Google Developers

V8: The Oz Story

Solving Performance Mysteries

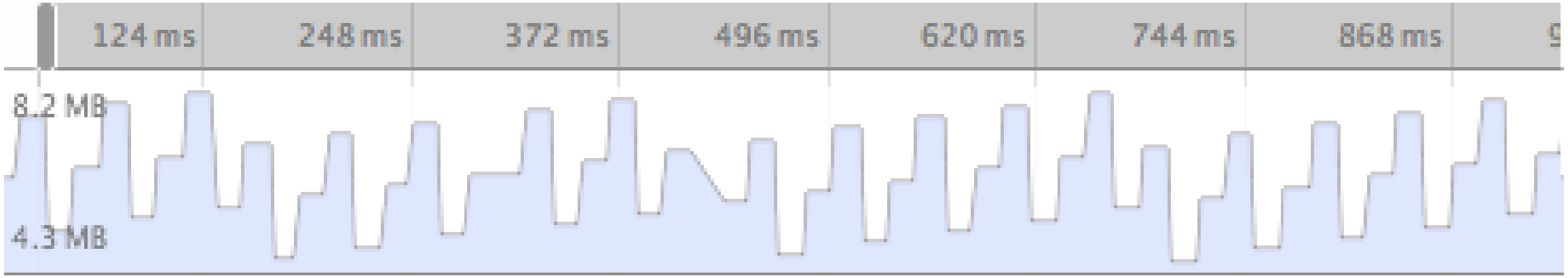
John McCutchan


MUSIC BOX

Find Your Way to Oz

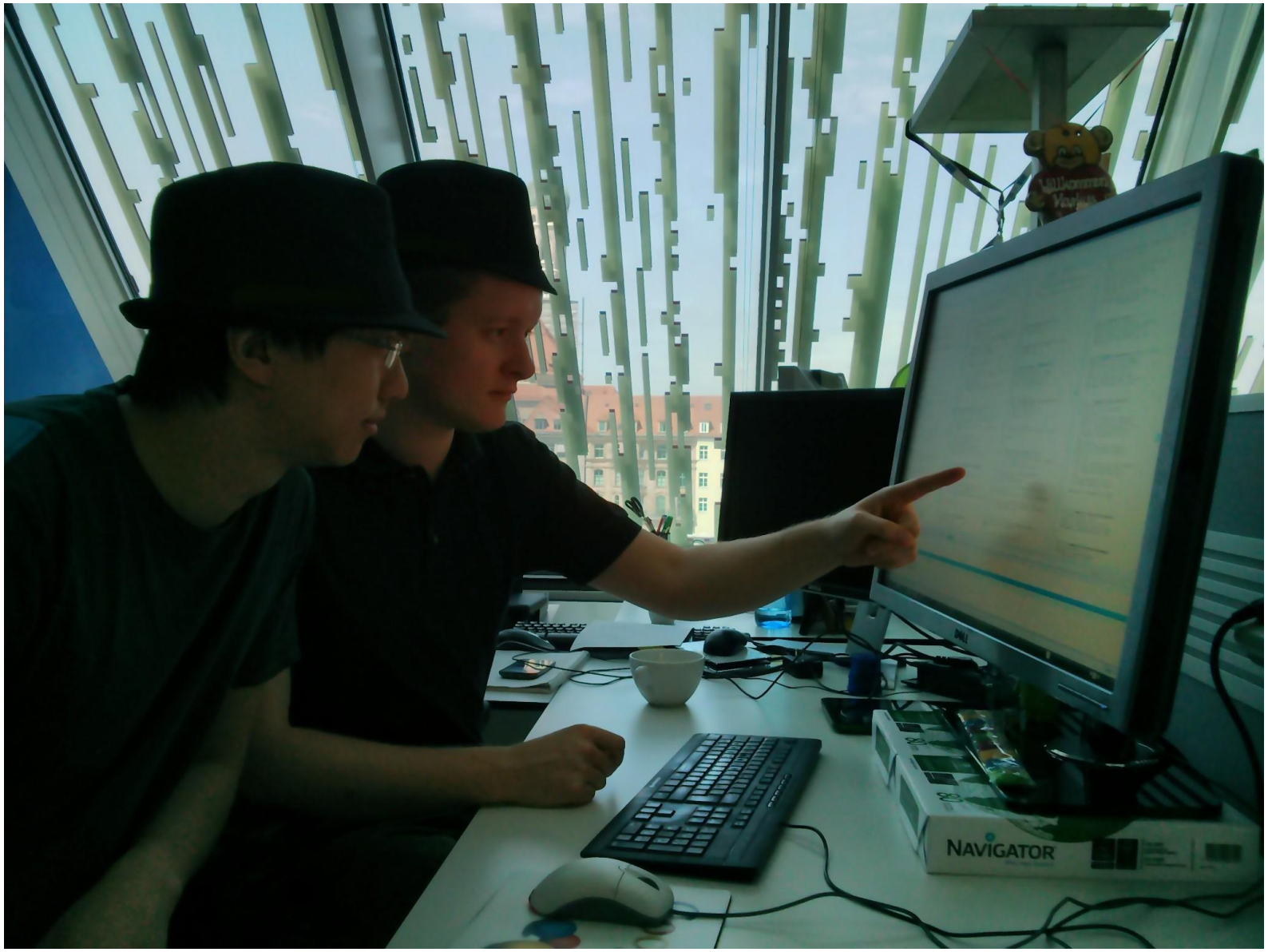
Late in development...

We've got a problem!



... didn't want to delay ...

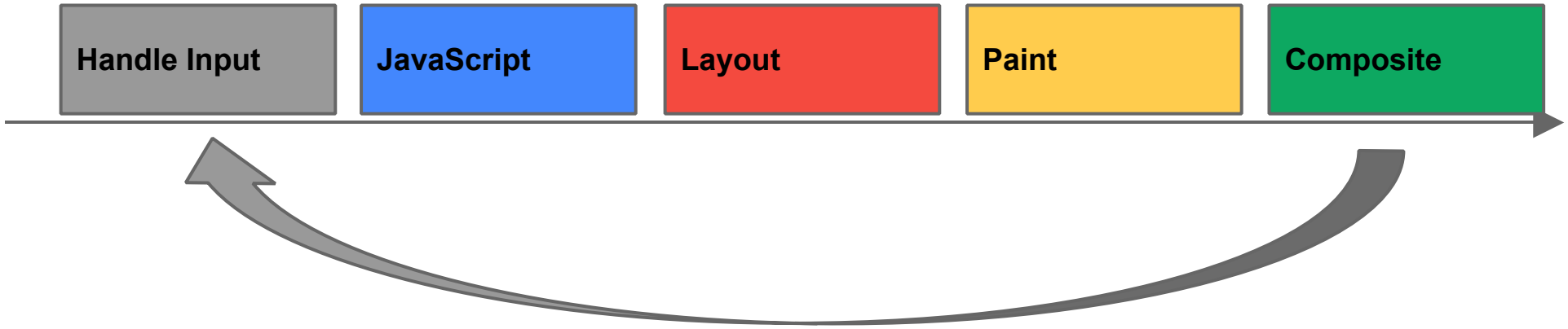
... called in performance detectives



Why Performance Matters

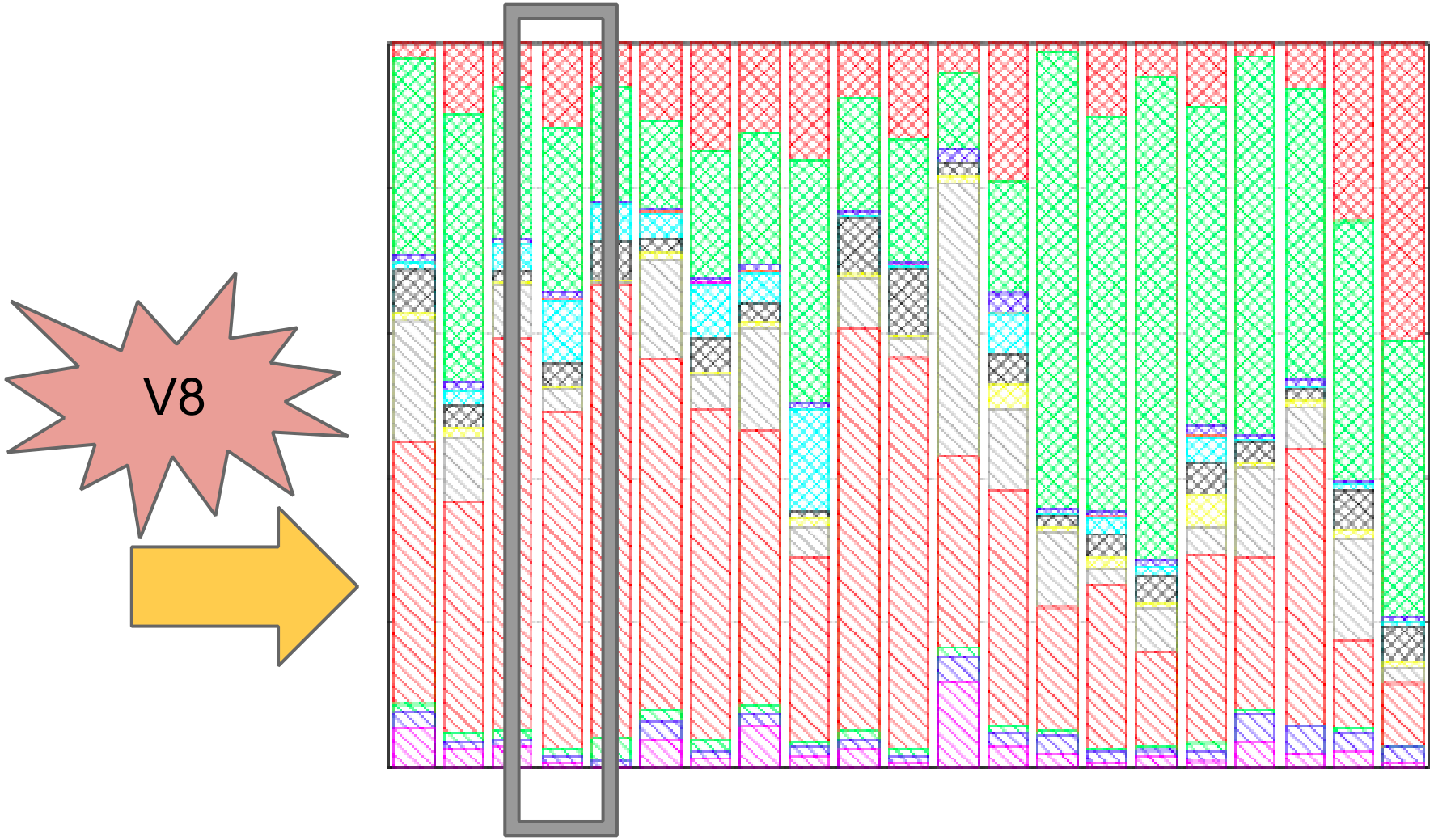
60 times a second

16 Milliseconds



JANK

Performance Matters



JavaScript execution time:

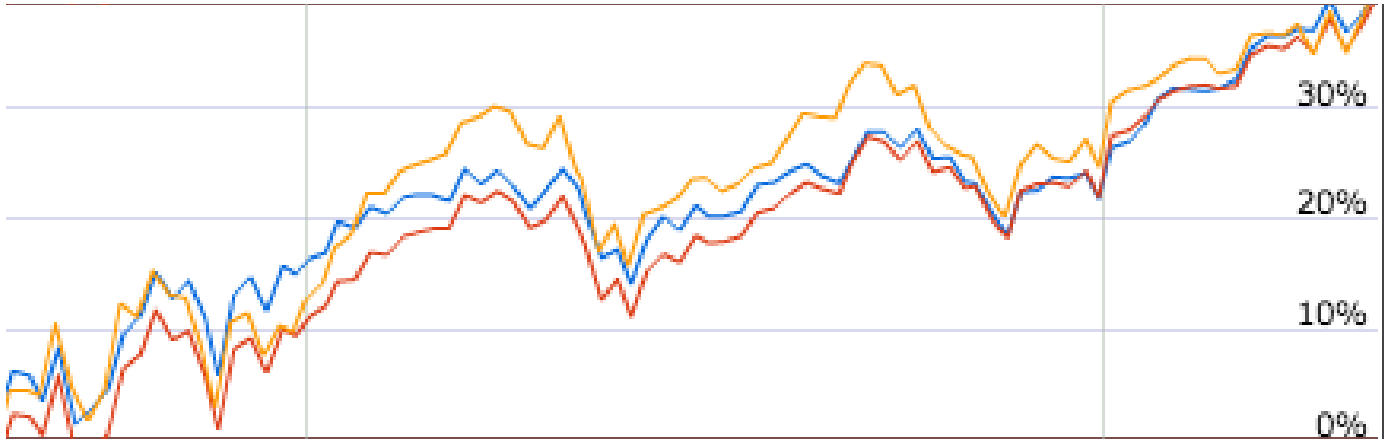
Google apps spend 50-70% of time in V8

Popular sites 20%-40% of time in V8

Longer Battery Life

Smoother Applications

More Features



Performance Mysteries

Solving Crimes, Performance Crimes.

