

LeakPair: *Proactive* Repairing of Leaks in Single Page Web Applications

with  ACM Distinguished Paper Award

Arooba Shahoor

Kyungpook National University, Korea

Askar Yeltayuly Khamit

UNIST, Korea

Jooyong Yi

UNIST, Korea

Dongsun Kim

Kyungpook National University, Korea



The 38th IEEE/ACM International Conference on Automated Software Engineering (ASE 2023)
14 September 2023

Low disk space

You're running out of space on this PC. Manage storage to view usage and free up some space.

Don't remind me again

Manage storage

Close

Introduction to Investment Funds from a Legal and Finance Perspective



y, 27 September



17:00 - 18:00



JFK, room E004/005



Aw, Snap!

Something went wrong while displaying this webpage.

Error code:

Out of Memory

[Learn more](#)



Reload



MOTIVATION

Why Are Memory Leaks Critical in the **SPA Architecture?**

Multi Page Application Architecture

Inside the Heap

On user interactions



Object 2

Object 4

Object 1

Object 3

Client

User Interface

Page 1



Page 2



Page 3



Page switching with reload



HTTP Request for
initial page load



New page for each request

Server

Presentation Layer



<template>



Base &
Business Logic

Single Page Application Architecture

Unintentional references
will keep piling up!

Client

Server

Object 5

On user interactions

Object 4

Object 2

Object 1

Object 3

This page is never refreshed... Framework
(Angular/React/Vue..)

View 1



View 2



View 3



View switching **without**
reload

Single .html file

<template 1>

<template 2>

<template 3>

No reload required



HTTP Request
for **initial page**
load



AJAX Request for
page transition



Database &
Business
Logic

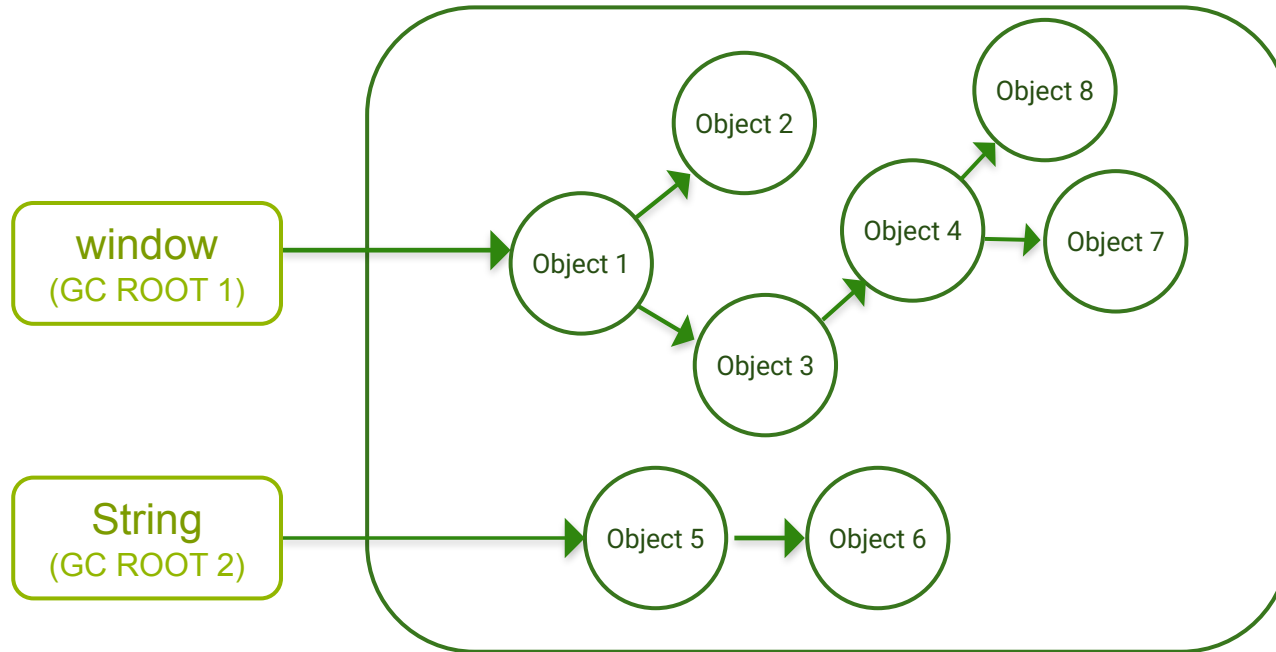
JavaScript Devs Can't Rely on Garbage Collectors



Garbage Collection in JS (Mark and Sweep)

