Lessons Learned from the Memory Roadshow

•••

Michael Hablich, hablich@chromium.org
Hannes Payer, hpayer@chromium.org
V8

Why memory matters to Chrome …

30 % of all renderer crashes on **Windows** are because of out of memory situations

… so, we can make Chrome crash 30 % less often? Sweet!

What’s the reason for all these OOM crashes?

Is it Chrome?

Aw, Snap!

Is it the Web?

Hypothesis

The majority of OOMs are caused by webpages?*

*proof pending :-P

How are we going to do it?

Let's ask the web developers!

- Why do web pages use so much memory?
- How do web developers reason about memory?
- How can Chromium improve?
- Where are all the leaks?

Conduct a roadshow with popular web applications.

31
facebook

Where do I start?

How much memory
do I really use?

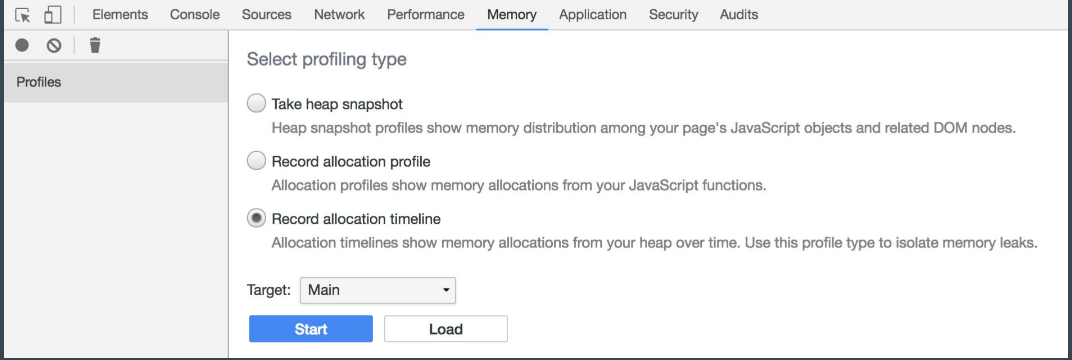
Why do I use so much memory?

How slim
should a
webpage be?

How do I convince my manager
to work on memory reduction?

Where do I start?

- DevTools is the start
- There is no clear goal for the web developer to accomplish



Select profiling type

○ Take heap snapshot

Heap snapshot profiles show memory distribution among your page's JavaScript objects and related DOM nodes.

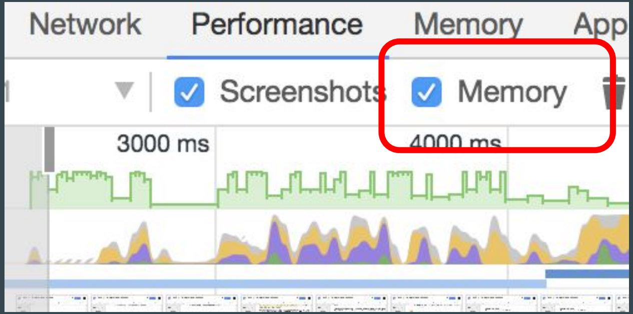
○ Record allocation profile

Allocation profiles show memory allocations from your JavaScript functions.

● Record allocation timeline

Allocation timelines show memory allocations from your heap over time. Use this profile type to isolate memory leaks.

