DEVIEW
2019

눈발자국

박한범
Kosslab
NAVER

CONTENTS

DEVIEW
2019

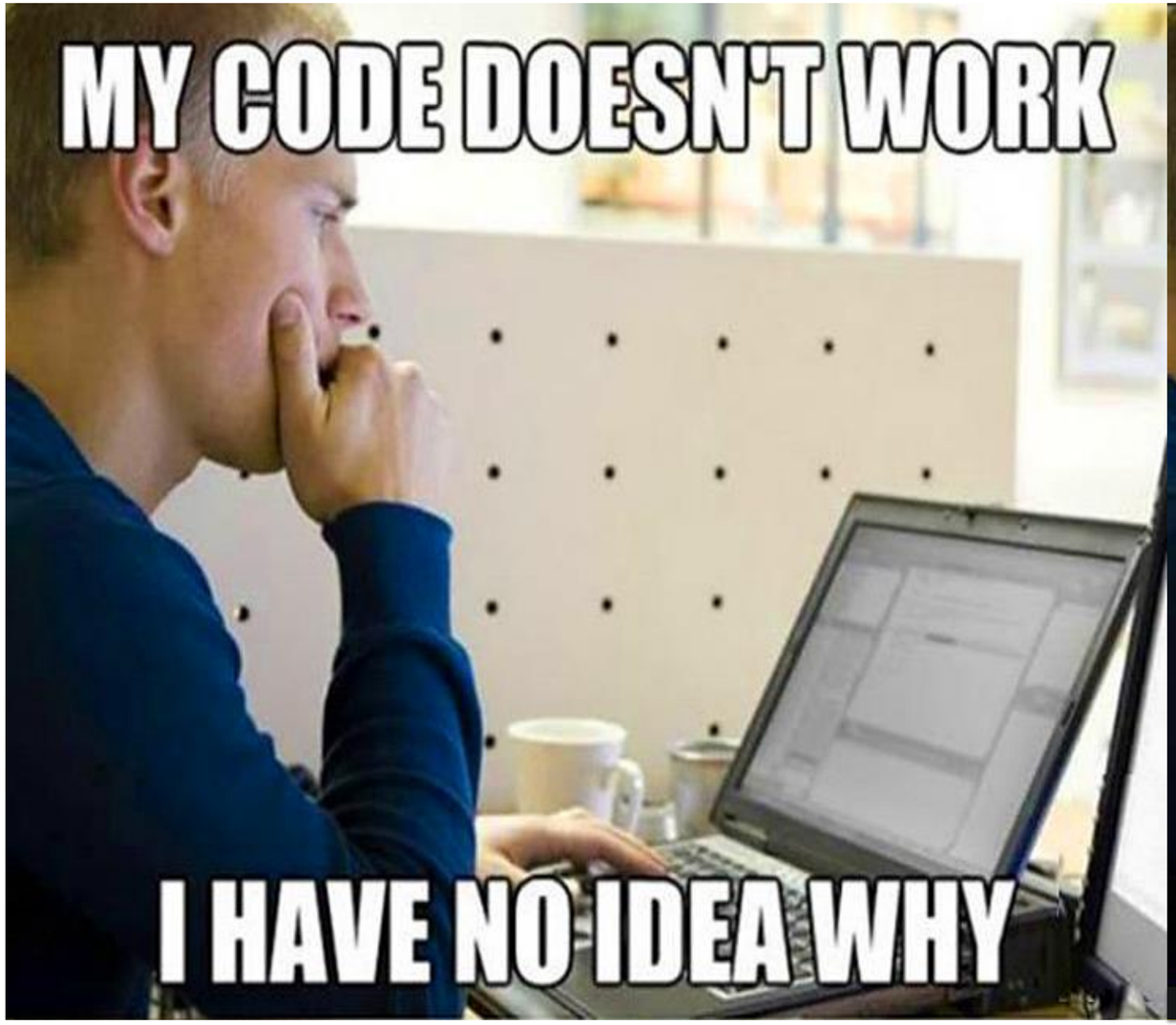
1. Tracing 소개, Printf is nothing or everything.
2. Tracing 에 도움이 되는 요소들
3. Tracing 오픈소스 접근성을 올려주는 요소
4. Tracing 에 배경지식과 개발자 메시지 입히기
5. 눈발자국

