that effectively forms a barrier, which gestures will not cross into from the parent. That frame can include its own children, and can choose to use the "gesture" attribute for them, in which case a gesture in it will be usable by those children, but not a gesture from the top-level page.

[IDL]

```
partial interface HTMLIFrameElement {
    [CEReactions, Reflect] attribute boolean delegate-activation;
};
```

Examples

Simple example:

In this example, a site includes a map as well as an ad. When the user is interacting with the page, the map is allowed to use geolocation, vibrate, go fullscreen, etc. The ad gets none of those features until it is clicked directly.

HTML:

```
<body>
  <p>Some text</p>
```

```
  <iframe src="https://mapservice.com/" delegate-activation></iframe>
  <iframe src="https://adnetwork.com"></iframe>
</body>
```

Complex example:

In this case, it is not the top-level page which grants gestures to its subframes, but a third-party framed site. Clicking on the top page won't count as a gesture in the third-party content, but clicking inside of that frame will be treated as a gesture in *its* subframe.

Top-level HTML document:

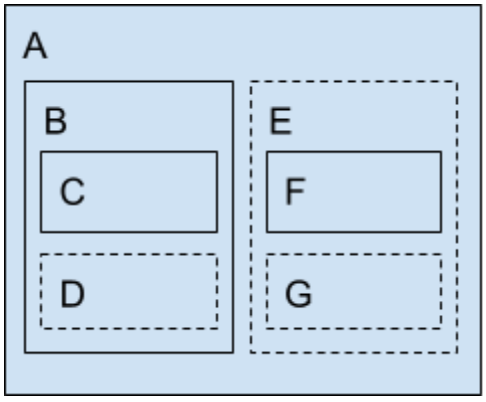
```
<body>
  <h1>Portal Site</h1>
  <iframe src="https://example.com/"></iframe>
</body>
```

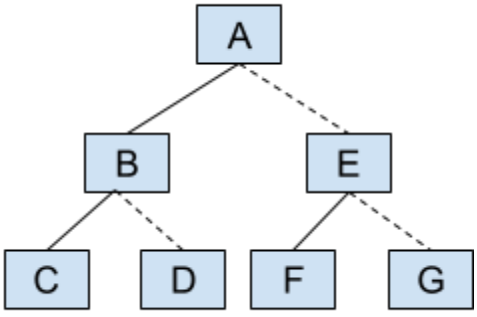
Framed site:

```
<body>
  <h1>Example Site</h1>
  <iframe src="https://mapservice.com/" delegate-activation></iframe>
</body>
```

Complete example: (frame tree diagrams)

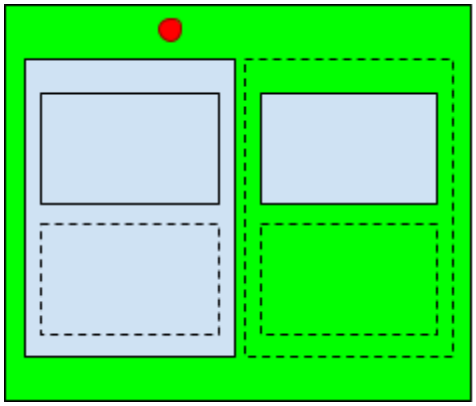
As a complex example, consider the following page structure: (Each nested box is an iframe; solid lines denote iframes without the gesture attribute; dashed lines denote iframes with the gesture attribute.)