Target: Main



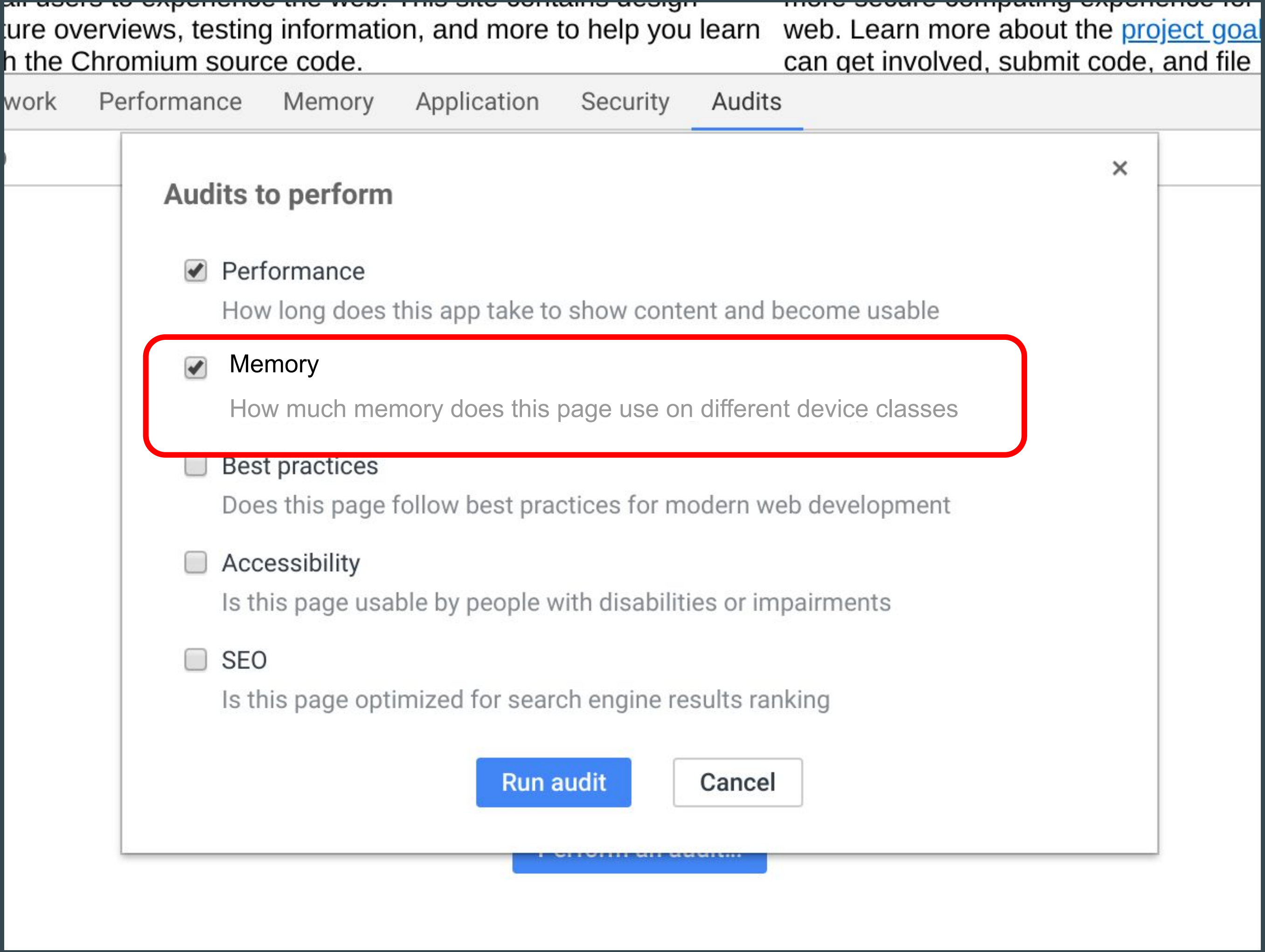
☑ Screenshots ☑ Memory

Where do I start?

- DevTools is the start
- There is no clear goal for the web developer to accomplish

What we can do:

- Engage and provide guidance to web developers earlier in the process
- Apply methodology used in loading project:
    - Surface memory consumption in Lighthouse


ure overviews, testing information, and more to help you learn
h the Chromium source code.
web. Learn more about the project goal
can get involved, submit code, and file
work
Performance
Memory
Application
Security
Audits
×
Run audit
Cancel
Perform an audit...