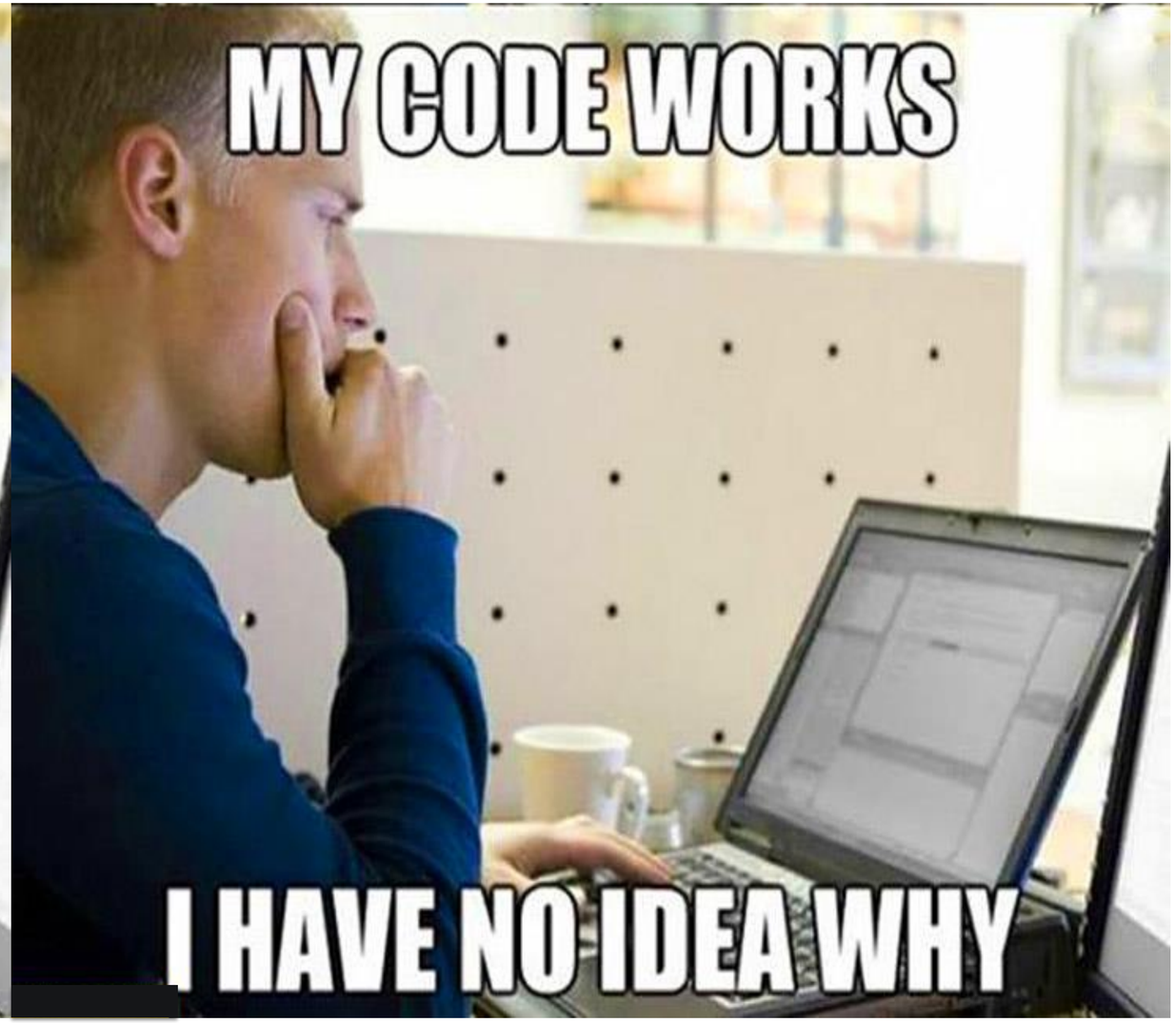
DEVIEW
2019

Tracing 소개

Printf is nothing, or everything

프로그래밍은 쉬운 적이 없죠





코드의 시각화를 위한 Printf 문의 발전

DEVIEW
2019

```
Printf(“function entry : %d %d\n”, a, b);
```

```
Static bool debug = true;
if (debug)
  Printf(“function entry : %d %d\n”, a, b);
```

```
#ifdef DEBUG
  #define DBGPR(a, args...) printf("%s(%s:%d) " a, __func__,__FILE__, __LINE_
#else
  #define DBGPR(a, args...)
#endif
DBGPR("%d %d\n", a, b);
```