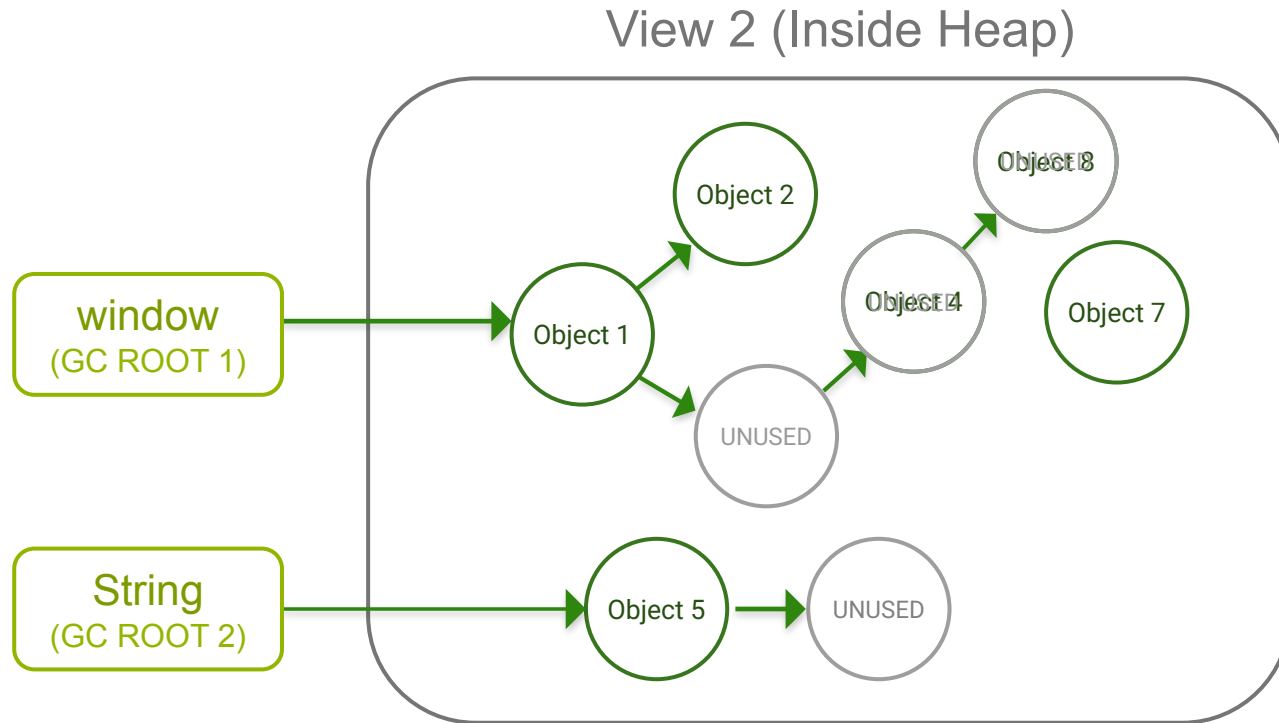
Reachability from Root

View 1 (Inside Heap)



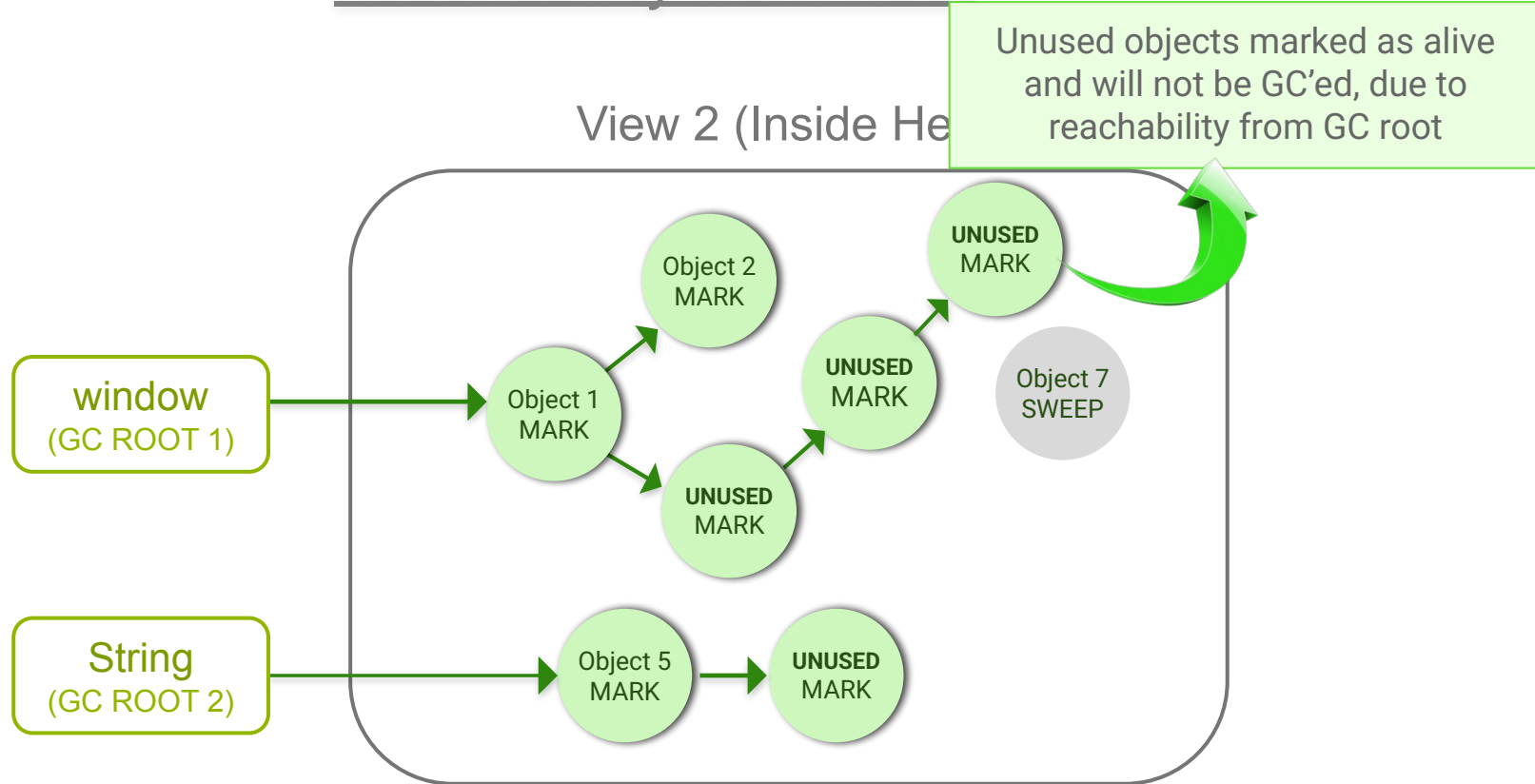
Garbage Collection in JS (Mark and Sweep)