Audits to perform

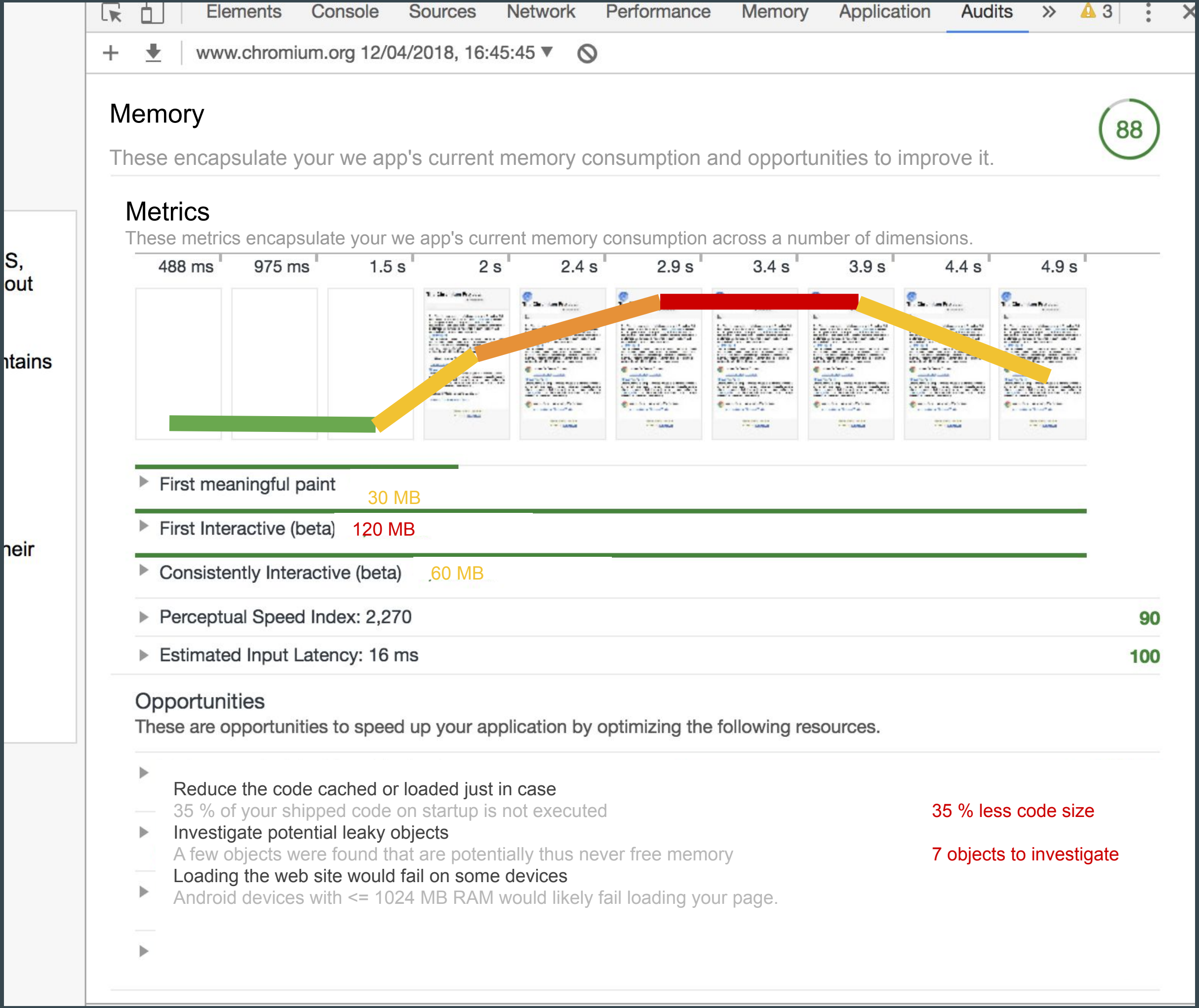
☑ Performance
How long does this app take to show content and become usable

☑ Memory
How much memory does this page use on different device classes

□ Best practices
Does this page follow best practices for modern web development

□ Accessibility
Is this page usable by people with disabilities or impairments

□ SEO
Is this page optimized for search engine results ranking



Memory

88

These encapsulate your we app's current memory consumption and opportunities to improve it.

Metrics

These metrics encapsulate your we app's current memory consumption across a number of dimensions.

▸ First meaningful paint 30 MB

▸ First Interactive (beta) 120 MB

▸ Consistently Interactive (beta) 60 MB

▸ Perceptual Speed Index: 2,270 90

▸ Estimated Input Latency: 16 ms 100

Opportunities

These are opportunities to speed up your application by optimizing the following resources.

▸ Reduce the code cached or loaded just in case
35 % of your shipped code on startup is not executed — 35 % less code size

▸ Investigate potential leaky objects
A few objects were found that are potentially thus never free memory — 7 objects to investigate

▸ Loading the web site would fail on some devices
Android devices with <= 1024 MB RAM would likely fail loading your page.

Where do I start?

- DevTools is the start
- There is no clear goal for the web developer to accomplish

What we can do:

- Engage and provide guidance to web developers earlier in the process
- Apply methodology used in loading speed project:
    - Surface memory consumption in Lighthouse

