

Memory Info API

(public doc)

igrigorik@chromium.org

Web applications need a mechanism to track memory usage of the document, and each associated worker, to optimize performance and detect memory leaks and regressions within their code.

[Use cases](#)

[Where we are today](#)

[API requirements](#)

[Questions](#)

[Strawman proposal](#)

[Motivation](#)

Use cases

“We (Inbox) track jsheap memory in prod. The problem is that only captures memory for the main tab. In Inbox we have a worker that manages the bulk of our data model. For example most of our network access happens in the worker, and lots of proto manipulation happens there. So there is a huge opportunity for us to go off the tracks and chew through memory in the worker.” - Garrick Toubassi, Google

“Docs needs memory monitoring for the following cases:

- *Shared worker where the bulk background syncer lives.*
- *Calculation engine lives in a traditional web worker.” - Steven Saviano, Google*

“When we started shipping 128 MB devices, I pointed out that not only does a dev have to be mindful of their peak JS Heap Size, but things like bitmaps, audio, video, and probably even complexity of the DOM affect total memory usage.” - [Nick Desaulniers, Mozilla](#)

“We’ve had a lot of people implement TV’s that run youtube with subpar performance that was making us look bad so we needed some base line for memory usage.” - YouTube

“When developing native web applications in a web-based OS like Firefox OS, having access to more native memory measurements such as PSS, RSS, and USS are important to developers as well.” Eli Perelman, Mozilla

Note: add yours here.

Where we are today

- **Chrome**
 - Exposes a rate-limited and quantized view via `performance.memory`.
 - https://bugs.webkit.org/show_bug.cgi?id=80444
 - <http://trac.webkit.org/changeset/123856>
 - [Intent to ship in Worker](#) blocked due to lack of standard/spec.
 - <https://code.google.com/p/chromium/issues/detail?id=326370>
- **Mozilla**
 - Intent to implement is blocked due to lack of standard/spec.
 - https://bugzilla.mozilla.org/show_bug.cgi?id=1124223
- **IE**
 - Positive
- **Safari**
 - No signals

API requirements

1. Standard API available for main-frame and worker processes.
2. Low-overhead monitoring via [Performance Observer](#): no polling; UA is responsible for emitting records when new data is available; interop with other monitoring APIs, such as Frame Timing, User Timing, etc.
 - a. **Optional?** Method to query current memory usage at a point in time.
 - i. e.g. `performance.memory`
3. Memory usage metrics:
 - a. JS heap size. estimate of size of live objects on the JS heap.
 - b. DOM size, bitmaps, audio video? These are important and have known to cause many problems in the past - e.g. implementing a performant infinite scroll requires active management and recycling of memory used by images, canvas, etc. Further this is critical for lower-end devices which have limited memory - e.g. Android One hardware, etc.
 - c. <https://code.google.com/p/chromium/issues/detail?id=484664>
4. Consider security and privacy implications - e.g. side channel attacks, information about cross origin resources, etc.

Questions

1. **What data can we expose?**
 - a. JS heap is straightforward and useful.

- b. Developers also want memory usage of other components: DOM size, image, video, etc. What can we expose in a cross-UA manner? At a minimum, can we define a “total”, for which JS heap is a subset?
- c. How do Shared Array Buffers get reported?
 - i. <https://groups.google.com/a/chromium.org/forum/#!msg/blink-dev/d-0ibJwCS24/E7oB7slaLccJ>
- d. How are iframes accounted for?