Reachability from Root



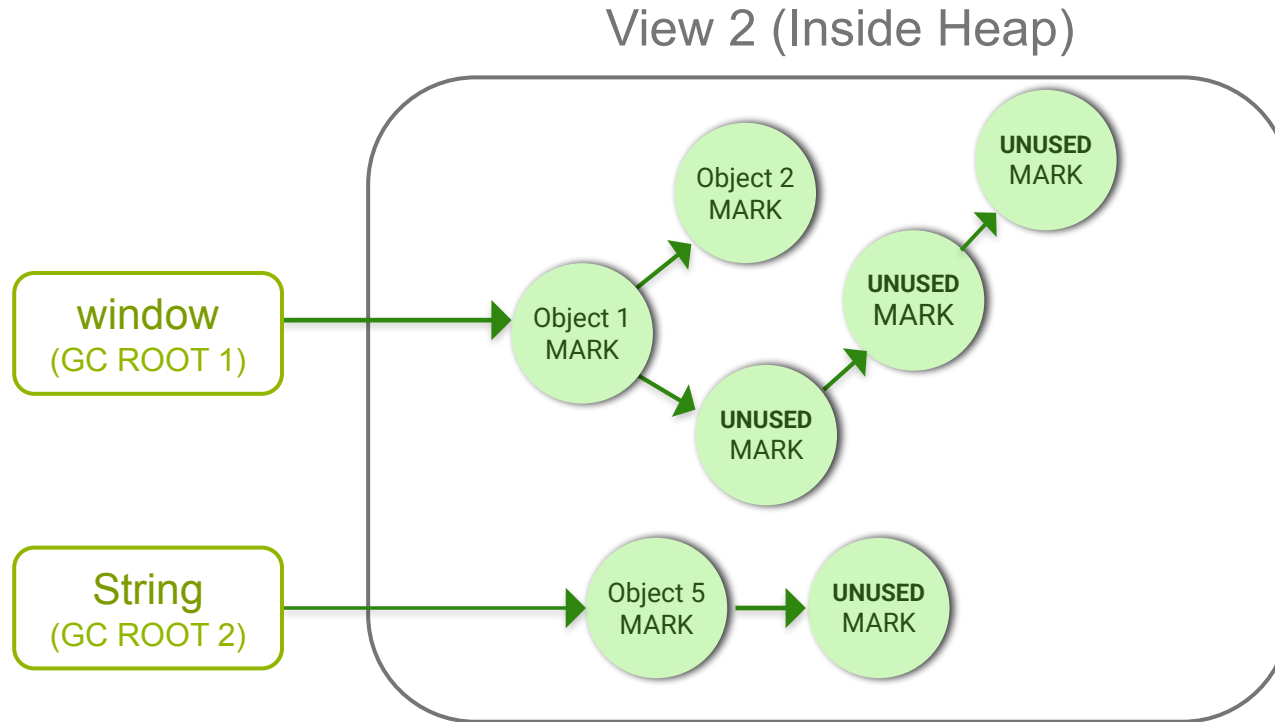
Garbage Collection in JS (Mark and Sweep)

Reachability from Root



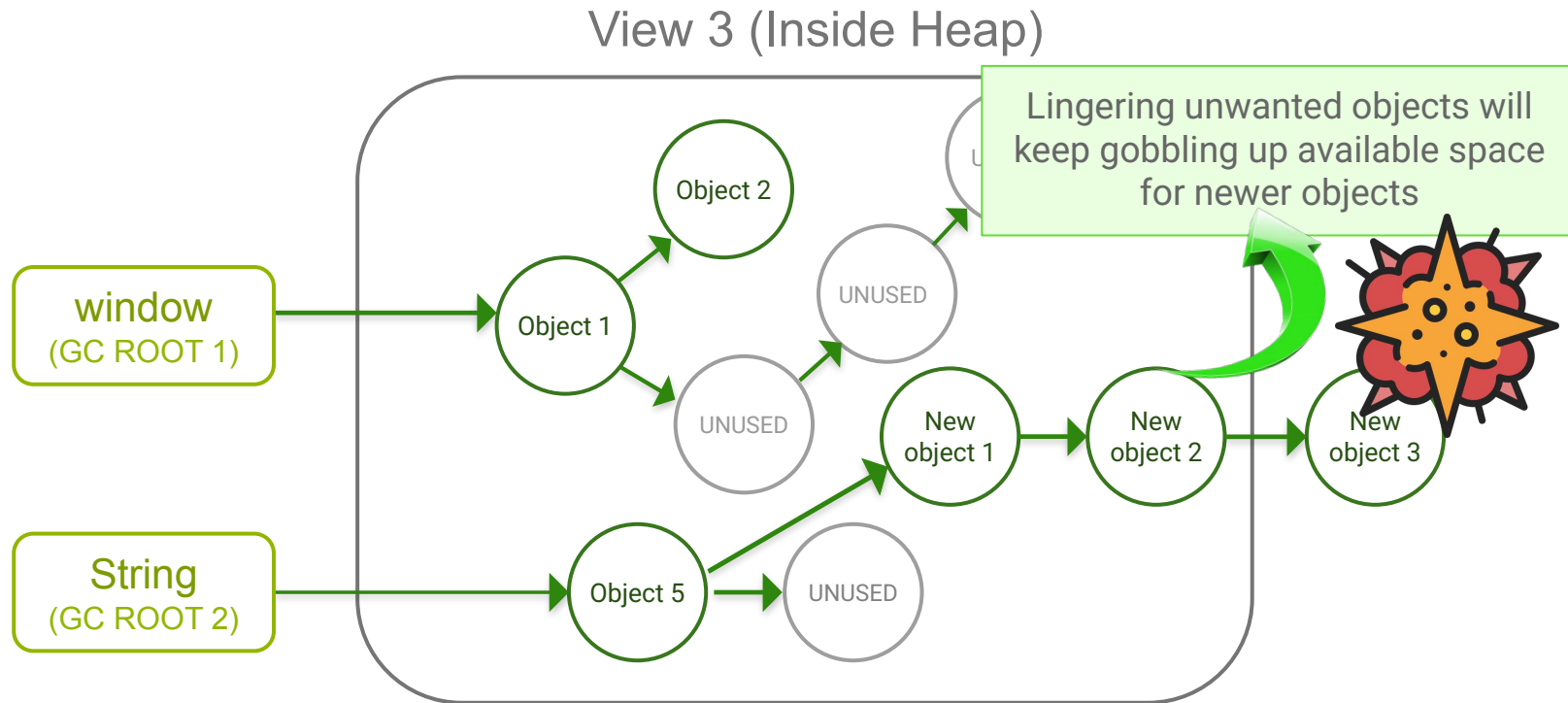
Garbage Collection in JS (Mark and Sweep)