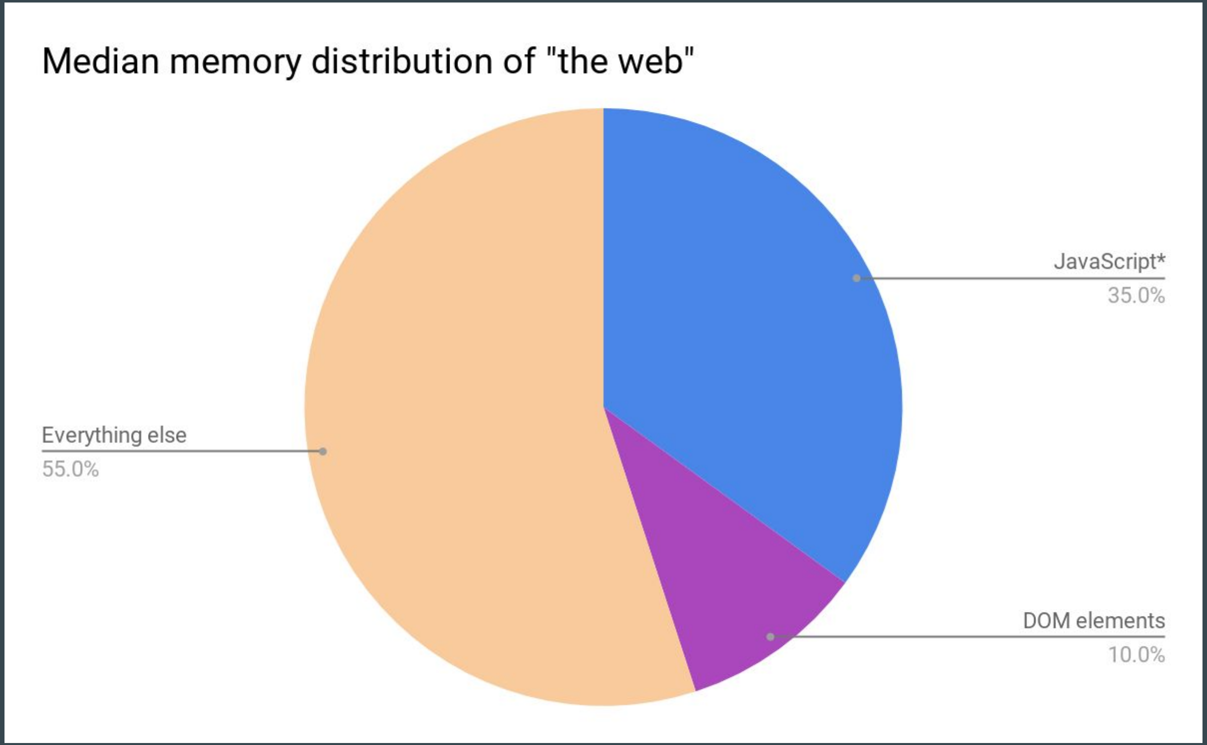
Where do I start?

- DevTools is the start
- There is no clear goal for the web developer to accomplish

What we can do:

- Engage and provide guidance to web developers earlier in the process
- Apply methodology used in loading speed project:
    - Surface memory consumption in Lighthouse
    - Add memory consumption to Chrome UX report

How much memory do I really use?

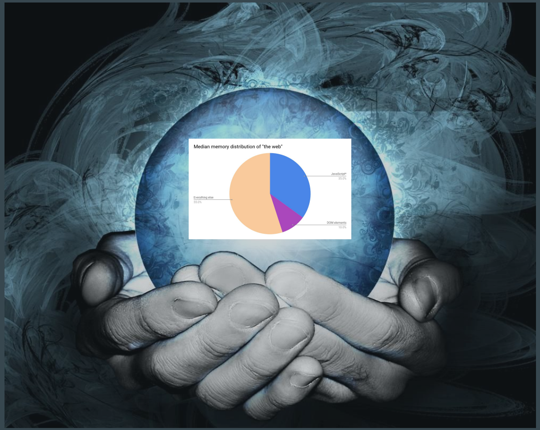


How much memory do I really use?

- performance.memory API only shows blurred V8 heap sizes (blue)
- DevTools profiling only shows you the JavaScript related memory (blue+purple)
- about:tracing (+some magic) has the full picture

What we can do:

- Make performance.memory recognize the whole memory
- Surface **full** memory consumption in DevTools



performance.memory

Today:

- Displays only memory allocated on the V8 heap
- Updates every 20 minutes
- Only for the main thread

performance.memory

Today:

- Displays only memory allocated on the V8 heap
- Updates every 20 minutes
- Only for the main thread

