Browser Exploitation

Javascript Engine PWN Beginner

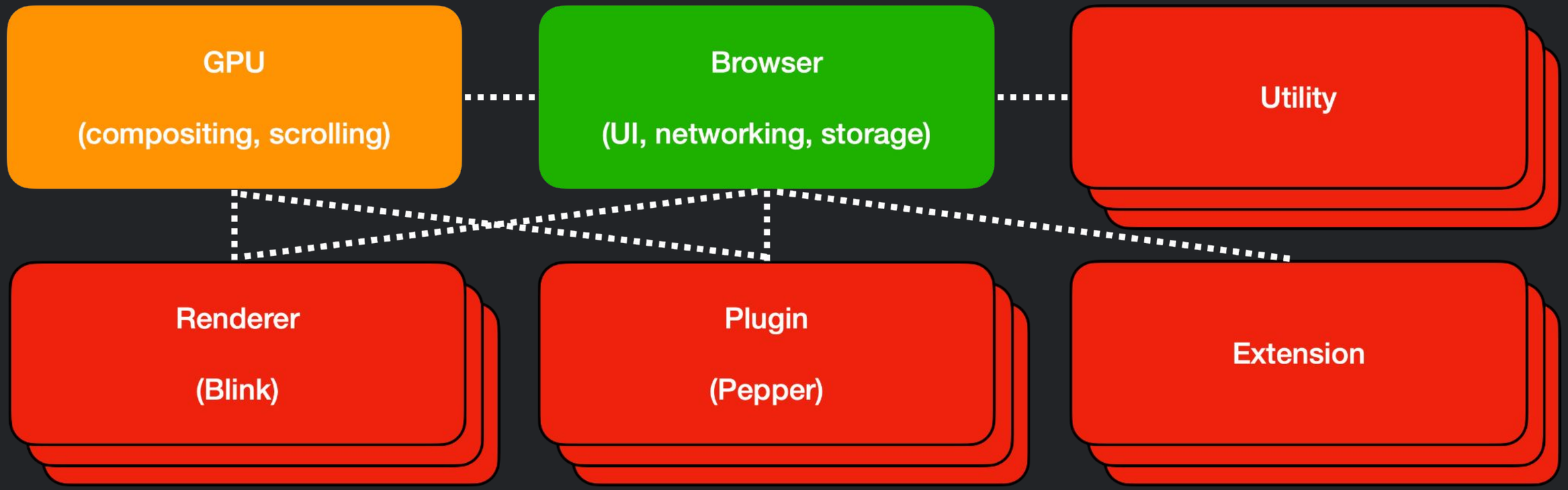
K1a

PWN Week 3

TABLE OF CONTENTS

INTRODUCTION

$ Multi-process Architecture



Slide credit: John Abd-El-Malek's Anatomy of a Browser

$ Renderer Process

- Blink
  - Rendering Engine
  - Parse HTML
  - Build DOM Tree
  - Embed V8

$ V8

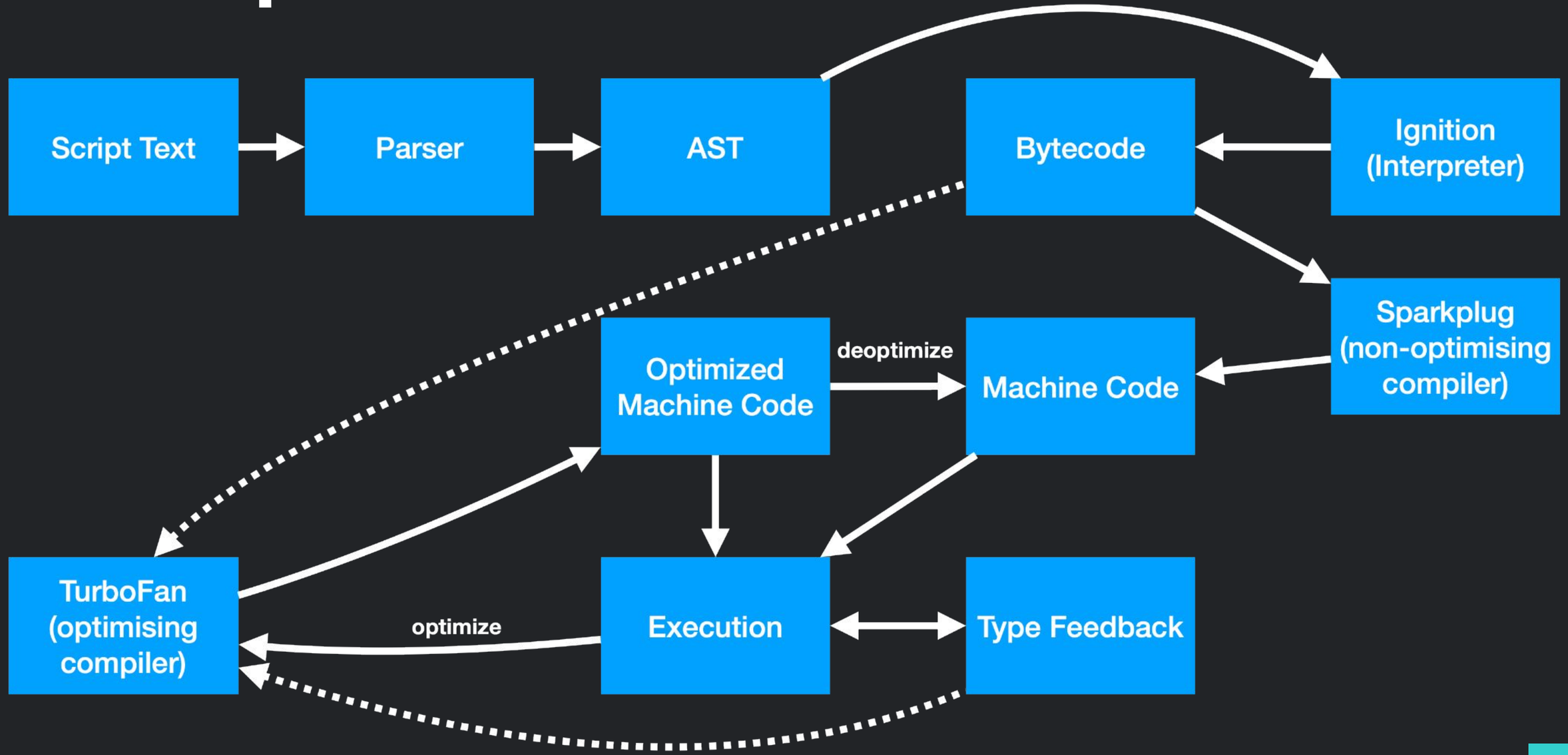
- Google’s open source high-performance JavaScript and WebAssembly engine

- compiles and executes Javascript source code

$ V8’s Public API

- Example
  - https://chromium.googlesource.com/v8/v8.git/+/HEAD/samples/hello-world.cc

$ V8 Pipeline


Script Text
Parser
AST
Bytecode
Ignition (Interpreter)
Sparkplug (non-optimising compiler)
Optimized Machine Code
deoptimize
Machine Code
TurboFan (optimising compiler)
optimize
Execution
Type Feedback

