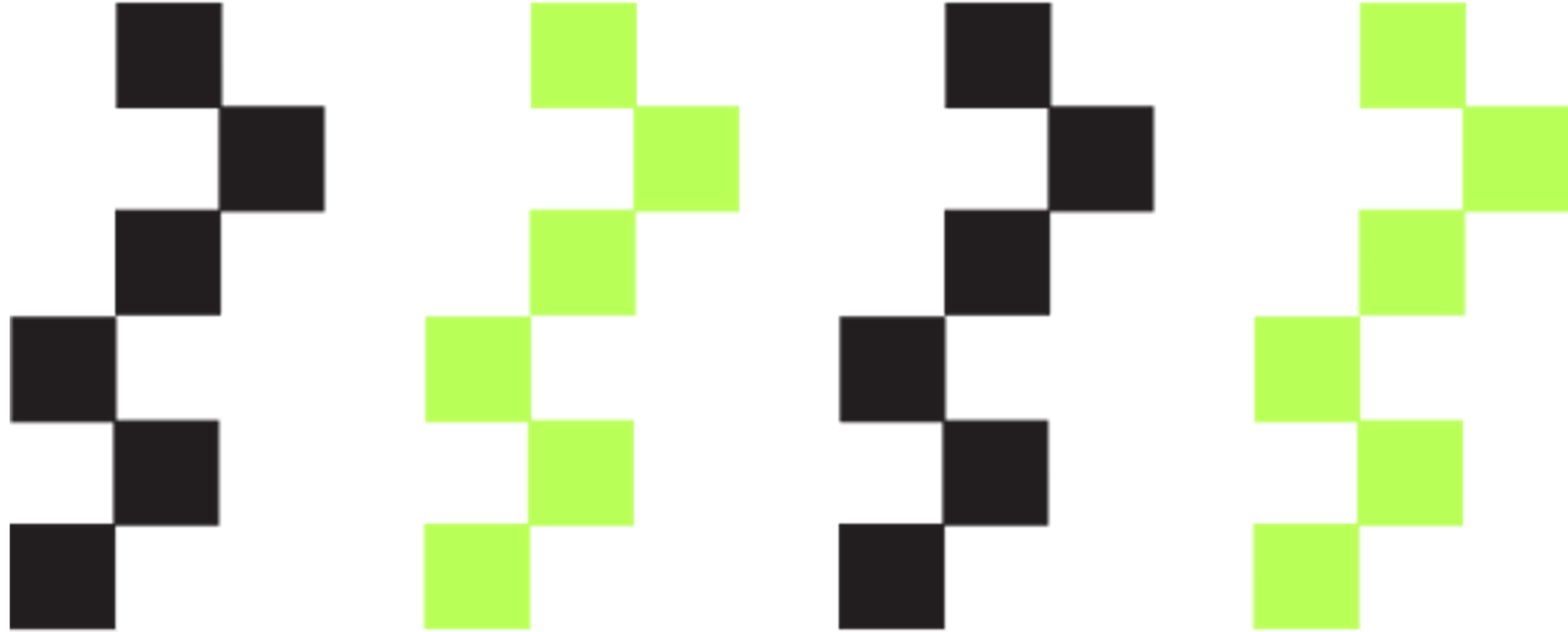


Партнерство с Garbage Collector. Память в V8 и работа с утечками



А что такое Garbage Collector?

Что такое Garbage Collector?

 Garbage Collector - одна из форм **автоматического управления памяти** программы.

Что такое **мусор**?

Какие-то данные в памяти приложения, которые уже не нужны для работы.

Зачем **удалять** мусор?

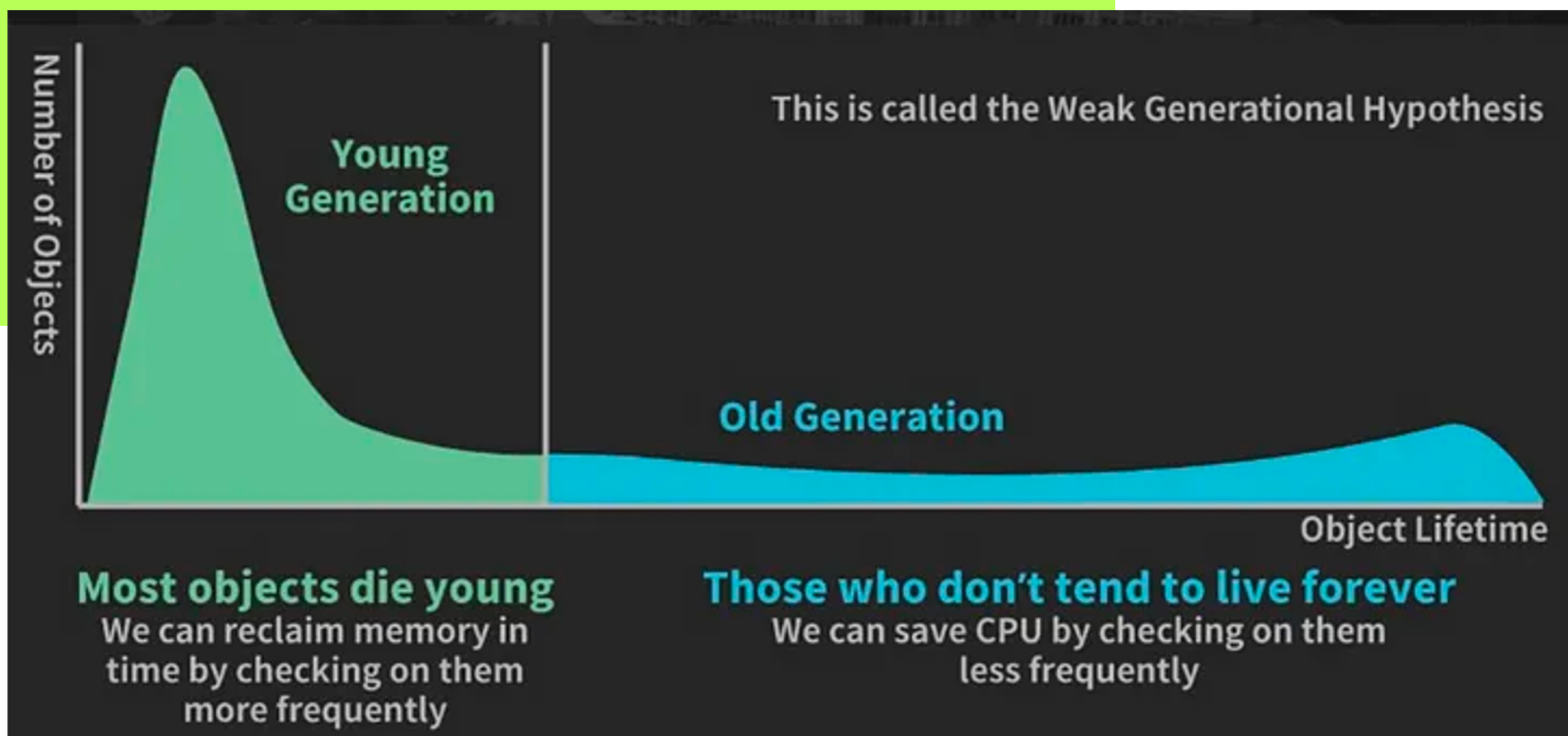
Если мы будем бесконечно создавать новые объекты, то рано или поздно упрямся в лимит памяти.

Как GC может **понять**, что это мусор?

