[Node-internals]
V8 & GC

Hello!

Vincent Vallet

NodeJs & performance evangelist

@Vince_Vallet

Introduction

Garbage Collector .... What is this ?

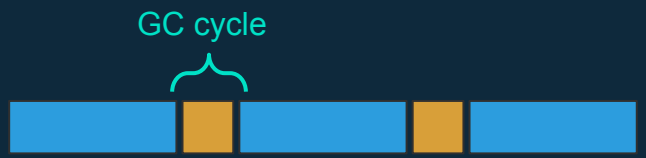
Good point : With NodeJS, no need to manage memory !

=> Garbage Collector (GC) = take care of memory !

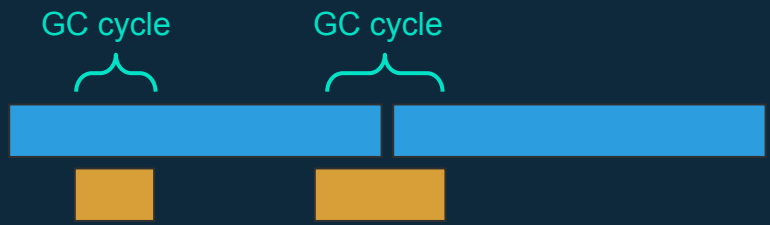
Type of GC

Type	Description
1	Scavenge (Minor GC)
2	Mark Sweep compact (Major GC)
4	Incremental marking / Lazy sweeping
8	Weak/Phantom callback processing
15	All