EVIDENCE COLLECTION

SUSPECTS

FORENSICS LAB

Evidence Collection

What Kind of Application is Oz?

Real-time interactive 3D game

Are the Developers Following Best Practices?

Yes

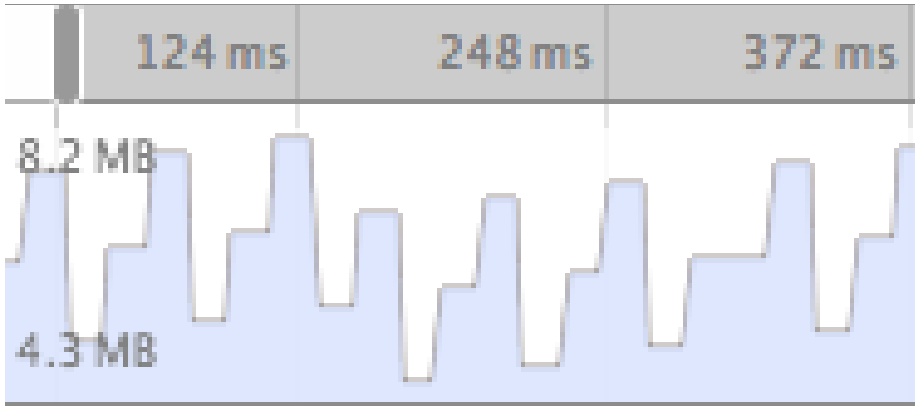
What Kind of Performance Problem are we Seeing?

Frame rate drop, once per second.

Correlated with GC activity

Is 10MB/sec of Garbage Expected?

No



What Triggers a Garbage Collection?

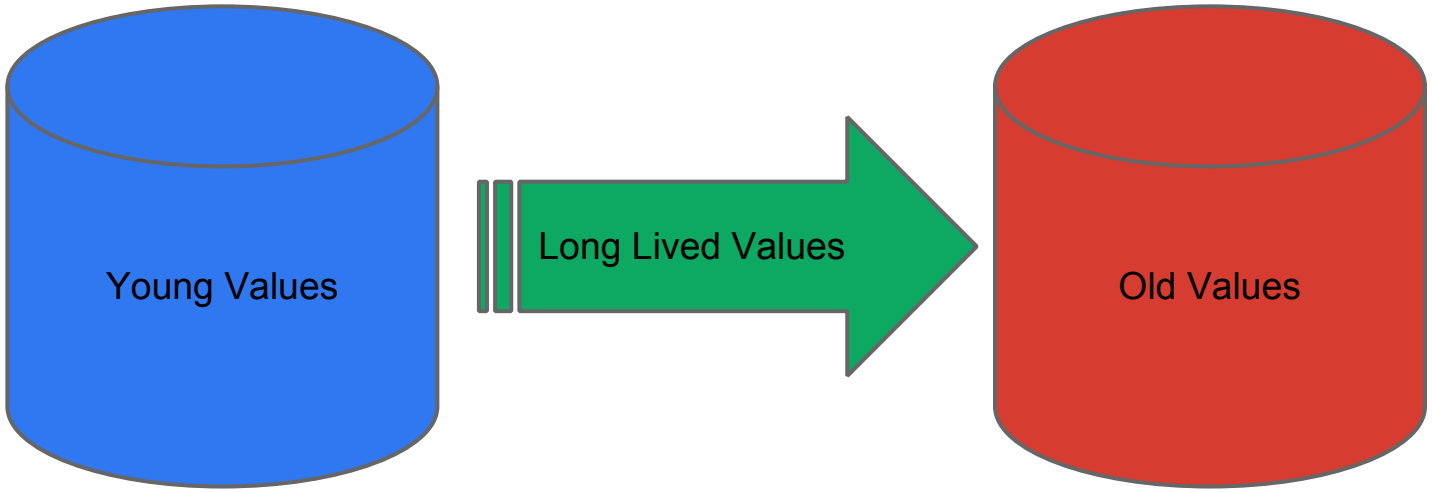
V8 Memory Management

A GC Pause Walkthrough

Where is the cost in allocating memory?

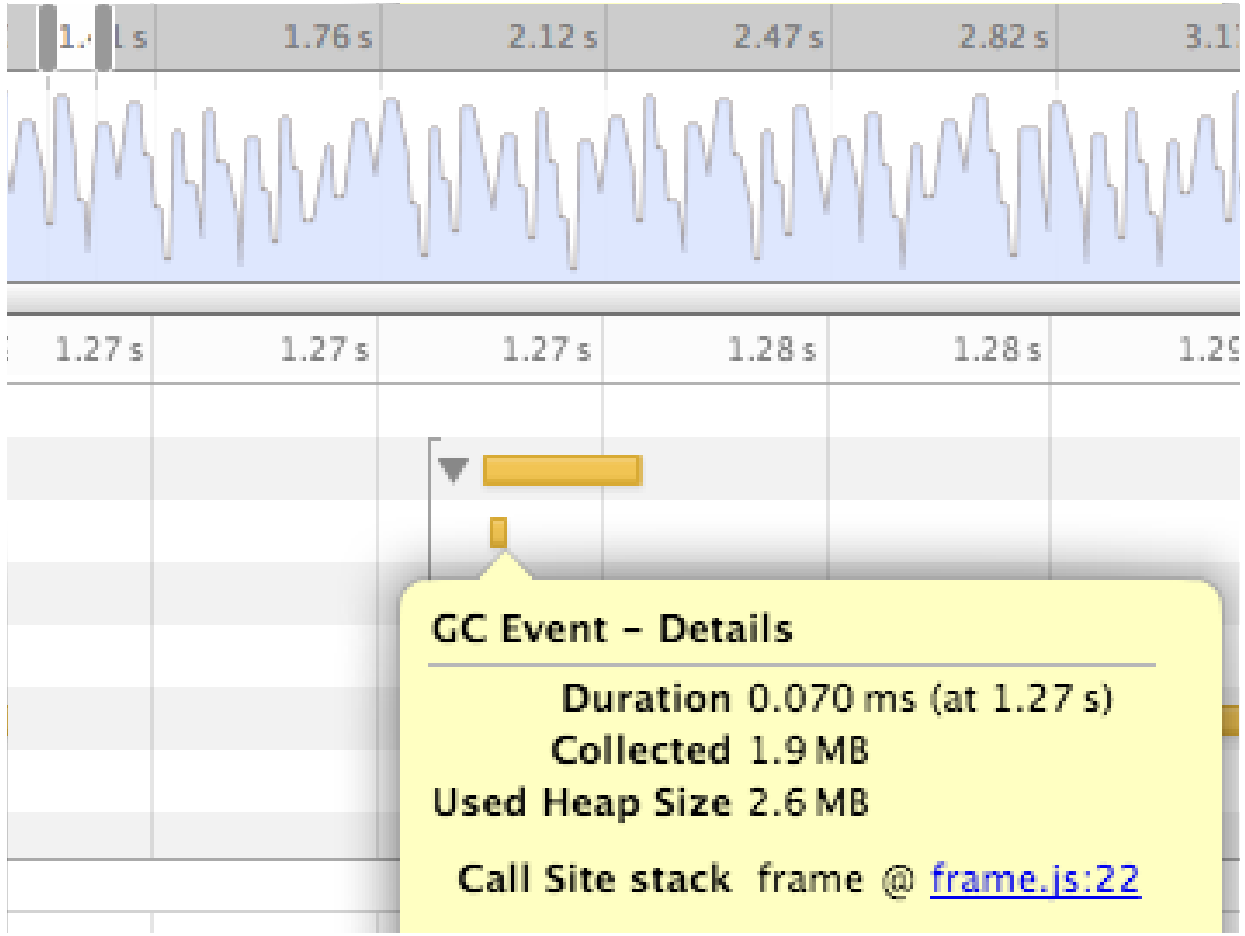
- Every call to new or implicit memory allocation
  - Reserves memory for object
  - Cheap until...
- Memory pool **exhausted**
  - Runtime forced to perform a garbage collection
  - Can take milliseconds (!)
- Applications must be careful with object allocation patterns
  - Every allocation brings you closer to a GC pause

- Generational
  - Split values between young and old
  - Overtime young values promoted to old



How does V8 manage memory?

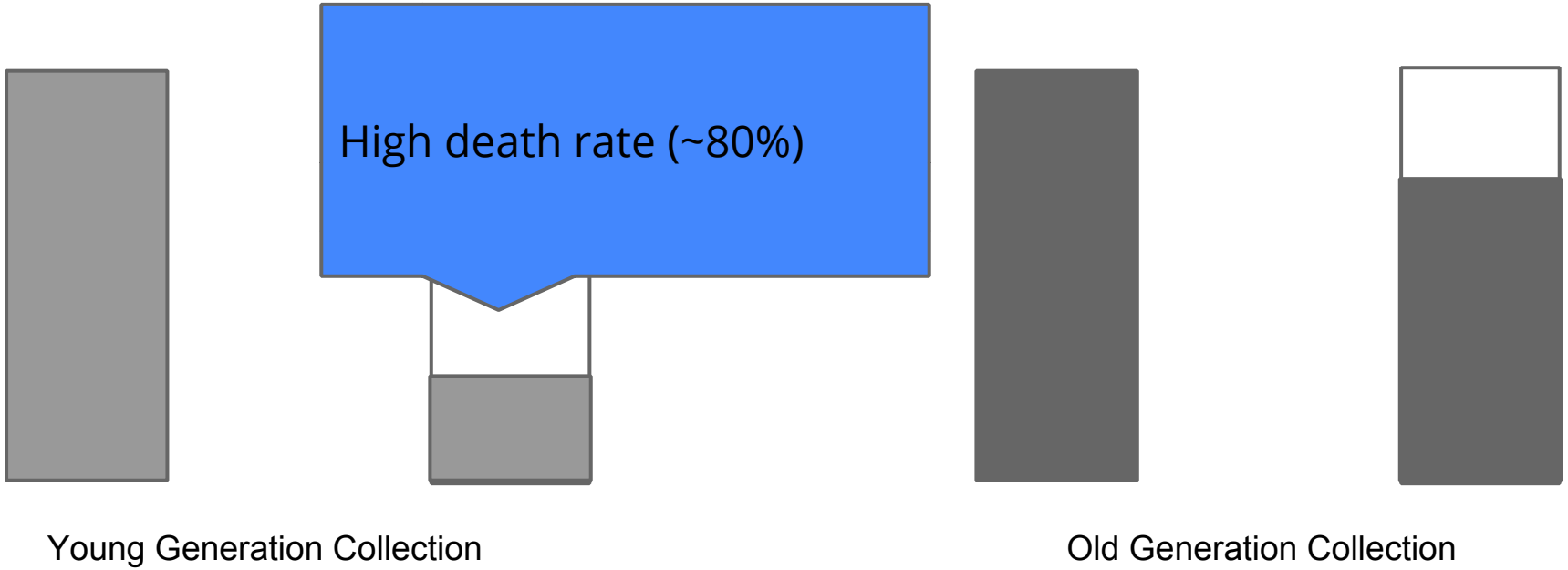
- Young Generation
  - Fast allocation
  - Fast collection
  - Frequent collection