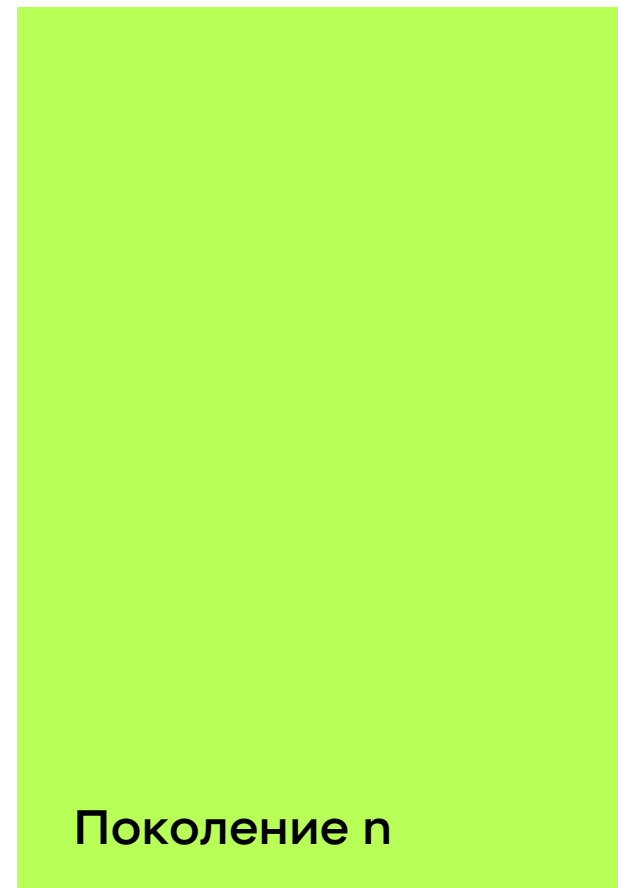
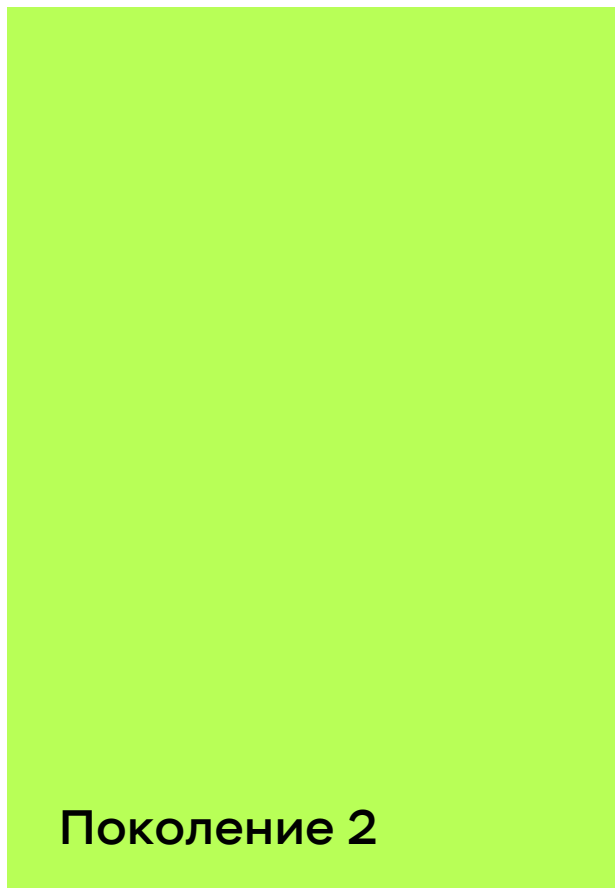
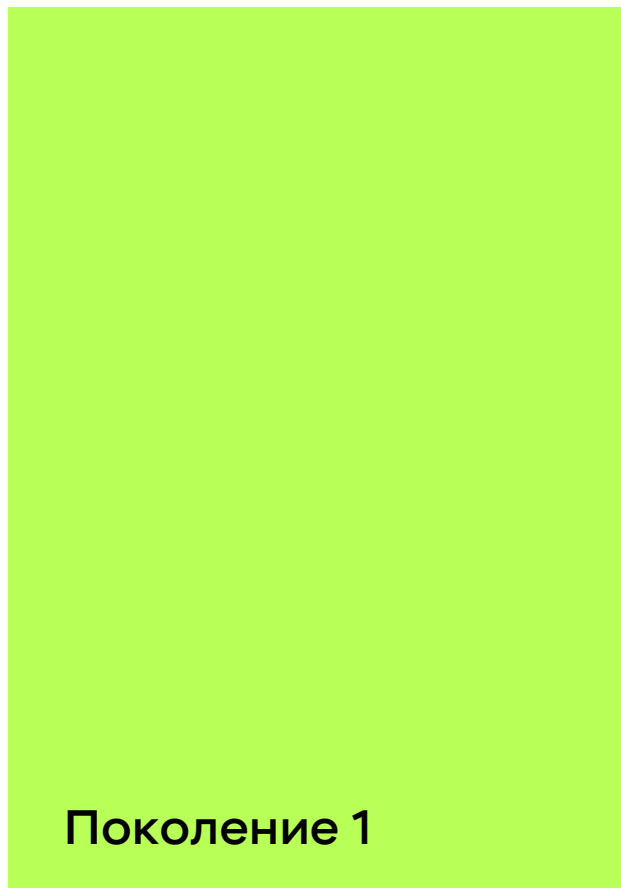
На него не осталось ссылок в приложении, то есть эти данные уже не могут быть использованы кодом.

Garbage Collector. Паттерны и реализация

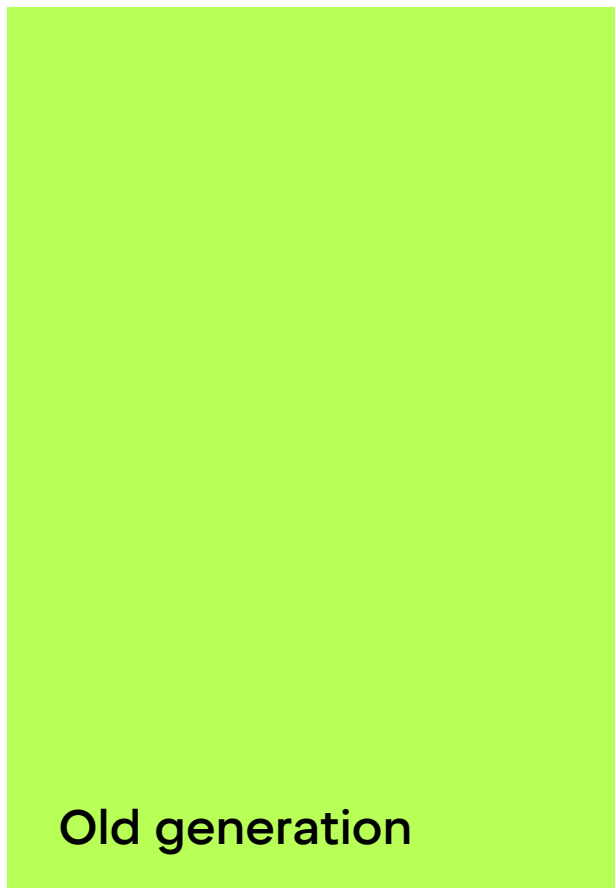
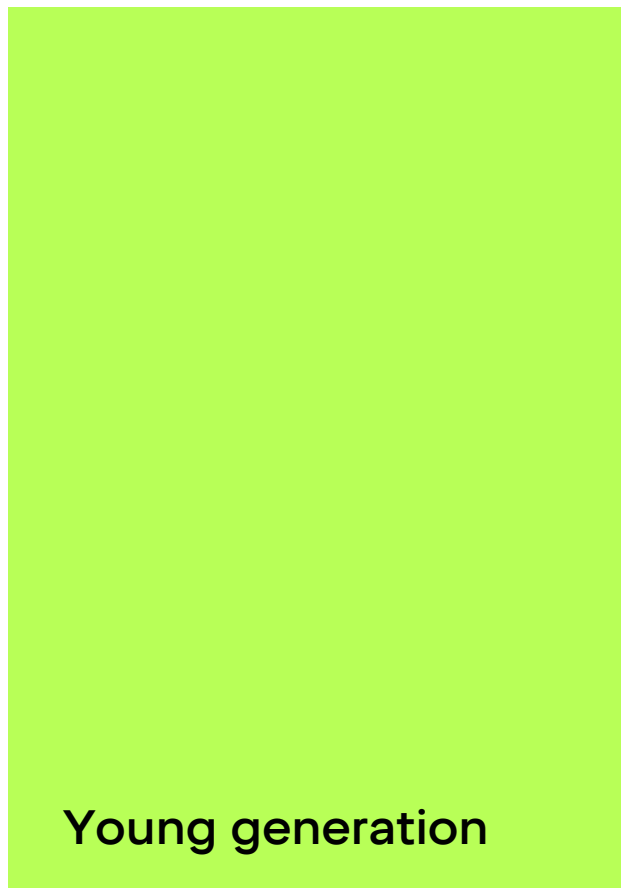
В далеком 1984 году было доказано, что **большинство объектов умирает молодыми.**