Printf is nothing, or everything

DEVIEW 2019

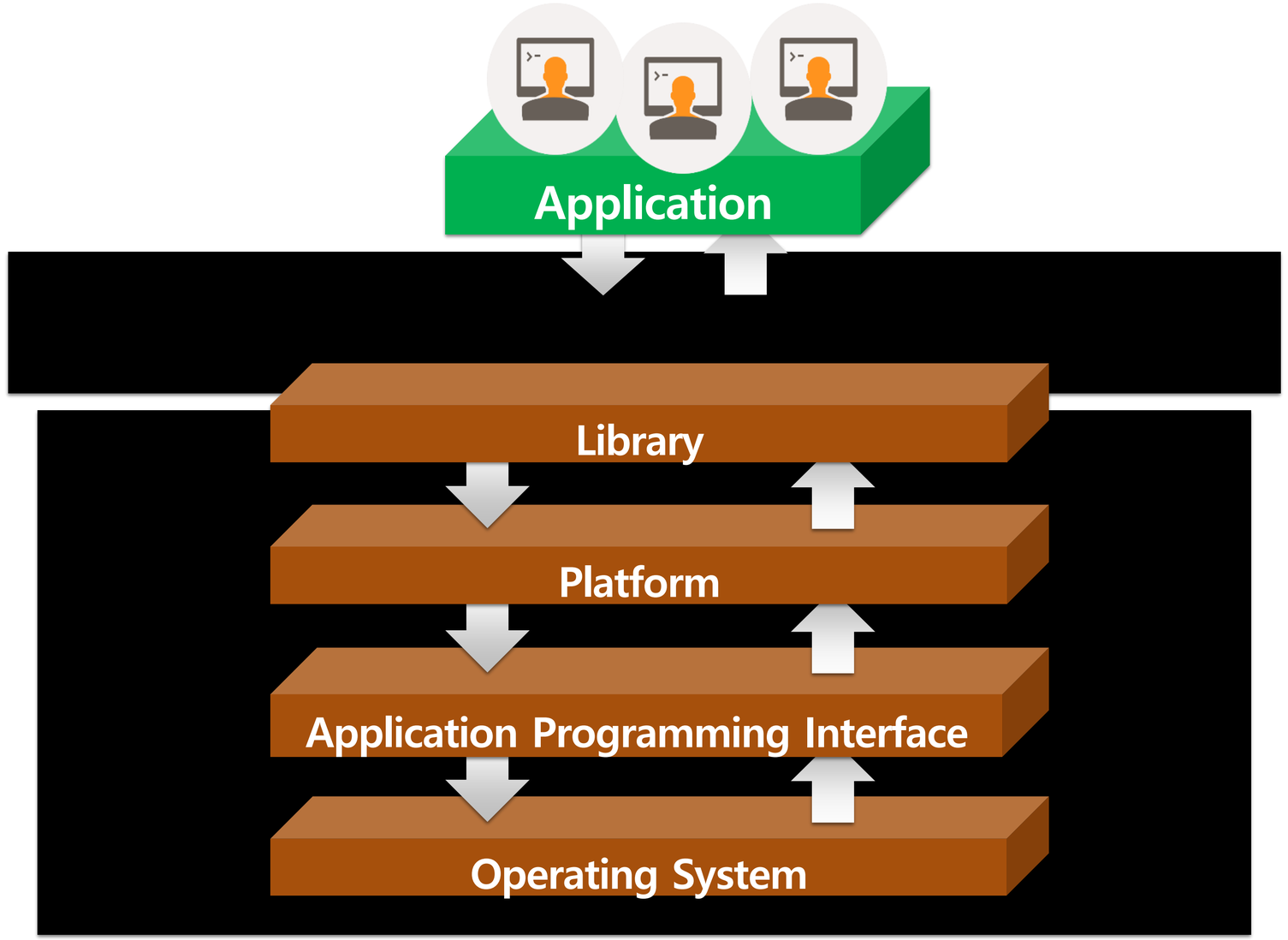
개발자가 Printf 를 활용하는 방식 = Tracing

> Tracing, “프로그램의 실행 동안의 변화를 다각도로 기록하고 이를 검토하는 일”