Reachability from Root



Garbage Collection in JS (Mark and Sweep)

Reachability from Root



Case in Point

```
constructor(props: SidePaneProps) {  
  super(props);  
  this.state = {  
    // Reachable from root  
    plugins: props.plugins[0],  
  };  
  
  window.addEventListener('hashchange', this.updateStateFromHash);  
}
```

```
componentDidMount() {  
  this.updateStateFromHash();  
}
```

Mem

```
23  
24       window.addEventListener('hashchange', this.updateStateFromHash);  
25     }  
26  
27     componentDidMount() {  
28       this.updateStateFromHash();  
29     }  
30  
31 +   componentWillUnmount() {  
32 +     window.removeEventListener('hashchange', this.updateStateFromHash);  
33 +   }
```

The fix

Leak Detection in JS

Automation Efforts (2015-2022)

2015. MemInsight [3] Reports stale objects, by computing object lifetimes.

2016. LeakSpot [4] Reports all allocation and reference sites of leaked objects.

2018. BLeak [5] Identifies and reports leaked objects by running the website in a headless browser.

2022. Memlab [7] Reports retainer traces of leaked objects by running the website in a headless browser.



The Problem With Automated Detection Tools

MemInsight

LeakSpot



Based on **staleness** criteria + report leak-related data rather than actual leak locations

Bleak

Memlab



Require manually-written scenario file

Bleak



Takes at least 10 mins to execute

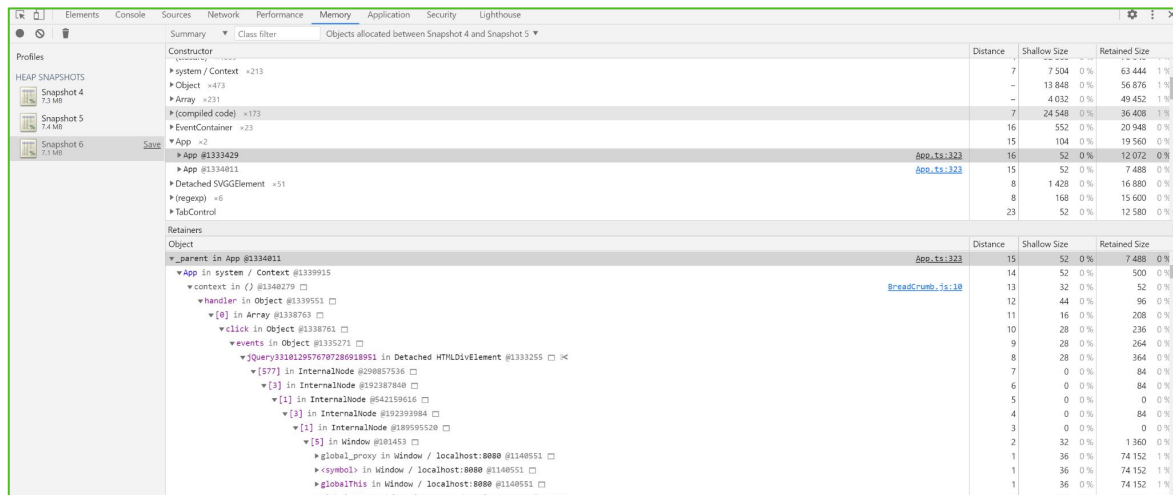
Memlab



Reported retainer traces contain **metadata** and **internal objects**, making root cause detection arduous

State of the Art

- Manual diagnosis via heap snapshots
- Three snapshot technique
- First introduced by Gmail team in 2012
- Detecting root cause can be **laborious** and **inaccurate**



The screenshot shows the Chrome DevTools Memory tab. The left sidebar displays 'HEAP SNAPSHOTS' with three snapshots: Snapshot 4 (7.3 MB), Snapshot 5 (7.4 MB), and Snapshot 6 (7.1 MB). The main panel shows a comparison of objects allocated between Snapshot 4 and Snapshot 5. The 'Summary' tab is active, showing a table of objects with columns for Distance, Shallow Size, and Retained Size. The table lists various objects, including a system context, an array, compiled code, an event container, and an application object. The application object is expanded, showing its internal structure and references to other objects like a context, handler, and events.