- Old Generation
  - Fast allocation
  - Slower collection
  - **Infrequently** collected

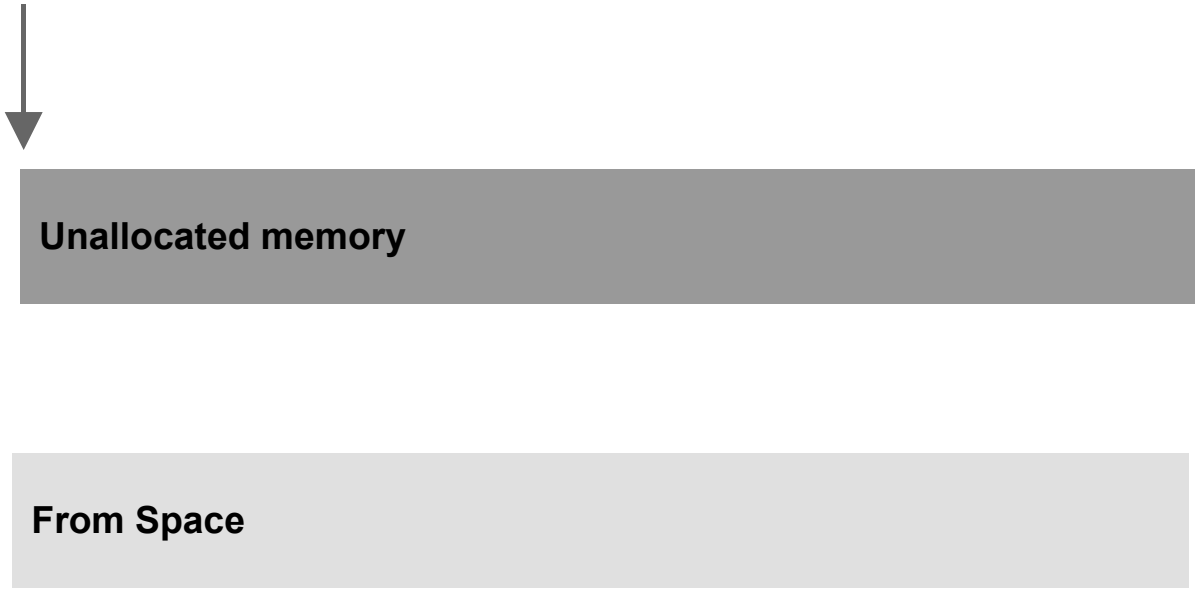
- Parts of collection run concurrently with mutator
  - Incremental Marking
- Mark-sweep
  - Return memory to system
- Mark-compact
  - Move values

- Why is collecting the young generation faster
  - Cost of GC is proportional to the number of live objects

