



Sealing the Gaps

A Deep Dive into JavaScript Memory
Leak Detection





Aw, Snap!

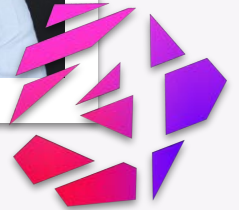
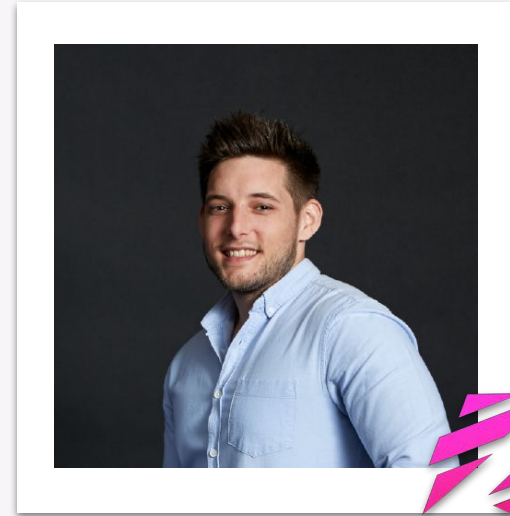
Something went wrong while displaying this webpage.

[Learn more](#)

Send feedback

Julian Jandl

Performance Engineer



@hoebbelsB

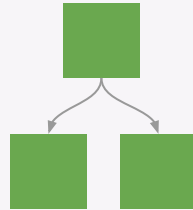
Memory consumption
what consumes memory?



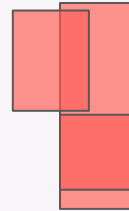
javascript

```
script.js  
  
// your code
```

<DOM>



Composition Layers



what consumes memory

```
script.js  
  
let movie = {  
  title: 'Spiderman'  
};  
  
// ...  
  
let movie = null;
```

Store object in
memory

Release object
from memory

```
script.js
let movie = {
  title: 'Spiderman'
};

let movieCopy = { ...movie };
```

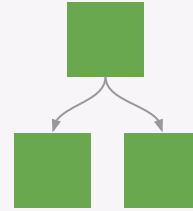
create a
shallow-copy of
an object



2x memory consumption

```
template.html

<body>
  <app-root>
    <app-title></app-title>
  </app-root>
</body>
```