Stop the world

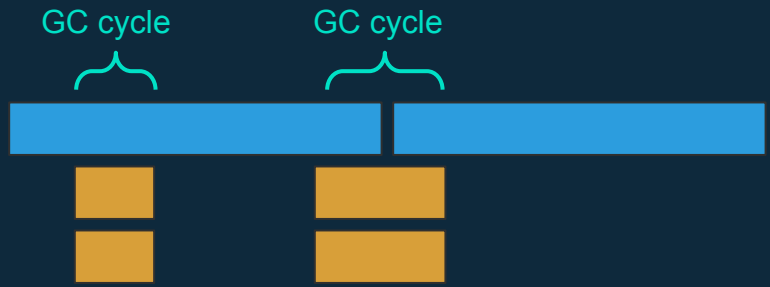


Concurrent



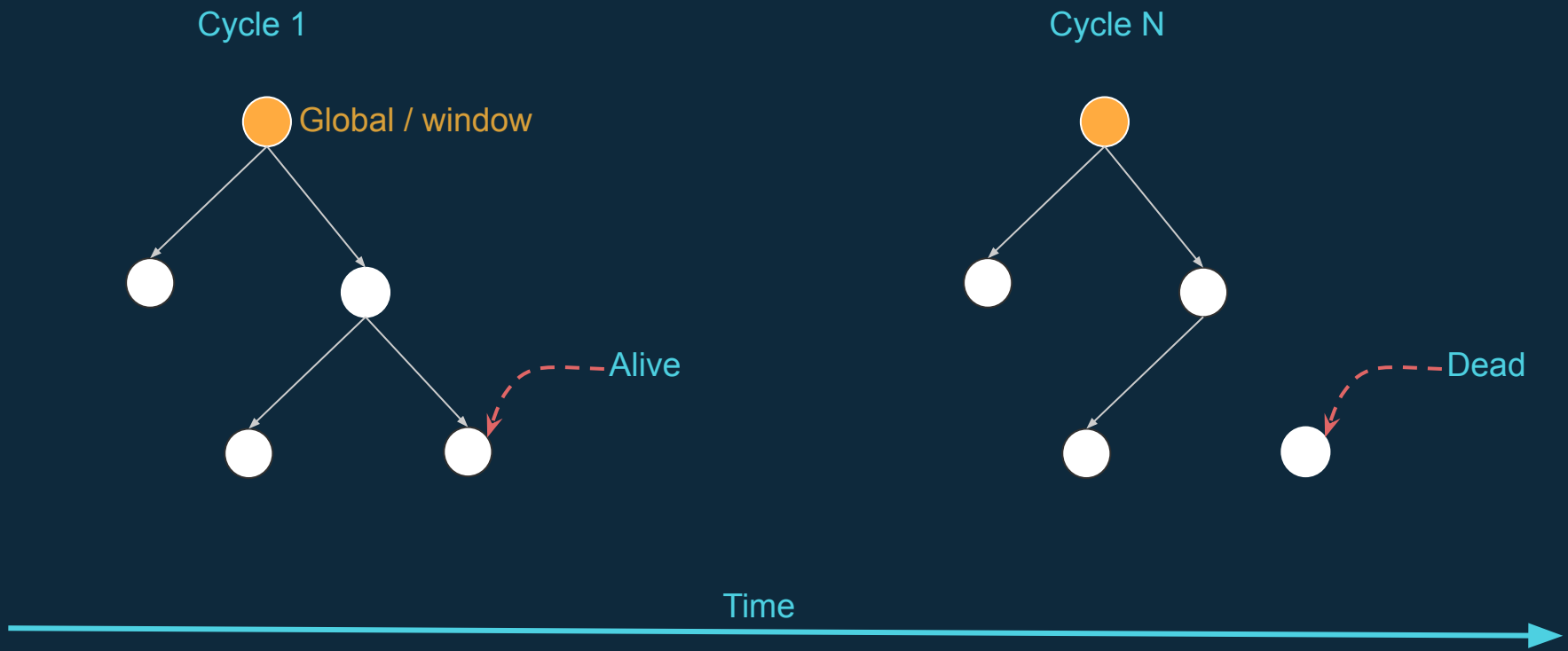
Incremental

Parallel + concurrent

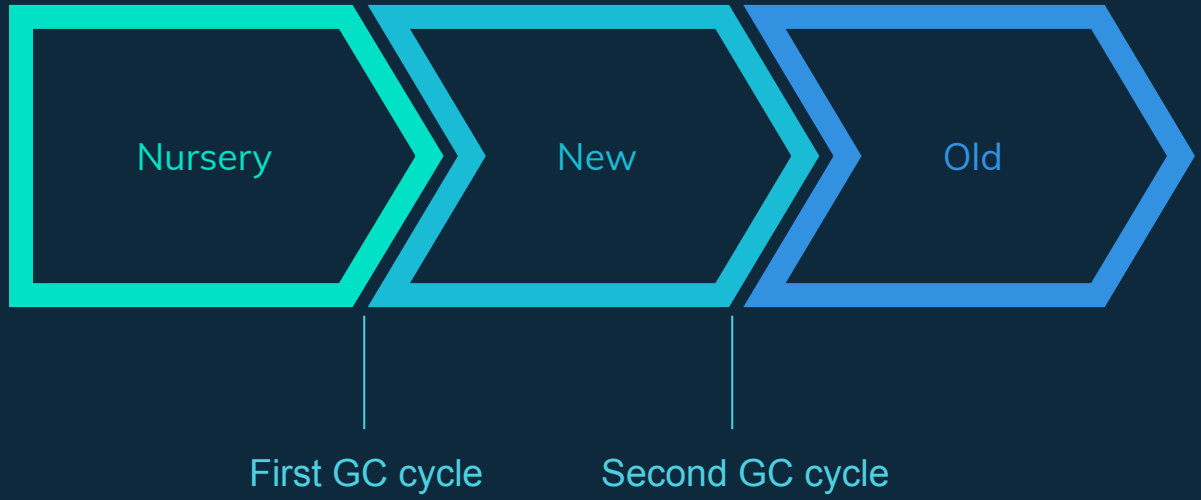


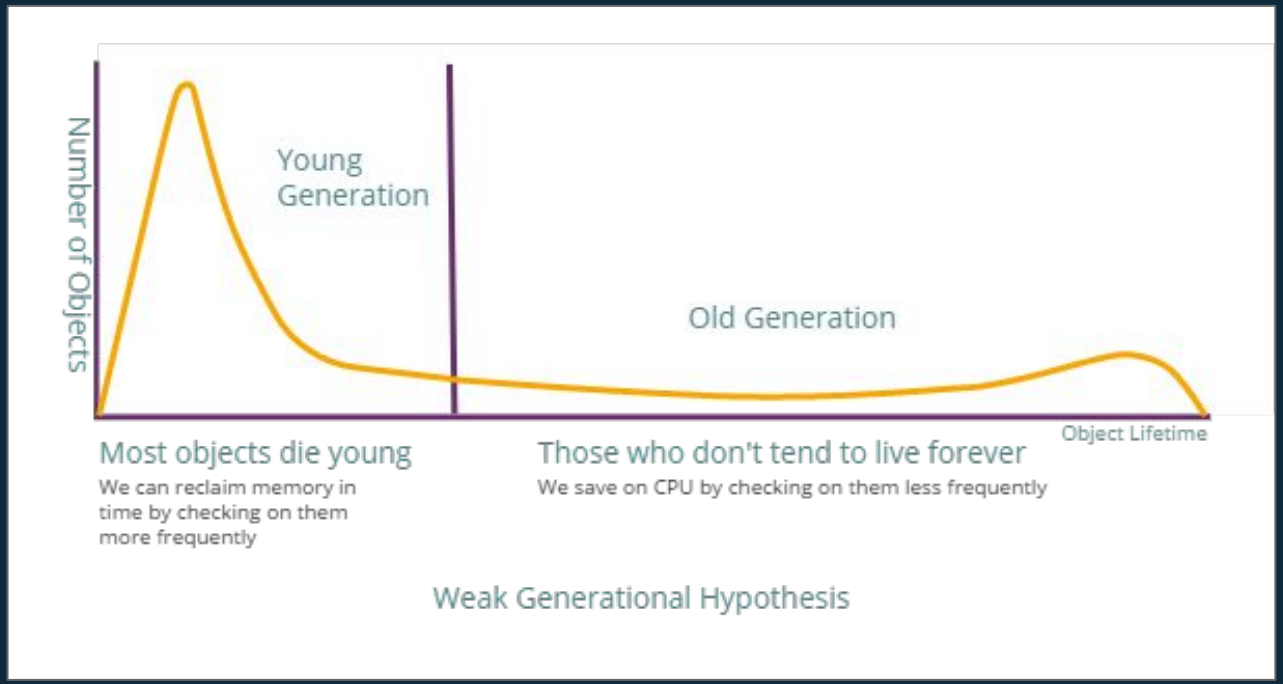
Dead or Alive ?

When an object is dead (not reachable) GC can remove it !


Cycle 1
Global / window
Alive
Cycle N
Dead
Time

Generations




Number of Objects
Young Generation
Old Generation
Object Lifetime
Most objects die young
We can reclaim memory in time by checking on them more frequently
Those who don't tend to live forever
We save on CPU by checking on them less frequently
Weak Generational Hypothesis

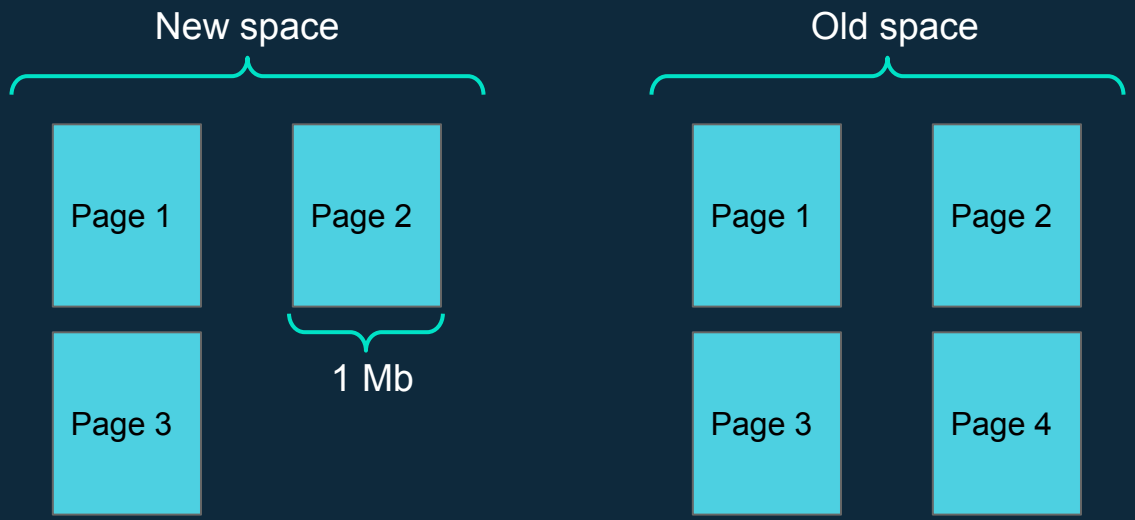
Spaces

V8 uses spaces for a better memory organization !

Spaces

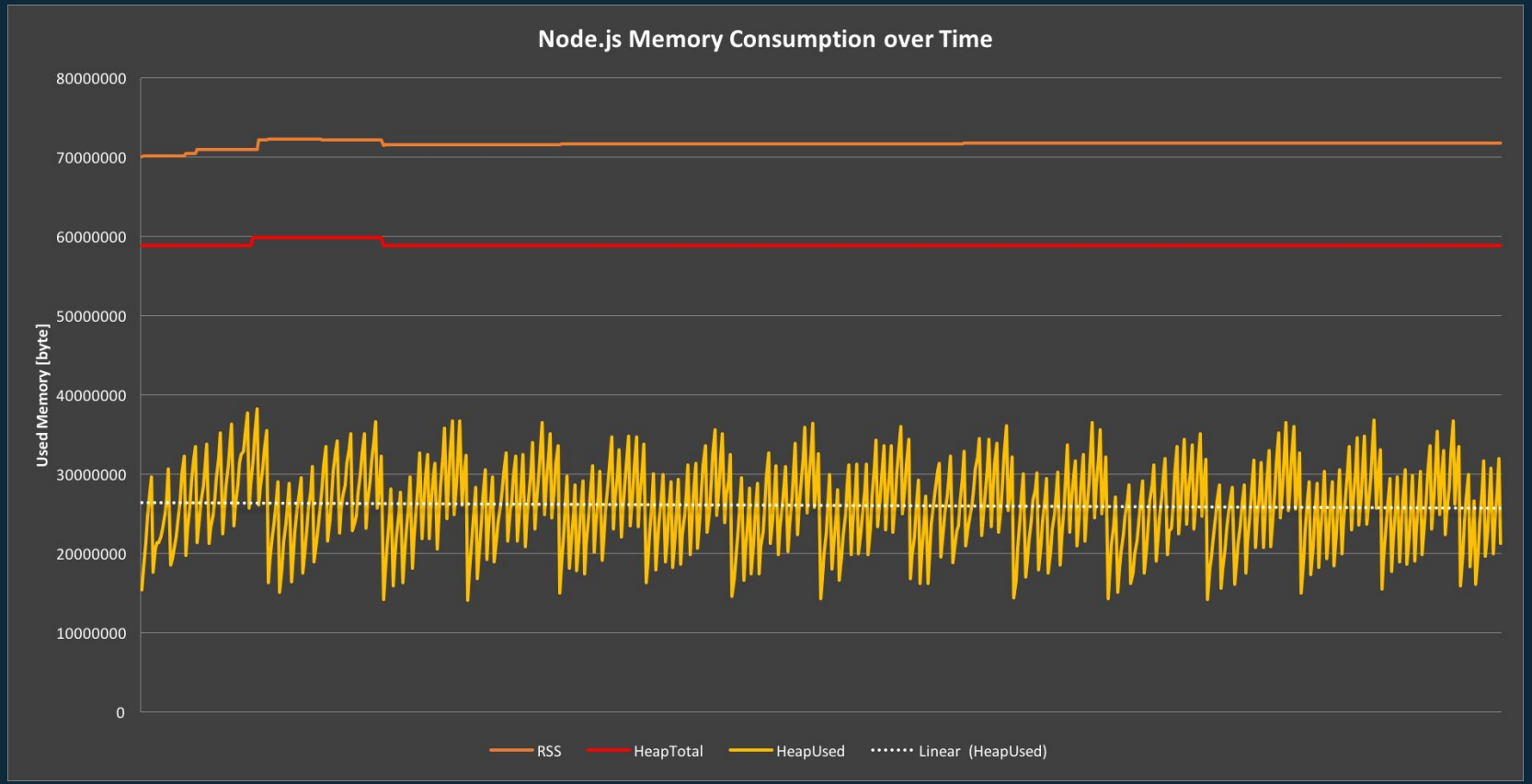
Type	Description
New-space	Most objects are allocated here.
Old space	Moved here after surviving in new-space for a while.
Large space	This space contains objects which are larger than the size limits of other spaces. Large objects are never moved by the garbage collector.
Code space	Code objects, which contain JITed instructions, are allocated here.
Map space	Cells, PropertyCells, and Maps