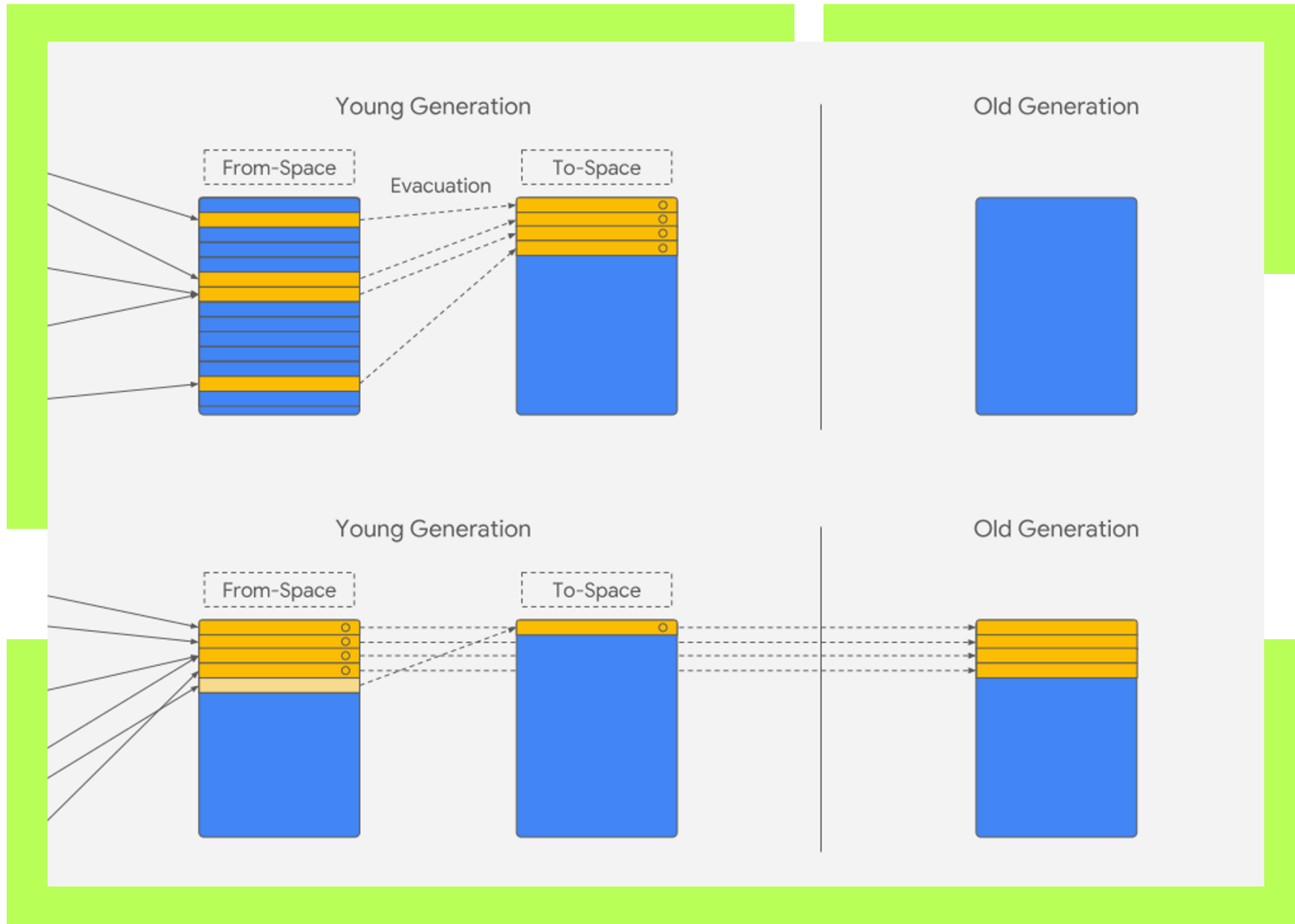
Сборка мусора поколениями



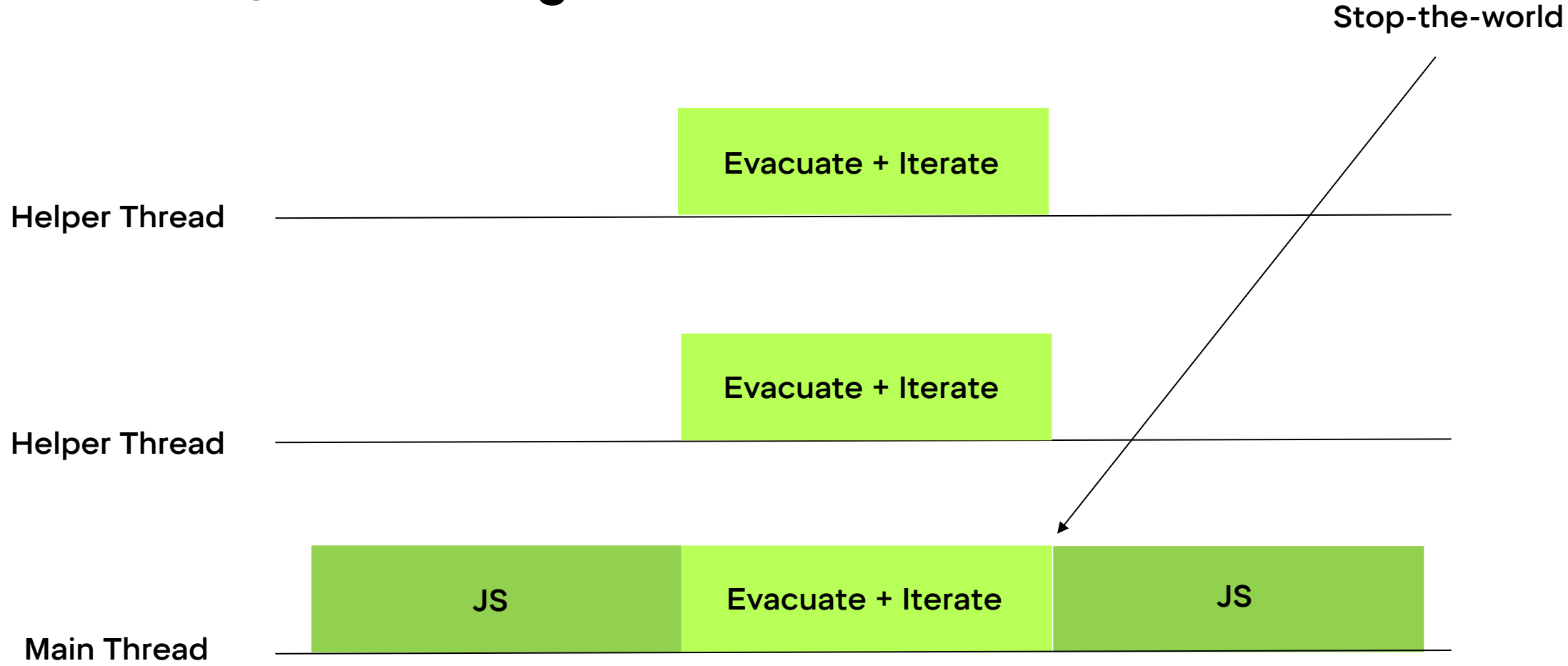
Сборка мусора поколениями в V8



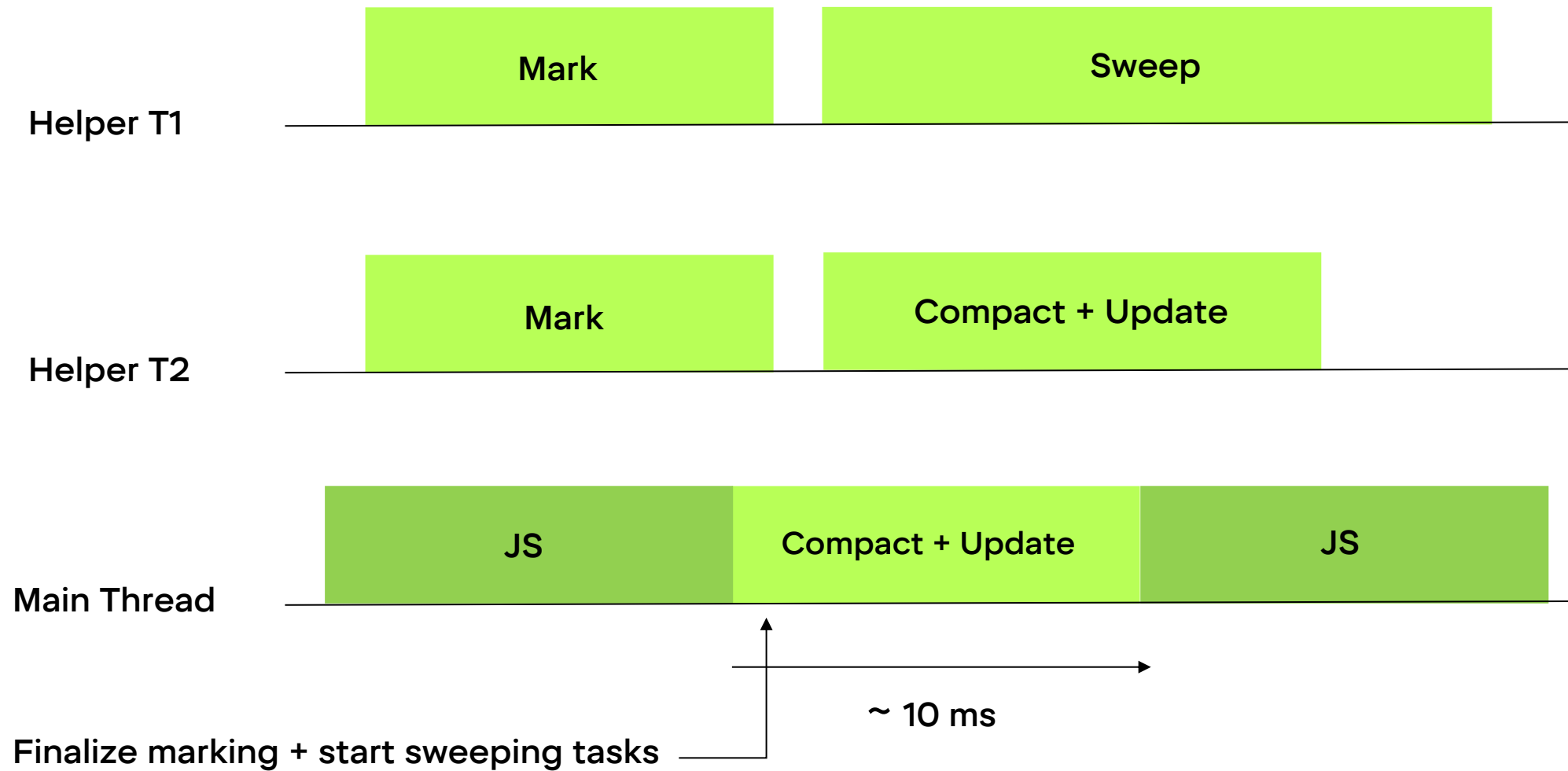
Реализация scavenger



Реализация scavenger



Реализация mark-sweep