Every DOM Node is consuming memory

Inspect DOM & Javascript Memory



birdseye view



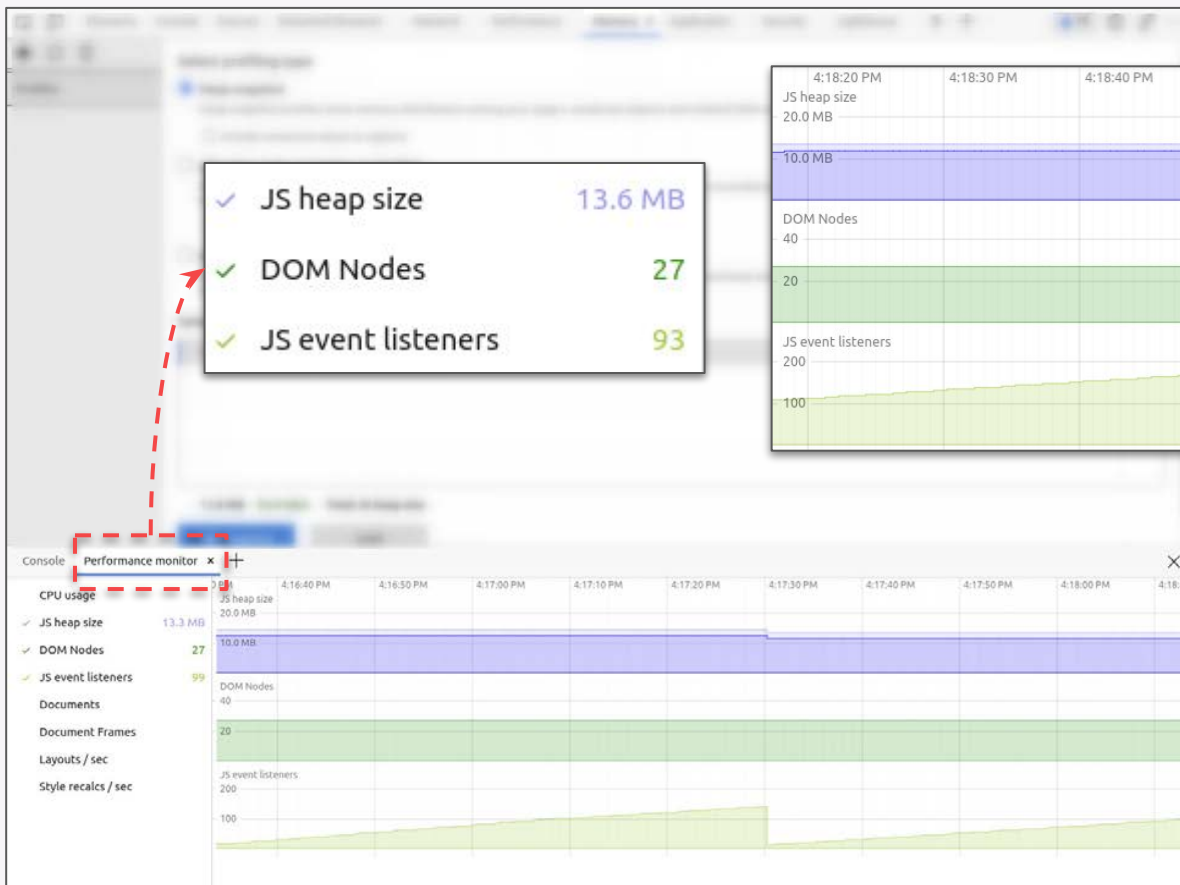
in-depth analysis



Birdseye View



Performance Monitor



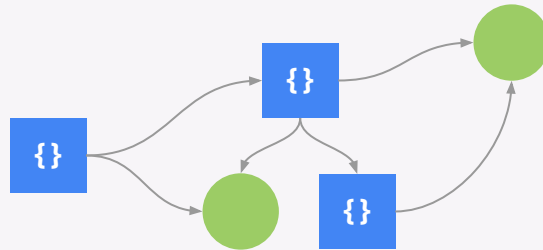
get an **overview** about memory usage over time

In-Depth Analysis



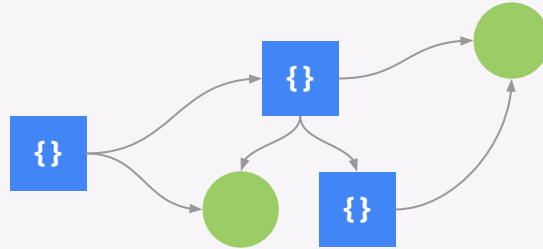
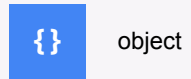
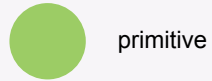
Memory inspection

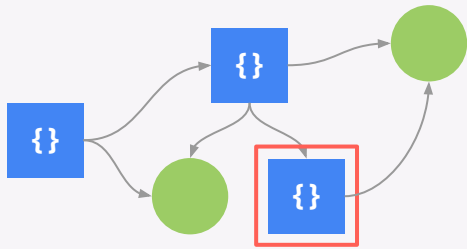
Terminology



Memory Heap

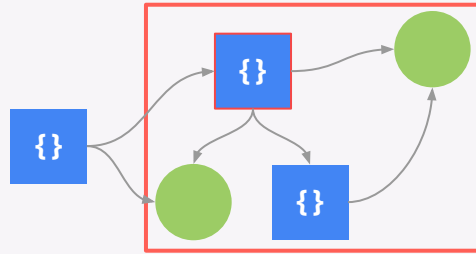
the **memory heap** is an interconnected **graph**





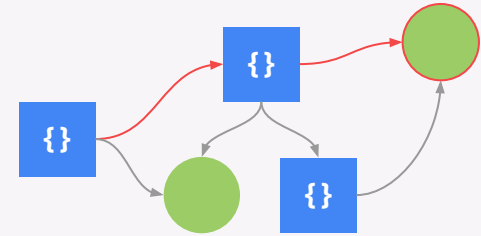
Shallow Size

Size held by the object **itself**



Retained Size

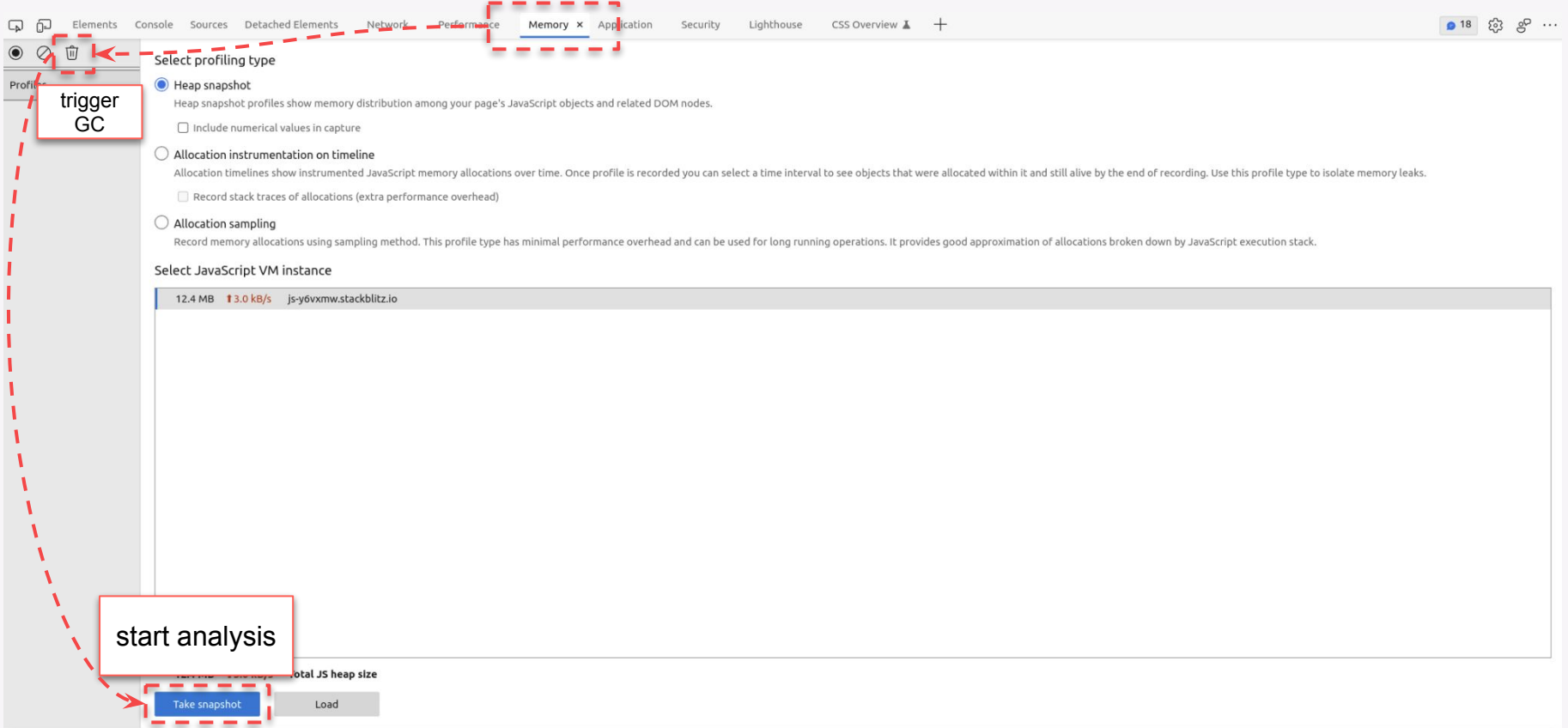
Size that is **freed** when the object gets removed from memory