Spaces & pages

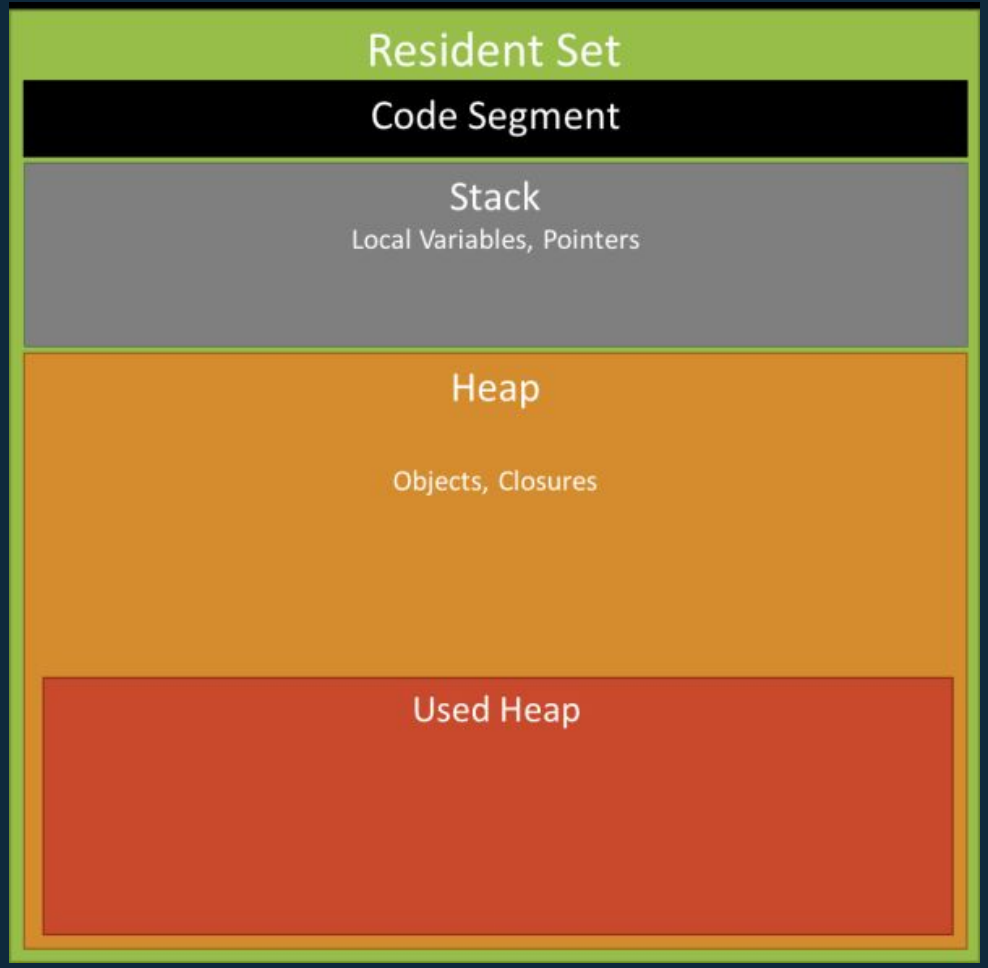


Memory Leak

Nightmare !


Node.js Memory Consumption over Time
Used Memory [byte]
80000000
70000000
60000000
50000000
40000000
30000000
20000000
10000000
0
RSS
HeapTotal
HeapUsed
Linear (HeapUsed)