Как регулировать поколения и влиять на gc?

--expose-gc, можно вызывать его из global gc && gc()

--trace-gc (смотреть за выполнением)

-gc-interval=ms

--max-semi-space-size=mb

--max-old-space-size=mb



Выводы

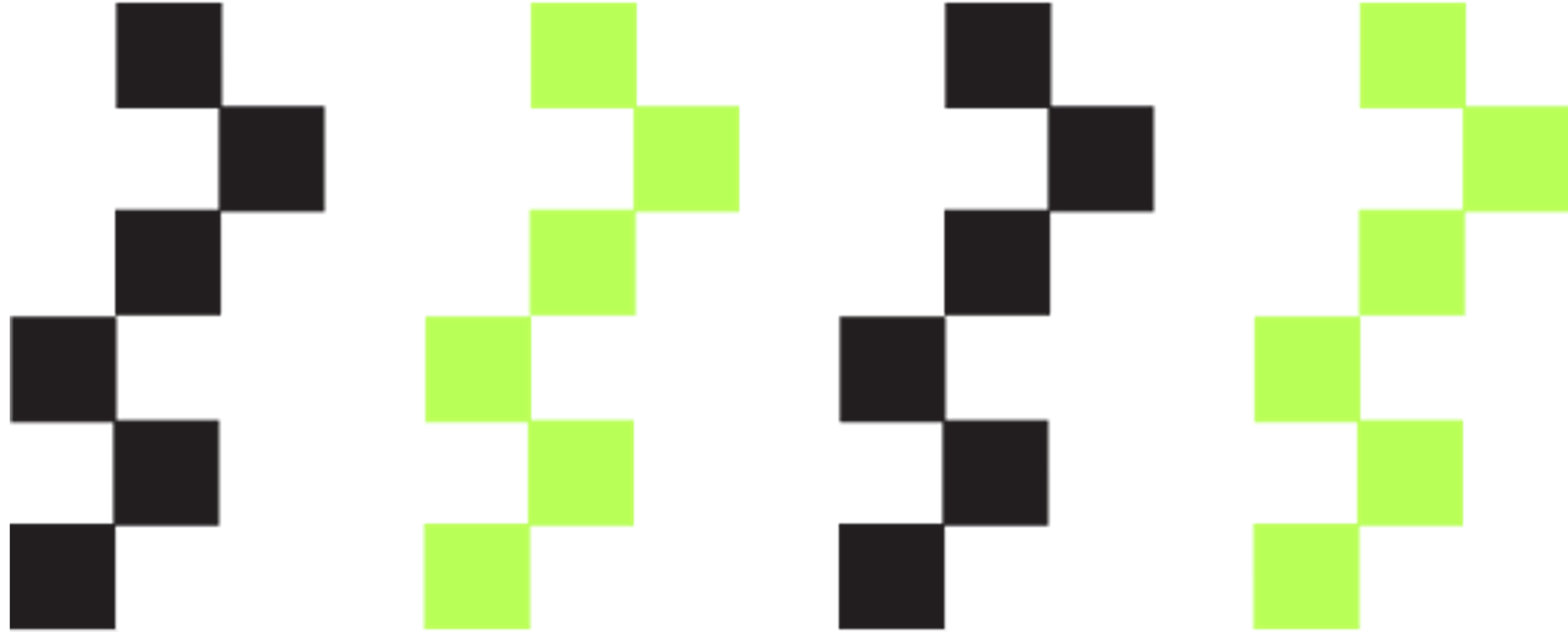
Понимаем, что такое мусор и что такое сборка;