Distance

Distance the **GC** needs to travel to clear the object

inspect DOM & Javascript Memory



The screenshot shows the Chrome DevTools interface with the Memory tab selected. A red dashed line highlights the 'Performance' tab in the top navigation bar and the 'Take snapshot' button at the bottom. A red box labeled 'trigger GC' points to the 'Heap snapshot' option, and another red box labeled 'start analysis' points to the 'Take snapshot' button. The 'Heap snapshot' option is selected, and the 'Select JavaScript VM instance' section shows 'js-y6vxmw.stackblitz.io' with a memory usage of 12.4 MB and a rate of 3.0 kB/s.

Elements Console Sources Detached Elements Network Performance **Memory** x Application Security Lighthouse CSS Overview +

18 ⚙️ 🔍 ⋮

Select profiling type

Heap snapshot
Heap snapshot profiles show memory distribution among your page's JavaScript objects and related DOM nodes.
 Include numerical values in capture

Allocation instrumentation on timeline
Allocation timelines show instrumented JavaScript memory allocations over time. Once profile is recorded you can select a time interval to see objects that were allocated within it and still alive by the end of recording. Use this profile type to isolate memory leaks.
 Record stack traces of allocations (extra performance overhead)

Allocation sampling
Record memory allocations using sampling method. This profile type has minimal performance overhead and can be used for long running operations. It provides good approximation of allocations broken down by JavaScript execution stack.

Select JavaScript VM instance

12.4 MB	3.0 kB/s	js-y6vxmw.stackblitz.io
---------	----------	-------------------------

Total JS heap size

Take snapshot Load

inspect DOM & Javascript Memory

Elements Console Sources Detached Elements Network Performance **Memory** x Application x Security Lighthouse CSS Overview +

237 25

Summary Class filter All objects Default

Profiles

HEAP SNAPSHOTS

- Snapshot 1 3.5 MB
- Snapshot 2 Save 11.9 MB

Overview of everything that is kept in memory

Save all to file

Constructor	Distance	Shallow Size	Retained Size
Window / https://js-y6vxmw.stackblitz.io	1	36 0 %	11 049 766 93 %
(array) x1263	2	8 384 124 70 %	8 547 304 72 %
Array x656	2	10 496 0.09 %	8 174 060 69 %
(compiled code) x35127	3	1 888 460 16 %	2 045 044 17 %
(closure) x7979	2	247 136 2 %	1 162 104 10 %
Object x1747	2	48 932 0.4 %	828 382 7 %
(system) x20597	2	459 720 4 %	666 948 6 %
system / Context x1111	3	34 392 0.3 %	570 148 5 %
(string) x12917	3	378 864 3 %	378 864 3 %
(object shape) x6747	2	343 844 3 %	369 920 3 %
t x268	2	9 776 0.08 %	97 122 0.8 %
(regexp) x189	4	5 292 0.04 %	74 228 0.6 %
Object /	1	20 0 %	74 032 0.6 %
(concatenated string) x2506	4	50 120 0.4 %	64 488 0.5 %
e x37	4	2 276 0.02 %	52 110 0.4 %
Window x5	2	1 248 0.01 %	44 792 0.4 %

Retainers Filter edges

Object	Distance	Shallow Size	Retained Size
--------	----------	--------------	---------------

inspect DOM & JavaScript Memory

The screenshot shows the Chrome DevTools Memory tab. The 'Class filter' is set to 'FindMe'. A table displays the search results for the 'FindMe' constructor.