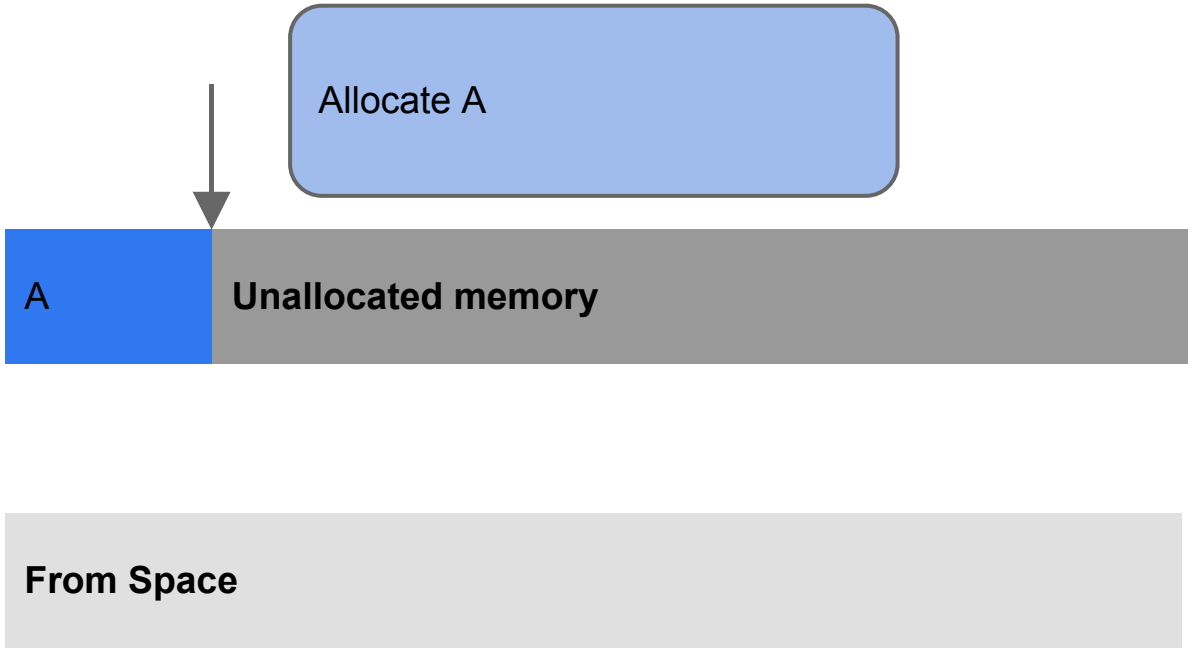


Young Generation In Action

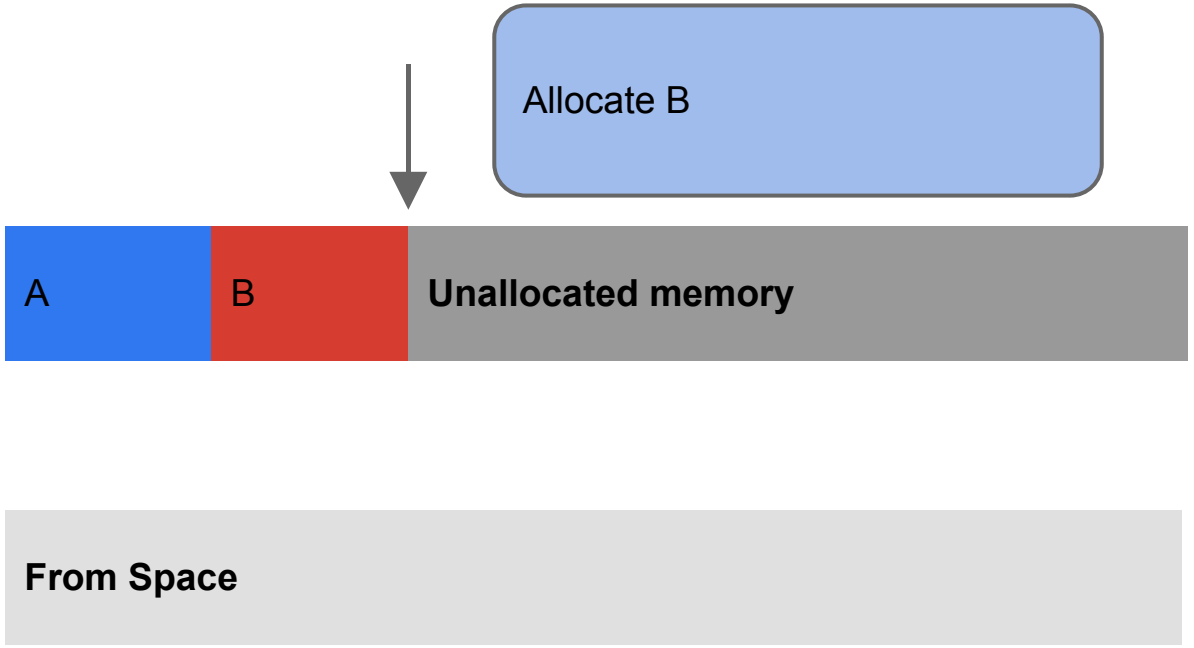
Young Generation In Action



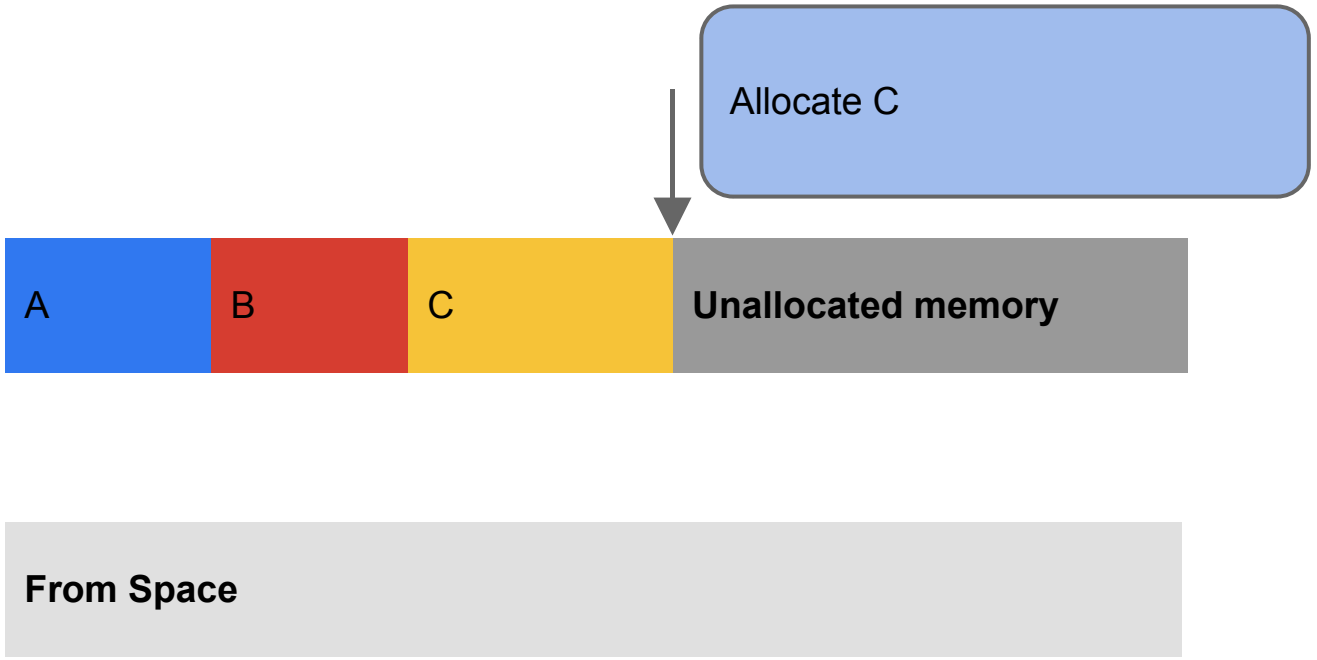
Young Generation In Action



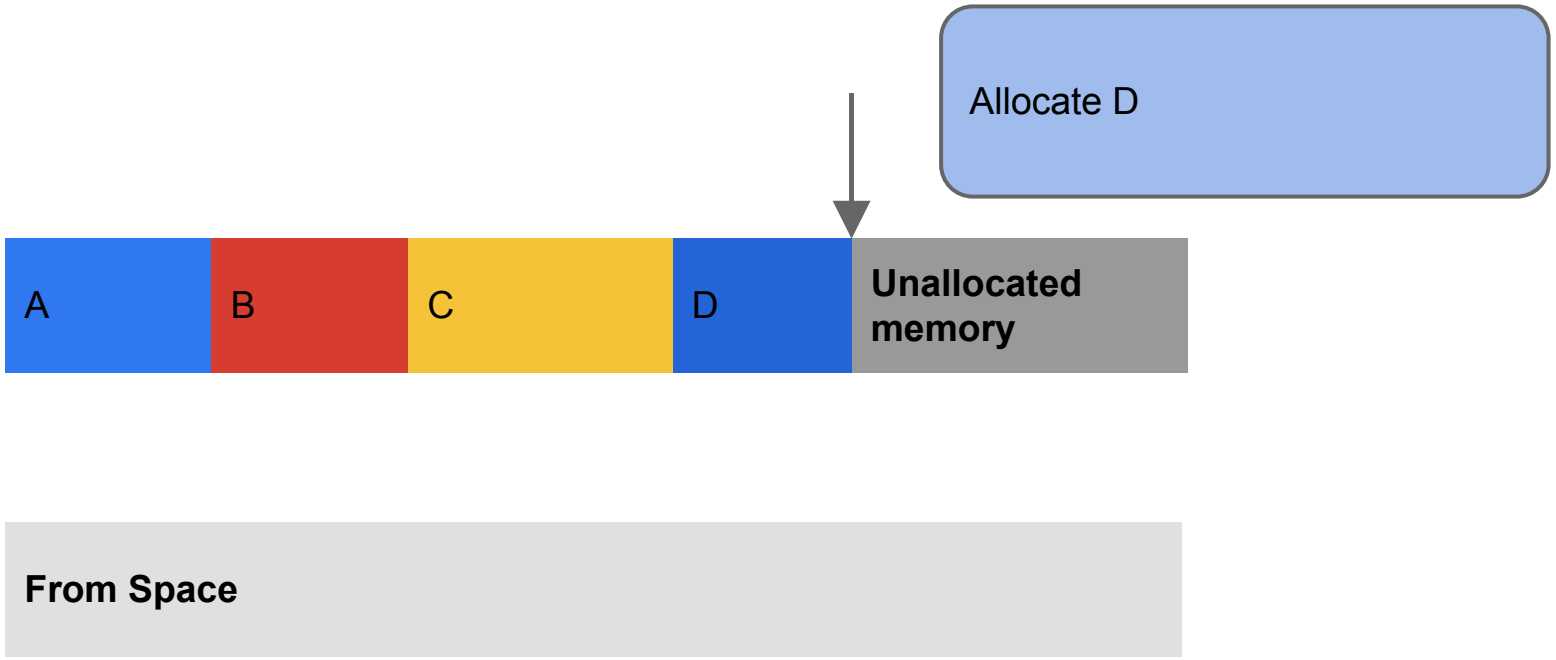
Young Generation In Action



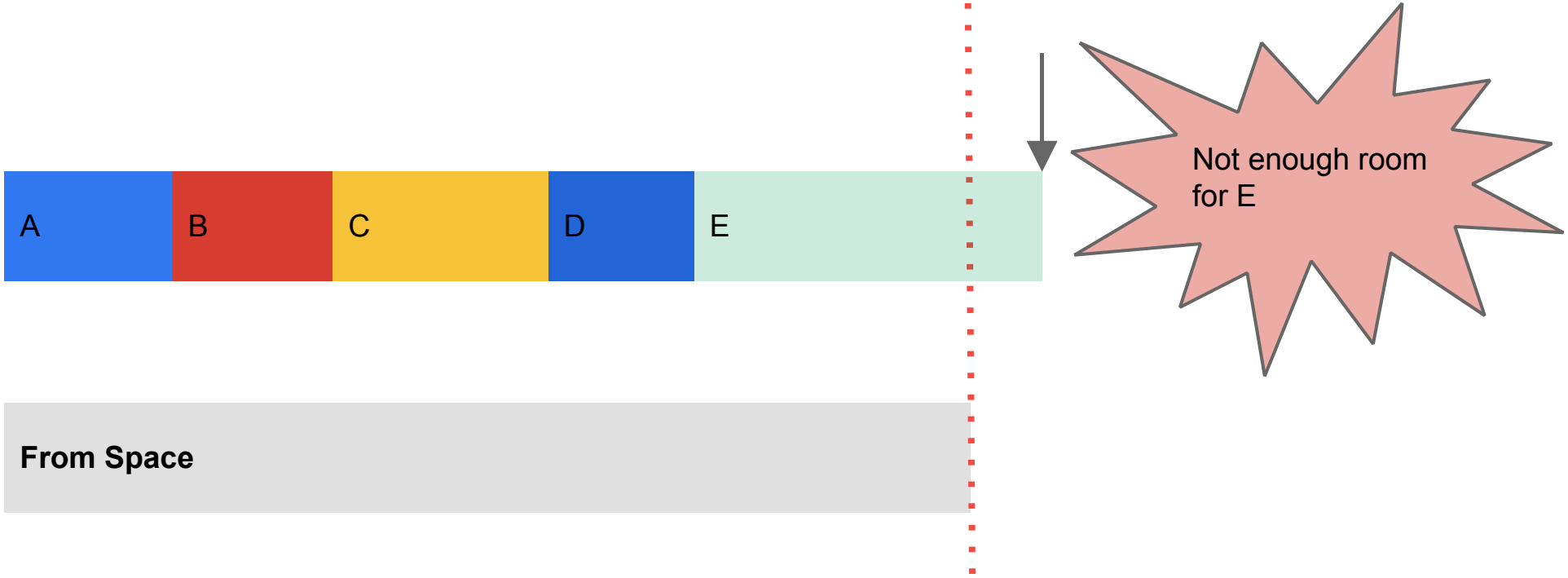
Young Generation In Action



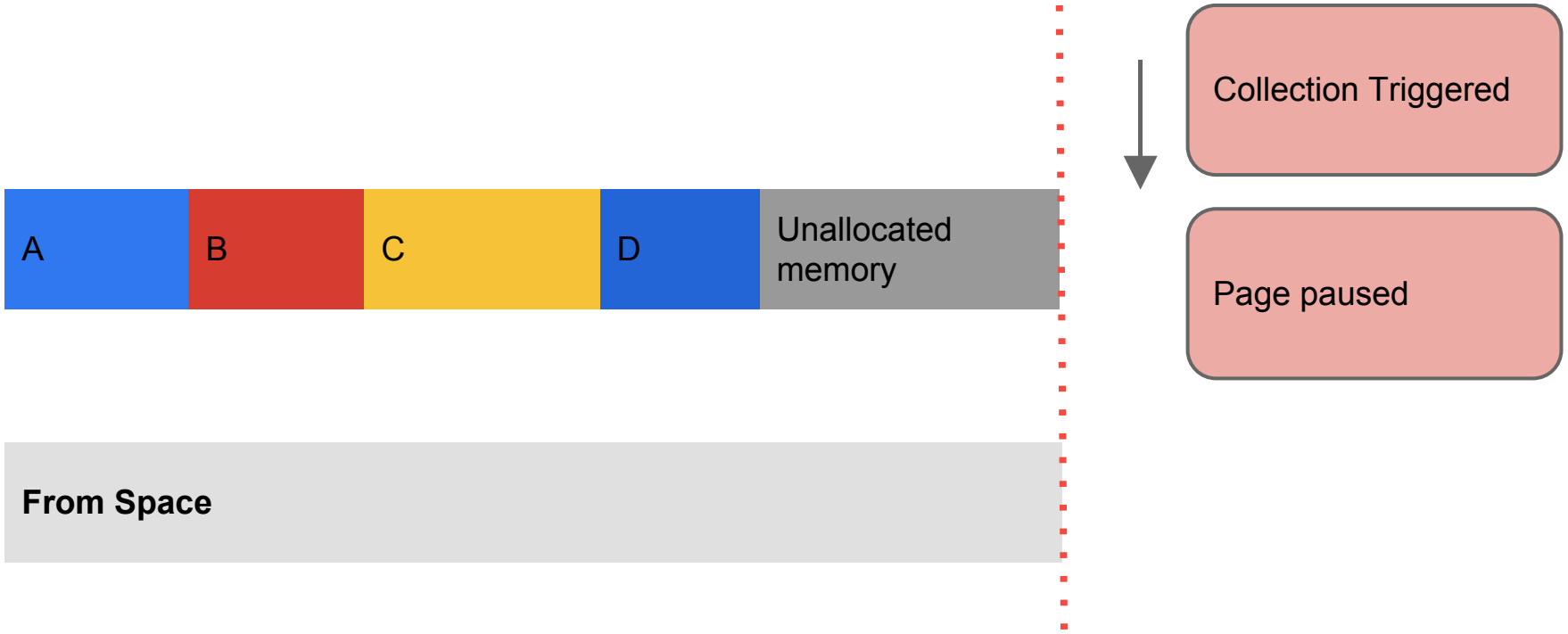
Young Generation In Action



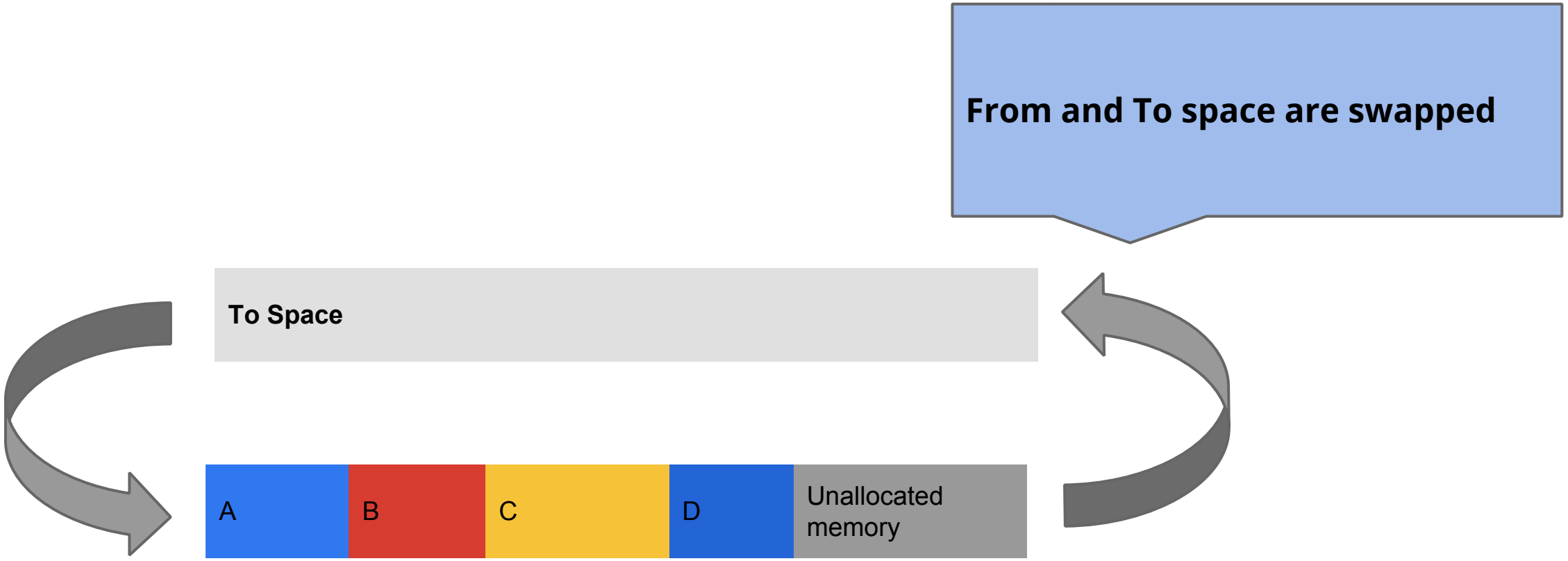
Young Generation In Action



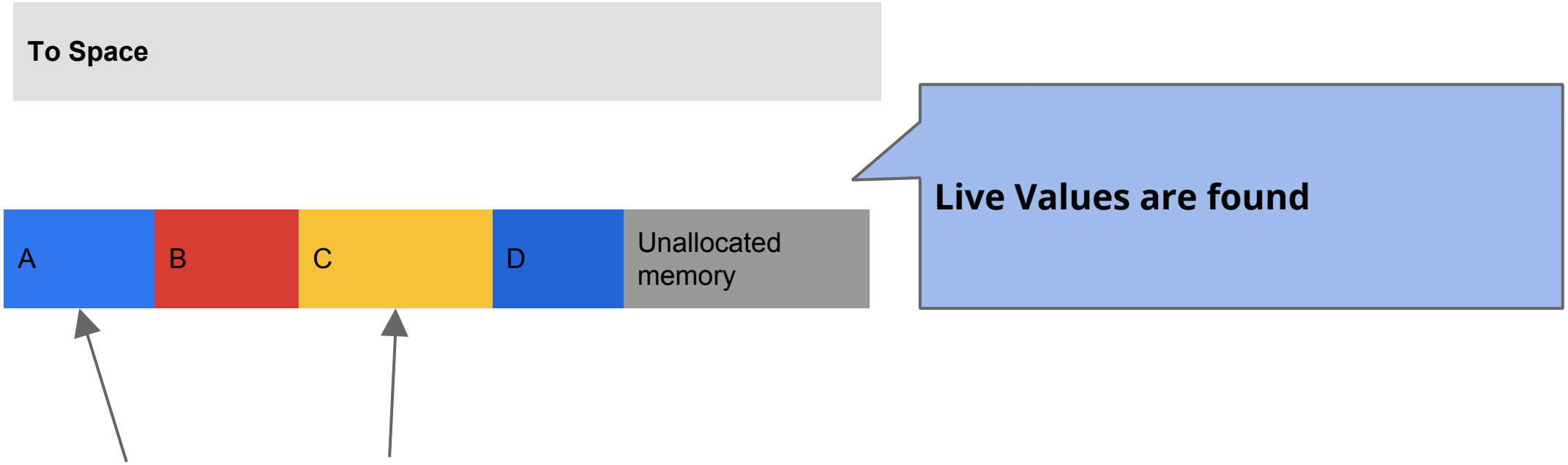
Young Generation In Action



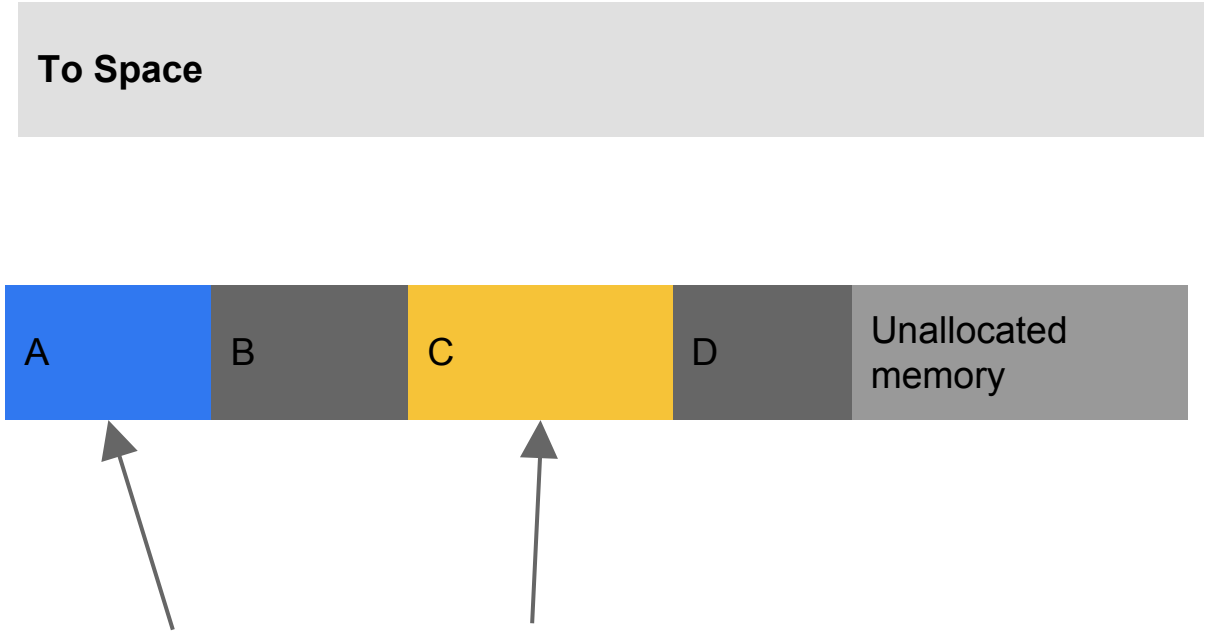
Young Generation In Action



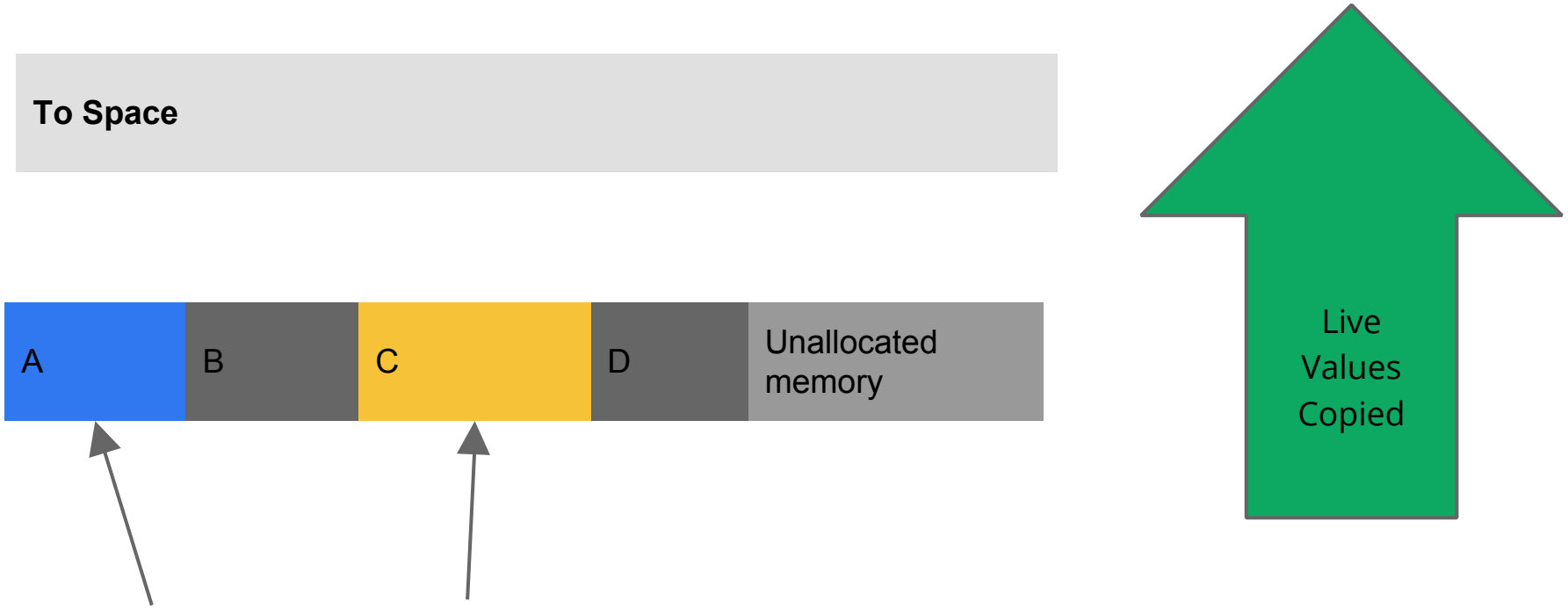
Young Generation In Action