Object	Distance	Shallow Size	Retained Size
System / Context	7	7,504 B	63,444 B
Object	~	13,848 B	56,876 B
Array	~	4,032 B	49,452 B
Compiled code	7	24,540 B	36,408 B
EventContainer	16	552 B	20,948 B
App	15	104 B	19,560 B
App #1333429	16	52 B	12,072 B
App #1334811	15	52 B	7,488 B
Detached SVGElement	8	1,428 B	16,880 B
Image	8	168 B	15,600 B
TabControl	23	52 B	12,580 B

Object	Distance	Shallow Size	Retained Size
App #1334811	15	52 B	7,488 B
App in system / Context	14	52 B	500 B
Context in ()	13	32 B	52 B
Handler in Object	12	44 B	96 B
Array	11	16 B	208 B
Click in Object	10	28 B	236 B
Events in Object	9	28 B	264 B
QuerySelector	8	28 B	364 B
InternalNode	7	0 B	84 B
InternalNode	6	0 B	84 B
InternalNode	5	0 B	0 B
InternalNode	4	0 B	84 B
InternalNode	3	0 B	0 B
Window	2	32 B	1,360 B
Global proxy in Window	1	36 B	74,152 B
Symbol in Window	1	36 B	74,152 B
GlobalThis in Window	1	36 B	74,152 B

What we need is...

*A change of
perspective*



Proactive Approach

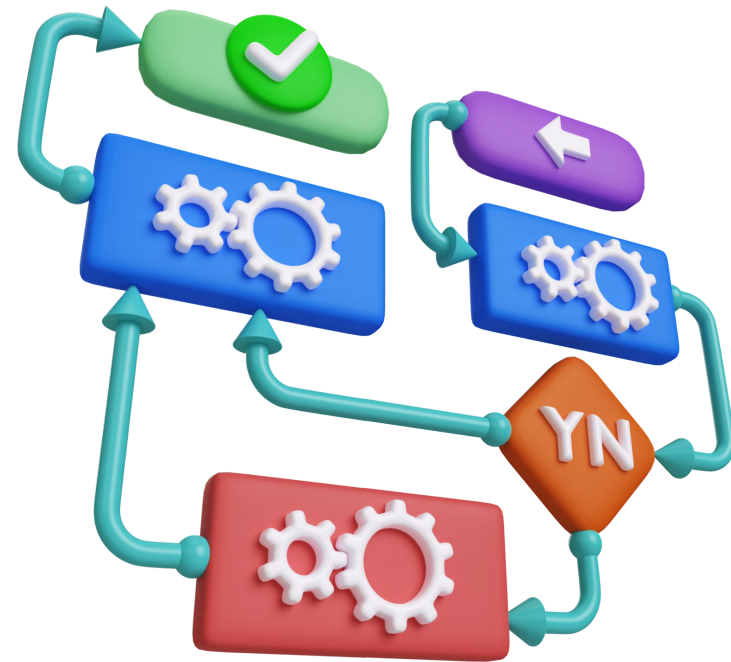
Based on fix patterns [1].

[1] D. Kim, J. Nam, J. Song, and S. Kim, "Automatic Patch Generation Learned from Human-written Patches," in *Proceedings of the 2013 International Conference on Software Engineering*, in ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 802–811.



- Fix first without detection.
- Patches that are simple, non-intrusive and recurring.
- Enable repairs *before* fault localization.
- Avoiding detection step is a huge advantage.

LeakPair



Leak Pattern Mining

- Search targets: **GitHub** and **Stack Overflow**.
- Reported at least 5 times.
- Acknowledged by at least 2 developers.
- Replicable and testable.



Fix Pattern Mining

- Stack Overflow answers accepted at least in 2 posts.
- GitHub commits merged in at least 2 different projects.
- Verified by comparing memory footprints before and after.



4 Leaks and (Corresponding) Fix Patterns

1. Uncleared Timing Events (**setTimeout** and **setInterval**)

If component unmounts before time, these events will *still* execute, retaining references of all enclosing objects.

Leak Pattern

```
{
  ...
  setTimeout(() => {...})
  ...
}
```

Fix Pattern

```
{
  ...
  setTimeout(() => {...})
  timeOutID = setTimeout(() => {...})
  ...
  destructorMethodDeclaration() {
    ...
    clearTimeout(timeOutID)
  }
}
```

Other leak patterns

2. Unreleased subscription
3. Unremoved event listener
4. Uncanceled animation frame request

Applying Fix Patterns



```
ClassMethod(path) {
  path.get('body')
  path.get('body')
}

60   ngOnDestroy(): void {
61   +   this._destroy$.next();
62   +   this._destroy$.complete();
63     this.authSubscription.unsubscribe();
64   }

  // ...
}
```

EVALUATION



Research Questions



RQ1. Effectiveness How effective is the tool at minimizing memory leaks?

RQ2. Acceptance How useful are generated patches, as perceived by developers?

RQ3. Correctness What is the impact of our tool on test suite execution results?

Subjects

10 open source SPA projects with already known memory leaks: Microsoft's Rooster, Fundamental Library for Angular, Material UI...