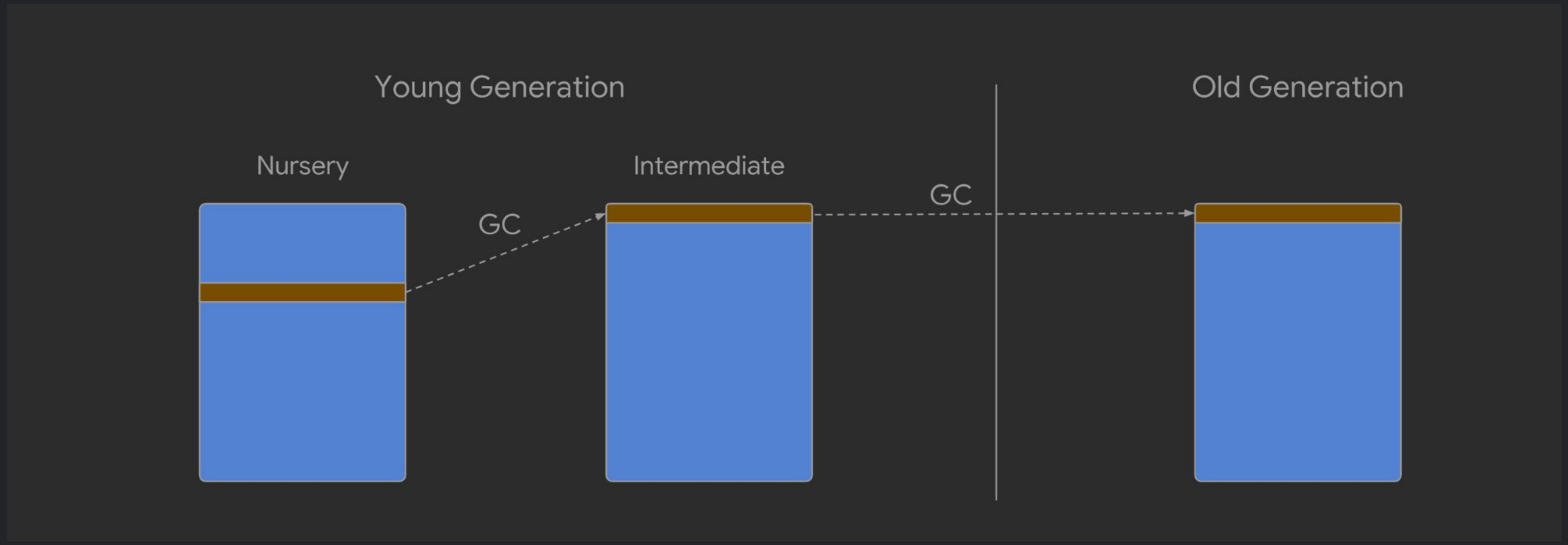
BACKGROUND

$ Isolate

- V8 VM instance
- contain v8 heap
- Size
  - 4GB
- Align
  - 4GB

```
0x2e8700000000     0x2e8700003000 rw-p     3000 0
0x2e8700003000     0x2e8700004000 ---p     1000 0
0x2e8700004000     0x2e8700023000 r-xp    1f000 0
0x2e8700023000     0x2e870003f000 ---p    1c000 0
0x2e870003f000     0x2e8707e80000 ---p  7e41000 0
0x2e8707e80000     0x2e8707fe0000 r-xp   160000 0
0x2e8707fe0000     0x2e8708000000 ---p    20000 0
0x2e8708000000     0x2e870802b000 r--p    2b000 0
0x2e870802b000     0x2e8708040000 ---p    15000 0
0x2e8708040000     0x2e870814d000 rw-p   10d000 0
0x2e870814d000     0x2e8708180000 ---p    33000 0
0x2e8708180000     0x2e8708183000 rw-p     3000 0
0x2e8708183000     0x2e87081c0000 ---p    3d000 0
0x2e87081c0000     0x2e8708240000 rw-p    80000 0
0x2e8708240000     0x2e8800000000 ---p f7dc0000 0
```