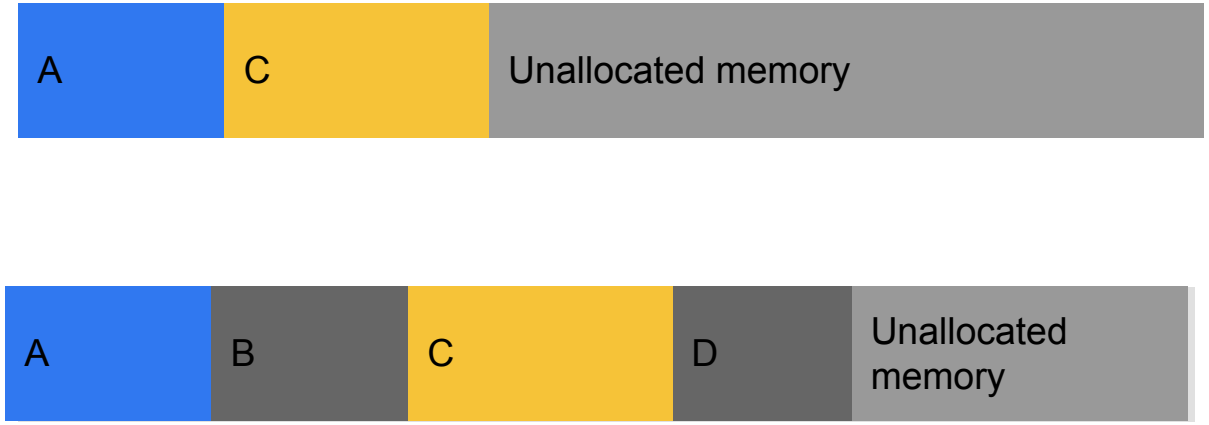
Young Generation In Action



Young Generation In Action

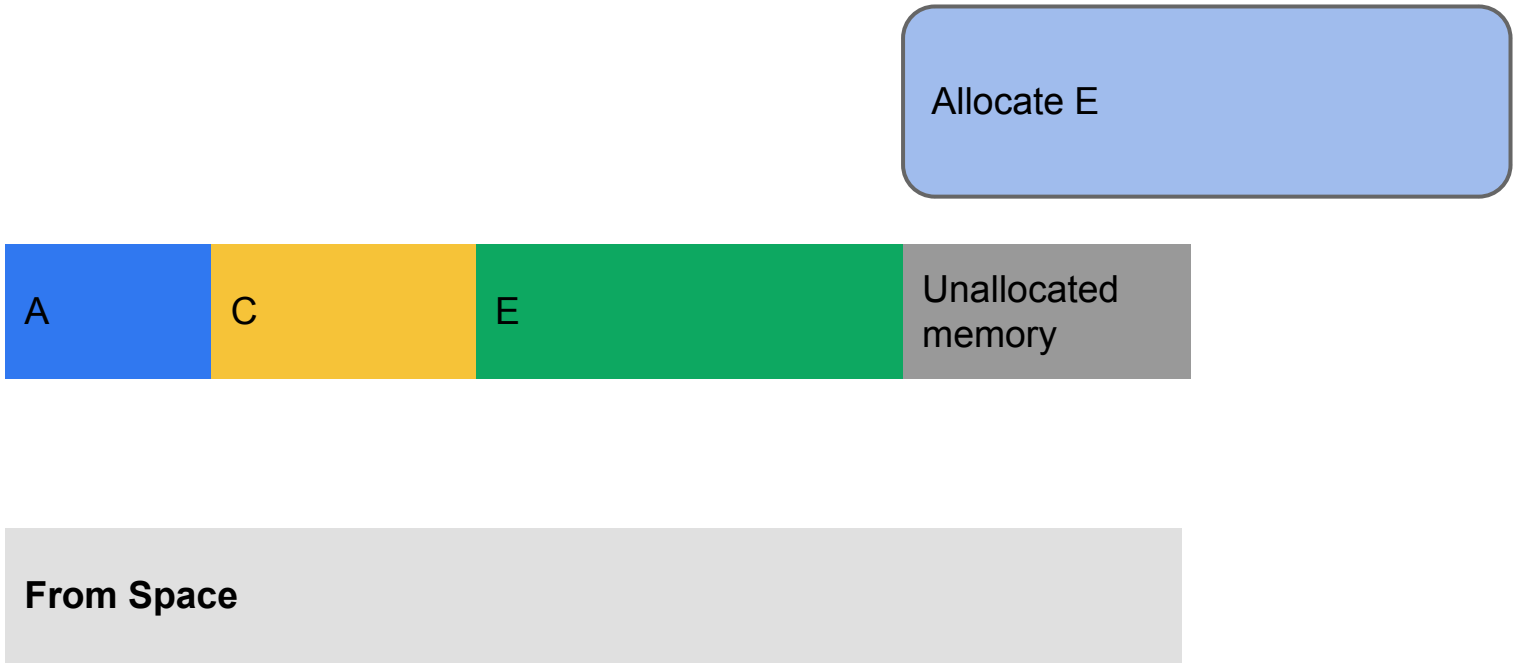


Young Generation In Action



Young Generation In Action

Young Generation In Action



- Each allocation moves you closer to a collection
  - Not always obvious when you are allocating
- Collection pauses your application
  - Higher latency
  - Dropped frames
  - Unhappy users

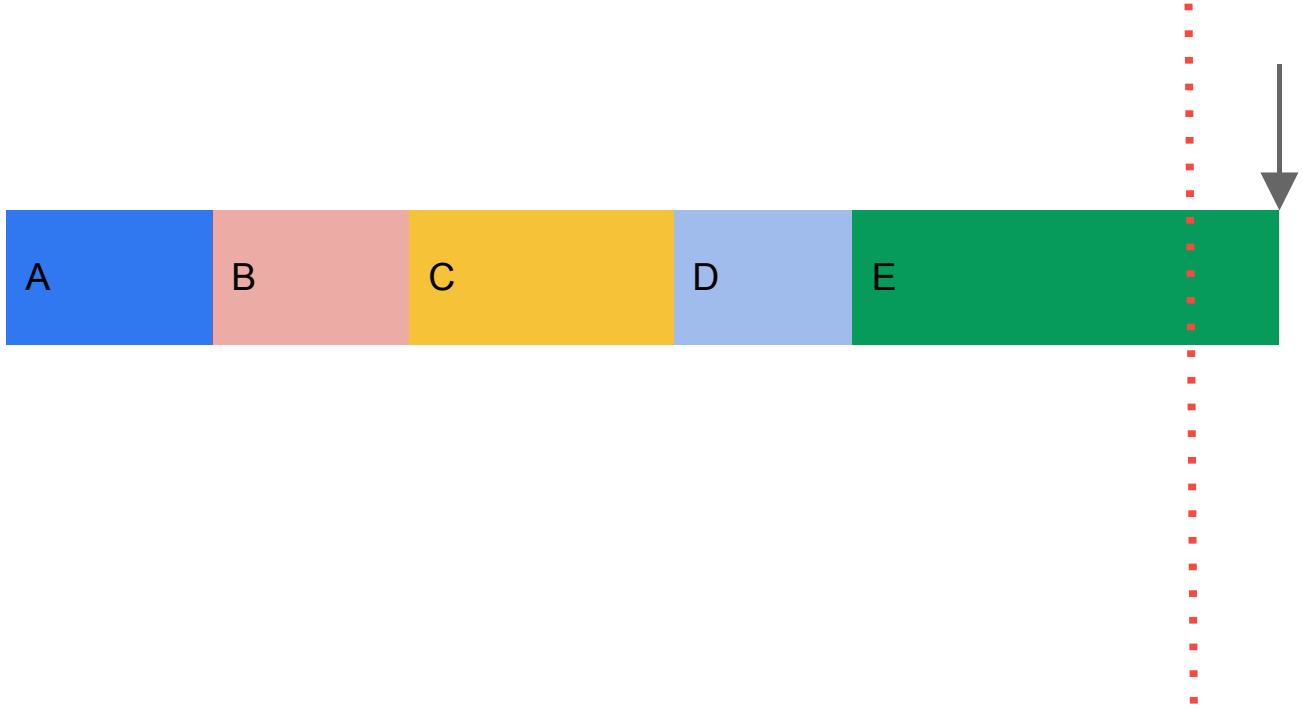
Suspects

What could be causing frequent GC pauses?

SUSPECTS

SUSPECT #1

Calling new



SUSPECT #1

Audit confirmed no calls to new within frame

"That would have been embarrassing" - UNIT9

SUSPECT #2

Code running in un-optimized mode

JavaScript

```
var a = p * d;