Мы лучше понимаем, как работает сборка мусора в наших приложениях;

Понимаем, какие паттерны сборки мусора реализованы в V8;

Понимаем, как мы можем с этим взаимодействовать;





А как устроена память в V8?

Как устроена память

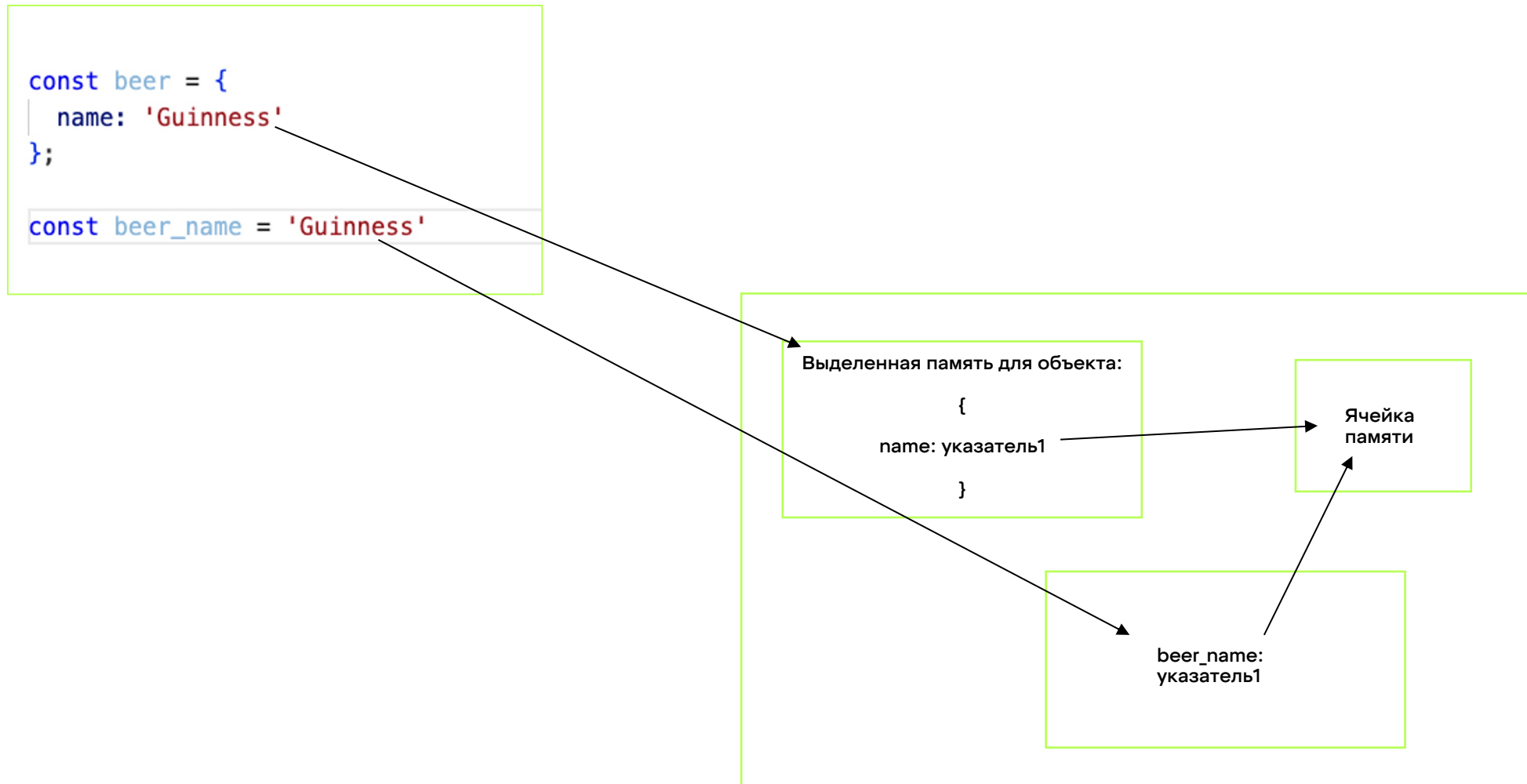


stack

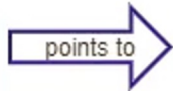
heap

external

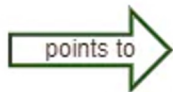
Как v8 распределяет память ссылками



Javascript code:
`let obj = {};`
'obj' is a pointer to the structure



JS OBJECT
Map
Properties
Elements
In-Object Property 1
In-Object Property 2
...
In-Object Property N