Constructor	Distance	Shallow Size	Retained Size
▼ FindMe	5	52 0%	868 0%
▶ FindMe @175889	5	52 0%	868 0%

Below the table, the text 'Search for **Classes** or **Nodes**' is displayed.

inspect DOM & JavaScript Memory

Elements Console Sources Detached Elements Network Performance Memory x Application x Security Lighthouse CSS Overview +

Summary find All objects Default

Profiles

HEAP SNAPSHOTS

- Snapshot 1 3.5 MB
- Snapshot 2 11.9 MB Save

Constructor	Distance	Shallow Size	Retained Size
FindMe			
FindMe @175889	5	52 0%	868 0%
FindMe @175889 index.js:9	5	52 0%	868 0%
context → Array @233665	6	16 0%	8 000 024 67%
map → system / Map @275367	6	40 0%	164 0%
__proto__: Object @275363	6	12 0%	148 0%
type :: "findMe" @176445	6	20 0%	20 0%

Retainers list

superLargeObject in system / Context @273913	8	20 0%	20 0%
context → @177307	7	28 0%	232 0%
[1] in V8Function @184003520	6	48 0%	280 0%
[2] in ScheduledAction @184003840	5	56 0%	336 0%
[1] in DOMTimer @184034368	4	160 0%	496 0%

references to selected node



Demo Time!

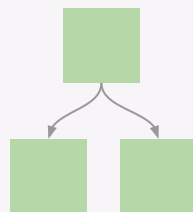
[DOM & JS Memory Inspection](#)



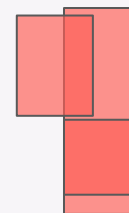
javascript

```
script.js  
// your code
```

<DOM>



Composition Layers



Composition Layers

You'll never let a question pass you by when starting your next project is as easy as clicking a button.

New Dashboard Analysis Report

Jumpstart your explorations

Start by forking any project from your teammates or Observable's global community of visualization and analysis experts.

Ping chart Tham Nguyen	What's That Noise? Fabian Isard in Observable Ambassadors	NOAA Weather Data Tom Johnson in Observable	Correlation over time Mike Freeman in Observable	How Has Men's Tennis Changed From 1973-2021? Kifui Cheung
Most Popular Programming Languages, 2004-2021 Mike Boslock	Why Visualization helps Developers Mindy MacIsaac in Observable	Mapping gridded data with a Voronoi diagram Harry Stevens	Streamgraph Transitions Mike Boslock in D3	Tidy Stacked Area Chart Mike Boslock in D3

Report

Jumpstart your explorations

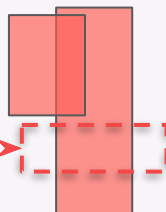
Start by forking any project from your teammates or Observable's global community of visualization and analysis experts.

Building blocks and beyond

Jumpstart your explorations

```
template.css
.class {
  will-change: transform;
}
.class {
  transform: translateZ(0);
}
```

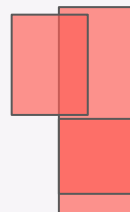
promote a new **layer**



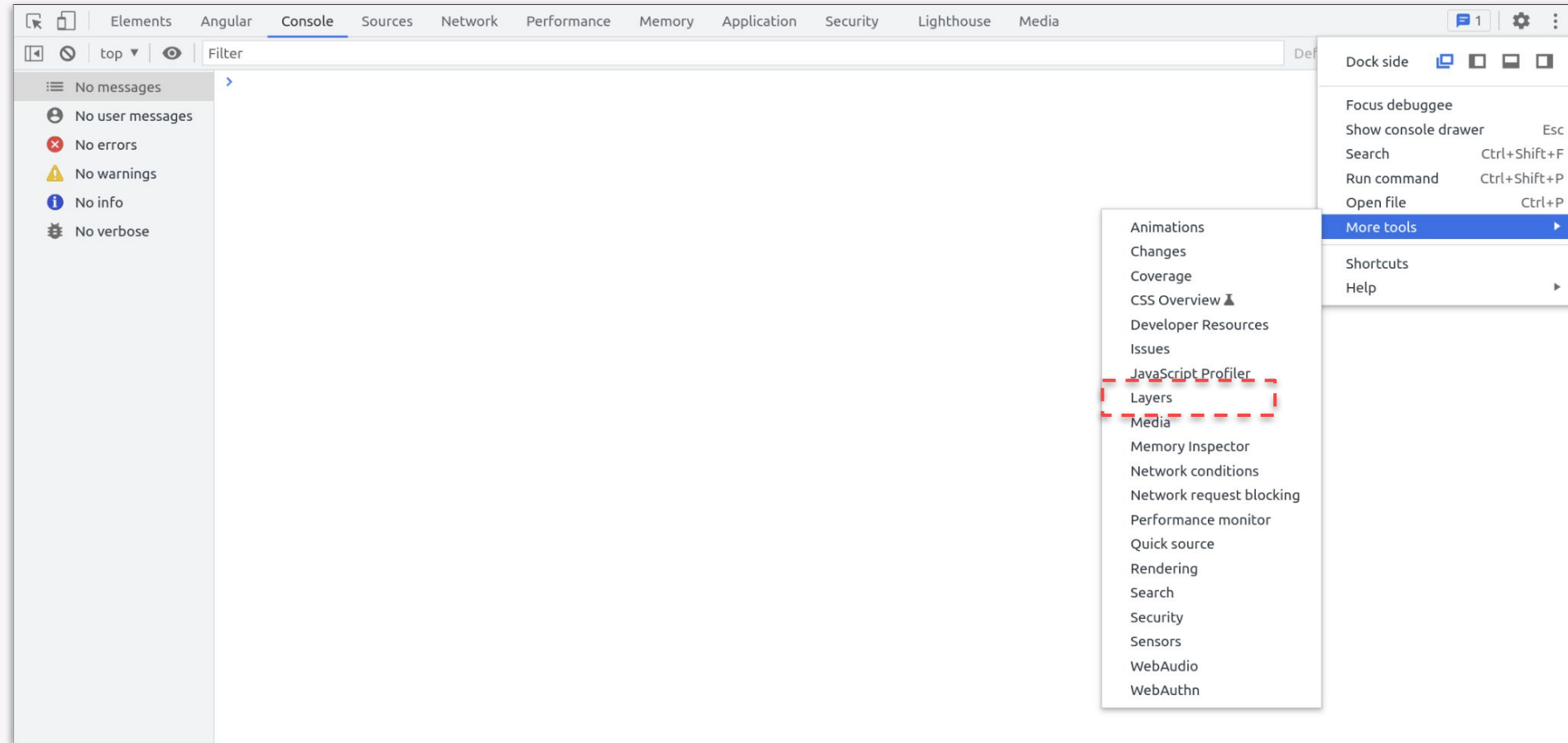
Every composition layer consumes memory on the GPU