$ V8 Heap Overview



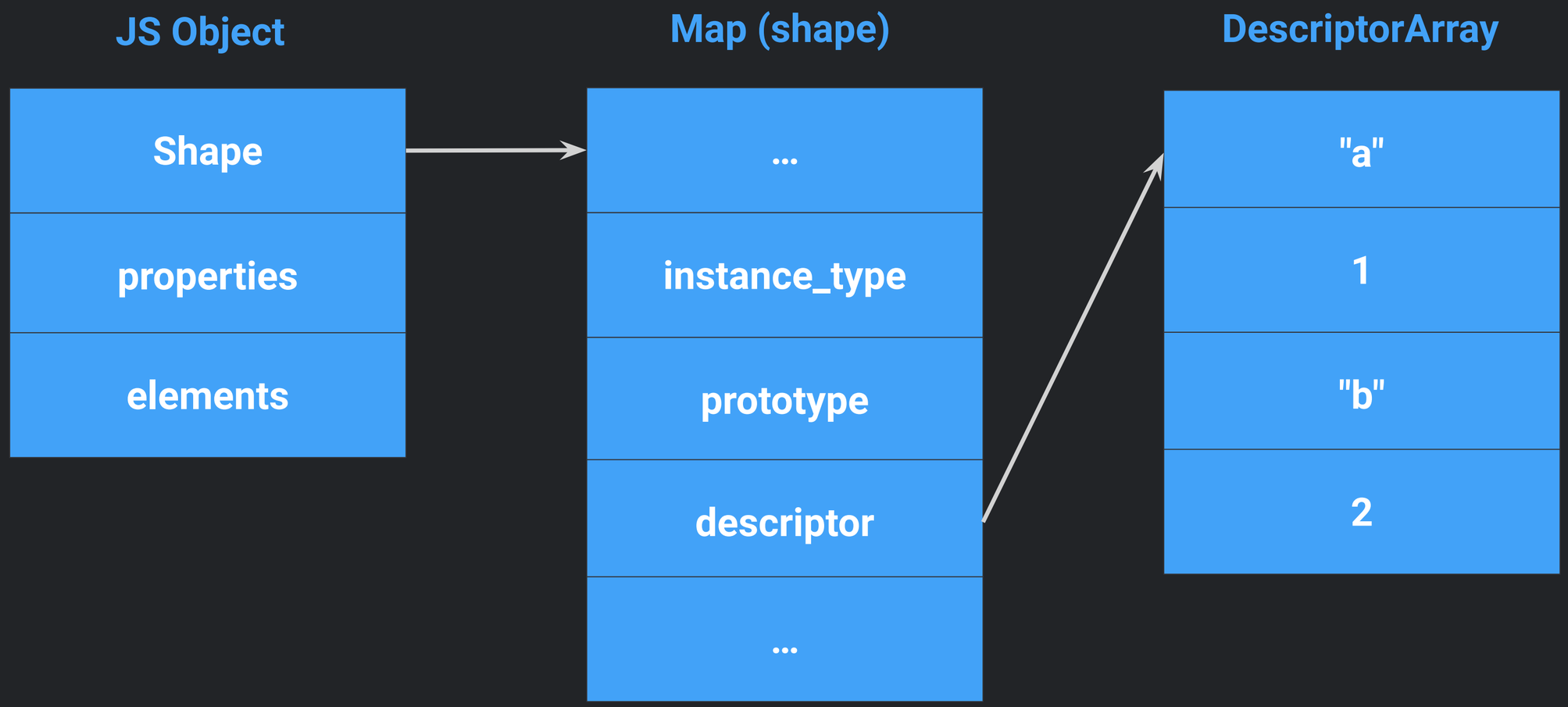
Reference: [Trash talk: the Orinoco garbage collector]

$ Javascript Object

- Javascript is weakly typed language
  - How to know the shape of an object?