var b = c + 3;
var c = 3.3 * dt;

point.x = a * b * c;
```

SUSPECT #2

Unoptimized mode

JavaScript

```
var a = p * d;
var b = c + 3;
var c = 3.3 * dt;

point.x = a * b * c;
```

Implicit memory allocation

SUSPECT #2

Optimized Mode

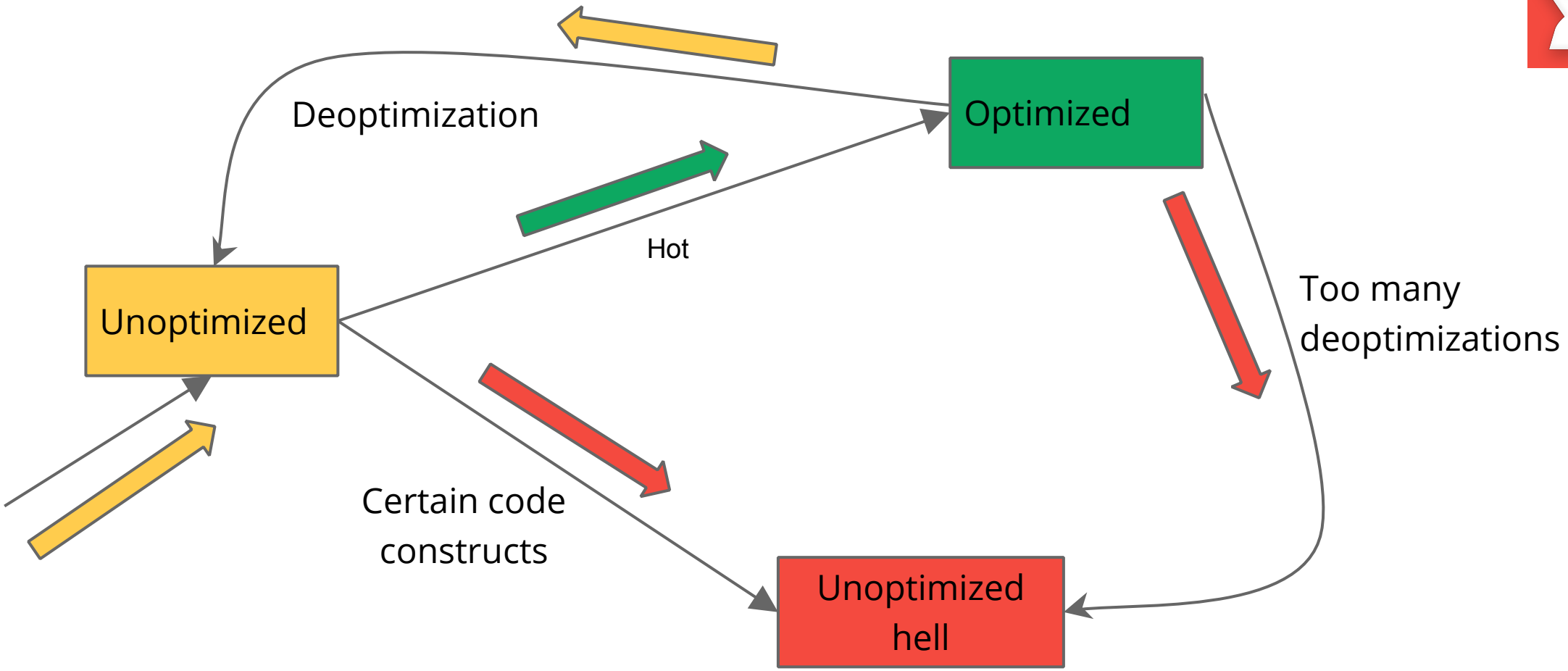
JavaScript

```
var a = p * d;
var b = c + 3;
var c = 3.3 * dt;

point.x = a * b * c;
```

Implicit memory allocation

V8 recently optimized for this case

Transitions between optimized and unoptimized mode



SUSPECT #2

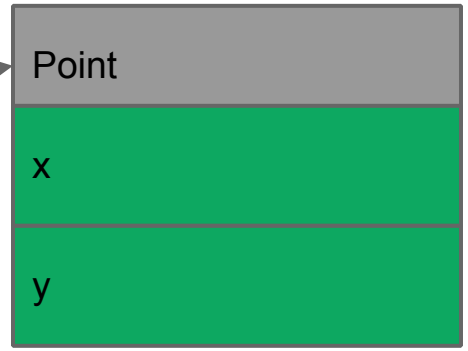
Potential Suspect!

SUSPECT #3

Modifying Object Shape

JavaScript

```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

var p = new Point(2.3, 4.5);

p.z = 9.9;
```

