Chrome 64 to deprecate the chrome.loadTimes() API

To see all deprecations and removals for this and previous versions of Chrome, visit the deprecations page.

By Philip Walton
Engineer at Google working on the Web Platform

`chrome.loadTimes()` is a non-standard API that exposes loading metrics and network information to developers in order to help them better understand their site's performance in the real world.

Since this API was implemented in 2009, all of the useful information it reports can be found in standardized APIs such as:

- Navigation Timing 2
- Paint Timing
- The `nextHopProtocol` addition to Navigation Timing 2 and Resource Timing 2.

These standardized APIs are being implemented by multiple browser vendors. As a result, `chrome.loadTimes()` is being deprecated in Chrome 64.

The deprecated API

The `chrome.loadTimes()` function returns a single object containing all of its loading and network information. For example, the following object is the result of calling `chrome.loadTimes()` on www.google.com:

```
{
  "requestTime": 1513186741.847,
  "startLoadTime": 1513186741.847,
  "commitLoadTime": 1513186742.637,
  "finishDocumentLoadTime": 1513186742.842,
  "finishLoadTime": 1513186743.582,
  "firstPaintTime": 1513186742.829,
  "firstPaintAfterLoadTime": 0,
  "navigationType": "Reload",
  "wasFetchedViaSpdy": true,
  "wasNpnNegotiated": true,
  "npnNegotiatedProtocol": "h2",
  "wasAlternateProtocolAvailable": false,
  "connectionInfo": "h2"
}
```

Standardized replacements

You can now find each of the above values using standardized APIs. The following table matches each value to its standardized API, and the sections below show code examples of how to get each value in the old API with modern equivalents.

`chrome.loadTimes()` feature	Standardized API replacement
`requestTime`	Navigation Timing 2
`startLoadTime`	Navigation Timing 2
`commitLoadTime`	Navigation Timing 2
`finishDocumentLoadTime`	Navigation Timing 2
`finishLoadTime`	Navigation Timing 2
`firstPaintTime`	Paint Timing
`firstPaintAfterLoadTime`	N/A
`navigationType`	Navigation Timing 2
`wasFetchedViaSpdy`	Navigation Timing 2
`wasNpnNegotiated`	Navigation Timing 2
`npnNegotiatedProtocol`	Navigation Timing 2
`wasAlternateProtocolAvailable`	N/A
`connectionInfo`	Navigation Timing 2

Note: some of the features of the original API were never implemented or only apply to deprecated features. These are marked "N/A" in the table above.

The code examples below return equivalent values to those returned by `chrome.loadTimes()`. However, for new code these code examples are not recommended. The reason is `chrome.loadTimes()` gives times values in epoch time in seconds whereas new performance APIs typically report values in milliseconds relative to a page's time origin, which tends to be more useful for performance analysis.

Several of the examples also favor Performance Timeline 2 APIs (e.g. `performance.getEntriesByType()`) but provide fallbacks for the older Navigation Timing 1 API as it has wider browser support. Going forward, Performance Timeline APIs are preferred and are typically reported with higher precision.

requestTime

```
function requestTime() {
  // If the browser supports the Navigation Timing 2 and HR Time APIs, use
  // them, otherwise fall back to the Navigation Timing 1 API.
  if (window.PerformanceNavigationTiming && performance.timeOrigin) {
    const ntEntry = performance.getEntriesByType('navigation')[0];
    return (ntEntry.startTime + performance.timeOrigin) / 1000;
  } else {
    return performance.timing.navigationStart / 1000;
  }
}
```

startLoadTime

```
function startLoadTime() {
  // If the browser supports the Navigation Timing 2 and HR Time APIs, use
  // them, otherwise fall back to the Navigation Timing 1 API.
  if (window.PerformanceNavigationTiming && performance.timeOrigin) {
    const ntEntry = performance.getEntriesByType('navigation')[0];
    return (ntEntry.startTime + performance.timeOrigin) / 1000;
  } else {
    return performance.timing.navigationStart / 1000;
  }
}
```