$ Javascript Object

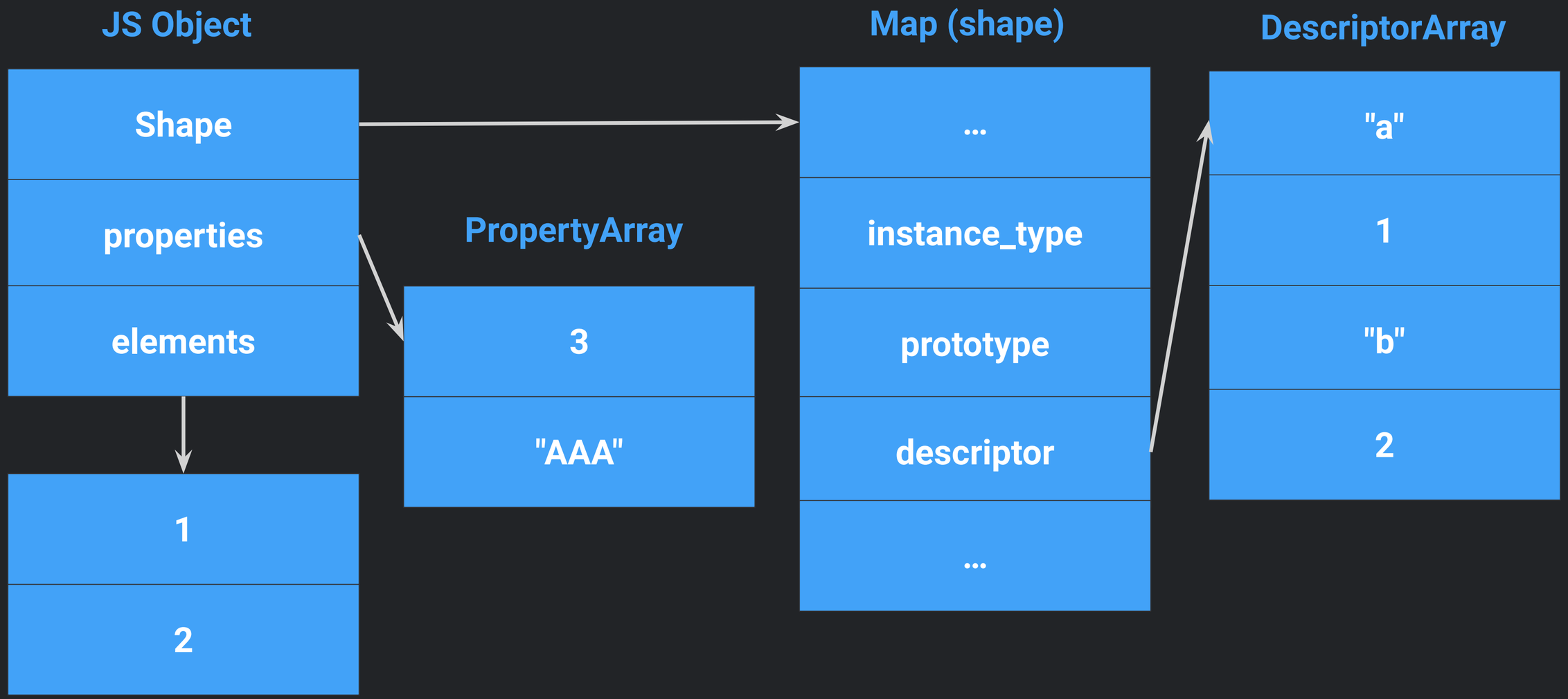
```
let a = [1, 2];
a.a = 3;
a.b = "AAA";
```