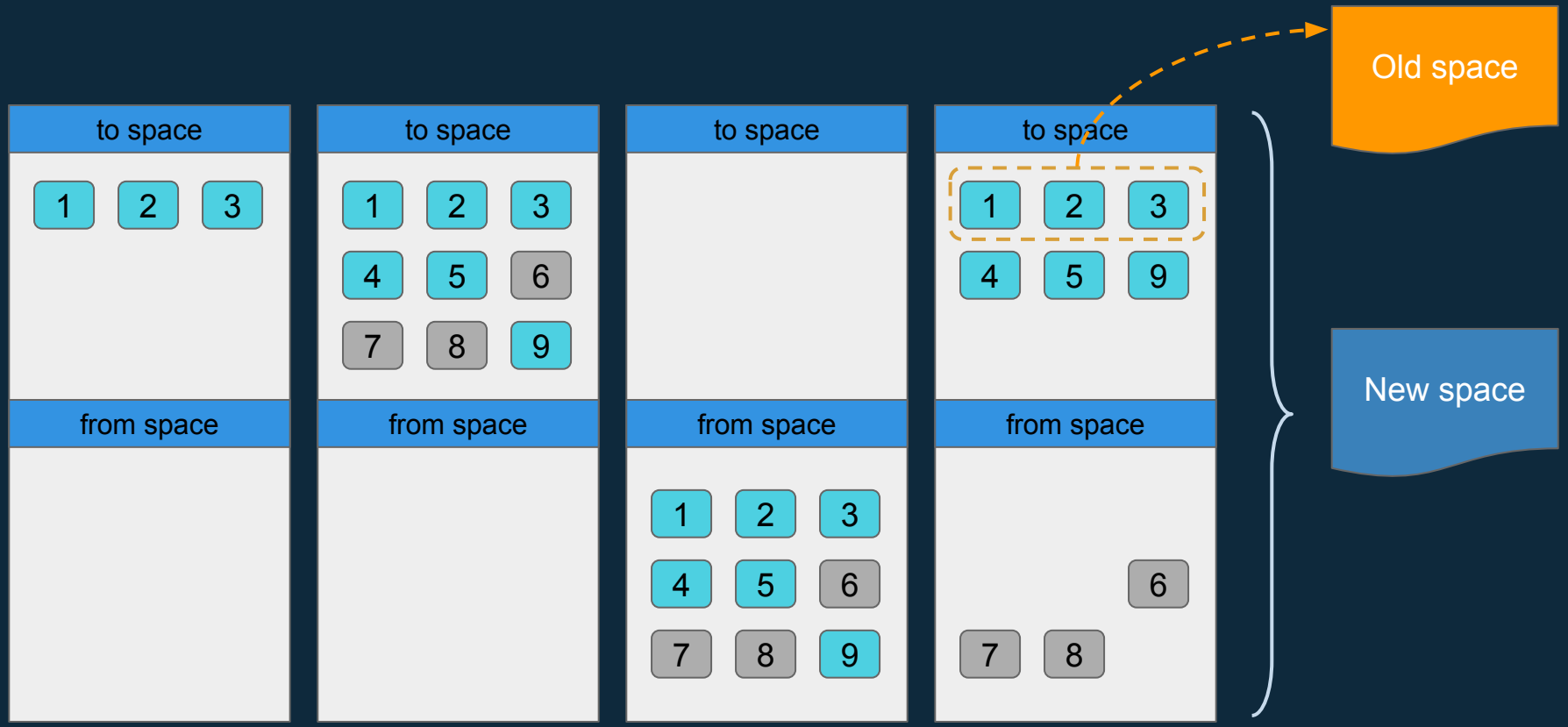

Node.js Memory Consumption over Time
120000000
100000000
80000000
60000000
40000000
20000000
0
Used Memory [byte]
MEMORY LEAK
MEMORY LEAK EVERYWHERE
RSS
HeapTotal
HeapUsed
Linear (HeapUsed)


Resident Set
Code Segment
Stack
Local Variables, Pointers
Heap
Objects, Closures
Used Heap

Algorithm

Scavenge vs Mark / Sweep / Compact
White -> gray -> back ...

Scavenge : Minor GC algorithm.
It’s used only in new space.

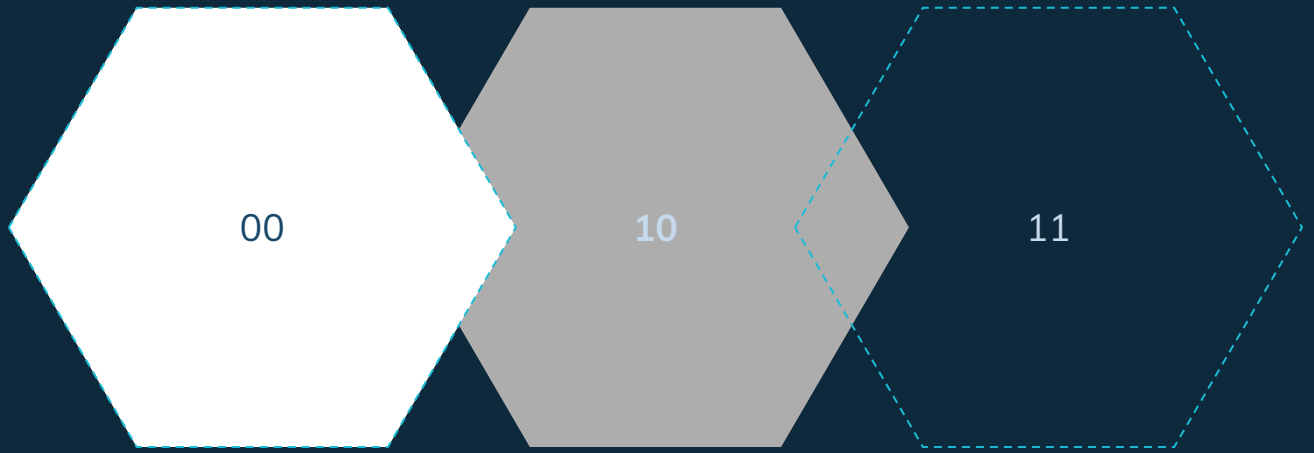

to space
1
2
3
from space
to space
1
2
3
4
5
6
7
8
9
from space
to space
from space
1
2
3
4
5
6
7
8
9
to space
1
2
3
4
5
9
from space
6
7
8
Old space
New space

Major GC

Mark : Recursive procedure of marking reachable objects !

V8 use the white/gray/black marking system.