기록을 위한 장치 = 계측기 = Printf = instrument

Printf 보다 고도화 된 Instrument의 필요성

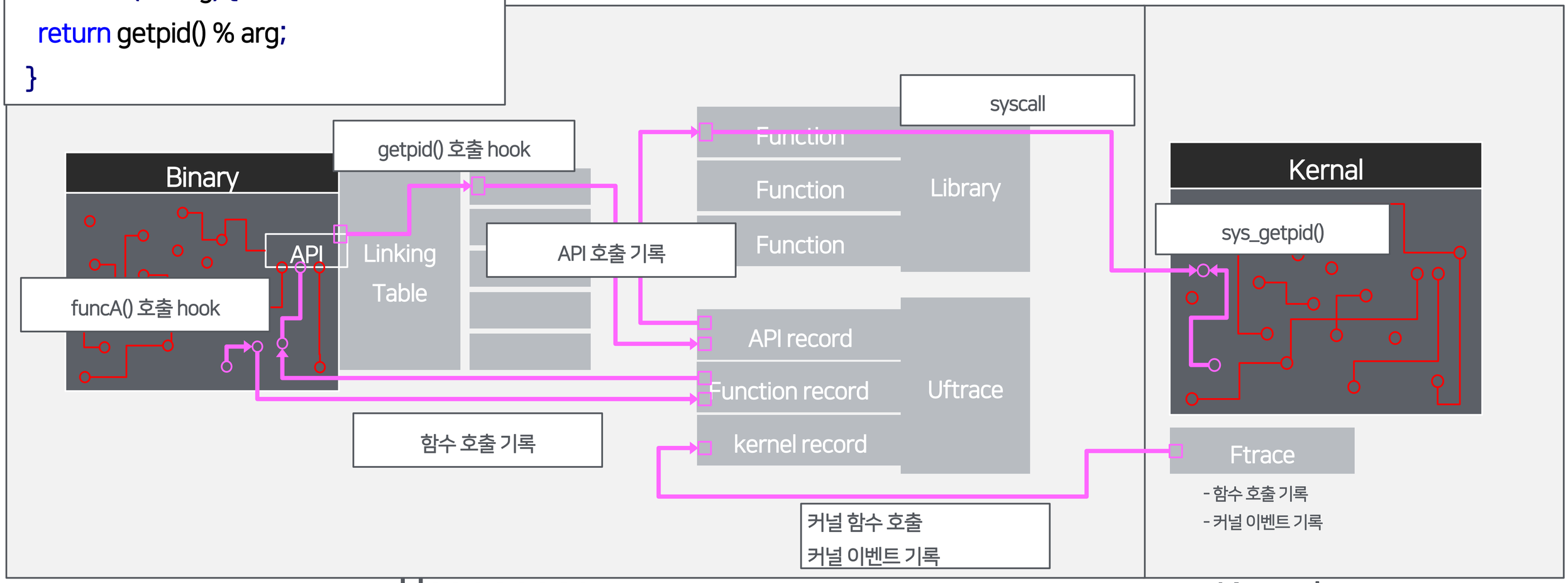
계측기=Printf
필요



Printf, Printf, Printf, Printf 를 가능케 하라

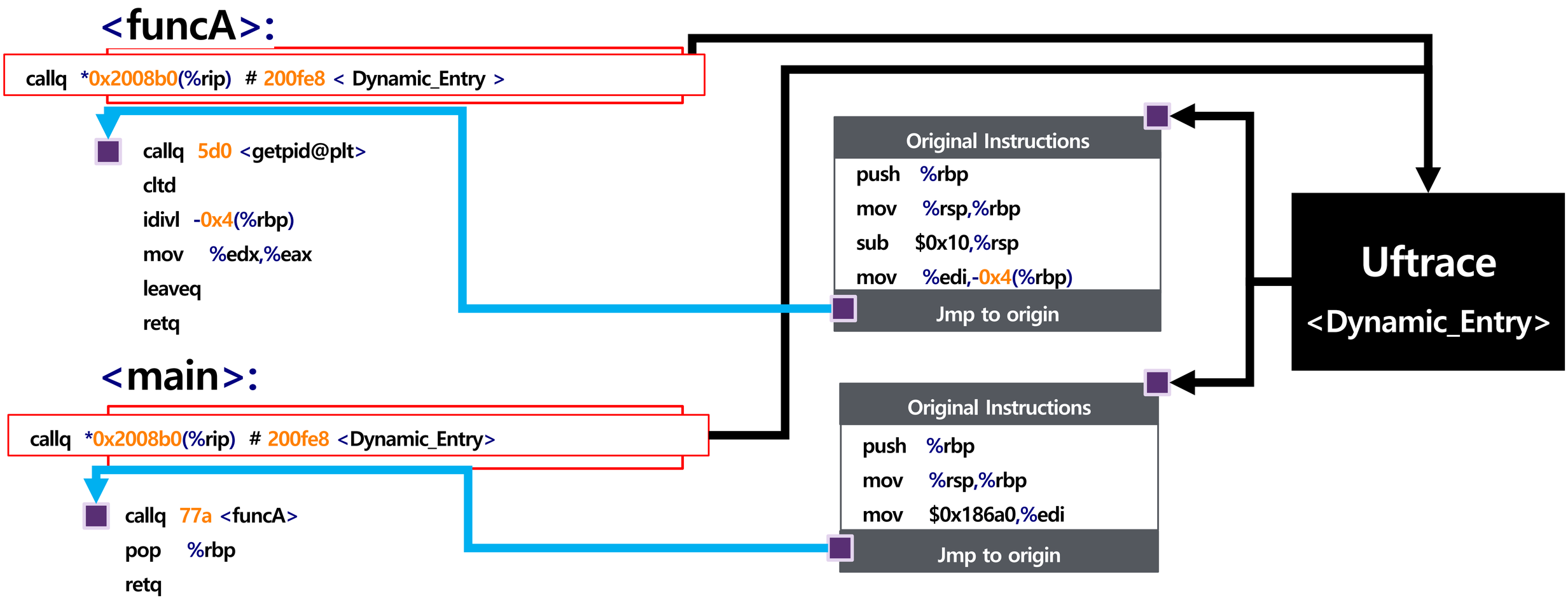
DEVIEW 2019

```
int funcA(int arg) {
  return getpid() % arg;
}
```