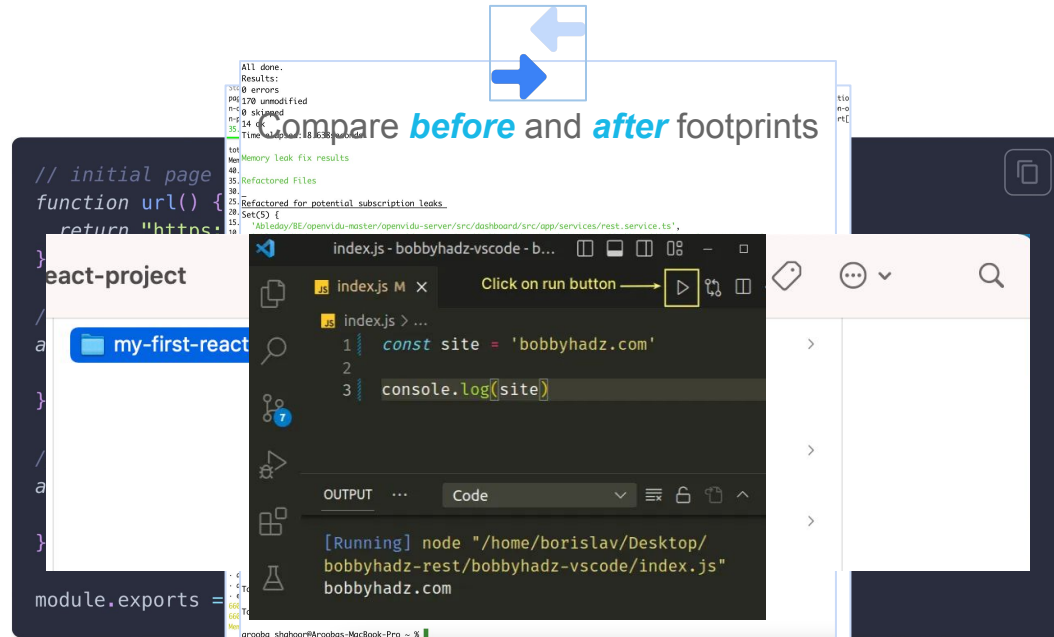
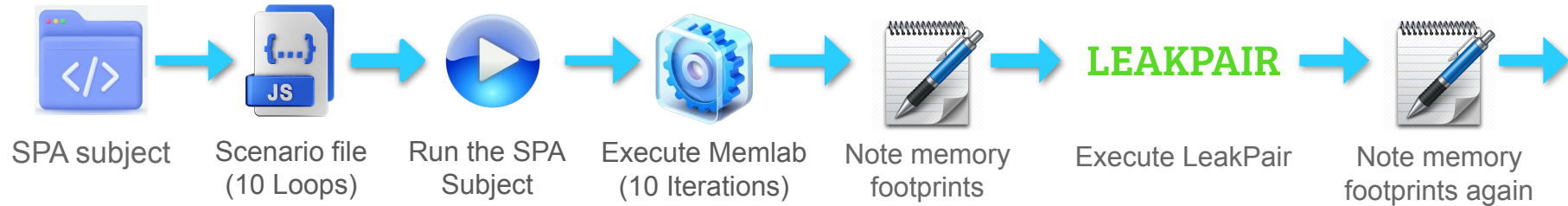
10 projects with unfound leaks at the time of evaluation: Angular Extensions Elements, react-multi-carousel, ngx-bootstrap...

Known leaks Leaks already detected and fixed by developers.

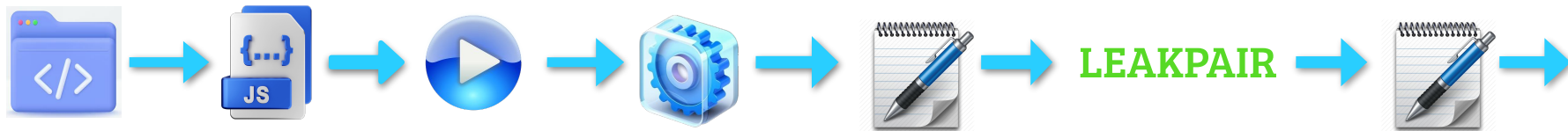
Unknown leaks Leaks developers were not aware of.



Effectiveness Evaluation (RQ 1)



Effectiveness Evaluation (RQ 1)



SPA subject

Scenario file
(10 Loops)

Run the SPA Subject

Execute Memlab (10 Iterations)

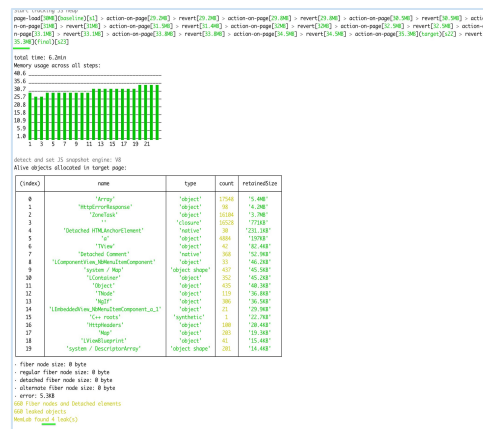
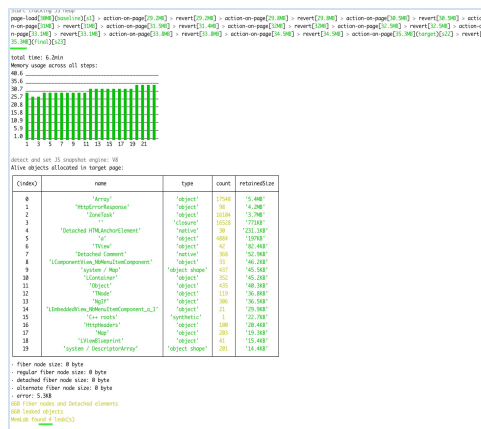
Note memory footprints