White / Gray / Black

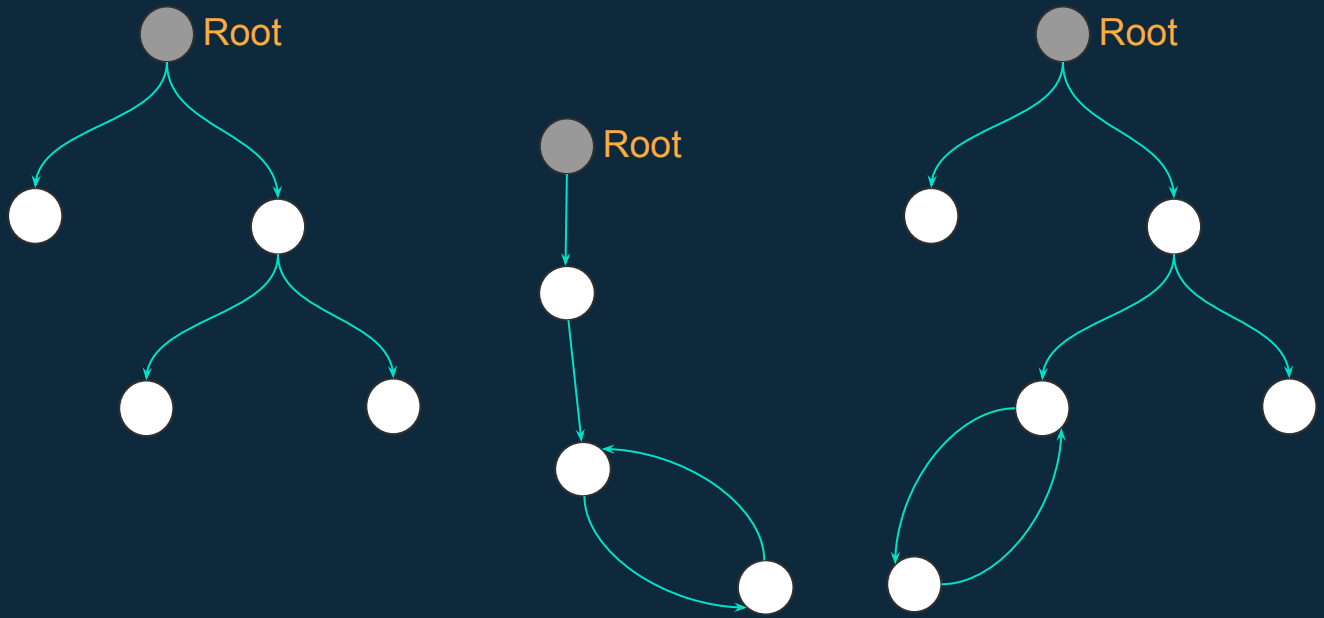


Not yet been discovered

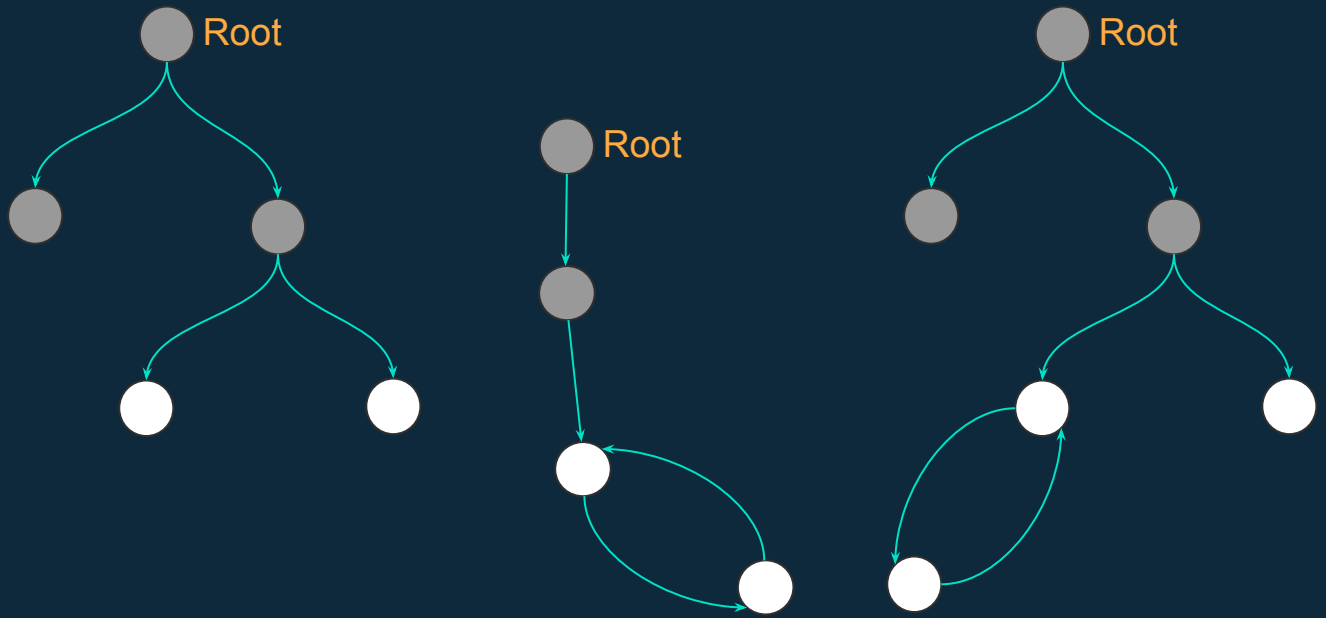
Discovered

Discovered and all of its neighbors also.

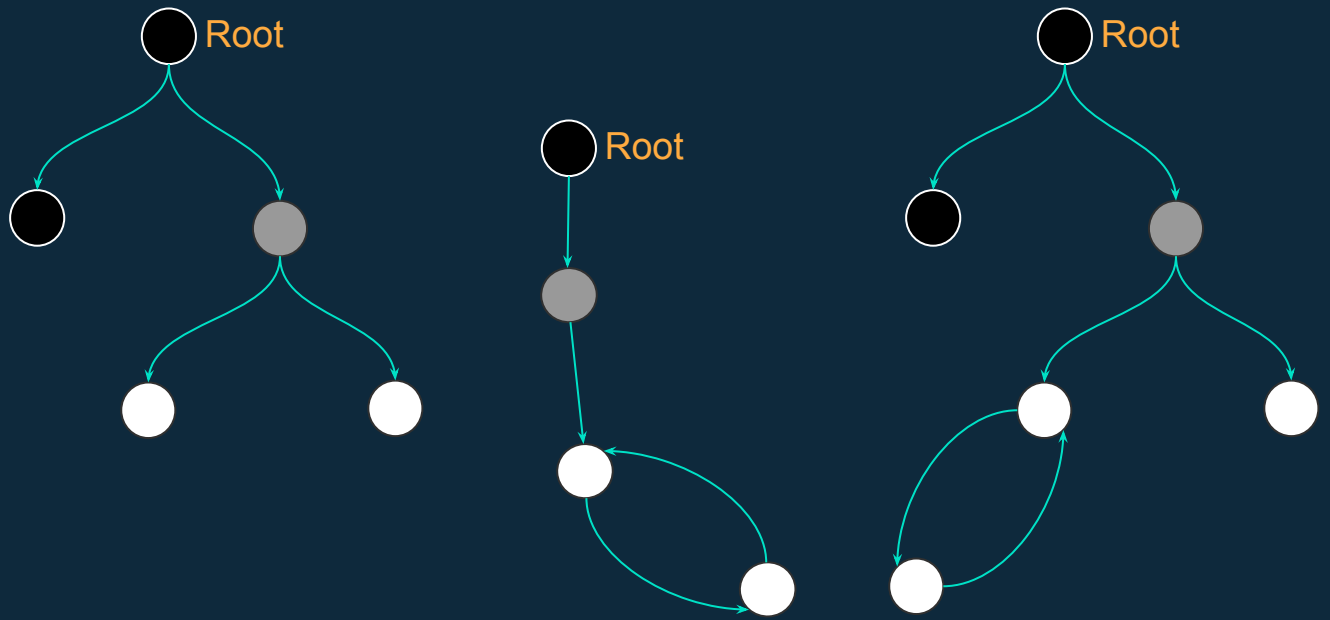
Define root nodes + mark them as discovered !