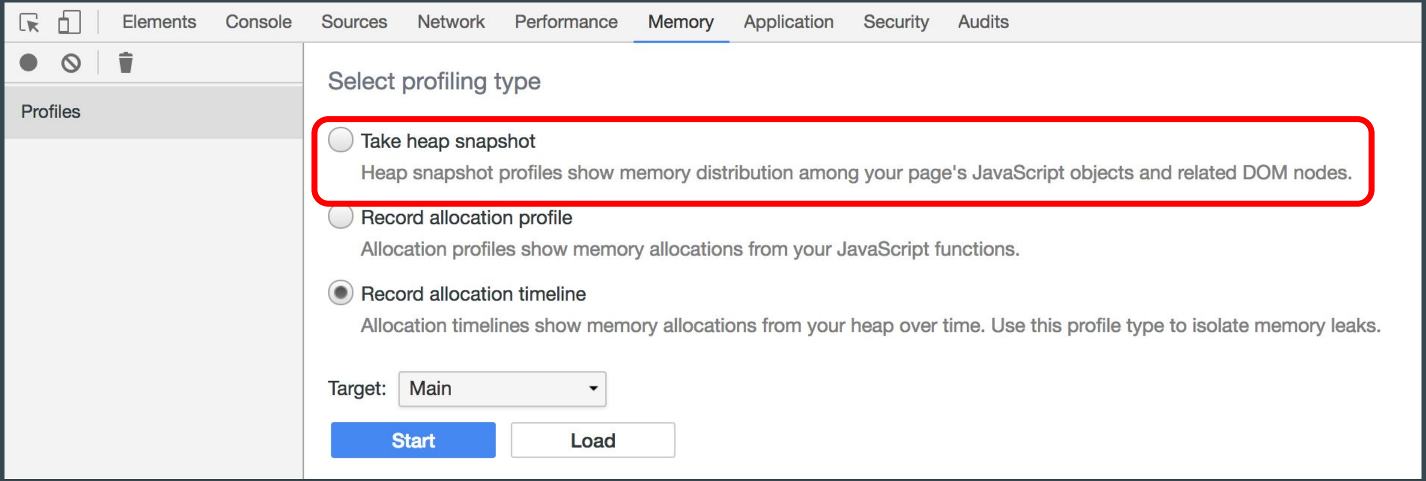
The (maybe) future:

- Precise
- Process-wide private memory
- Information about shared tabs and workers
- Expose all mem-infra memory metrics
- Allow it to be called from main thread and workers

https://github.com/erikchen/performance-memory

Why do I use so much memory?

- DevTools profiling only shows you the JavaScript related memory



Select profiling type

○ Take heap snapshot

Heap snapshot profiles show memory distribution among your page's JavaScript objects and related DOM nodes.

○ Record allocation profile

Allocation profiles show memory allocations from your JavaScript functions.

● Record allocation timeline

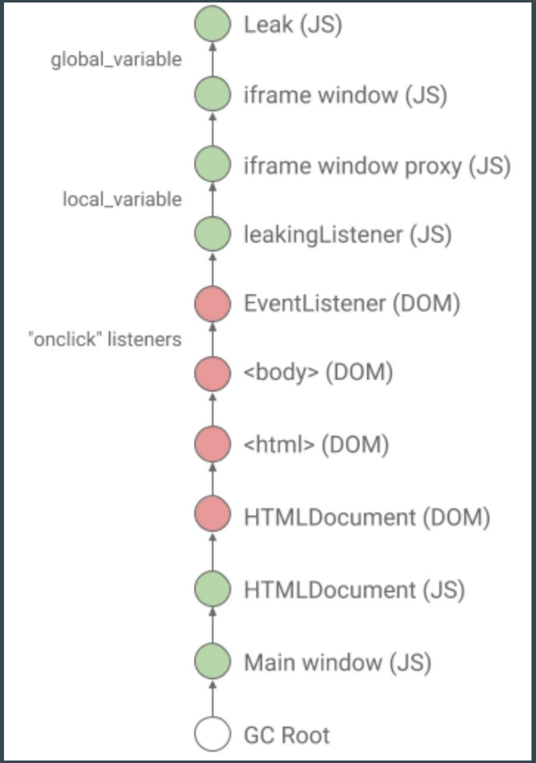
Allocation timelines show memory allocations from your heap over time. Use this profile type to isolate memory leaks.

