API Deprecations and Removals in Chrome 54

By Joseph Medley
Technical Writer

By Paul Kinlan
Paul is a Developer Advocate

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 54, which is in beta as of September 15. This list is subject to change at any time.

Disable navigations in the unload handler

TL;DR: All cross-origin navigations will be disallowed in `window.onunload` event handlers to bring Chrome inline with the HTML spec as well as Firefox and Safari.

Intent to Remove | Chromestatus Tracker | Chromium Bug

Previous versions of Chrome allowed cross-origin navigation to be interrupted inside `window.onunload`. by setting `window.location.href = '#fragment'`. According to the HTML spec, only in-page navigations are allowed in the unload handlers, and in previous versions of Chrome other methods of navigating were blocked as required by the spec. Starting in Chrome 54, such navigations will be disallowed to bring us in line with the spec as well as Firefox and Safari.

HTTP/0.9 deprecated

TL;DR: HTTP/0.9 is deprecated. Developers should move to a later version, preferably HTTP/2.

Intent to Remove | Chromestatus Tracker | Chromium Bug

HTTP/0.9 is the predecessor to HTTP/1.x. It lacks many features of its successors. A particular concern for the modern web is its lack of response headers. Without them, there's no way to verify that an HTTP/0.9 response is really an HTTP/0.9 response. This can cause several problems. Examples include, among other problems:

- Clients that treat certain error responses as valid HTTP/0.9 responses.
- Servers that fail to close the request socket causing clients to treat responses as a hanging GET which either stays alive eternally or until a user navigates from a page that made the request.
- Servers that are unable to indicate to the browser that a request failed, which can cause problems with caching heuristics.

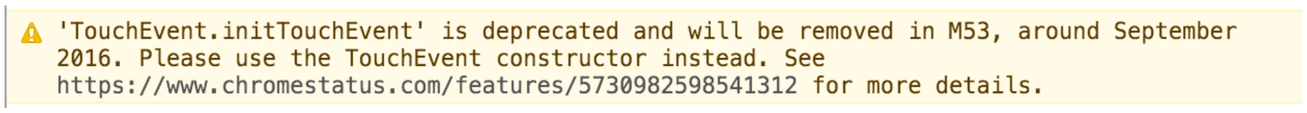
The only foolproof way to fix issues with HTTP/0.9 is to remove support altogether. Which is why support for HTTP/0.9 is removed in Chrome 54.

Use of `initTouchEvent` is removed

TL;DR: `initTouchEvent` has been deprecated in favor of the `TouchEvent` `constructor` to improve spec compliance and will be removed altogether in Chrome 54.

Intent to Remove | Chromestatus Tracker | Chromium Bug

For a long time developers have been able to create synthetic touch events in Chrome using the `initTouchEvent` API. These are frequently used to simulate Touch Events either for testing or automating some UIs in your site. Since Chrome 49, this deprecated API has displayed the following warning .



```
'TouchEvent.initTouchEvent' is deprecated and will be removed in M53, around September
2016. Please use the TouchEvent constructor instead. See
https://www.chromestatus.com/features/5730982598541312 for more details.
```

`TouchEvent.initTouchEvent` is deprecated and will be removed in M53, around September 2016. Please use the `TouchEvent` constructor instead. See https://www.chromestatus.com/features/5730982598541312 for more details.

Aside from not being in the Touch Events spec, there are a number of reasons why this change is good. The Chrome implementation of `initTouchEvent` was not compatible at all with Safari's `initTouchEvent` API and was different to Firefox on Android's. And finally, the `TouchEvent` constructor is a lot easier to use.

For these reasons we decided to follow the spec rather than maintain an API that is neither specced nor compatible with the only other implementation. Developers needing an alternative should use the `TouchEvent` constructor.