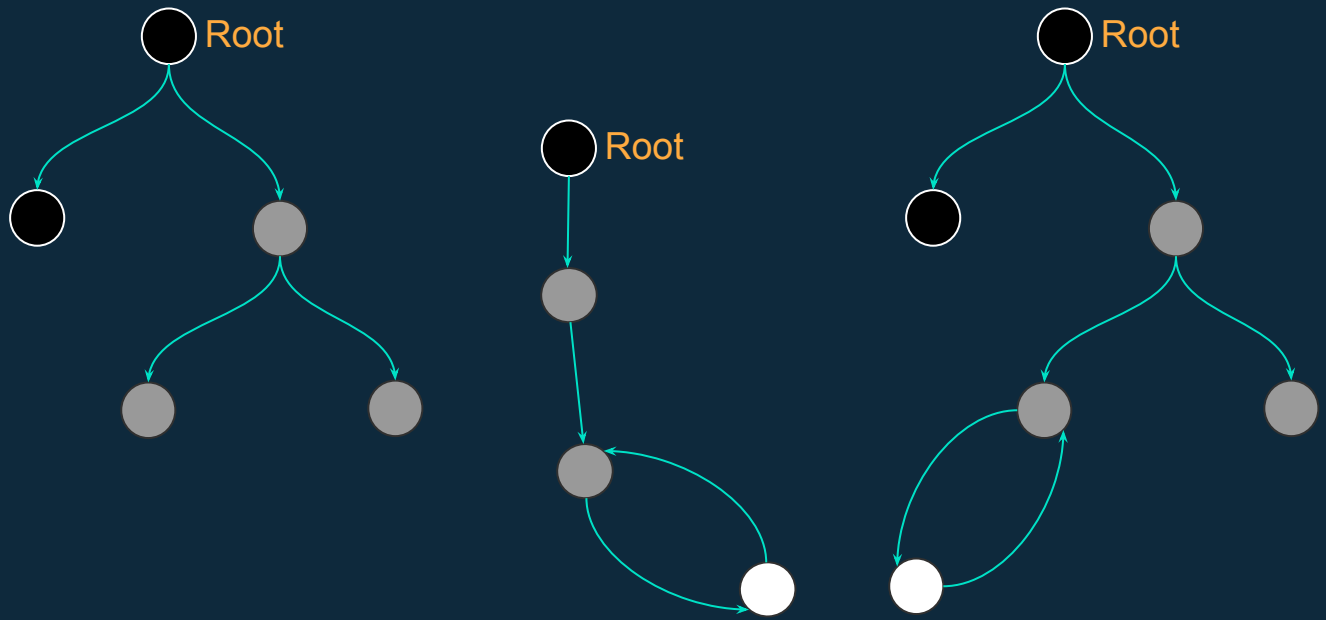
Direct neighbors discovery



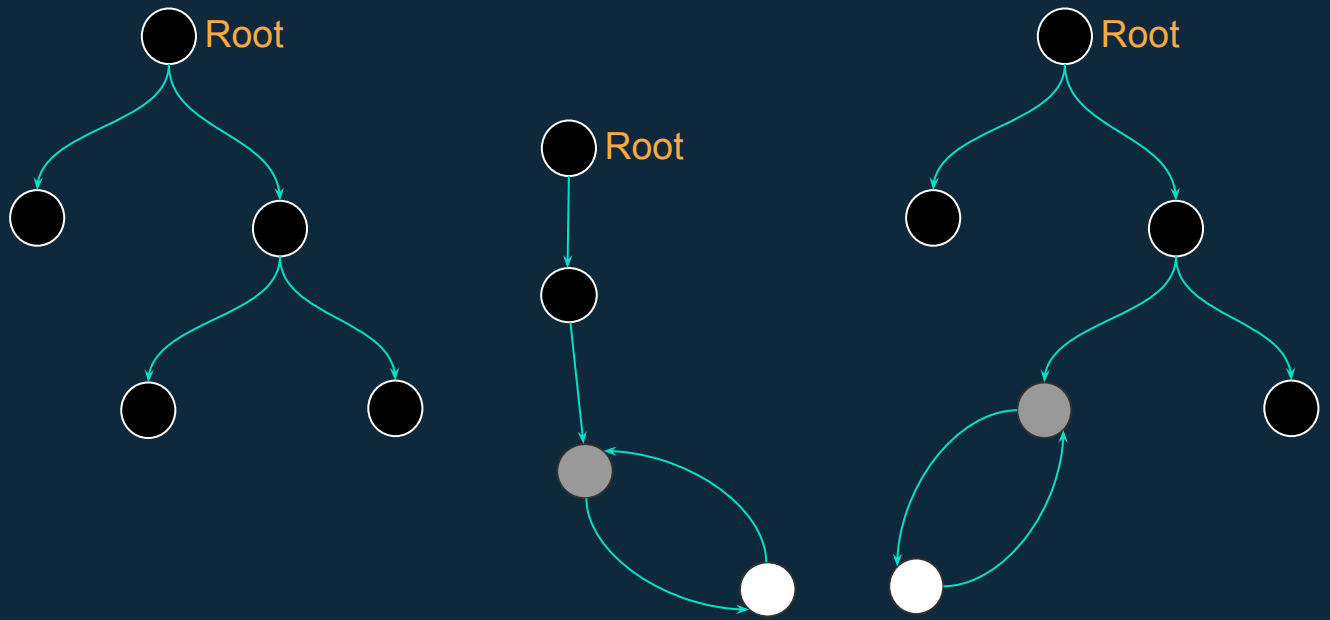
Mark node in black if they are discovered and all their neighbors also.



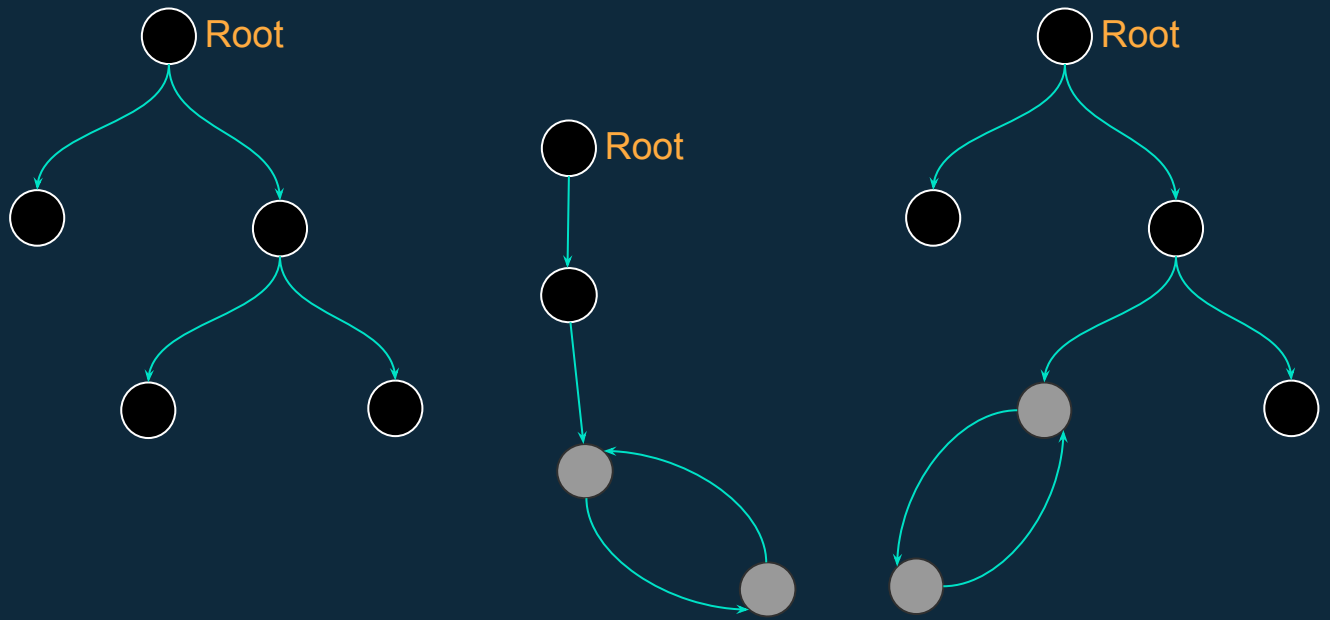
Mark node in black if they are discovered and all their neighbors also.



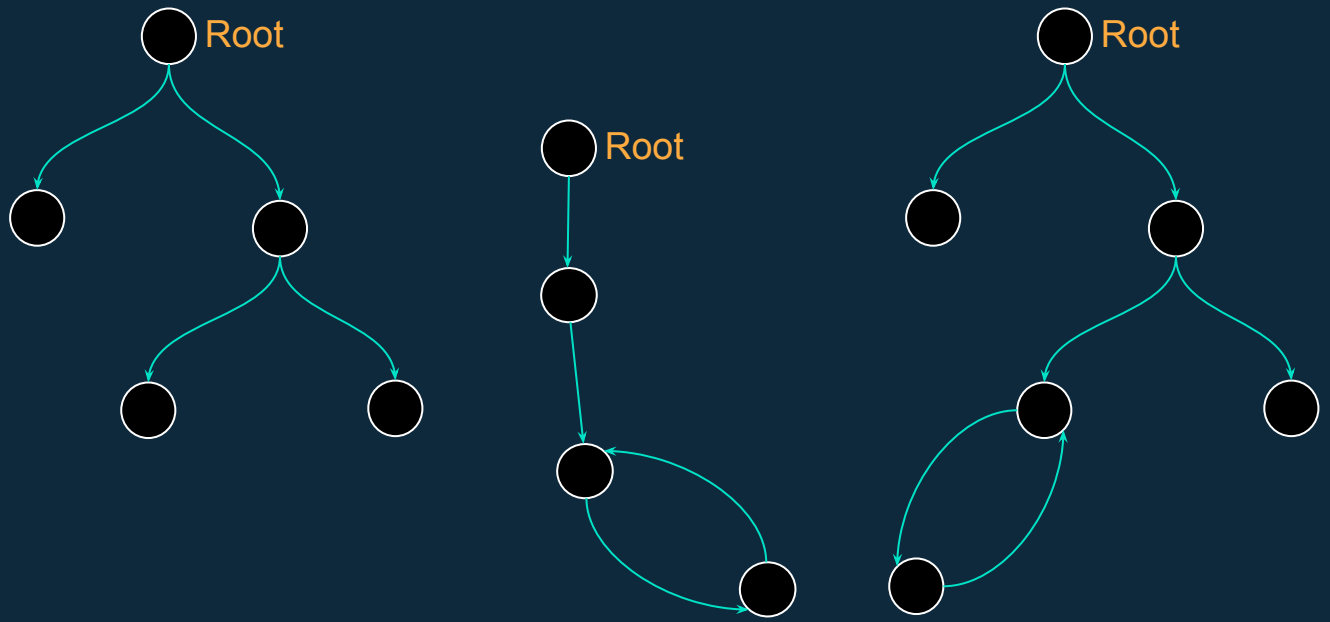
Mark node in black if they are discovered and all their neighbors also.