Target: Main

Fix the retaining path in DevTools

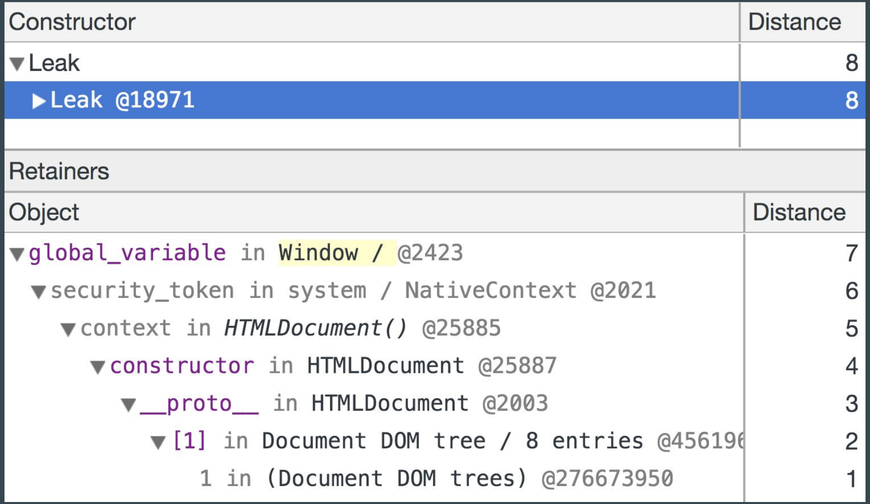
```
// Main window:
const iframe = document.createElement('iframe');
iframe.src = 'iframe.html';
document.body.appendChild(iframe);
iframe.addEventListener('load', function() {
  const local_variable = iframe.contentWindow;
  function leakingListener() {
    // Do something with `local_variable`.
    if (local_variable) {}
  }
  document.body.addEventListener('my-debug-event', leakingListener);
  document.body.removeChild(iframe);
  // BUG: forgot to unregister `leakingListener`.
});
```

```
// iframe.html:
class Leak {};
window.global_variable = new Leak();
```



Fix the retaining path in DevTools

M65



Constructor	Distance
▼Leak	8
▶Leak @18971	8

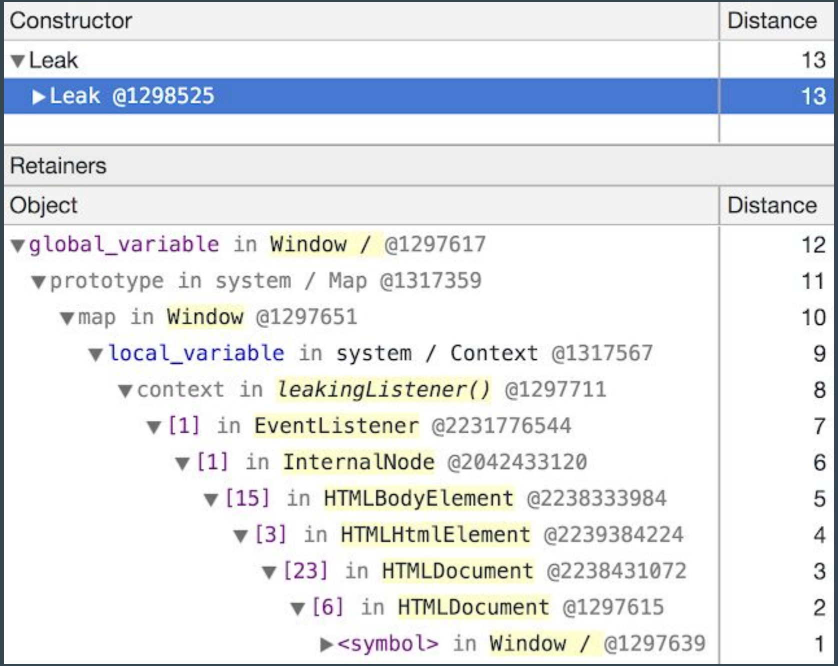
Retainers

Object	Distance
▼global_variable in Window / @2423	7
▼security_token in system / NativeContext @2021	6
▼context in *HTMLDocument()* @25885	5
▼constructor in HTMLDocument @25887	4
▼__proto__ in HTMLDocument @2003	3
▼[1] in Document DOM tree / 8 entries @45619	2
1 in (Document DOM trees) @276673950	1

We fixed the glitch.