Execute LeakPair

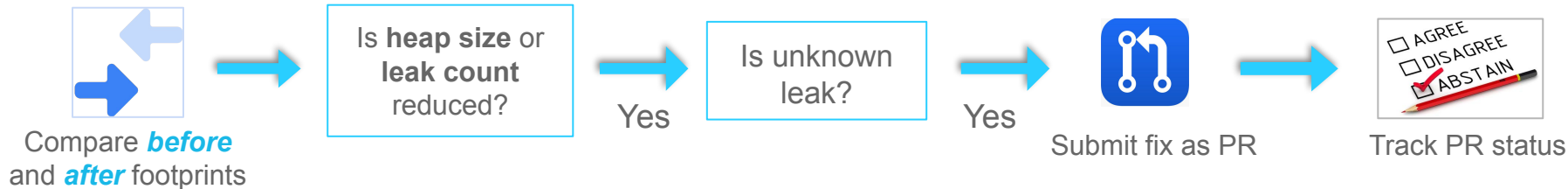
Note memory
footprints again



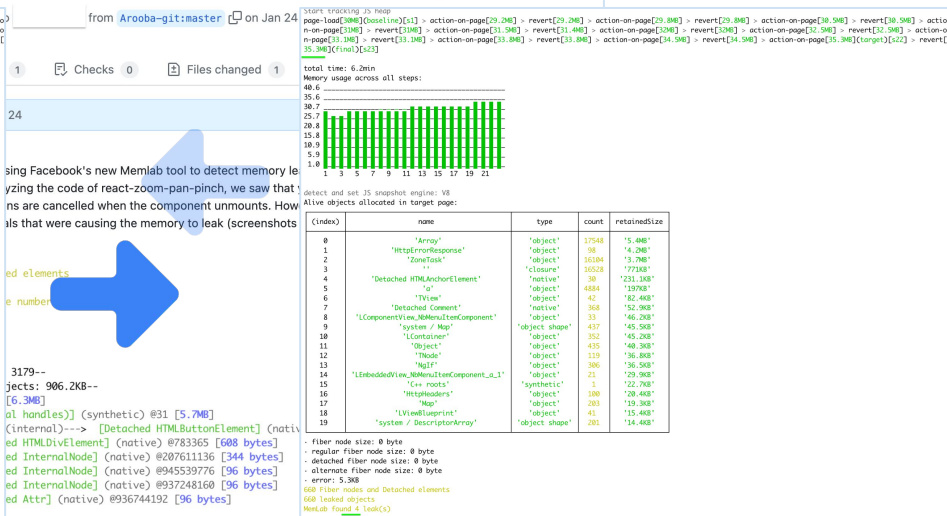
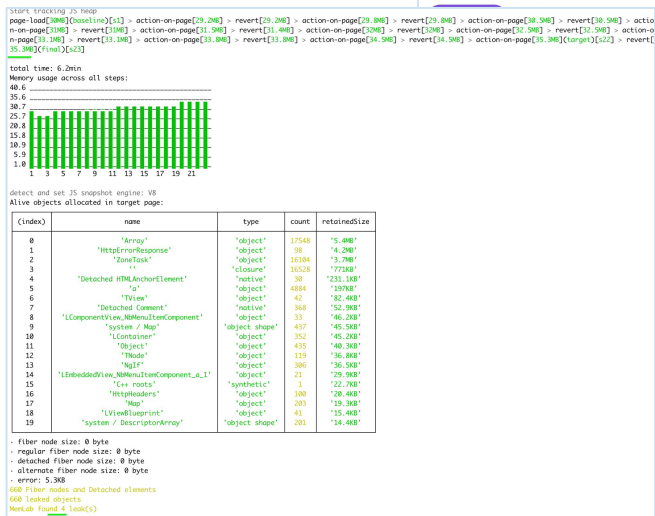
Compare *before* and *after* footprints



Acceptance Evaluation (RQ 2)



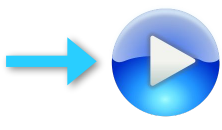
Clear intervals to avoid detached components and memory leaks #331



Correctness Evaluation (RQ 3)



SPA subject



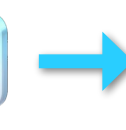
Run the SPA
Subject



Note build/
compile time



Execute test suite
(If available)



Note passing/failing
test cases



LEAKPAIR

Execute LeakPair



Note **build/compile**
time and **test case**
results again



Compare before and after **test cases results** and **execution times**

```
DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL
[PASS] src/components/Items.test.js
✓ renders correctly when there are no items (8ms)

> 1 snapshot written.
Snapshot Summary
  1 snapshot written from 1 test suite.

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:  1 written, 1 total
Time:        0.889s, estimated 1s
Ran all test suites related to changed files.

Watch Usage
  Press a to run all tests.
  Press f to run only failed tests.
  Press q to quit watch mode.
  Press p to filter by a filename regex pattern.
  Press t to filter by a test name regex pattern.
  Press Enter to trigger a test run.
```

```
DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL
[PASS] src/components/Items.test.js
✓ renders correctly when there are no items (8ms)

> 1 snapshot written.
Snapshot Summary
  1 snapshot written from 1 test suite.

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:  1 written, 1 total
Time:        0.889s, estimated 1s
Ran all test suites related to changed files.

Watch Usage
  Press a to run all tests.
  Press f to run only failed tests.
  Press q to quit watch mode.
  Press p to filter by a filename regex pattern.
  Press t to filter by a test name regex pattern.
  Press Enter to trigger a test run.
```

```
Version: webpack 2.2.1
Time: 1639ms
Asset      Size  Chunks  [emitted]  [big]  main
chunk      0     index.js (main) 1 MB [entry] [rendered]
[99]      ./~/react/react.js 56 bytes {0} [built]
[114]     (webpack)/hot/emitter.js 77 bytes {0} [built]
[115]     ./main.js 470 bytes {0} [built]
[116]     (webpack)-dev-server/client?http://localhost:8080 5.28 kB {0} [built]
[117]     (webpack)/hot/dev-server.js 1.57 kB {0} [built]
[120]     ./App.jsx 2.99 kB {0} [built]
[136]     ./~/html-entities/index.js 231 bytes {0} [built]
[141]     ./~/react-dom/index.js 59 bytes {0} [built]
[155]     ./~/react-dom/lib/ReactDOM.js 5.14 kB {0} [built]
[251]     ./~/strip-ansi/index.js 161 bytes {0} [built]
[253]     (webpack)-dev-server/client/overlay.js 3.6 kB {0} [built]
[254]     (webpack)-dev-server/client/socket.js 856 bytes {0} [built]
[256]     (webpack)/hot/log-apply-result.js 1.02 kB {0} [built]
[262]     (webpack)/~/node-libs-browser~/url/url.js 23.3 kB {0} [built]
[264]     multi (webpack)-dev-server/client?http://localhost:8080 webpack/hot/dev-server ./main.js 52 bytes {0} [built]
+ 250 hidden modules
webpack: Compiled successfully.
```

```
Version: webpack 2.2.1
Time: 1639ms
Asset      Size  Chunks  [emitted]  [big]  main
chunk      0     index.js (main) 1 MB [entry] [rendered]
[99]      ./~/react/react.js 56 bytes {0} [built]
[114]     (webpack)/hot/emitter.js 77 bytes {0} [built]
[115]     ./main.js 470 bytes {0} [built]
[116]     (webpack)-dev-server/client?http://localhost:8080 5.28 kB {0} [built]
[117]     (webpack)/hot/dev-server.js 1.57 kB {0} [built]
[120]     ./App.jsx 2.99 kB {0} [built]
[136]     ./~/html-entities/index.js 231 bytes {0} [built]
[141]     ./~/react-dom/index.js 59 bytes {0} [built]
[155]     ./~/react-dom/lib/ReactDOM.js 5.14 kB {0} [built]
[251]     ./~/strip-ansi/index.js 161 bytes {0} [built]
[253]     (webpack)-dev-server/client/overlay.js 3.6 kB {0} [built]
[254]     (webpack)-dev-server/client/socket.js 856 bytes {0} [built]
[256]     (webpack)/hot/log-apply-result.js 1.02 kB {0} [built]
[262]     (webpack)/~/node-libs-browser~/url/url.js 23.3 kB {0} [built]
[264]     multi (webpack)-dev-server/client?http://localhost:8080 webpack/hot/dev-server ./main.js 52 bytes {0} [built]
+ 250 hidden modules
webpack: Compiled successfully.
```