Printf 안되면 되게 하라!

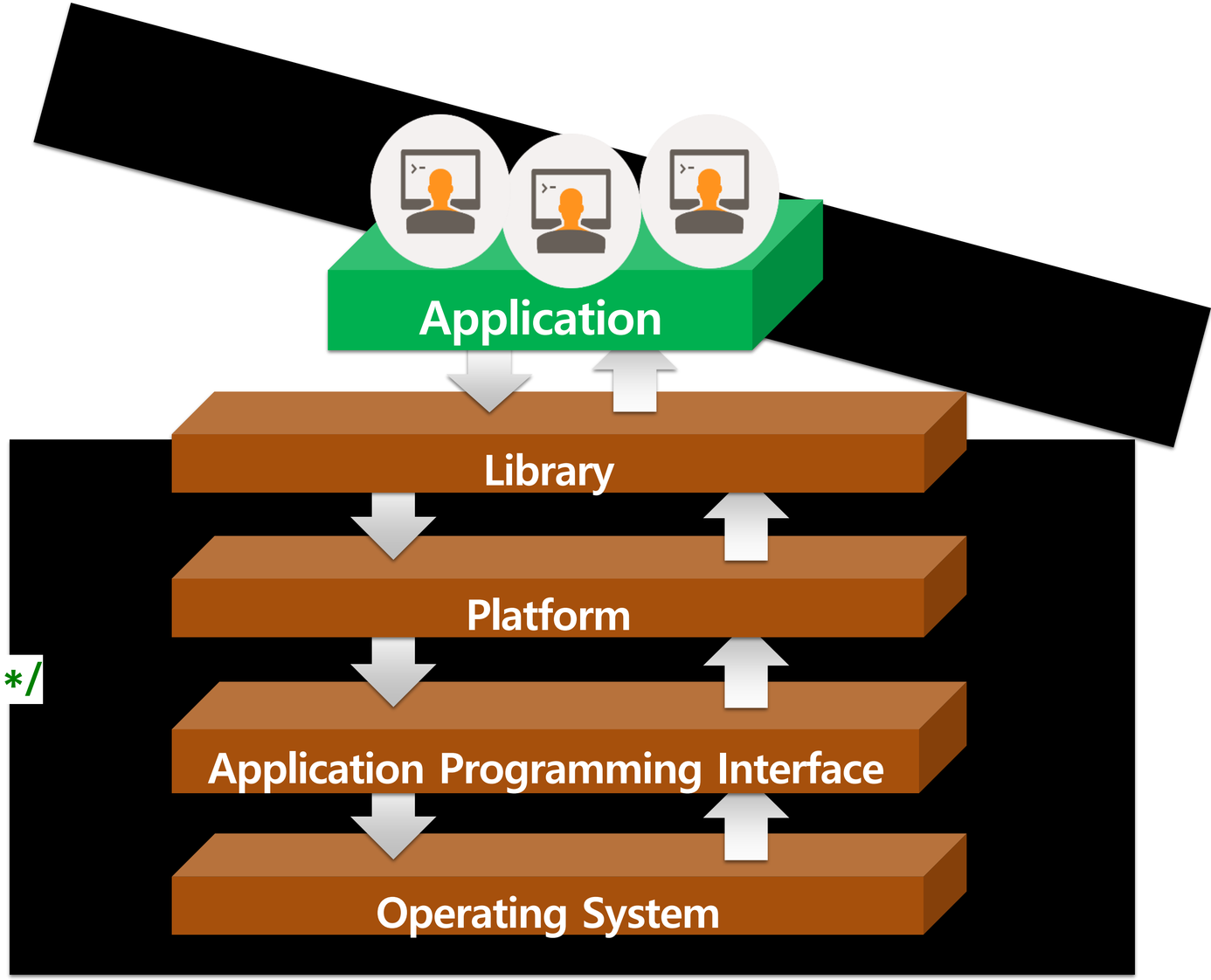
DEVIEW 2019



모든 영역에서 Printf-ing 하다

DEVIEW 2019

```
$ sudo ./uftrace -L. -k -a test
# DURATION     TID     FUNCTION
            [  1598] ¦ main() {
            [  1598] ¦   funcA(100000) {
            [  1598] ¦     getpid() {
   0.269 us [  1598] ¦       sys_getpid();
   1.375 us [  1598] ¦     } = 1598; /* getpid */
   1.375 us [  1598] ¦   } = 1598; /* funcA */
   2.355 us [  1598] ¦ } = 1598; /* main */
```



기록했으니 검토를 시작해야죠…

DEVIEW 2019

Tracing, “프로그램의 실행 동안의 변화를 다각도로 기록하고 이를 검토하는 일”

- 함수 호출
- API 호출
- 커널 함수 호출
- 함수 호출 인자
- 함수 반환 인자

- 호출 기록 검토
- 시각화

DEVIEW
2019

Tracing
도움이 되는 요소들

Gcc에서 만난 문제를 추적했던 경우


```
static inline __attribute__((always_inline)) char* get_msg() {
  char str[64] = "Hello world\n";
  return (char*)str;
}
int main() {
  char* p;
  p = get_msg();
  puts(p);
  return 0;
}
```

```
$ ./compile_by_gcc
Segmentation fault (core dumped)
```

```
$ ./compile_by_clang
Hello world
```

Mitigation도 좋지만 해결책을 찾아볼까?


```
static inline __attribute__((always_inline)) char* get_msg() {
  char* str = "Hello world\n";
  return str;
}
```

Mitigation 방안 #1 - 문자를 수정 못함

```
static inline __attribute__((always_inline)) char* get_msg() {
  const char* str = "Hello world\n";
  char* p = malloc(strlen(str));
  strcpy(p, str);
  return p;
}
```

Mitigation 방안 #2 - 메모리 해제가 강제됨

개발자가 남긴 #1 Warning Message


```
$ gcc inline.c
inline.c: In function ‘get_msg:
inline.c:6:9: warning: function returns address of local variable
[-Wreturn-local-addr]
return (char *)str;
```

Warning Message 로 찾은 문제지점


```
if (DECL_P (inner)
    && !DECL_EXTERNAL (inner)
    && !TREE_STATIC (inner)
    && !DECL_DECLARED_INLINE_P(current_function_decl)
    && DECL_CONTEXT (inner) == cu
    warning_at(loc, OPT_Wreturn_local_addr,
    "function returns address of local variabl
    tree zero = build_zero_cst(TREE_TYPE(res
    t = build2(COMPOUND_EXPR, TREE_TYP
  }
}
```

return 될 tree가 속한 함수가
inline 으로 선언되어 있으면
NULL 로 바꾸지 마세요!

Current_function_decl.function_decl	
static_flag = 1	declared_inline_flag = 1

개발자가 남긴 #2 Debug 용 로그

$ gcc –O1 –fdump-tree-all-details inline.c

```
;; Function get_msg (get_msg, funcdef_no=0, decl_uid=1792,
cgraph_uid=0, symbol_order=0)
Deleted dead store: str = "Hello world₩n";
__attribute__((a                inline)) get_msg ()
{
    char str[64];
    <bb 2> [0.00
    str ={v} {CLOB
    return &str;
}
```

inline.c.040t.dse1

Deleted dead store: str = "Hello world₩n";

메모리 최적화 때문일까?


```
if (DECL_P (inner) && !DECL_EXTERNAL (inner)
    && !TREE_STATIC (inner)
    && !DECL_DECLARED_INLINE_P(current_function_decl)
    && DECL_CONTEXT (inner) == current_function_decl)
```

```
warning_at(loc, OPT_Wreturn_local_addr,
"function returns address of local variable");
```

```
else if (DECL_DECLARED_INLINE_P(current_function_decl))
  inner->base.volatile_flag = true;
```