Fix the retaining path in DevTools

M66



Constructor	Distance
▼Leak	13
▶Leak @1298525	13

Retainers

Object	Distance
▼global_variable in Window / @1297617	12
▼prototype in system / Map @1317359	11
▼map in Window @1297651	10
▼local_variable in system / Context @1317567	9
▼context in *leakingListener()* @1297711	8
▼[1] in EventListener @2231776544	7
▼[1] in InternalNode @2042433120	6
▼[15] in HTMLBodyElement @2238333984	5
▼[3] in HTMLHtmlElement @2239384224	4
▼[23] in HTMLDocument @2238431072	3
▼[6] in HTMLDocument @1297615	2
▶<symbol> in Window / @1297639	1

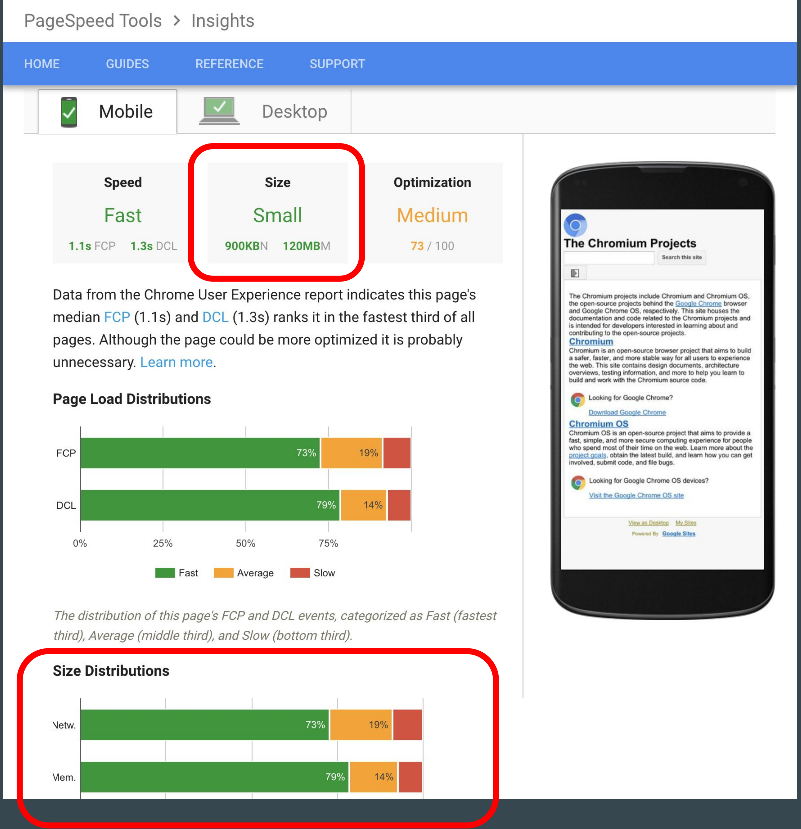
How slim should a web page be?

- There is no accessible way to find out what is slim enough
- There is no "standard"

What we can do:

- Apply methodology used in loading speed project:
  - Publish case studies on memory consumption
  - Get memory data being used in PageSpeed Insights

PageSpeed+Memory Insights



PageSpeed Tools > Insights

HOME GUIDES REFERENCE SUPPORT

Mobile Desktop

Speed	Size	Optimization
Fast	Small	Medium
1.1s FCP 1.3s DCL	900KBN 120MBM	73 / 100

Data from the Chrome User Experience report indicates this page's median FCP (1.1s) and DCL (1.3s) ranks it in the fastest third of all pages. Although the page could be more optimized it is probably unnecessary. Learn more.

Page Load Distributions

The distribution of this page's FCP and DCL events, categorized as Fast (fastest third), Average (middle third), and Slow (bottom third).

Size Distributions

The Chromium Projects

Search this site

The Chromium projects include Chromium and Chromium OS, the open-source projects behind the Google Chrome browser and Google Chrome OS, respectively. This site houses the documentation and code related to the Chromium projects and is intended for developers interested in learning about and contributing to the open-source projects.

Chromium

Chromium is an open-source browser project that aims to build a safer, faster, and more stable way for all users to experience the web. This site contains design documents, architecture overviews, testing information, and more to help you learn to build and work with the Chromium source code.

Looking for Google Chrome?

Download Google Chrome

Chromium OS

Chromium OS is an open-source project that aims to provide a fast, simple, and more secure computing experience for people who spend most of their time on the web. Learn more about the project goals, obtain the latest build, and learn how you can get involved, submit code, and file bugs.

Looking for Google Chrome OS devices?