HIDDEN CLASS



PROPERTIES



ELEMENTS



Как регулировать память

- max-old-space-size = \${NumberMB}
- max-semi-space-size = \${NumberMB}



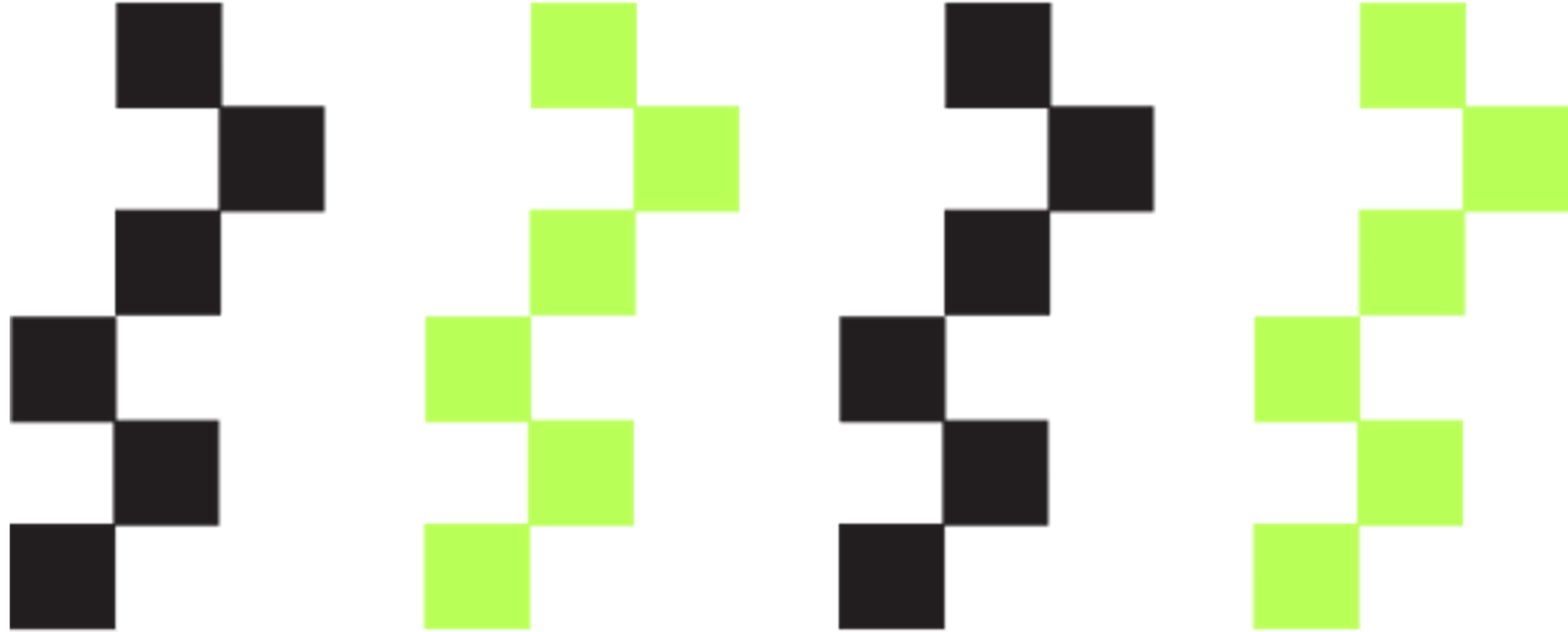
Выводы

Мы лучше понимаем, как **работает** наш инструмент;

Понимаем, как мы можем **влиять** на него.

Понимаем, как **хранится** код, который мы пишем





Личный опыт работы с утечками памяти

Что же такое сама утечка памяти?

Задумался?



Утечки – это нахождение в оперативной памяти объектов, которые не нужны для работы приложения.

А зачем утечки вообще убирать?

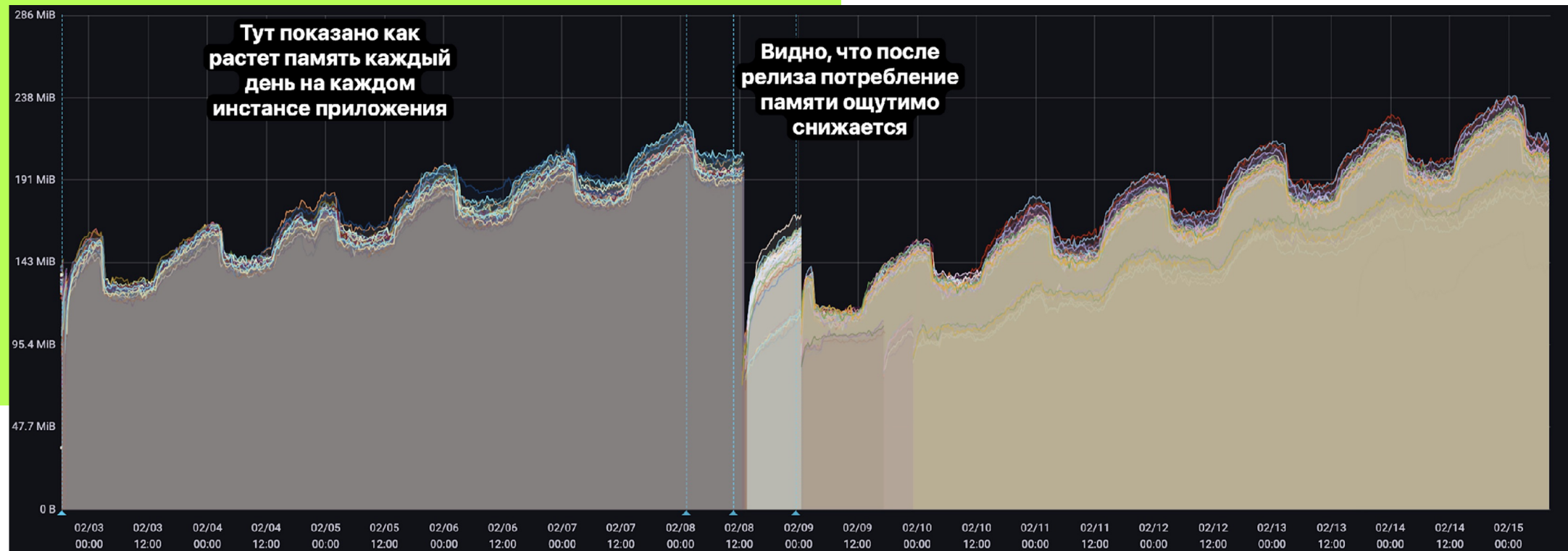
Экономия денег;

Устойчивость системы;

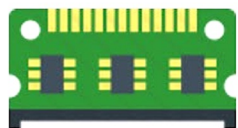
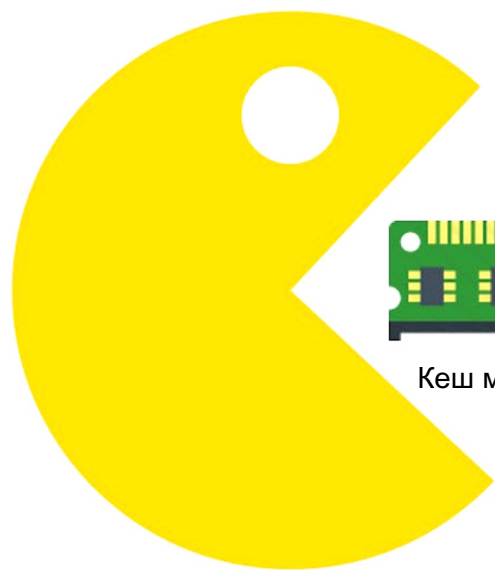
Рост как специалиста;