2. Is “memory used” the right thing to surface?

- a. It’s hard to reason about used memory across different architectures (both OS and UA), and precise accounting is both expensive and opens security + privacy questions.
 - i. https://groups.google.com/a/chromium.org/forum/#!msg/blink-dev/_MOh3lZ8zxo/DilhJlInH3gJ
 - 1. “Plus, perhaps prior to killing the page we should dispatch an event to the page to let it know when memory is getting tight. Maybe a web developer could respond to that by clearing their own caches, etc.”
 - ii. [OOM event \(Mozilla\)](#)
 - iii. [Low Memory Event \(whatwg\)](#)
 - 1. converges on simple event with no details; as a hint.
- b. Perhaps a better signal is a simpler “**you’re under memory pressure**”, e.g...
 - i. A callback / notification that fires when the UA is under pressure
 - ii. A callback / notification based on application defined threshold
 - 1. Eliminates polling, allows you to invoke logic to cleanup, etc.
 - iii. Chrome implementation notes
 - 1. <https://docs.google.com/document/d/1Zz6ksh6l0Q7iW4toIL3gKSAIkjQkPJPAqUve8OJf4hU/edit#>

3. Privacy and security

- a. Current Chrome implementation buckets memory use, and also rate limits it to “once every 20 minutes”, which restricts its use to long-running pages and is too restrictive to be useful for dynamic use cases like infinite scroll, working with canvas, Service Workers, etc.
 - b. Can we define some bucketing logic and trigger new memory events when a threshold is exceeded, omitting the time-based rate limits?
 - c. Any other privacy / security issues to look out for?
-

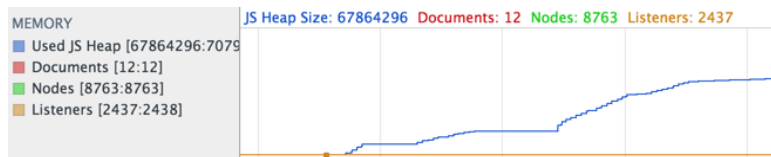
~very rough sketch

```
interface MemoryEntry : PerformanceEntry {
    readonly attribute unsigned long heapSize;      // used heap size
    readonly attribute unsigned long nodeCount;     // DOM-node count
    readonly attribute unsigned long listenerCount; // ...
    readonly attribute unsigned long documentCount; // ...
    readonly attribute boolean underPressure;       // ...
    serializer = {inherit, attribute};
};

var observer = new PerformanceObserver(function(list) {
    var observedMemoryEntries = list.getEntries();
    observedMemoryEntries.forEach(function(e) {
        // e: {heap: ..., ..., total: ...}
    });
});

observer.observe({eventTypes: ['memory']});
```

- **heapSize**: number of bytes consumed by the current navigation or worker context's
 - the user agent MUST quantize the value (function TBD)
 - the user agent SHOULD deliver new event whenever new quantized value differs from the one that was reported previously
 - the quantized value MAY include
 - the number of bytes used by allocated JavaScript objects
 - the number of bytes used by the DOM (e.g. [IE reports retained size](#))
 - the number of bytes used by images, video, and other objects
 - OR, alternatively...



- Can (should?) we report doc / node / listener counts?
 - How would we roll this up or quantize? =/
 - Counting DOM elements can be done via JS...
- **underPressure**: boolean signal indicating that the user agent is under memory pressure.
 - The application should, if possible, release memory and/or adapt its logic - e.g. disable expensive animations, release held buffers, etc.

- Implementation: there is no cross-platform signal / implementation for this today. On Android the app gets a notification when “under pressure” (i.e. you might get killed), but there is no reverse signal.
 - You can’t query this synchronously, this is a notification
 - As a first approximation: broadcast to all renderers

Motivation

- Detailed memory reporting is expensive and very hard due to architectural and implementation reasons. As such, the intent here is to provide a high-level metric that applications can use as an approximation, not as an exact measure:
 - The reported amounts may differ based on architecture
 - The reported amounts may differ based on user agent
- Quantized values are reported due to security reasons (obfuscate impact of a single resource, etc), and to make clear that the advertised value **is an estimate**.
- “Under Pressure” flag can be used by application to adapt its logic - e.g. memory use can be high, but if we’re not under pressure that’s OK.. higher memory use can yield better performance.

Very, very rough first draft:

<https://rawgit.com/igrigorik/cd4e152d8f01268e5363/raw/d4504a9f7fb66bc59dd22c84ba891d2abf5f2482/index.html>