Visit the Google Chrome OS site

View as Desktop My Sites

Powered By Google Sites

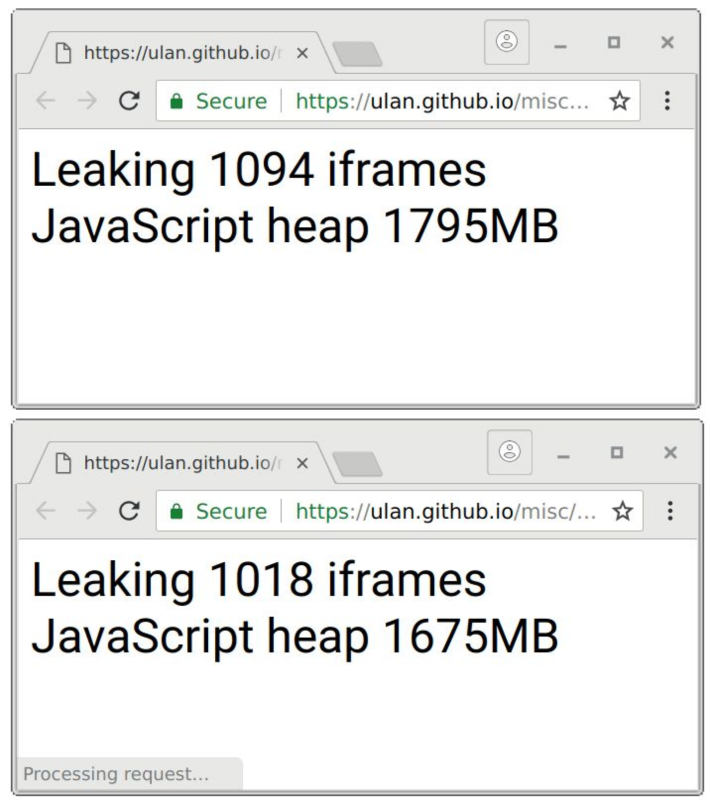
How do I convince my manager ...?

- It is mostly unknown how memory consumption affects business metrics
- OOMs result in breaking a web application's use cases

Memory Pressure & OOM Handling

- NearOOM interventions
- Tab discarding & LifeCycle API
- prOOMpt

prOOMpt


https://ulan.github.io/r
Secure
https://ulan.github.io/misc...

Leaking 1094 iframes
JavaScript heap 1795MB

https://ulan.github.io/r
Secure
https://ulan.github.io/misc/...

Leaking 1018 iframes
JavaScript heap 1675MB

Processing request...

How do I convince my manager ...?

- It is mostly unknown how memory consumption affects business metrics
- OOMs result in breaking a web application's use cases

What we can do:

- Surface OOMs happening to web applications
  (e.g. LifeCycle API, Reporting API, performance.memory)
  - Surface discarding metric
  - Surface proompt metric
- Tie memory consumption to performance and business metrics

Hypothesis: The majority of OOMs are caused by webpages?

We found memory leaks in all web pages of the memory roadshow attendees.

Hypothesis: The majority of OOMs are caused by webpages?

We found memory leaks in all web pages of the memory roadshow attendees.

That is not a proof...

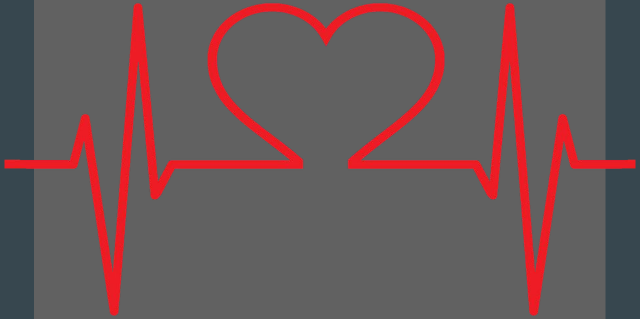
Want to get involved? Have an opinion? Think we are wrong? Good! Participate in the relevant unconference sessions and talks this BlinkOn:

- The Future of Real World Benchmarks
- Providing developers with a well-lit path
- Update: Unified JavaScript and Blink Heap

… or simply talk to us!

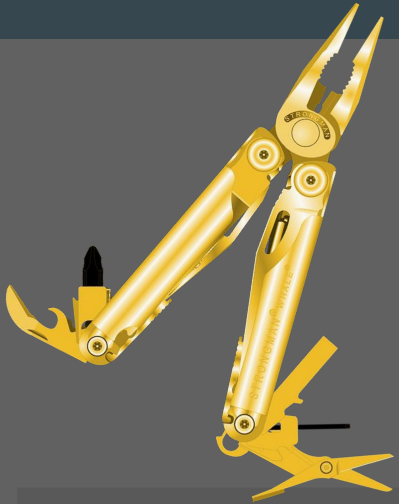
WOULD YOU LIKE TO KNOW MORE?

The Wrap a.k.a. Key Takeaways



We need to **incentivise web developers** to care more about memory

We need to provide them with the means to reliably **measure and benchmark memory consumption**



We need to give them the **right tools** to fix the problems on their own