Other reasons for layer promotion:

- 3d or perspective changes
- `<video>`
- `<canvas>`
- animated opacity or transform
- sibling with lower z-index + stacking index



Layer Inspection



The image shows the Chrome DevTools interface with the 'Console' tab selected. The 'More tools' menu is open, and the 'Layers' option is highlighted with a red dashed box. The 'Layers' tool is used for inspecting the visual structure of the page, including the Document Object Model (DOM) and the Computed Style of elements.

Elements Angular Console Sources Network Performance Memory Application Security Lighthouse Media

top Filter

No messages

- No user messages
- No errors
- No warnings
- No info
- No verbose

More tools

- Animations
- Changes
- Coverage
- CSS Overview
- Developer Resources
- Issues
- JavaScript Profiler
- Layers**
- Media
- Memory Inspector
- Network conditions
- Network request blocking
- Performance monitor
- Quick source
- Rendering
- Search
- Security
- Sensors
- WebAudio
- WebAuthn

Dock side

- Focus debuggee
- Show console drawer Esc
- Search Ctrl+Shift+F
- Run command Ctrl+Shift+P
- Open file Ctrl+P
- Shortcuts
- Help



Demo Time!

[Inspect Composition Layers](#)



Memory Leaks



Memory that is **allocated** and **not used** by the application anymore

No 1 reason for crashing browser sessions

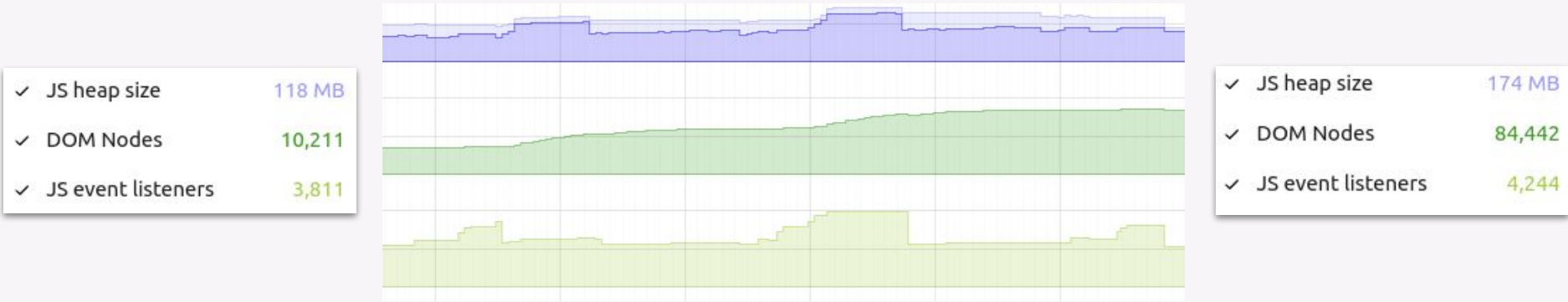


Aw, Snap!

Something went wrong while displaying this webpage.

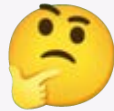
Error code: SIGSEGV

worst case: **repeatedly allocate** without cleanup



t▶

What causes memory leaks?




```
any.component.js

let obj = {};

console.log(obj);

obj = null;
```

 The browser has to **keep a reference** to `obj` to display it in the console

```
any.component.js

function create() {

  window.propKey = 'store';
}
```

```
any.component.ts

const cfg = {};

@Component()
export class Component {

  private _cfg = cfg;
}
```



global variables allocate memory that cannot be cleaned up

```
script.js
class Foo {
  constructor() {

    const data = hugeData();

    setInterval(() => {
      console.log(JSON.stringify(data))
    }, 1500);

    timer(1500).subscribe(() => {
      console.log(JSON.stringify(data));
    })
  }
}

let foo = new Foo();
foo = null;
```



foo stays in memory even after destroying it

```
script.js
class Foo {
  constructor() {

    const data = hugeData();

    setInterval(() => {
      console.log(JSON.stringify(data));
    }, 1500);

    timer(1500).subscribe(() => {
      console.log(JSON.stringify(data));
    })
  }
}

let foo = new Foo();
foo = null;
```



all references of foo stay in memory as well

```
script.js

const btn =
  document.getElementById('btn');

function clickIt() {
  btn.click();
}

btn.remove();
```

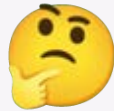


btn stays in memory because there is still a reference to it



btn becomes a **detached element**

How to detect memory leaks?