Reference: *Fast properties* in V8

$ Javascript Object

```
let a = [1, 2];
a.a = 3;
a.b = "AAA";
```



Reference: *Fast properties* in V8

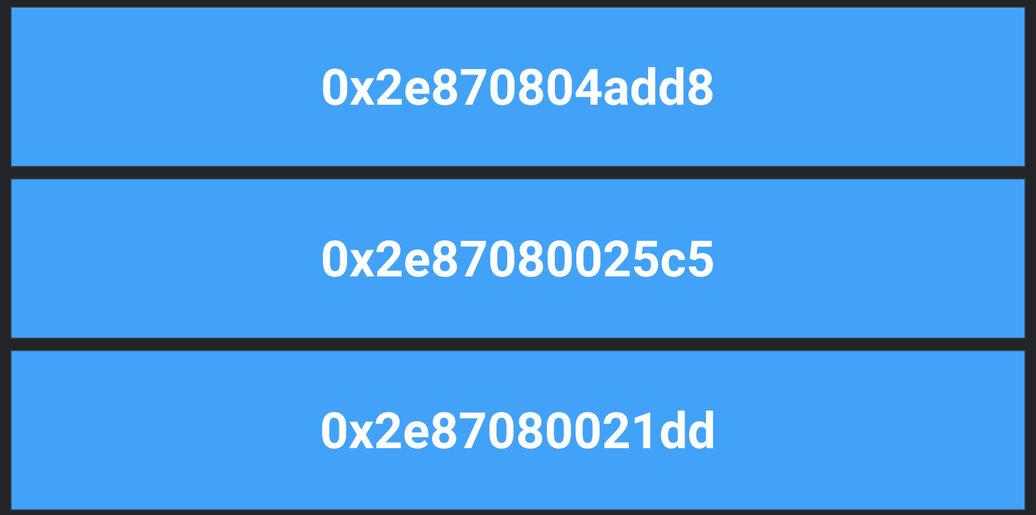
$ Map

- Map / HiddenClass / Shape
- contains information about
  - Instance_type
  - prototype
  - Instance_descriptors
  - …

- v8/src/objects/map.h

$ Pointer Compression

- Upper 32 bits are the same



Reference: Pointer Compression in V8

$ Pointer Compression

- Upper 32 bits are the same

0x2e87	0x0804add8
0x2e87	0x080025c5
0x2e87	0x080021dd

Reference: Pointer Compression in V8

$ Pointer Compression

- Store only the lower 32 bits

Reference: Pointer Compression in V8

$ Pointer Compression

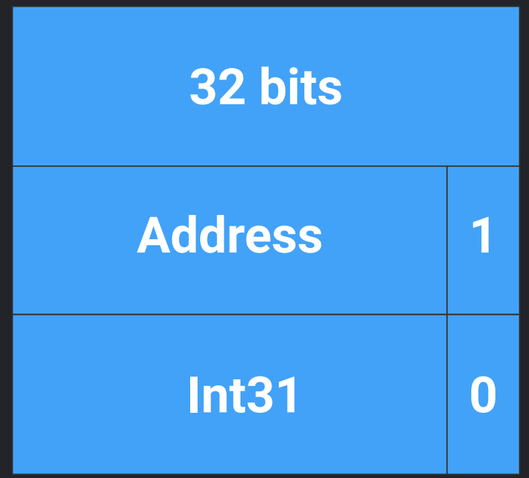
- Real Address = Base Address + Offset

	0x2e87	0x00000000
+		0x0804add8
=	0x2e87	0x0804add8

Reference: Pointer Compression in V8

$ Value tagging

- SMI
  - Small integer
  - lsb: 0
- HeapObject
  - lsb: 1



$ Example: MemDump

```
let a = [1, 2];
a.a = 3;
a.b = "AAA";
```

```
DebugPrint: 0x2e870804d935: [JSArray]
 - map: 0x2e8708207ce9 <Map(PACKED_SMI_ELEMENTS)> [FastProperties]
 - prototype: 0x2e87081cc151 <JSArray[0]>
 - elements: 0x2e87081d50f5 <FixedArray[2]> [PACKED_SMI_ELEMENTS (COW)]
 - length: 2
 - properties: 0x2e870804da45 <PropertyArray[3]>
 - All own properties (excluding elements): {
    0x2e8708004cc9: [String] in ReadOnlySpace: #length: 0x2e870814426d <AccessorInfo> (const accessor descriptor), location: descriptor
    0x2e87080082f5: [String] in ReadOnlySpace: #a: 3 (data field 0), location: properties[0]
    0x2e8708008391: [String] in ReadOnlySpace: #b: 0x2e87081d47b5 <String[3]: #AAA> (const data field 1), location: properties[1]
 }
 - elements: 0x2e87081d50f5 <FixedArray[2]> {
           0: 1
           1: 2
 }
```

```
0x2e870804d934: 0x08207ce9      0x0804da45      0x081d50f5      0x00000004
```