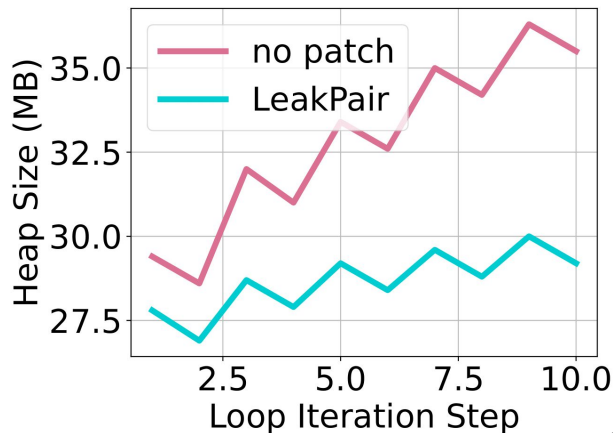
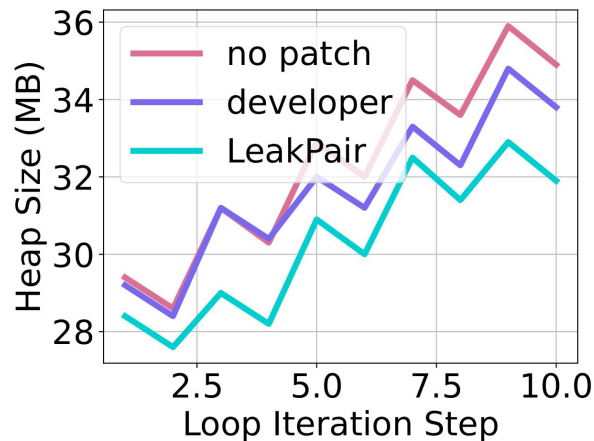
RESULTS



Effectiveness (RQ 1)

- 8 of 10 subjects with *known* leaks had noticeable heap reductions.
- 7 of 10 subjects with *unknown* leaks had **significant** heap reductions (**4 -18%**).
- All subjects had memory leak reductions.

LeakPair's patches can successfully prevent **leaks** and reduce **heap**, without leak detection.



Correctness (RQ 3)

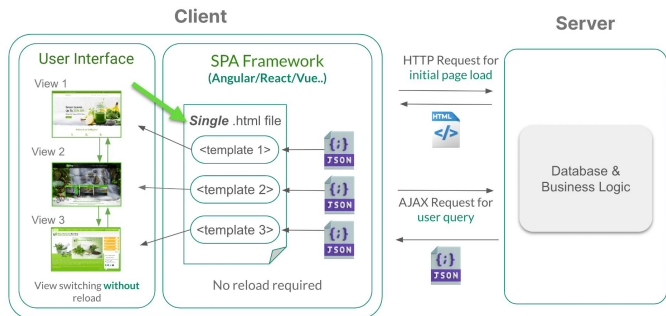
- Successful build/compilation
- Count of passing/failing test cases of 14 of 14 unchanged
- Execution times before and after LeakPair execution shared the same range for all 14 projects.

The patches are **non-intrusive**; they neither break functionality, nor incur execution time delays.



Summary

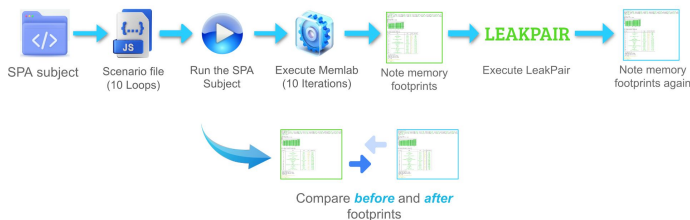
Single Page Application Architecture



Applying Fix Patterns



Effectiveness Evaluation



Effectiveness

- 8 of 10 subjects with *known* leaks had noticeable heap reductions
- 7 of 10 subjects with *unknown* leaks had **significant** heap reductions (4 -18%)
- All subjects had memory leak reductions

LeakPair patches can successfully prevent **leaks** and reduce **heap**, without leak detection