Mark node in black if they are discovered and all their neighbors also.

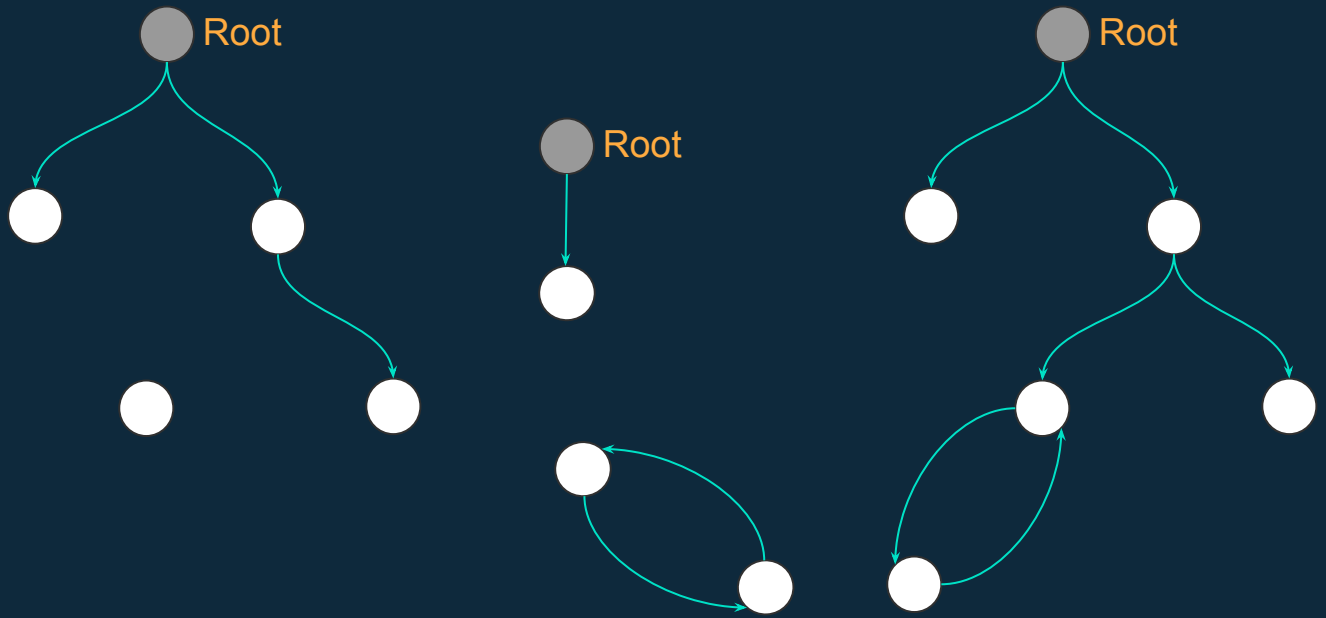


All reachable nodes are black : mark phase is done !!!!!!

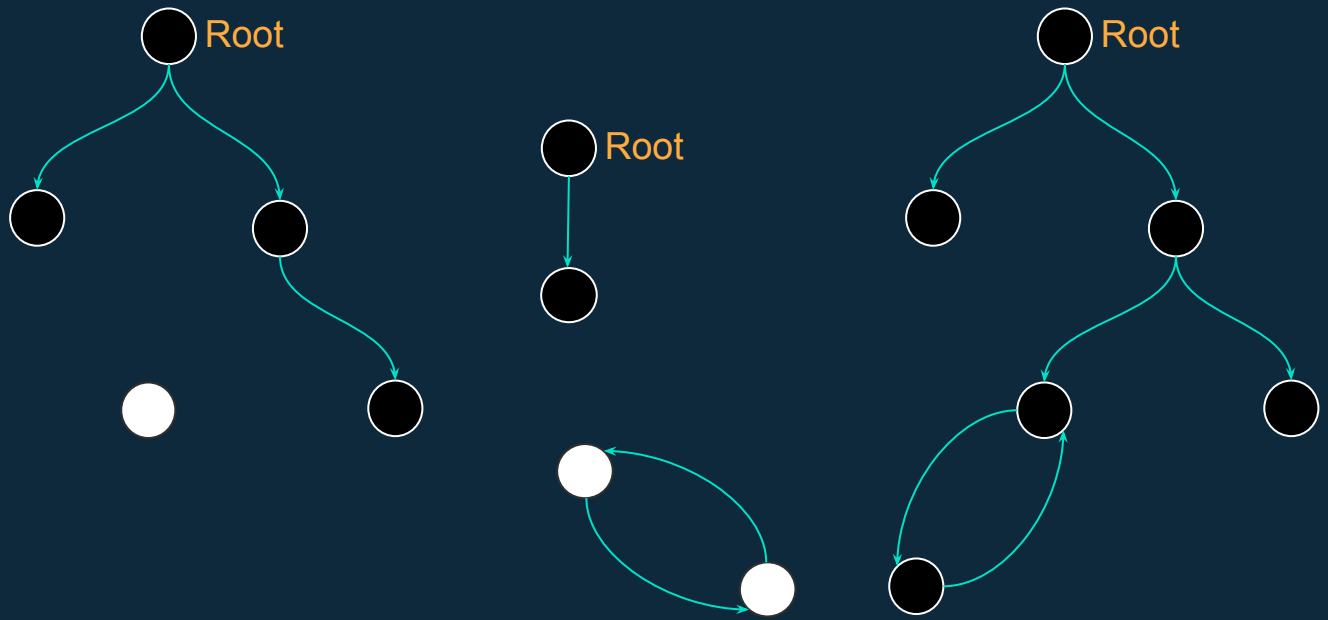


What about non-reachable ?

Some nodes are unreachable !



Some nodes stay in white, they will be removed by sweep phase.



Sweep : Remove all unused (white) objects.

Can be done during another GC pause !

Compact : moving all marked – and thus alive – objects to the beginning of the memory region.

