API Deprecations and Removals in Chrome 50

By Paul Kinlan
Paul is a Developer Advocate

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.

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- They are superseded by newer APIs.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.

In Chrome 50 (Estimated beta date: March 10 to 17) there are a number of changes to Chrome. This list is subject to change at any time.

AppCache deprecated on insecure contexts

TL;DR: To hinder cross-site scripting, we're deprecating AppCache on insecure origins. We expect that in Chrome 52 it will only work on origins serving content over HTTPS.

Intent to Remove | Chromestatus Tracker | Chromium Bug

AppCache is a feature that allows offline and persistent access to an origin, which is a powerful privilege escalation for an cross-site scripting attack. As part of a larger effort to remove powerful features on insecure origins.

Chrome is removing this attack vector by only allowing it over HTTPS. We're deprecating HTTP support in Chrome 50 and expect to remove it entirely in Chrome 52.

Document.defaultCharset is removed

TL;DR: `document.defaultCharset` has been removed to improve spec compliance.

Intent to Remove | Chromestatus Tracker | CRBug Issue

The `document.defaultCharset`, deprecated in Chrome 49, is a read-only property that returns the default character encoding of the user's system based on their regional settings. It's not been found to be useful to maintain this value because of the way that browsers use the character encoding information in the HTTP Response or in the meta tag embedded in the page.

Instead, use `document.characterSet` to get the first value specified in the HTTP header. If that is not present then you will get the value specified in the `charset` attribute of the `<meta>` element (for example, `<meta charset="utf-8">`). Finally if none of those are available the `document.characterSet` will be the user's system setting.

You can read more discussion of the reasoning not to spec this out in this github issue

Subresource attribute removed from link element

TL;DR: Remove support for the `subresource` value for the `rel` attribute of `HTMLLinkElement`.

Intent to Remove | Chromestatus Tracker | Chromium Bug

The intent of the `subresource` attribute on <link> was to prefetch a resource during a browser's idle time. After a browser downloaded a page, it could then pre-download resources such as other pages so that when they were requested by users, they could simply be retrieved from the browser cache.

The `subresource` attribute suffered from a number of problems. First, it never worked as intended. Referenced resources were downloaded with low priority. The attribute was never implemented on any browser other than Chrome. The Chrome implementation had a bug that caused resources to be downloaded twice.

Developers looking to improve the user experience through preloading of content have a number of options, the most customizable of which is to build a service worker to take advantage of precaching and the Caches API. Additional solutions include other values for the `rel` attribute including `preconnect`, `prefetch`, `preload`, `prerender`. Some of these options are experimental and may not be widely supported.

Remove insecure TLS version fallback

TL;DR: Remove a mechanism for forcing servers to return data using less- or non-secure versions of TLS.

Intent to Remove | Chromestatus Tracker | Chromium Bug

Transport layer security (TLS) supports a mechanism for negotiating versions, allowing for the introduction of new TLS versions without breaking compatibility. Some servers implemented this in such a way that browsers were required to use insecure endpoints as a fallback. Because of this, attackers could force *any* website, not just those that are incorrectly configured, to negotiate for weaker versions of TLS.

Affected sites will fail to connect with `ERR_SSL_FALLBACK_BEYOND_MINIMUM_VERSION`. Administrators should ensure their server software is up-to-date. If still unresolved, contact the server software vendor to see if a fix is available.

Remove KeyboardEvent.prototype.keyLocation

TL;DR: Remove an unneeded alias for the `Keyboard.prototype.location` attribute.

Intent to Remove | Chromestatus Tracker | Chromium Bug

This attribute is simply an alias to the `Keyboard.prototype.location` attribute, which allows disambiguation between keys that are located multiple places on a keyboard. For example, both attributes allow developers to distinguish between the two `Enter` keys on an extended keyboard.

Error and success handlers required in RTCPeerConnection methods

TL;DR: The WebRTC RTCPeerConnection methods `createOffer()` and `createAnswer()` now require an error handler as well as a success handler. Previously it had been possible to call these methods with only a success handler. That usage is deprecated.

Intent to Remove | Chromestatus Tracker | Chromium Bug

In Chrome 49we added a warning if you call `setLocalDescription()` or `setRemoteDescription()` without supplying an error handler. The error handler argument is mandatory as of Chrome 50.

This is part of clearing the way for introducing promises on these methods, as required by the WebRTC spec.

Here's an example from the WebRTC RTCPeerConnection demo (main.js, line 126):

```
function onCreateOfferSuccess(desc) {
  pc1.setLocalDescription(desc, function() {
     onSetLocalSuccess(pc1);
  }, onSetSessionDescriptionError);
  pc2.setRemoteDescription(desc, function() {
    onSetRemoteSuccess(pc2);
  }, onSetSessionDescriptionError);
  pc2.createAnswer(onCreateAnswerSuccess, onCreateSessionDescriptionError);
}
```

Note that both `setLocalDescription()` and `setRemoteDescription()` have an error handler. Older browsers expecting only a success handler will simply ignore the error handler argument if it's present; calling this code in an older browser will not cause an exception.

In general, for production WebRTC applications we recommend that you use `adapter.js`, a shim, maintained by the WebRTC project, to insulate apps from spec changes and prefix differences.

The XMLHttpRequestProgressEvent is no longer supported

TL;DR: The `XMLHttpRequestProgressEvent` interface will be removed, together with the attributes `position` and `totalSize`.

Intent to Remove | Chromestatus Tracker | Chromium Bug

This event existed to support the Gecko compatibility properties `position` and `totalSize`. Support for all three was dropped in Mozilla 22 and the functionality has long been superceded by the `ProgressEvent`.

```
var progressBar = document.getElementById("p"),
     client = new XMLHttpRequest()
 client.open("GET", "magical-unicorns")
 client.onprogress = function(pe) {
   if(pe.lengthComputable) {
     progressBar.max = pe.total
     progressBar.value = pe.loaded
   }
 }
```

Remove prefixed Encrypted Media Extensions

TL;DR: Prefixed encrypted media extensions have been removed in favor of a spec-based, unprefixed replacement.

Intent to Remove | Chromestatus Tracker | Chromium Bug

In Chrome 42, we shipped a specification-based, unprefixed version of encrypted media extensions. This API is used to discover, select, and interact with Digital Rights Management systems for use with `HTMLMediaElement`.

That was nearly a year ago. And since the unprefixed version has more capabilities than the prefixed version, it's time to remove the prefixed version of the API.

Remove support for SVGElement.offset properties

TL;DR: Offset properties for SVGElement have been dropped in favor of the more widely-supported properties on `HTMLElement`.

Intent to Remove | Chromestatus Tracker | Chromium Bug

Offset properties have long been supported by both `HTMLElement` and `SVGElement`; however, Gecko and Edge only support them on `HTMLElement`. To improve consistency between browsers these properties were deprecated in Chrome 48 and are now being removed.

Though equivalent properties are part of `HTMLElement`, developers looking for an alternative can also use `getBoundingClientRect()`

Feedback

Was this page helpful?

YES NO

Great! Thank you for the feedback.

Sorry to hear that. Please open an issue and tell us how we can improve.

Last updated July 24, 2018.