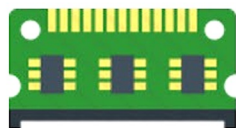
Как наглядно выглядит утечка



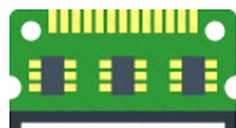
Что не очищается в памяти



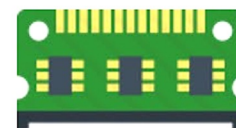
Кеш модулей



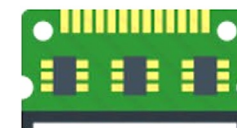
Глобальные
переменные



Управление
кешем

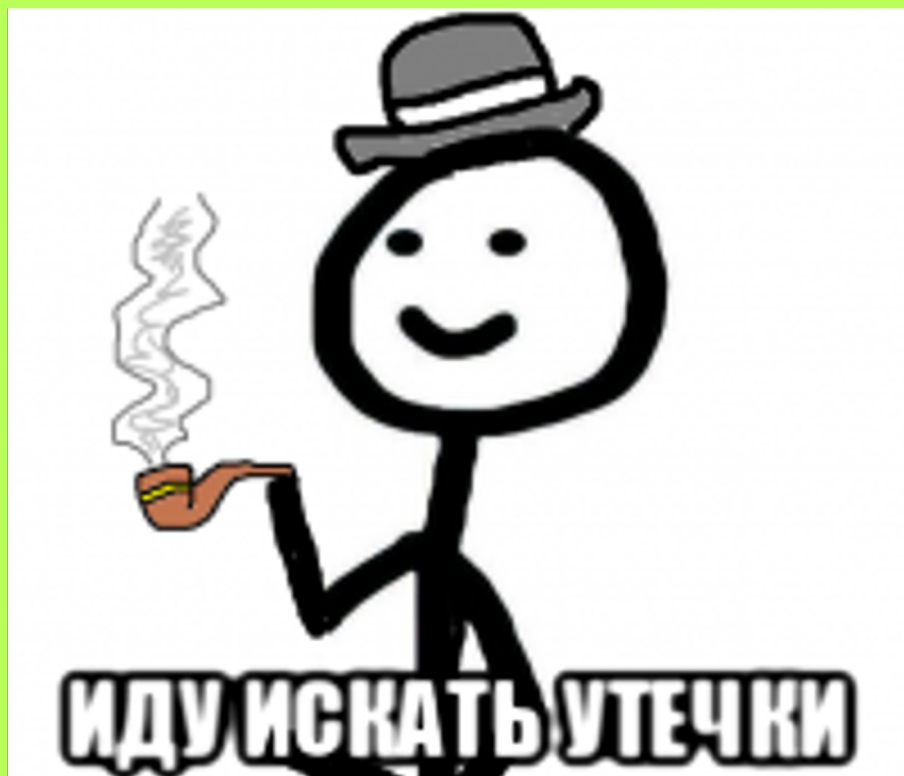


Замыкания



Соединения

А как искать эти ваши утечки?



Инструменты для поиска утечек

Возможно, проще:

Clinic js

Возможно, сложнее:

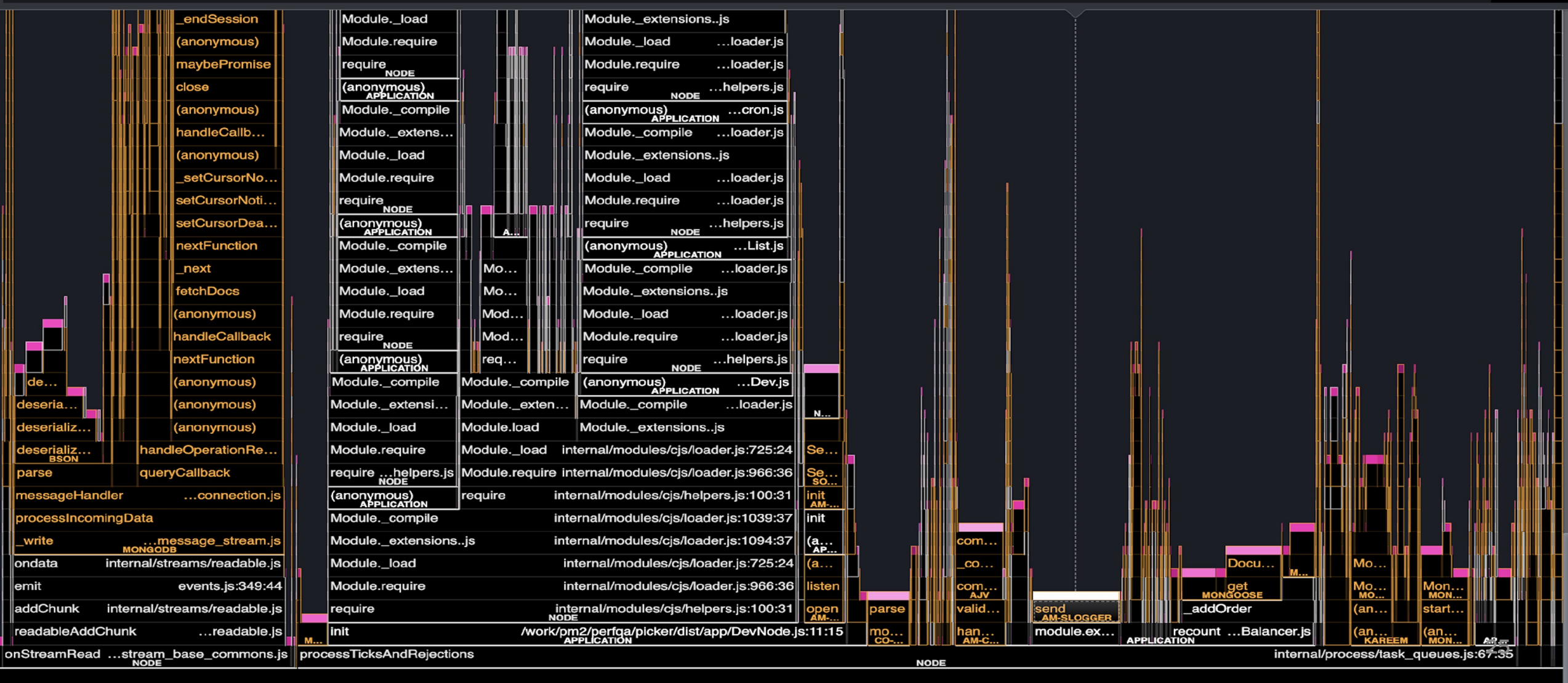
Chrome DevTools

(для снятия снепшота в node
можно использовать апи v8)



1 biggest allocation, of 3072 Next biggest > >|

send /work/pm2/perfqa/picker/node_modules/am-slogger/index.js line:52 column:6 In Dependency (am-slogger). 5.58 %



Тип данных

Сколько циклов
сборки мусора
пережил

Сколько занимает
в памяти сама
структура

Сколько
занимает в
памяти структура
с ссылками

Constructor	Distance	Shallow Size	Retained Size
Object x70334	1	4 070 624 1 %	183 309 527 52 %
system / Context x39746	3	2 383 048 1 %	162 286 067 46 %
(string) : 000100	2	1 12 642 008 41 %	1 12 642 158 41 %