Detect Memory Leaks



birdseye view



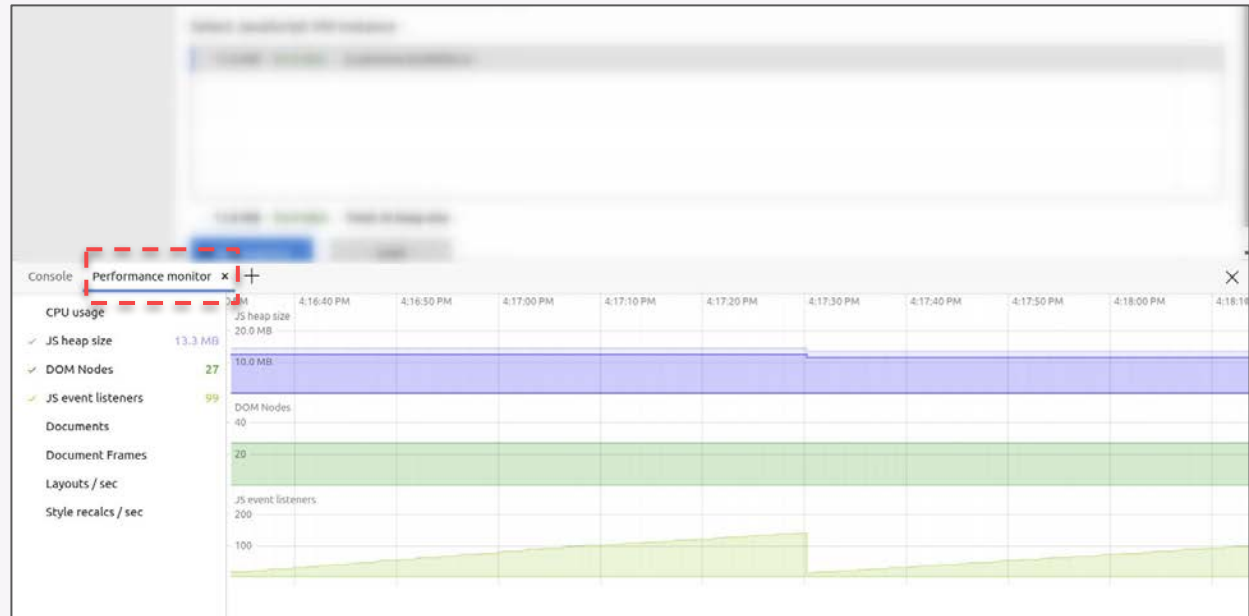
in-depth analysis



Birdseye View



use the **Performance Monitor** to observe memory consumption **over time**



t>

search for **patterns** while repeating various tasks in your application

JS heap size
DOM Nodes
JS event listeners



with leak



without leak

t▶

don't forget to trigger GC



The screenshot shows the Chrome DevTools interface with the Memory tab selected. The 'Performance' tab is highlighted with a red dashed box, and a red arrow points from it to the 'trigger GC' button in the left sidebar. The Memory tab shows the following options:

- Heap snapshot**
Heap snapshot profiles show memory distribution among your page's JavaScript objects and related DOM nodes.
 - Include numerical values in capture
- Allocation instrumentation on timeline**
Allocation timelines show instrumented JavaScript memory allocations over time. Once profile is recorded you can select a time interval to see objects that were allocated within it and still alive by the end of recording. Use this profile type to isolate memory leaks.
 - Record stack traces of allocations (extra performance overhead)
- Allocation sampling**
Record memory allocations using sampling method. This profile type has minimal performance overhead and can be used for long running operations. It provides good approximation of allocations broken down by JavaScript execution stack.

Select JavaScript VM instance

12.4 MB	3.0 kB/s	js-y6vxmw.stackblitz.io
---------	----------	-------------------------

In-Depth Analysis





remember **detached elements** ?