Because the iOS and Android/Chrome implementations of the `initTouchEvent` API were so wildly different, sites would often have code along the lines of (frequently forgetting Firefox)

```
var event = document.createEvent('TouchEvent');

if(ua === 'Android') {
  event.initTouchEvent(touchItem, touchItem, touchItem, "touchstart", window,
    300, 300, 200, 200, false, false, false, false);
} else {
  event.initTouchEvent("touchstart", false, false, window, 0, 300, 300, 200,
    200, false, false, false, false, touches, targetTouches, changedTouches, 0, 0
}

document.body.dispatchEvent(touchEvent);
```

This is bad because it looks for "Android" in the User-Agent and Chrome on Android will match and hit this deprecation. It can't be removed just yet though because there will be other WebKit and older Blink based browsers on Android for a while that you will still need to support the older API.

To correctly handle **`TouchEvents`** on the web you should change your code to support Firefox, IE Edge, and Chrome by checking for the existence of **`TouchEvent`** on the **`window`** object and if it has a positive "length" (indicating it's a constructor that takes an argument) you should use that.

```
if('TouchEvent' in window && TouchEvent.length > 0) {
  var touch = new Touch({
    identifier: 42,
    target: document.body,
    clientX: 200,
    clientY: 200,
    screenX: 300,
    screenY: 300,
    pageX: 200,
    pageY: 200,
    radiusX: 5,
    radiusY: 5
  });

  event = new TouchEvent("touchstart", {
    cancelable: true,
    bubbles: true,
    touches: [touch],
    targetTouches: [touch],
    changedTouches: [touch]
  });
}
else {
  event = document.createEvent('TouchEvent');
  if(ua === 'Android') {
    event.initTouchEvent(touchItem, touchItem, touchItem, "touchstart", window,
      300, 300, 200, 200, false, false, false, false);
  } else {
    event.initTouchEvent("touchstart", false, false, window, 0, 300, 300, 200,
      200, false, false, false, false, touches, targetTouches,
      changedTouches, 0, 0);
  }
}

document.body.dispatchEvent(touchEvent);
```

KeyboardEvent.keyIdentifier attribute removed

TL;DR: The little-supported `keyboardEvent.keyIdentifier` property is being removed in favor the standards-based `KeyboardEvent.key` property.

Intent to Remove | Chromestatus Tracker | Chromium Bug

The `keyboardEvent.keyIdentifier` attribute was briefly part of a W3C specification in 2009 and 2010. However, it was only ever implemented in WebKit.

Developers needing to replace this attribute can use either the standards-based `KeyboardEvent.key` property or the `KeyboardEvent.code` property (as described in an article we did last spring). The former has the widest implementation base, being supported on all major desktop browsers except Safari. The later is currently supported on Chrome, Firefox, and Opera. Removing this feature is intended to drive adoption of `KeyboardEvent.key` property. There is no word from Apple as to whether will support this; however the also deprecated (but not yet removed from Chrome) `KeyboardEvent.keyCode` and `KeyboardEvent.charCode` properties are still available on Safari.

Remove MediaStream ended event and attribute and onended attribute

TL;DR: The `ended` event and attribute and the `onended` event handler are being removed because they have been removed from the Media Capture and Streams spec.

Intent to Remove | Chromestatus Tracker | Chromium Bug

Neither the `ended` event, nor the `onended` event handler have been part of the WebRTC spec for about three years. Developers wanting to watch events should use `MediaStreamTracks` instead of `MediaStreams`.

Deprecate SVGSVGElement.viewPort

The implementation has not worked in Chrome since 2012. The attribute is not present at all in other browsers and it has been removed from the specification. For these reasons the property is being deprecated. Removal is anticipated in Chrome 55.

Intent to Remove | Chromestatus Tracker | Chromium Bug

Deprecate SVGViewElement.viewTarget

The `SVGViewElement.viewTarget` attribute is not part of the SVG2.0 specification and it's usage is small or nonexistent. This attribute is deprecated in Chrome 54. Removal is anticipated in Chrome 56.

Intent to Remove | Chromestatus Tracker | Chromium Bug

Remove SVGZoomEvent

The `SVGZoomEvent` is not part of the SVG2.0 specification and does not function in Chromium. Despite that it's still feature detectable, leading to potential confusion by developers. It will be removed.

Intent to Remove | Chromestatus Tracker | Chromium Bug

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.