▶ Object x71417	1	4 129 424	1 %
▶ system / Context x41139	3	2 449 872	1 %
▼ (string) x224864	2	143 961 104	40 %
▶ "{Object.<anonymous>":function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com */ /*	49	1 826 784	1 %
▶ "{Object.<anonymous>":function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com */ /*	49	1 826 784	1 %
▶ "{Object.<anonymous>":function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com */ /*	15	1 826 784	1 %
▶ "/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com */ /* vim: set ts=2: */ /*exported XLSX */ /*global global, exports, module, require:fa	11	1 826 608	1 %
▶ "/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com */ /* vim: set ts=2: */ /*exported XLSX */ /*global global, exports, module, require:fa	10	1 826 608	1 %
▶ "{Object.<anonymous>":function(module,exports,require,__dirname,__filename,jest){/** * @license * Lodash <https://lodash.com/> * Copyright OpenJS	26	1 088 384	0 %
▶ "{Object.<anonymous>":function(module,exports,require,__dirname,__filename,jest){/** * @license * Lodash <https://lodash.com/> * Copyright OpenJS	25	1 088 384	0 %
▶ "{Object.<anonymous>":function(module,exports,require,__dirname,__filename,jest){/** * @license * Lodash <https://lodash.com/> * Copyright OpenJS	14	1 088 384	0 %
▶ "{ "version": "2023c", "zones": ["Africa/Abidjan LMT GMT g.8 0 01 -2ldXH.Q 48e5", "Africa/Nairobi LMT +0230 EAT +0245 -2r.g -2u -30 -2J 012132 -2u	10	776 496	0 %
▶ "/* * @license * Lodash <https://lodash.com/> * Copyright OpenJS Foundation and other contributors <https://openjsf.org/> * Released under MIT lic	11	544 112	0 %
▶ "/* * @license * Lodash <https://lodash.com/> * Copyright OpenJS Foundation and other contributors <https://openjsf.org/> * Released under MIT lic	10	544 112	0 %
▶ "{Object.<anonymous>":function(module,exports,require,__dirname,__filename,jest){/*! cpexcel.js (C) 2013-present SheetJS -- http://sheetjs.com */	53	459 408	0 %
▶ "{Object.<anonymous>":function(module,exports,require,__dirname,__filename,jest){/*! cpexcel.js (C) 2013-present SheetJS -- http://sheetjs.com */	53	459 408	0 %
▶ "{Object.<anonymous>":function(module,exports,require,__dirname,__filename,jest){/*! cpexcel.js (C) 2013-present SheetJS -- http://sheetjs.com */	16	459 408	0 %
▶ "/*! cpexcel.js (C) 2013-present SheetJS -- http://sheetjs.com */ /*jshint -W100 */ var cptable = {version:"1.15.0"}; cptable[437] = (function(){ v	11	459 232	0 %
▶ "/*! cpexcel.js (C) 2013-present SheetJS -- http://sheetjs.com */ /*jshint -W100 */ var cptable = {version:"1.15.0"}; cptable[437] = (function(){ v	10	459 232	0 %
▶ "{Object.<anonymous>":function(module,exports,require,__dirname,__filename,jest){'use strict'; Object.defineProperty(exports, '__esModule', { val	25	437 448	0 %

Retainers

Object	Distance	Shallow Size
▼ source in /Users/aleksandr.zaytsev/WebstormProjects/picker/node_modules/xlsx/xlsx.js @1950815	48	136 0 %
▼ script_or_debug_info in parse_xlscfb @2114279	47	56 0 %
▼ shared in parse_xlscfb() @2698181	46	64 0 %
▼ parse_xlscfb in Object @2441203	45	56 0 %
▼ XLSX in system / Context @2510325	44	96 0 %
▼ previous in system / Context @2510321	43	32 0 %
▼ context in Parser() @2510313	42	64 0 %

Чуть подробнее о том, как читать снэпшот

The screenshot displays the Chrome DevTools Memory tab with a heap snapshot. The top section shows a table of constructors, and the bottom section shows the retainers for a selected object.

Constructor	Distance	Shallow Size	Retained Size
Object x70334	1	4 070 624 1 %	183 309 527 52 %
system / Context x39746	3	2 383 048 1 %	162 286 067 46 %
(string) x223199	2	143 642 096 41 %	143 642 168 41 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	49	1 826 784 1 %	1 826 784 1 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	49	1 826 784 1 %	1 826 784 1 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	15	1 826 784 1 %	1 826 784 1 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	11	1 826 608 1 %	1 826 608 1 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	10	1 826 608 1 %	1 826 608 1 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	25	1 088 384 0 %	1 088 384 0 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	26	1 088 384 0 %	1 088 384 0 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	14	1 088 384 0 %	1 088 384 0 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	10	776 496 0 %	776 496 0 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	11	544 112 0 %	544 112 0 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	10	544 112 0 %	544 112 0 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	53	459 408 0 %	459 408 0 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	53	459 408 0 %	459 408 0 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	16	459 408 0 %	459 408 0 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	11	459 232 0 %	459 232 0 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	10	459 232 0 %	459 232 0 %
Object.<anonymous>:function(module,exports,require,__dirname,__filename,jest){/*! xlsx.js (C) 2013-present SheetJS -- http://sheetjs.com /* vim: set ts=2: */	25	437 448 0 %	437 448 0 %

Retainers	Distance	Shallow Size	Retained Size
Object			
source in /Users/aleksandr.zaytsev/WebstormProjects/picker/node_modules/xlsx/xlsx.js @1195085	48	136 0 %	1 838 520 1 %
script_or_debug_info in parse_xlscfb @1198249	47	56 0 %	80 0 %
shared in parse_xlscfb() @1221165	46	64 0 %	144 0 %
parse_xlscfb in Object @1216237	45	56 0 %	13 647 203 4 %
XLSX in system / Context @1339341	44	96 0 %	13 789 427 4 %
previous in system / Context @1225299	43	32 0 %	13 789 459 4 %
context in Parser() @1225159	42	64 0 %	488 0 %
constructor in Object @1212165	41	24 0 %	3 024 0 %
__proto__ in Parser @1212163	40	104 0 %	13 808 827 4 %
default in Object @1212885	39	32 0 %	32 0 %
_amSheetParser in system / Context @1212769	38	96 0 %	20 088 0 %
context in MAP_IMPORT_DATA() @1212041	37	56 0 %	288 0 %
MAP_IMPORT_DATA in Object @1211551	36	24 0 %	51 672 0 %

Как может выглядеть утечка в тестах

PASS tests/integration/Inventories/Done/02_process_adjustments.test.ts (574 MB heap size)

Завершение корректировок по продуктам инвентаризации

- ✓ Успех с завершением корректировок (1195 ms)

PASS tests/unit/features/04_group.test.js (493 MB heap size)

Группировка фичей

- ✓ Группировка фичей из заказа (103 ms)
- ✓ Группировка фичей из заказа с enum (125 ms)
- ✓ Группировка фичей с родителями (104 ms)

PASS tests/integration/Products/App/Uncheck/04_uncheck_water_and_milk_in_cis.test.ts (568 MB heap size)

анчек молочки и воды с необходимостью анрезервировать в CIS

- ✓ анчек молочки (824 ms)

PASS tests/integration/Orders/Features/01_order_features.test.js (601 MB heap size)

Получение группы фичей по заказу

- ✓ Не найдено
- ✓ Успех

PASS tests/integration/CourierConsumer/01_courier_consumer_update.test.ts (1050 MB heap size)



Какие метрики помогут?

База:  Grafana

Потребление памяти

Алерты на потребление памяти

Для тестов через jest

--logHeapUsage (в других тестовых фреймворках тоже есть аналогичные команды)

Что использовать для построения графиков?

Perf hooks, promclient, cronjob with memoryUsage()

Как улучшить?

Можно разделить график потребления памяти на разные колонки, которые есть в memoryUsage()

Алерты можно посчитать математически, если память сильно растет день ото дня, то что-то пошло не так, а нагрузка вроде как не растет

Пара советов напоследок...

- Будьте аккуратнее с кешом;
- Избегайте глобальных переменных;
- Аккуратнее с зависимостями;
- Думайте об использовании замыканий;
- Создавайте мусор;
- Читайте issue о библиотеках, которые используете.



Интересные факты

Проблемы с jest

Jest делает monkey patching всех require в проекте

В 16 версии node js

был не оптимизирован GC и одно время запускался без необходимых флагов

Утечки в node js

До 18 версии
JSON.parse(undefined)
создавала 128кб мусора

V8 кеширует строки

больше чем из 13
элементов

Полезные ссылки



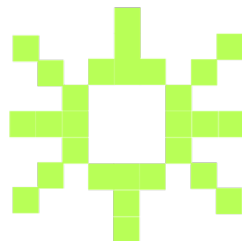
[Тимур Шемсединов
об утечках памяти](#)



[Документация от node js
по работе со
снелшотами памяти](#)



[V8 о Garbage Collector](#)



[Контакты](#)
<https://t.me/Yoshimuro1861>
kovachar@gmail.com