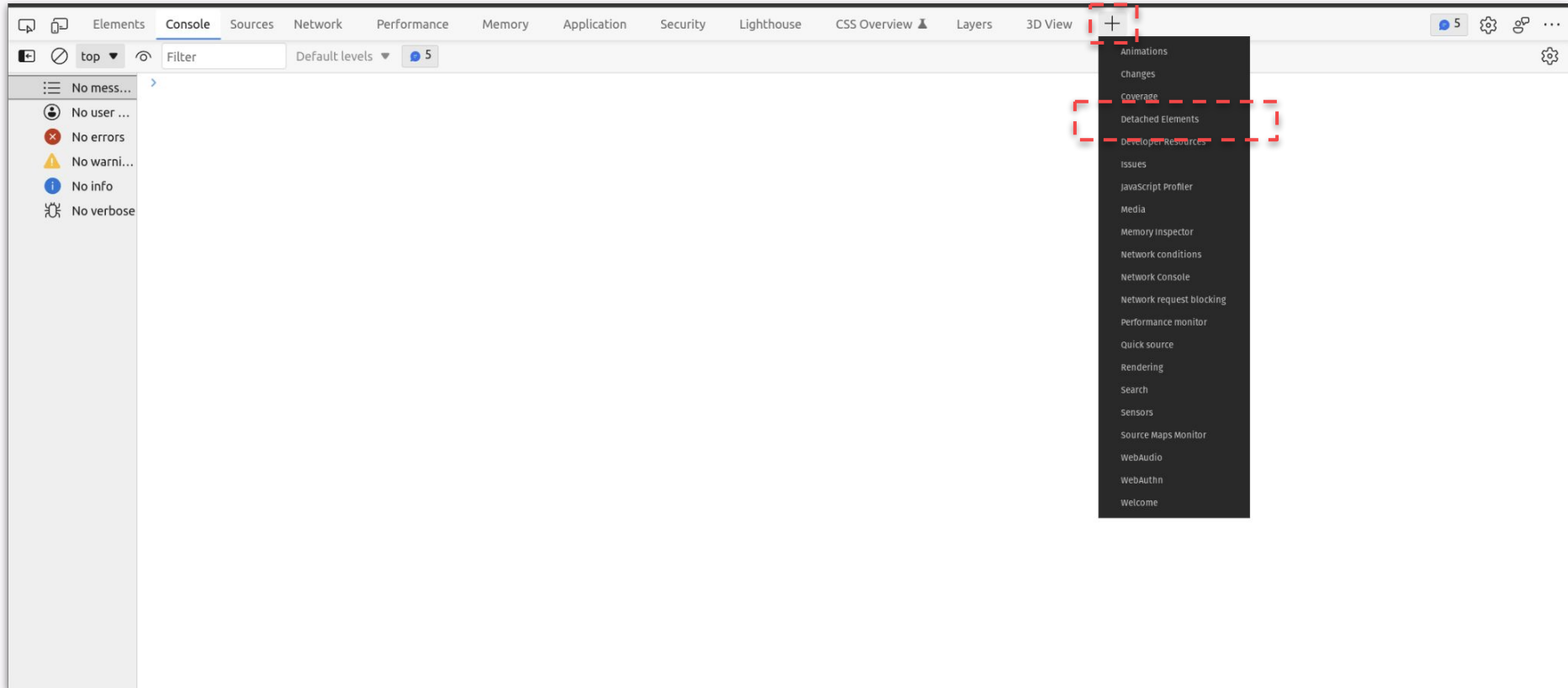
js frontend code is mostly coupled to **DOM Nodes**



v93: the new Detached Elements tool

[announcement](#)

Detached Elements Tools



Detached Elements Tool

The screenshot shows the Chrome DevTools interface with the 'Detached Elements' tool active. The browser address bar shows `https://js-pgj8he.stackblitz.io/`. The tool's toolbar contains three icons: a refresh icon (labeled 2), a trash icon (labeled 1), and an eye icon (labeled 3). Below these icons are the labels 'read detached elements', 'trigger GC', and 'analyze heap'. A red dashed box highlights the browser address bar and the toolbar. A red dashed arrow points from the address bar to the 'read detached elements' icon.

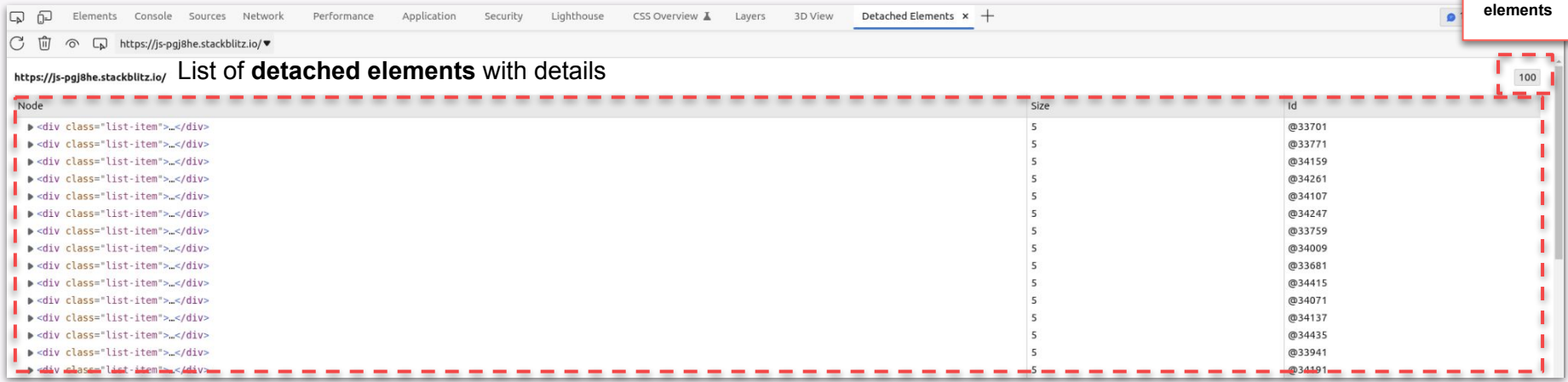
Below the screenshot is a workflow diagram illustrating the sequence of actions:

```
graph LR; A[1 trigger GC] --> B[2 read detached elements]; B --> C[3 analyze heap];
```

The diagram consists of three rectangular boxes connected by arrows. The first box is labeled '1 trigger GC', the second '2 read detached elements', and the third '3 analyze heap'. Arrows point from the first box to the second, and from the second box to the third.