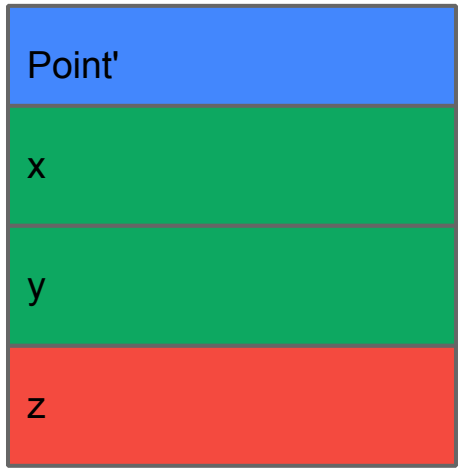


SUSPECT #3

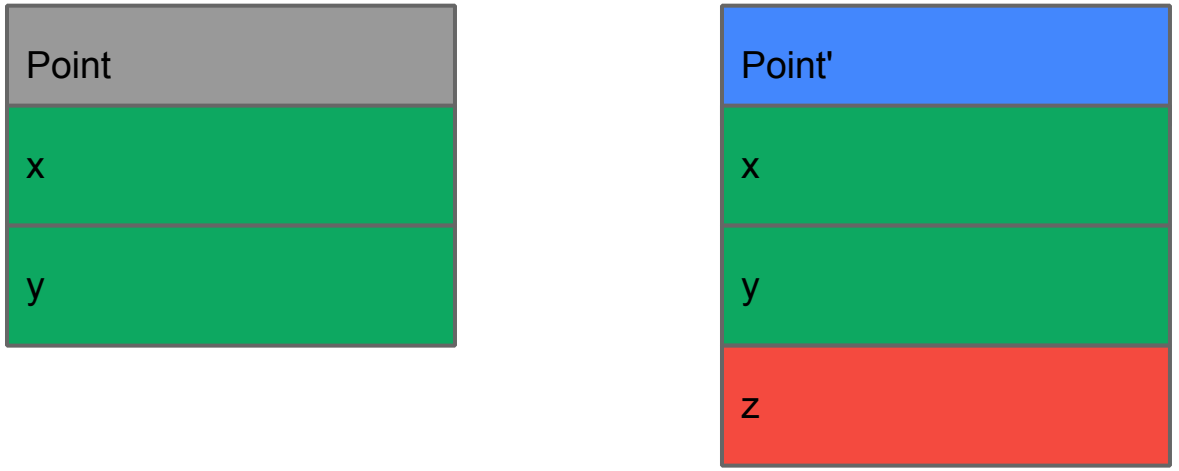
Modifying Object Shape

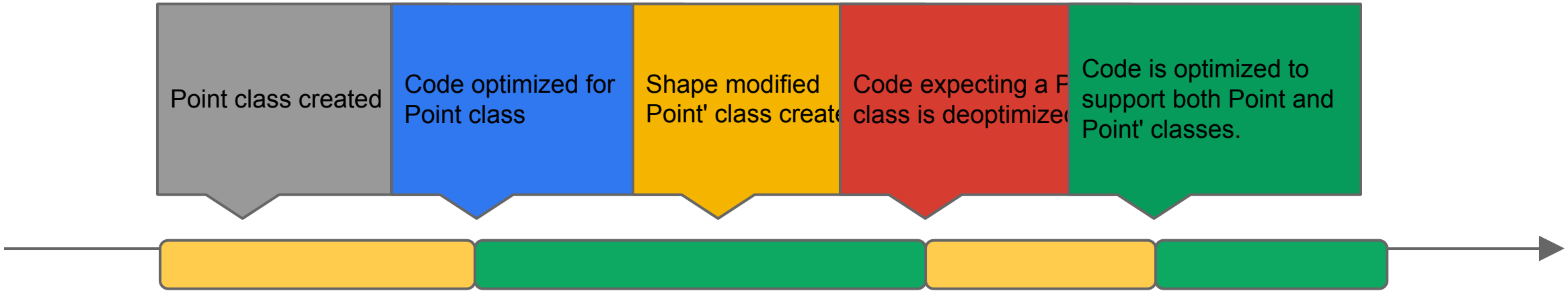
JavaScript

```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

var p = new Point(2.3, 4.5);

p.z = 9.9;
```



SUSPECT #3


Point
x
y
Point'
x
y
z


Point class created
Code optimized for Point class
Shape modified Point' class create
Code expecting a P class is deoptimize
Code is optimized to support both Point and Point' classes.

SUSPECT #3

Audit confirmed no shape changes

Suspect #1: Calling New

Alibi: Not at crime scene

Suspect #2: Unoptimized mode

Alibi: None

Suspect #3: Shape Change

Alibi: Not at crime scene

Forensics

Forensics

- **Chrom**

Forensics

- **Confirm unoptimized code is running**
- **Determine why code isn't optimized**

Forensics - Capturing V8 timeline

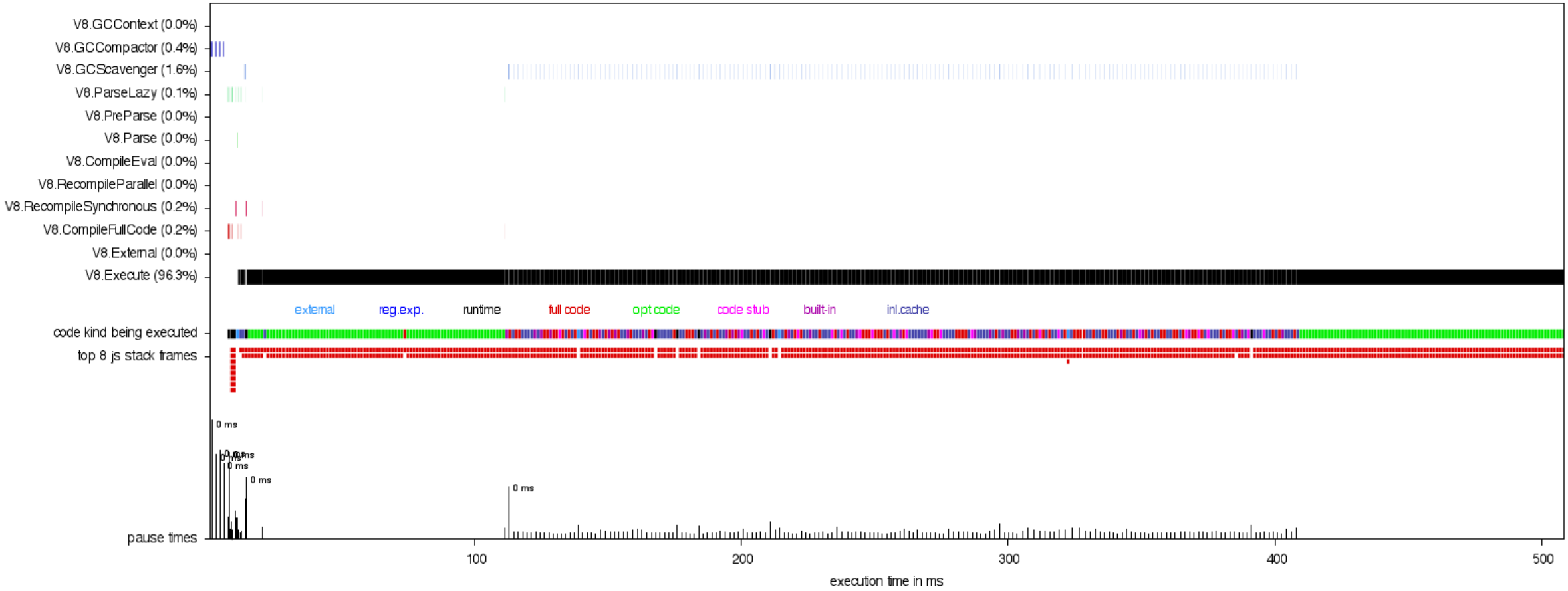
Command Line

```
$ Chrome --no-sandbox --js-flags="--prof --noprof-lazy --log-timer-events"
```

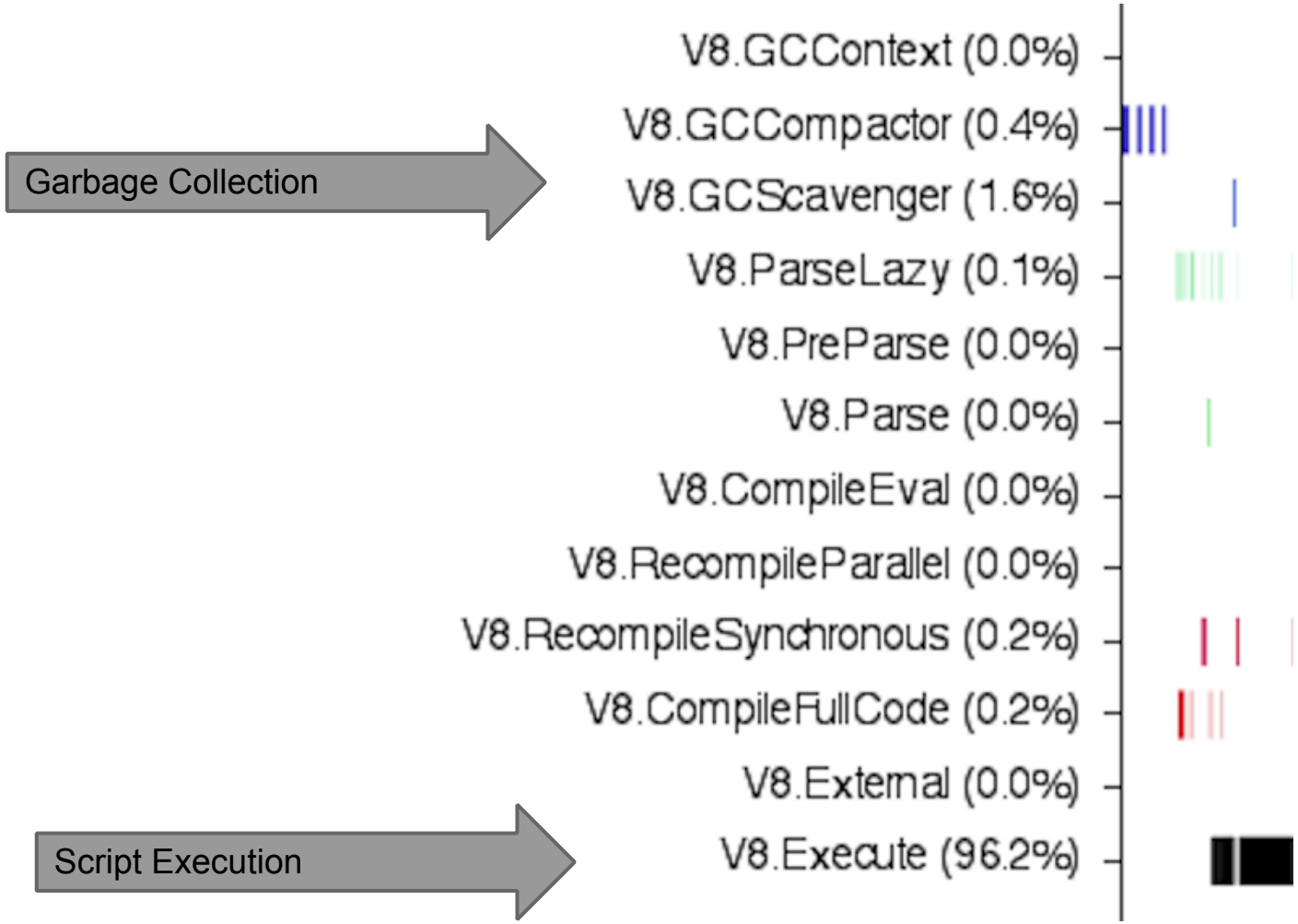
Command Line

```
$ tools/plot-timer-events /path/to/v8.log
```