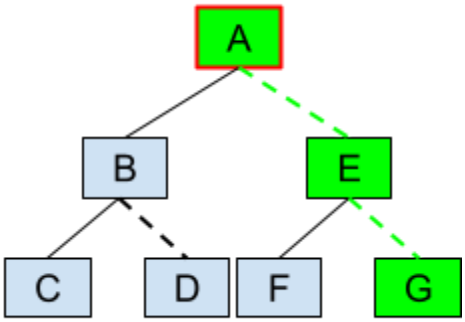




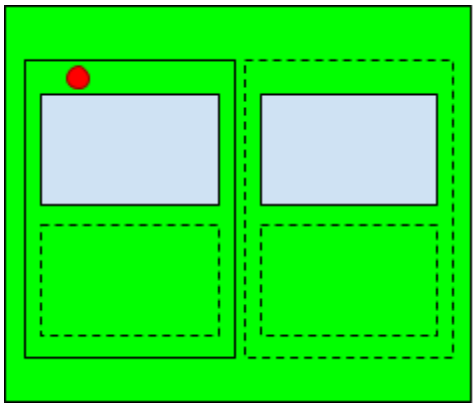
Clicking in various locations in the page would be considered a gesture in different frames, depending on the gesture attributes:

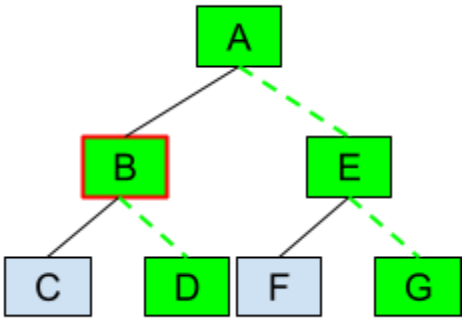
Clicking in the top-level frame (A) counts as a gesture in the two "gesture"-tagged iframes as well:



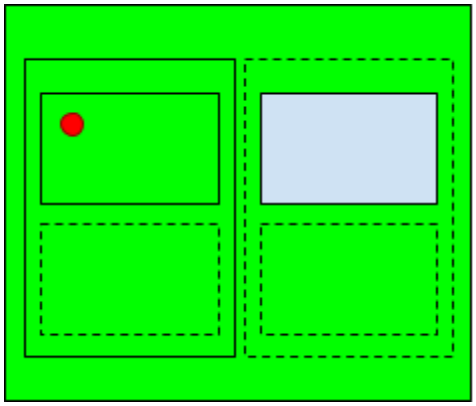


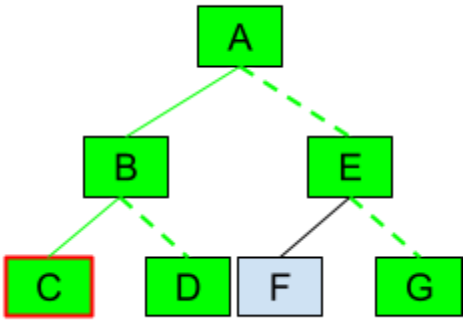
Clicking inside of the iframe "B" counts as a gesture inside of its embedded content as well. And because gestures also flow up to the top-level document, it also counts in the iframes which the top-level page has included with "gesture":



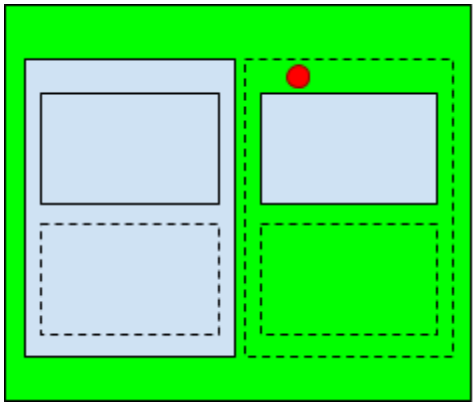


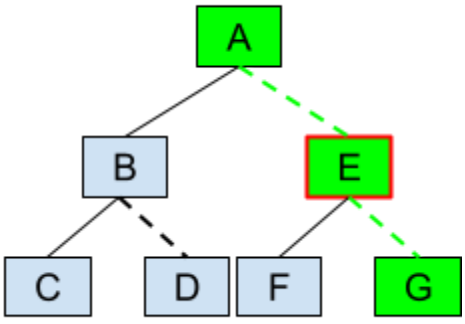
Clicking inside of C, which is embedded in B, also causes a gesture which flows up through the frame tree, and then down into any frames embedded with the "gesture" attribute:



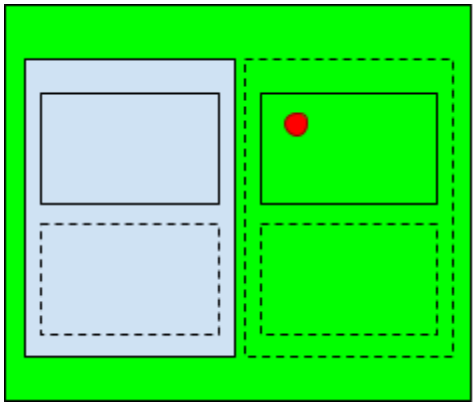


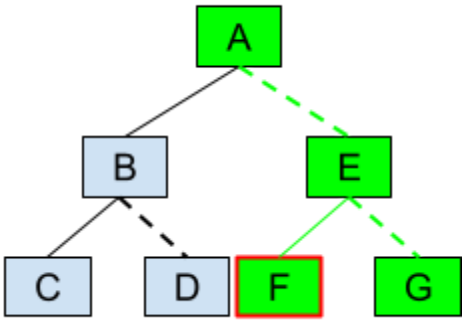
Clicking inside of frame E (or G) is exactly the same as clicking inside of frame A:





And finally, clicking inside of frame F allows F (and its parent frames) to see a gesture, as well as frame G:





Interaction with Feature Policy

One of the motivations for this proposal was the interaction between the Vibration gesture requirement and desire to control the vibration feature through feature policy. Vibration is currently allowed without a gesture in the main frame (consider the gesture to be implicit in that frame,) but a gesture is required in any cross-origin content.

Ideally, we would be able to allow gestureless use of vibrate in some iframes, as well as allowing a site to disable vibration completely in other frames, and still having the default gesture-required behavior available for other sites.

We can achieve each of these with this proposal:

For this example, vibrate is a feature which requires a gesture, but the gesture is implicit in the top-level frame. The feature policy for vibrate is that it is available in all frames by default. (In feature policy terms, this is a default allowlist of ["*"])

Example: Gestureless vibration allowed in top-level doc; gesture required in cross-origin frame (Default)

```
<iframe src="https://example.com/"></iframe>
```

Example: Gestureless vibration in cross-origin frame, using <iframe gesture>

```
<iframe src="https://example.com/" gesture></iframe>
```

Example: Vibration allowed in top-level origin, blocked in cross-origin frames

HTTP Header:
```
Feature-Policy: {"vibrate": ["self"]}
```

Example: Vibration blocked completely, using policy

HTTP Header:

```
Feature-Policy: {"vibrate": []}
```

Example: Vibration blocked in most frames, allowed with no gesture in "ad1" frame, allowed with gesture in "ad2" frame

HTTP Header:
```
Feature-Policy: {"vibrate": ["self"]}
```

HTML Document:
```
<iframe id="ad1" allow="vibrate" gesture src="..."></iframe>
<iframe id="ad2" allow="vibrate" src="..."></iframe>
```

Alternatives considered

[See original doc: Proposals for combining gesture delegation with feature policy]

1. "gesture" iframe feature

This was rejected because the nature of feature policy is that once disabled, it can never be re-enabled in a child frame. This would mean that a frame which does not receive gestures from its parents cannot choose to delegate its own gestures to its children.

2. Feature policy features for each gesture-gated-action which allows the action without gesture

Rejected because there would be no way to disable a gesture-activated feature completely. It would be allowed unconditionally in the main frame, and with gesture, or (with the appropriate policy,) without gesture in subframes.

3. Feature policy features for each gesture-gated-action (on by default) which can be blocked to turn the action off completely

Rejected because there would be no way to enable gestureless-actions in subframes; the only options would be allowed-with-gesture and denied.

4. Multiple feature policy features for each gesture-gated-action: "<action>-with-gesture" and "<action>-without-gesture"

Rejected as too complex, with poor developer ergonomics, and confusing when both features are mentioned in policy, especially when they contradict each other.