Sweep in action

Before

Sweep phase

After

Compact in action

Before Sweep

After Sweep

After Compact

Reduce marking pause

Parallel / concurrent

Stop the world

Incremental marking :

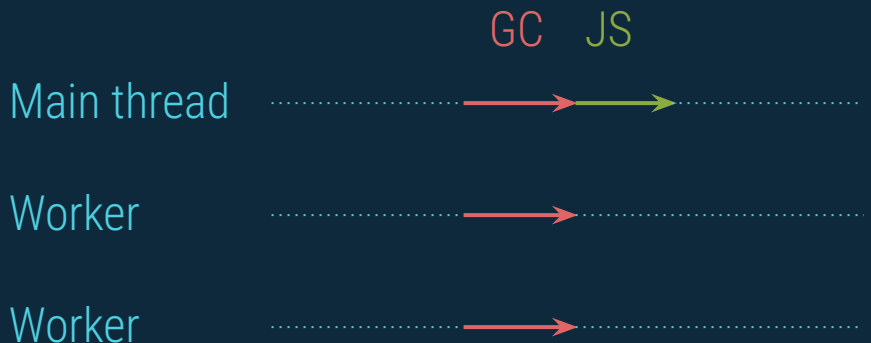
garbage collector splits up the marking work into smaller chunks

Incremental marking

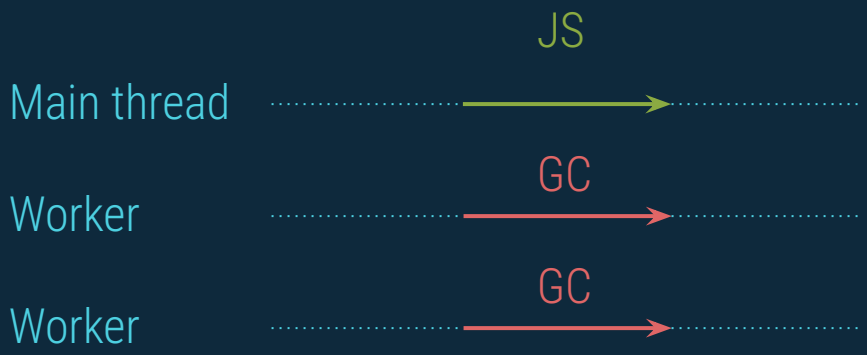
Lazy sweeping :

sweep pages on an as-needed basis
until all pages have been swept

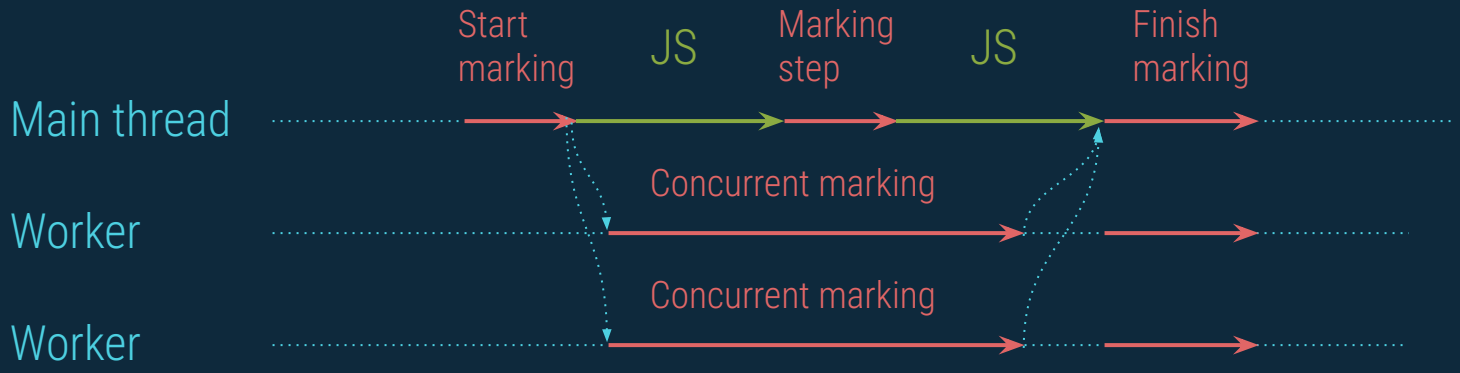
Concurrent



Parallel



All together



Monitoring tools

Modules : v8, gc stats
Options : --trace_gc


Demo
Module : V8

Module v8

The v8 module exposes APIs that are specific to the version of V8 built into the Node.js binary

- v8.getHeapSpaceStatistics()
- v8.getHeapStatistics()

```
{
  total_heap_size: 7326976,
  total_heap_size_executable: 4194304,
  total_physical_size: 7326976,
  total_available_size: 1152656,
  used_heap_size: 3476208,
  heap_size_limit: 1535115264,
  malloced_memory: 16384,
  peak_malloced_memory: 1127496,
  does_zap_garbage: 0
}
```

Gc Stats

Exposes stats about V8 GC after it has been executed.

```
var gc = (require('gc-stats'))();

gc.on('stats', function (stats) {
    console.log('GC happened', stats);
});
```

```
GC happened {
  startTime: 9426055813976,
  endTime: 9426057735390,
  pause: 1921414,
  pauseMS: 1,
  gctype: 1,
  before: {
     ...
  },
  after: {
    ...
  },
  diff: {
    ...
  }
}
```

Demo
--trace_gc

V8 options

--trace_gc : print one trace line following each garbage collection

--trace_gc_verbose : print details following each garbage collection

--trace_gc_nvp : print one detailed trace line in name=value format after each garbage collection

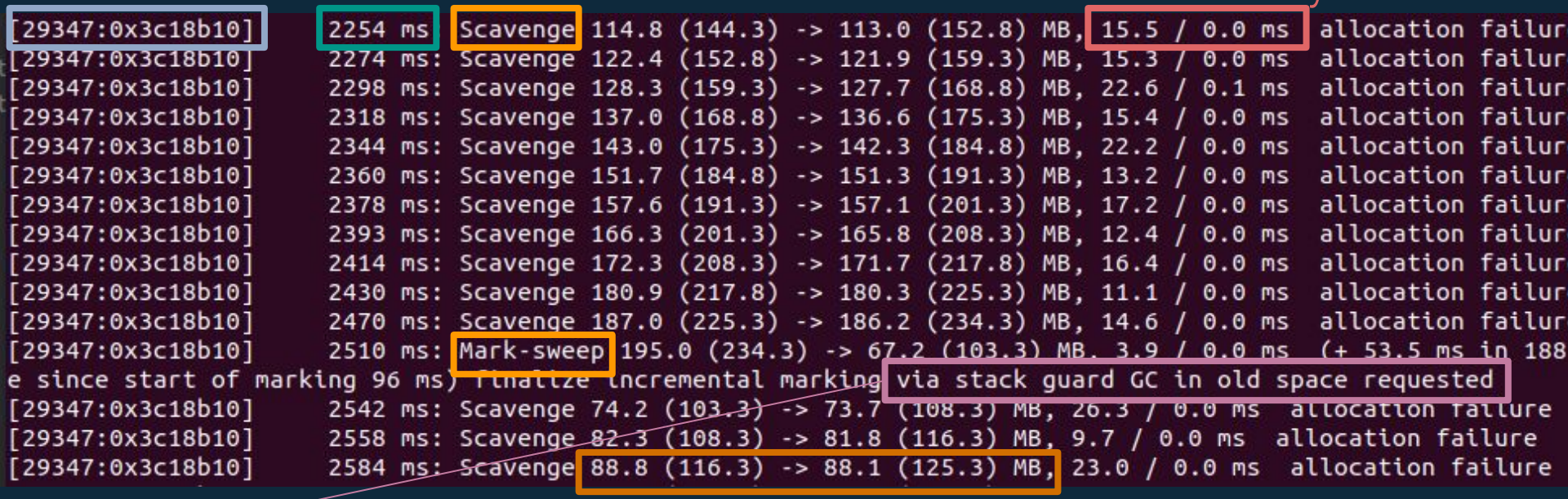
--expose_gc : expose gc extension

PID

Start

GC type

GC pause / time spent in external memory



Reason of GC

Size of all objects

Thanks!

Any questions?

You can find me at:

- @Vince_Vallet
- wallet77@gmail.com

Credits

- V8 GC overview
- Concurrent marking
- GC algorithm
- V8 options