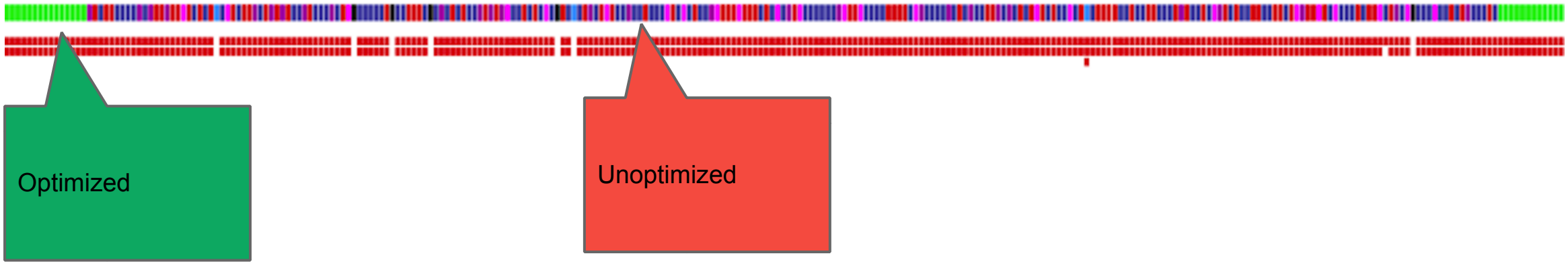
Forensics - Analyzing V8 timeline



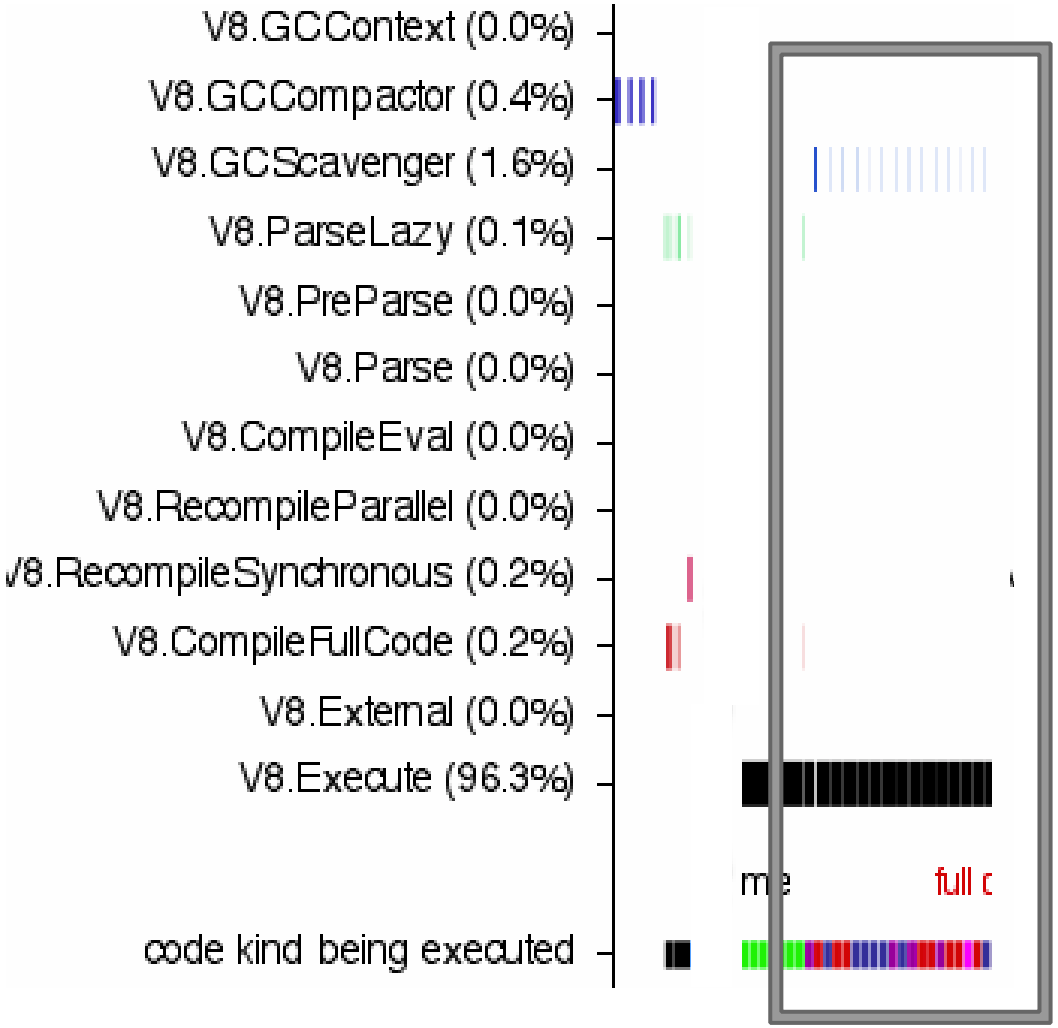
Forensics - Analyzing V8 timeline



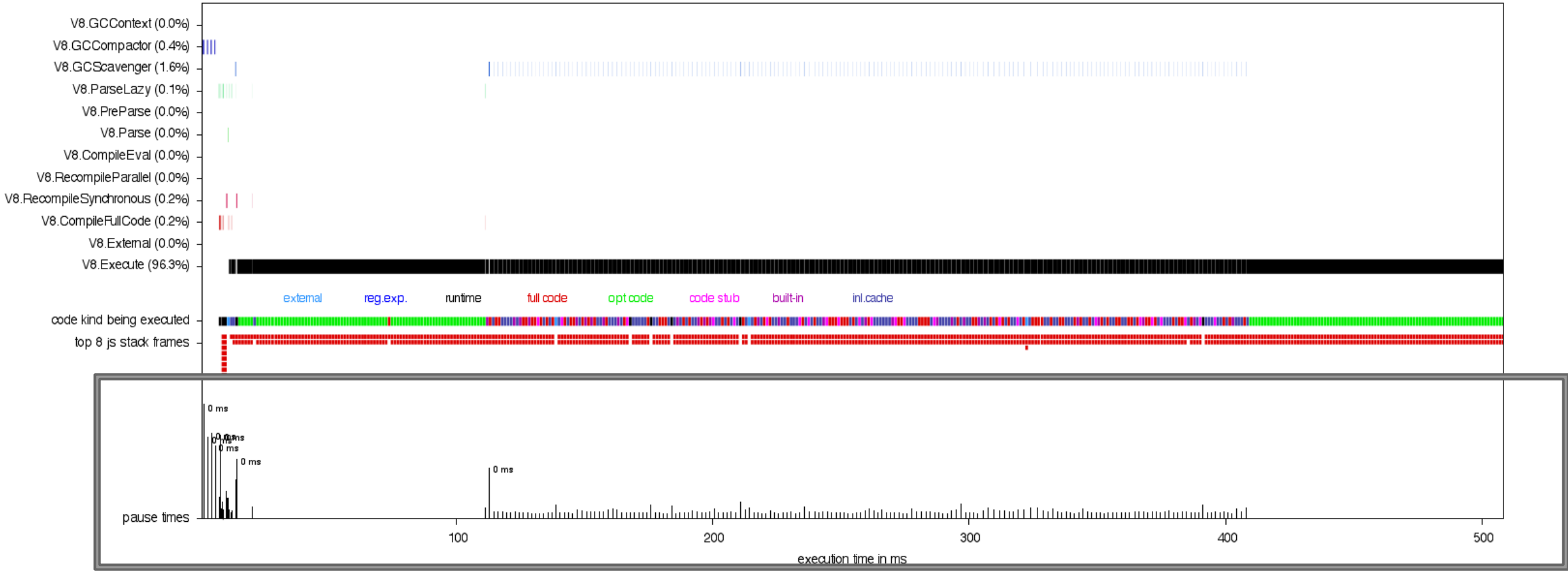
Code Kind



Forensics - Analyzing V8 timeline



Forensics - Analyzing V8 timeline



Forensics - Finding hot unoptimized functions

Command Line

```
$ Chrome --no-sandbox --js-flags="--prof --noprof-lazy --log-timer-events"
```

Command Line

```
$ tools/mac-tick-processor /path/to/v8.log
```

Forensics - Finding hot unoptimized functions

Command Line

```
[JavaScript]:
  ticks  total  nonlib   name
   167   61.2%   61.2%  LazyCompile: updateSprites source.js:12
    40   14.7%   14.7%  LazyCompile: *drawSprites source.js:20
    15    5.5%    5.5%  Stub: K
    13    4.8%    4.8%  Stub: B                                  nber+Smi
     6    2.2%    2.2%  Stub: BinaryOpStub_ADD_OverwriteRight_Number+Number
     4    1.5%    1.5%  Stub: KeyedStoreElementStub
     4    1.5%    1.5%  KeyedLoadIC:  {12}
     2    0.7%    0.7%  KeyedStoreIC:  {13}
     1    0.4%    0.4%  LazyCompile: ~main source.js:30
```

* indicates optimized function

Forensics - Determining why a function is not optimized

Command Line

```
$ Chrome --no-sandbox --js-flags="--trace-deopt --trace-opt-verbose"
```

Command Line

```
[disabled optimization for updateSprites, reason: ForInStatement is not fast case]
```

Equivalent of Oz problem code:

JavaScript

```
function updateSprites(dt) {
  for (var sprite in sprites) {
    sprite.position.x += sprite.velocity.x * dt; // update position
}
```

Function not optimized because of this loop construct.

Potential Fix

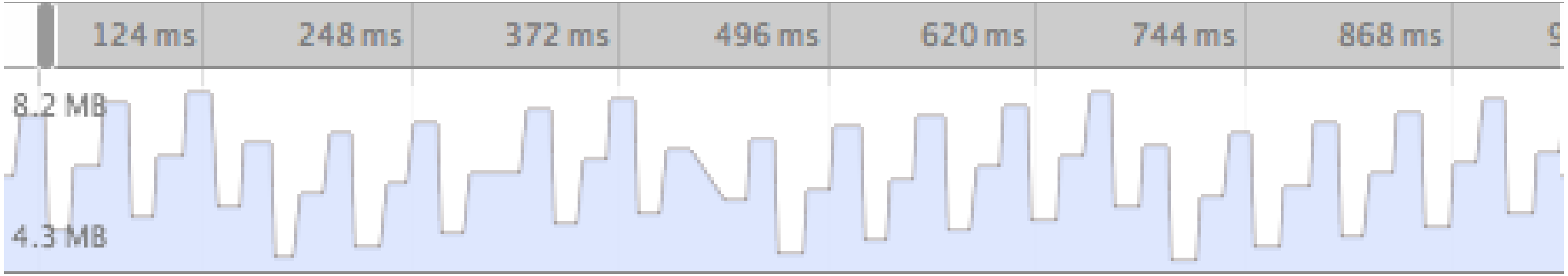
JavaScript

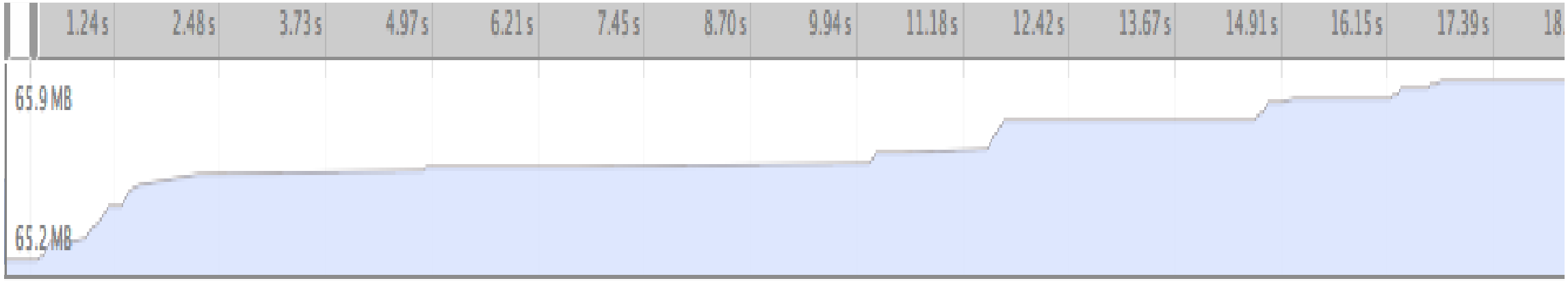
```
function updateSprite(sprite, dt) {
    sprite.position.x += sprite.velocity.x * dt; // update position
    // many more lines of arithmetic.
}

function updat
  for (sprite
    updateSpri
  }
}
```

Arithmetic in optimized function.

Unoptimized function now only calls function

Before and After






CASE
CLOSED

- **Simple fix**
  - **GC pause problem solved**
  - **Real problem was unoptimized code**
- **Oz devs understood how to look under the hood**
  - **Identified other functions in "deoptimization hell"**

Conclusion

- **Timeline plot**
  - **Birds eye view of V8 activity**
- **Tick processor**
  - **Table of hot functions**
- **Deoptimization log**
  - **Deep insight into optimization state machine**

Conclusion

- **Evidence Collection**
  - **Asking the right questions**
- **Suspects**
  - **Narrowing in on likely cause**
- **Forensics**
  - **Using tools to prove your case**

<Thank You!>

johnmccutchan@google.com

twitter.com/johnmccutchan

google.com/+JohnMcCutchan

Check out Perf Alley and Chrome Office Hours

Chrome DevTools:

- https://developers.google.com/chrome-developer-tools/

V8 Tools:

- https://code.google.com/p/v8/

Structural Profiling JS:

- http://www.youtube.com/watch?v=nxXkquTPng8


Google
Developers