메모리 관련 최적화
Disable

개발자가 남긴 메시지의 중요성


```
Dump of assembler code for function main:
<+4>:     movabs $0x6f77206f6c6c6548,%rax
<+14>:    mov    $0xa646c72,%edx
<+19>:    mov    %rax,(%rsp)
<+23>:    mov    %rdx,0x8(%rsp)
<+28>:    movq   $0x0,0x10(%rsp)
<+37>:    movq   $0x0,0x18(%rsp)
<+46>:    movq   $0x0,0x20(%rsp)
<+55>:    movq   $0x0,0x28(%rsp)
<+64>:    movq   $0x0,0x30(%rsp)
<+73>:    movq   $0x0,0x38(%rsp)
<+82>:    mov    %rsp,%rdi
<+85>:    callq  0x400400 <puts@plt>
```

O1

$ gcc –O1 inline.c

$ ./a.out

Hello world

```
Dump of assembler code for function main:
<+4>:     movdqa 0x1b4(%rip),%xmm0
<+12>:    mov    %rsp,%rdi
<+15>:    movaps %xmm0,(%rsp)
<+19>:    pxor   %xmm0,%xmm0
<+23>:    movaps %xmm0,0x10(%rsp)
<+28>:    movaps %xmm0,0x20(%rsp)
<+33>:    movaps %xmm0,0x30(%rsp)
<+38>:    callq  0x400400 <puts@plt>
```

O2

$ gcc –O2 inline.c

$ ./a.out

Hello world

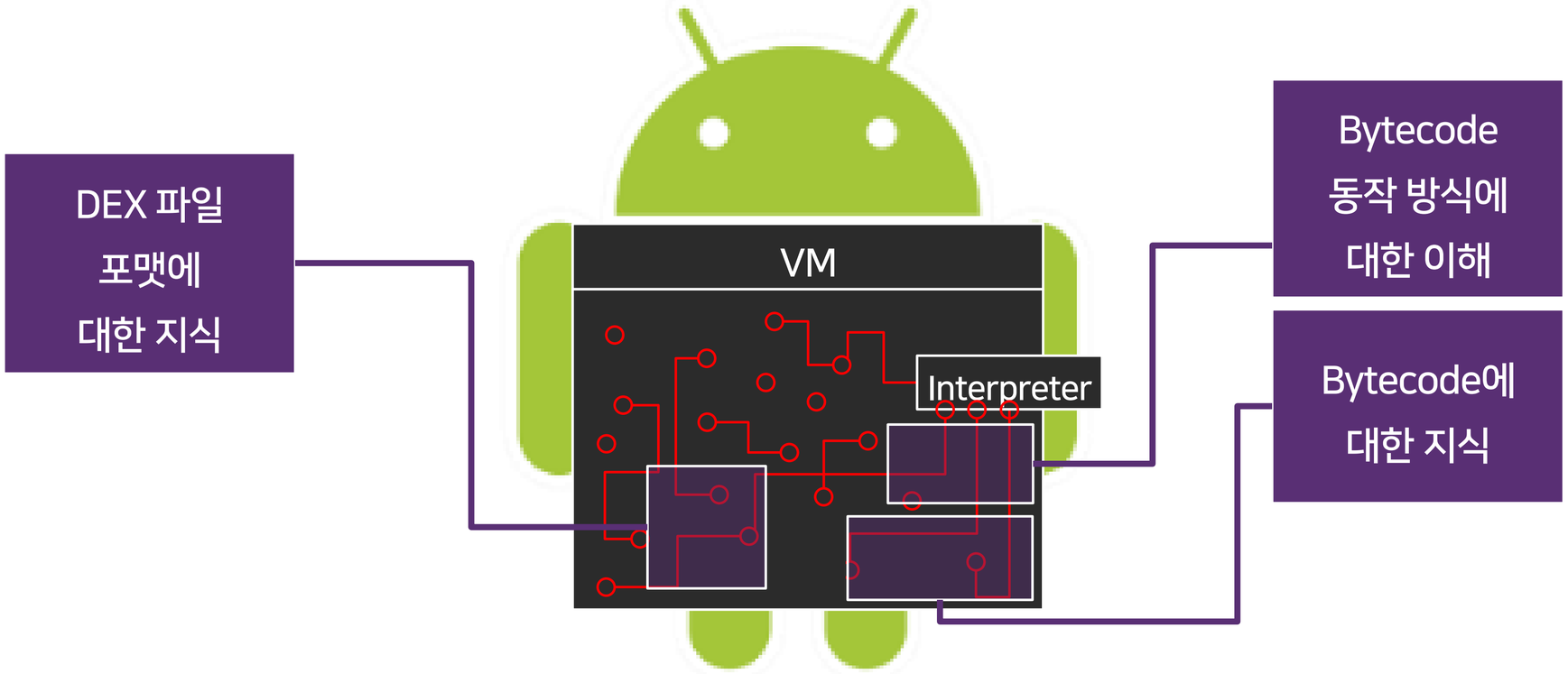
특정 오픈소스를 분석해야 하는 경우


```
public class Main {
    public static void main(String[] args) {
            System.out.println("Hello World!");
        }
}
```

```
$ javac Main.java
$ dx --dex --output=classes.dex Main.class
$ zip Main.apk classes.dex
$ uftrace record -a `which dalvikvm64` –cp Main.apk Main
```