commitLoadTime

```
function commitLoadTime() {
  // If the browser supports the Navigation Timing 2 and HR Time APIs, use
  // them, otherwise fall back to the Navigation Timing 1 API.
  if (window.PerformanceNavigationTiming && performance.timeOrigin) {
    const ntEntry = performance.getEntriesByType('navigation')[0];
    return (ntEntry.responseStart + performance.timeOrigin) / 1000;
  } else {
    return performance.timing.responseStart / 1000;
  }
}
```

finishDocumentLoadTime

```
function finishDocumentLoadTime() {
  // If the browser supports the Navigation Timing 2 and HR Time APIs, use
  // them, otherwise fall back to the Navigation Timing 1 API.
  if (window.PerformanceNavigationTiming && performance.timeOrigin) {
    const ntEntry = performance.getEntriesByType('navigation')[0];
    return (ntEntry.domContentLoadedEventEnd + performance.timeOrigin) / 1000;
  } else {
    return performance.timing.domContentLoadedEventEnd / 1000;
  }
}
```

finishLoadTime

```
function finishLoadTime() {
  // If the browser supports the Navigation Timing 2 and HR Time APIs, use
  // them, otherwise fall back to the Navigation Timing 1 API.
  if (window.PerformanceNavigationTiming && performance.timeOrigin) {
    const ntEntry = performance.getEntriesByType('navigation')[0];
    return (ntEntry.loadEventEnd + performance.timeOrigin) / 1000;
  } else {
    return performance.timing.loadEventEnd / 1000;
  }
}
```

firstPaintTime

```
function firstPaintTime() {
  if (window.PerformancePaintTiming) {
    const fpEntry = performance.getEntriesByType('paint')[0];
    return (fpEntry.startTime + performance.timeOrigin) / 1000;
  }
}
```

firstPaintAfterLoadTime

```
function firstPaintTimeAfterLoad() {
  // This was never actually implemented and always returns 0.
  return 0;
}
```

navigationType

```
function navigationType() {
  if (window.PerformanceNavigationTiming) {
    const ntEntry = performance.getEntriesByType('navigation')[0];
    return ntEntry.type;
  }
}
```

wasFetchedViaSpdy

```
function wasFetchedViaSpdy() {
  // SPDY is deprecated in favor of HTTP/2, but this implementation returns
  // true for HTTP/2 or HTTP2+QUIC/39 as well.
  if (window.PerformanceNavigationTiming) {
    const ntEntry = performance.getEntriesByType('navigation')[0];
    return ['h2', 'hq'].includes(ntEntry.nextHopProtocol);
  }
}
```

wasNpnNegotiated

```
function wasNpnNegotiated() {
  // NPN is deprecated in favor of ALPN, but this implementation returns true
  // for HTTP/2 or HTTP2+QUIC/39 requests negotiated via ALPN.
  if (window.PerformanceNavigationTiming) {
    const ntEntry = performance.getEntriesByType('navigation')[0];
    return ['h2', 'hq'].includes(ntEntry.nextHopProtocol);
  }
}
```

npnNegotiatedProtocol

```
function npnNegotiatedProtocol() {
  // NPN is deprecated in favor of ALPN, but this implementation returns the
  // HTTP/2 or HTTP2+QUIC/39 requests negotiated via ALPN.
  if (window.PerformanceNavigationTiming) {
    const ntEntry = performance.getEntriesByType('navigation')[0];
    return ['h2', 'hq'].includes(ntEntry.nextHopProtocol) ?
        ntEntry.nextHopProtocol : 'unknown';
  }
}
```

`wasAlternateProtocolAvailable`

```
function wasAlternateProtocolAvailable() {
  // The Alternate-Protocol header is deprecated in favor of Alt-Svc
  // (https://www.mnot.net/blog/2016/03/09/alt-svc), so technically this
  // should always return false.
  return false;
}
```

`connectionInfo`

```
function connectionInfo() {
  if (window.PerformanceNavigationTiming) {
    const ntEntry = performance.getEntriesByType('navigation')[0];
    return ntEntry.nextHopProtocol;
  }
}
```

Removal plan

The `chrome.loadTimes()` API will be deprecated in Chrome 64 and is targeted to be removed in late 2018. Developers should migrate their code as soon as possible to avoid any loss in data.

Intent to Deprecate | Chromestatus Tracker | Chromium Bug

Last updated July 2, 2018.