At the bottom left of the screenshot, the text 'All done' is visible.

amount
detached
elements



The screenshot shows the Chrome DevTools interface with the 'Detached Elements' panel open. The browser address bar shows 'https://js-pgj8he.stackblitz.io/'. The panel title is 'List of detached elements with details'. A table lists 100 detached elements, each with a 'Node' column showing the HTML tag, a 'Size' column with the value '5', and an 'Id' column with unique identifiers. A red dashed box highlights the table content, and a small box in the top right corner of the panel indicates '100' items.

Node	Size	Id
<div class="list-item">...</div>	5	@33701
<div class="list-item">...</div>	5	@33771
<div class="list-item">...</div>	5	@34159
<div class="list-item">...</div>	5	@34261
<div class="list-item">...</div>	5	@34107
<div class="list-item">...</div>	5	@34247
<div class="list-item">...</div>	5	@33759
<div class="list-item">...</div>	5	@34009
<div class="list-item">...</div>	5	@33681
<div class="list-item">...</div>	5	@34415
<div class="list-item">...</div>	5	@34071
<div class="list-item">...</div>	5	@34137
<div class="list-item">...</div>	5	@34435
<div class="list-item">...</div>	5	@33941
<div class="list-item">...</div>	5	@34491

Detached Elements Tools

The screenshot shows the Chrome DevTools interface with the following components:

- Detached Elements Panel:** Lists detached HTML elements. A red dashed box highlights an entry with ID `@33701`. A yellow mouse cursor points to this entry.
- Memory Panel:** Shows a summary of memory usage. A red dashed box highlights a row for `Detached HTMLDivElement @33701`. Below this, the **Retainers** section shows the object graph. A red dashed box highlights the `button` in `ListItem @162787`, which is the source of the leak. A box labeled **source of leak** points to the source code for `index.js:36` and `index.js:42`.

Constructor	Distance	Shallow Size	Retained Size
Detached HTMLDivElement @33701	-	140 0%	140 0%
Detached HTMLDivElement @33771	-	140 0%	140 0%
Detached HTMLDivElement @33719	-	140 0%	140 0%

Object	Distance	Shallow Size	Retained Size
[1] in InternalNode @256770368	-	0 0%	0 0%
[2] in InternalNode @256735552	-	0 0%	0 0%
[1] in Detached Text @256691744	-	96 0%	96 0%
[3] in Detached Text @182422976	-	96 0%	96 0%
[2] in Detached Text @182423456	-	96 0%	96 0%
button in ListItem @162787	-	32 0%	8 000 056 1%
this in system / Context @162809	-	20 0%	8 000 076 1%
context in () @33699	-	28 0%	8 000 104 1%
[1] in Detached V8EventListener @256717472	-	56 0%	8 000 160 1%
[5] in InternalNode @256770368	-	0 0%	0 0%



Demo Time!

[Demo: Detached Elements](#)



Web Performance Consulting

The screenshot displays the PUSH BASED website. At the top left is the PUSH BASED logo, and at the top right is a 'Contact us' button. Below the logo is a navigation menu with 'Get Trained', 'Services', 'About', and 'Insights', each followed by a dropdown arrow. The main content area has a dark purple background with a pattern of colorful code symbols on the right. The section is titled 'Performance Audits' in white. Below the title is a paragraph: 'Our professionals can tell you exactly why your product is slow and provide a detailed step-by-step guide on how to fix it.' A purple 'Schedule Meeting' button is positioned below the text. The lower section of the page has a white background and is titled 'Leading Experts in Performance Audits'. It contains a paragraph: 'As global leaders in performance audits, our team of industry experts conducts meticulous analysis, identifying bottlenecks, and delivering prioritized recommendations. From memory leaks to bundle size reduction, we optimize for a seamless user experience.' Below this is a carousel of client logos: Microsoft, ClickUp, SAP, and salesforce, with left and right navigation arrows. At the bottom of the carousel is the text 'Companies that love us for what we do...'.

PUSH BASED [Contact us](#)

[Get Trained](#) [Services](#) [About](#) [Insights](#)

Performance Audits

Our professionals can tell you exactly why your product is slow and provide a detailed step-by-step guide on how to fix it.

[Schedule Meeting](#)

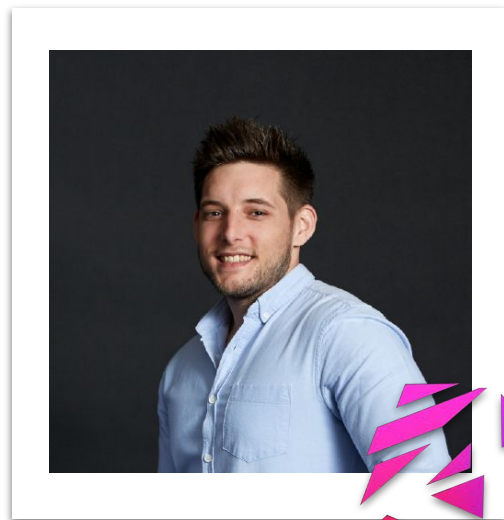
Leading Experts in Performance Audits

As global leaders in performance audits, our team of industry experts conducts meticulous analysis, identifying bottlenecks, and delivering prioritized recommendations. From memory leaks to bundle size reduction, we optimize for a seamless user experience.

Companies that love us for what we do...

Thanks for **your** time!
If you have any questions just **ping me!**

julian.jandl@push-based.io



@hoebbelsB