BUGs

$ Out-of-bounds

```
let l = [1, 2, 3]
let y = x();          // compiler: range(0,1), real: 5
l[y]                  // bounds-checking elimination
```

$ Out-of-bounds

```
let l = [1, 2, 3]
let y = x();          // compiler: range(0,1), real: 5
l[5]                  // out-of-bounds access
```

$ Type confusion

```
y = typeA() // Compiler: typeA, Real: typeA
x()         // Compiler: typeA, Real: typeB
y           // Type confusion
```

EXPLOIT

$ addrof / fakeobj Primitive

- origin: Attacking JavaScript Engines: A case study of JavaScriptCore and CVE-2016-4622 - saelo
- addrof
  - return the address of an object
- fakeobj
  - return an object point to given address

$ WASM page (rwx page)

```
let bytes = new
Uint8Array([0,97,115,109,1,0,0,0,1,133,128,128,128,0,1,96,0,1,127,3,130,128,128,128,0,
1,0,4,132,128,128,128,0,1,112,0,0,5,131,128,128,128,0,1,0,1,6,129,128,128,128,0,0,7,14
4,128,128,128,0,2,6,109,101,109,111,114,121,2,0,3,112,119,110,0,0,10,138,128,128,128,0
,1,132,128,128,128,0,0,65,42,11]);
let mod = new WebAssembly.Module(bytes);
let instance = new WebAssembly.Instance(mod);
let pwn = instance.exports.pwn;
```

```
0x2e87082400000    0x2e8800000000 ---p 17dc0000 0      [anon_2e870824]
0x38378ecdb000     0x38378ecdc000 rwxp     1000 0      [anon_38378ecdb]
```

$ ArrayBuffer Backing Store

```
DebugPrint: 0x2e870804e505: [JSArrayBuffer]
 - map: 0x2e8708203289 <Map(HOLEY_ELEMENTS)> [FastProperties]
 - prototype: 0x2e87081ca3c9 <Object map = 0x2e87082032b1>
 - elements: 0x2e8708002249 <FixedArray[0]> [HOLEY_ELEMENTS]
 - embedder fields: 2
 - backing_store: 0x56522070ab50
 - byte_length: 256
 - max_byte_length: 256
 - detachable
 - properties: 0x2e8708002249 <FixedArray[0]>
 - All own properties (excluding elements): {}
 - embedder fields = {
    0, aligned pointer: (nil)
    0, aligned pointer: (nil)
 }
```

```
0x2e870804e504: 0x08203289      0x08002249      0x08002249      0x00000100
0x2e870804e514: 0x00000000      0x00000100      0x00000000      0x2070ab50
0x2e870804e524: 0x00005652      0x2070ac60      0x00005652      0x00000002
```

$ OOB Exploit Flow

1. Trigger bugs to get OOB access (within v8 isolate)
2. Make addrof() / fakeobj() primitives
3. Use addrof() and fakeobj() to create a fake object
4. Use addrof() and fake_object to get address of wasm page (rwx page)
5. Modify ArrayBuffer’s backing_store to wasm page (arbitrary read/write)
6. Write shellcode to wasm page
7. Execute wasm function

Debugging Tools

$ Debugging Tools

- gdbinit
  - echo "source /PATH/TO/gdbinit" >> ~/.gdbinit
- d8
  - --allow-natives-syntax
- natives-syntax
  - https://gist.github.com/totherik/3a4432f26eea1224ceeb
  - %DebugPrint()
  - %SystemBreak()
- Chrome
  - --js-flags

LAB

$ LAB

-

THANK YOU FOR LISTENING!

ANY QUESTIONS?