Deprecations and Removals in Chrome 60

By Joseph Medley
Technical Writer

In nearly every version of Chrome, we see a significant number of updates and improvements to the product, its performance, and also capabilities of the Web Platform. This article describes the deprecations and removals in Chrome 60, which is in beta as of June 8. This list is subject to change at any time.

Security

crypto.subtle now requires a secure origin

The Web Crypto API which has been supported since Chrome 37 has always worked on non-secure origins. Because of Chrome's long-standing policy of preferring secure origins for powerful features, `crypto.subtle` is no only visible on secure origins.

Intent to Remove | Chromium Bug

Remove content-initiated top frame navigations to data URLs

Because of their unfamiliarity to non-technical browser users, we're increasingly seeing the `data:` scheme being used in spoofing and phishing attacks. To prevent this, we're blocking web pages from loading `data:` URLs in the top frame. This applies to `<a>` tags, `window.open`, `window.location` and similar mechanisms. The `data:` scheme will still work for resources loaded by a page.

This feature was deprecated in Chrome 58 and is now removed.

Intent to Remove | Chromestatus Tracker | Chromium Bug

Temporarily disable navigator.sendBeacon() for some blobs

The `navigator.sendBeacon()` function has been available since Chrome 39. As originally implemented, the function's `data` argument could contain any arbitrary blob whose type is not CORS-safelisted. We believe this is a potential security threat, though no one has yet

tried to exploit it. Because we do NOT have a reasonable immediate fix for it, temporarily, `sendBeacon()` can no longer be invokable on blobs whose type is NOT CORS-safelisted.

Although this change was implemented for Chrome 60, it is has since been merged back to Chrome 59.

Chromium Bug

CSS

Make shadow-piercing descendant combinator behave like descendent combinator

Note: This item was bumped from Chrome 60 to a later version some time after this article was originally published.

The shadow-piercing descendant combinator (`>>>`), part of CSS Scoping Module Level 1 , was intended to match the children of a particular ancestor element even when they appeared inside of a shadow tree. This had some limitations. First, per the spec, it could only be used in JavaScript calls such as `querySelector()` and did not work in stylesheets. More importantly, browser vendors were unable to make it work beyond one level of the Shadow DOM.

Consequently, the descendant combinator has been removed from relevant specs including Shadow DOM v1. Rather than break web pages by removing this selector from Chromium, we've chosen instead to alias the shadow-piercing descendent combinator to the descendant combinator. The original behavior was deprecated in Chrome 45. The new behavior is implemented in Chrome 61.

Intent to Remove | Chromestatus Tracker | Chromium Bug

JavaScript

Deprecate and remove RTCPeerConnection.getStreamById()

Nearly two years ago, `getStreamById()` was removed from the WebRTC spec. Most other browsers have already removed this from their implementations. Though this function is believed to be little-used, it's also believed there is some minor interoperability risk with Edge

and WebKit-based browsers *other than* Safari where `getStreamById()` is still supported. Developers needing an alternative implementation can find example code in the Intent to Remove, below.

Removal is in Chrome 62.

Intent to Remove | Chromestatus Tracker | Chromium Bug

Deprecate SVGPathElement.getPathSegAtLength

More than two years ago, `getPathSegAtLength()` was removed from the SVG spec. Since there are only a handful of hits for this method in httparchive, it is being deprecated in Chrome 60. Removal is expected to be in Chrome 62, which will ship some time in early or middle October.

Intent to Deprecate | Chromestatus Tracker | Chromium Bug

Move getContextAttributes() behind a flag

The `getContextAttributes()` function has been supported on `CanvasRenderingContext2D` since 2013. However the feature was not part of any standard and has not become part of one since that time. It should have been implemented behind the `--enable-experimental-canvas-features` command line flag, but was mistakenly not. In Chrome 60 this oversight has been corrected. It's believed that this change is safe, since there's no data showing that anyone is using the method.

Chromium Bug

Remove Headers.prototype.getAll()

The `Headers.prototype.getAll()` function is being removed per the latest version of the Fetch specification.

Intent to Remove | Chromestatus Tracker | Chromium Bug

Remove indexedDB.webkitGetDatabaseNames()

We added this feature when Indexed DB was relatively new in Chrome and prefixing was all the rage. The API asynchronously returns a list of existing database names in an origin, which seemed sensible enough.

Unfortunately, the design is flawed, in that the results may be obsolete as soon as they are returned, so it can really only be used for logging, not serious application logic. The github issue tracks/links to previous discussion on alternatives, which would require a different approach. While there's been on-and-off interest by developers, given the lack of cross-browser progress the problem has been worked around by library authors.

Developers needing this functionality need to develop their own solution. Libraries like Dexie.js for example use a global table which is itself another database to track the names of databases.

This feature was deprecated in Chrome 58 and is now removed.

Intent to Remove | Chromestatus Tracker | Chromium Bug

Remove WEBKIT_KEYFRAMES_RULE and WEBKIT_KEYFRAME_RULE

The non-standard **`WEBKIT_KEYFRAMES_RULE`** and **`WEBKIT_KEYFRAME_RULE`** constants are removed from CSS Rule. Developers should use **`KEYFRAMES_RULE`** and **`KEYFRAME_RULE`** instead.

Intent to Remove | Chromestatus Tracker | Chromium Bug

User Interface

Require user gesture for beforeunload dialogs

From Chrome 60 onward, the **`beforeunload`** dialog will only appear if the frame attempting to display it has received a user gesture or user interaction (or if any embedded frame has received such a gesture). To be clear, this is not a change to the dispatch of the **`beforeunload`** event. It is just a change to whether the dialog is shown.

The **`beforeunload`** dialog is an app-modal dialog box. As such, it is inherently user-hostile, meaning it responds to a user navigation by questioning the user's decision. There are positive uses for this feature. For example, it's often used to warn users when they will lose data by navigating.

While the ability for a page to provide text for the **`beforeunload`** dialog was removed a while ago, **`beforeunload`** dialogs remain a vector of abuse. In particular, **`beforeunload`** dialogs are an ingredient of scam websites, where autoplay audio and threatening text provide a context where the Chromium provided "are you sure you want to leave this page" message becomes worrisome.

We want to thread the needle, and only allow good uses of the `beforeunload` dialog. Good uses of the dialog are those where the user has state that might be lost. If the user never interacted with the page, then the user cannot have any state that might be lost, and therefore we do not risk user data loss by suppressing the dialog in that case.

Deprecation policy

To keep the platform healthy, we sometimes remove APIs from the Web Platform which have run their course. There can be many reasons why we would remove an API, such as:

- They are superseded by newer APIs.
- They are updated to reflect changes to specifications to bring alignment and consistency with other browsers.
- They are early experiments that never came to fruition in other browsers and thus can increase the burden of support for web developers.

Some of these changes will have an effect on a very small number of sites. To mitigate issues ahead of time, we try to give developers advanced notice so they can make the required changes to keep their sites running.

Chrome currently has a process for deprecations and removals of API's, essentially:

- Announce on the blink-dev mailing list.
- Set warnings and give time scales in the Chrome DevTools Console when usage is detected on the page.
- Wait, monitor, and then remove the feature as usage drops.

You can find a list of all deprecated features on chromestatus.com using the deprecated filter and removed features by applying the removed filter. We will also try to summarize some of the changes, reasoning, and migration paths in these posts.

Last updated July 2, 2018.