Tracing 에 도움이 되는 요소들

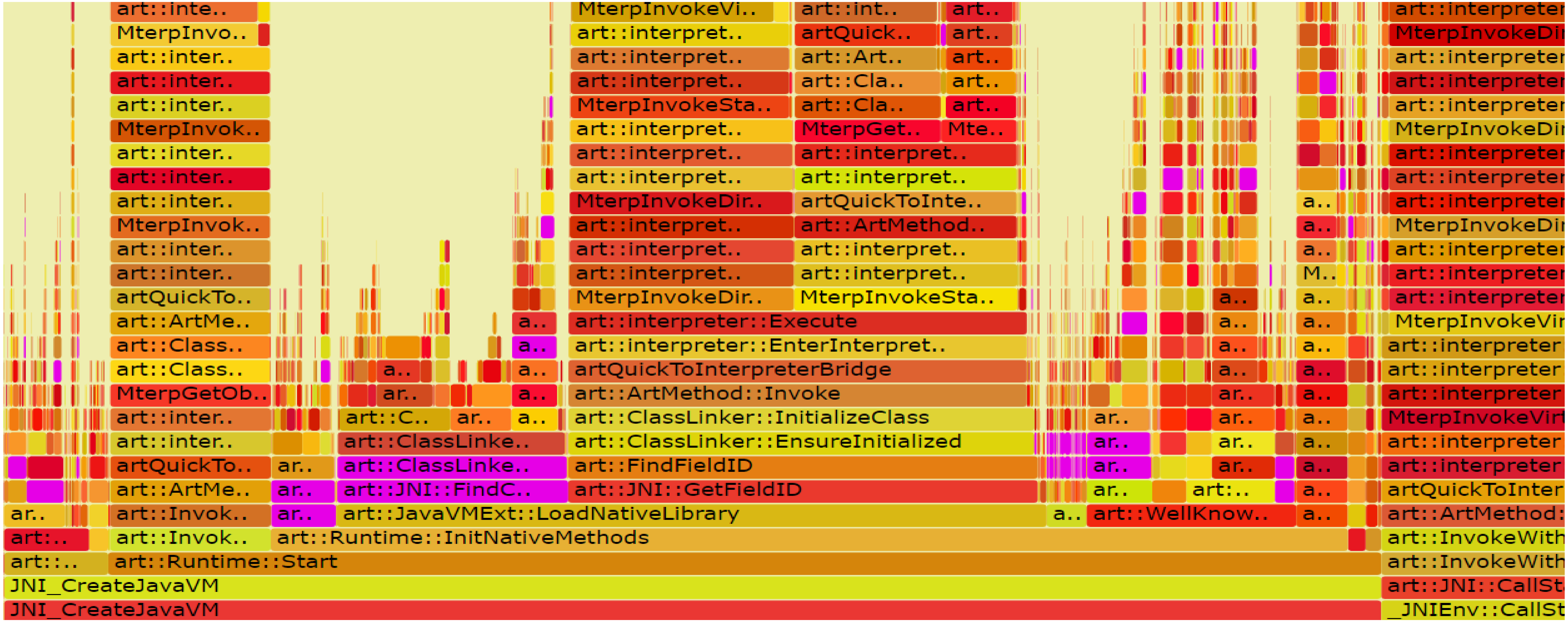
거대 오픈소스를 분석할 때는 관련 배경지식이 매우 중요합니다



Tracing 에 도움이 되는 요소들

DEVIEW 2019

Tracing 시각화의 유용성



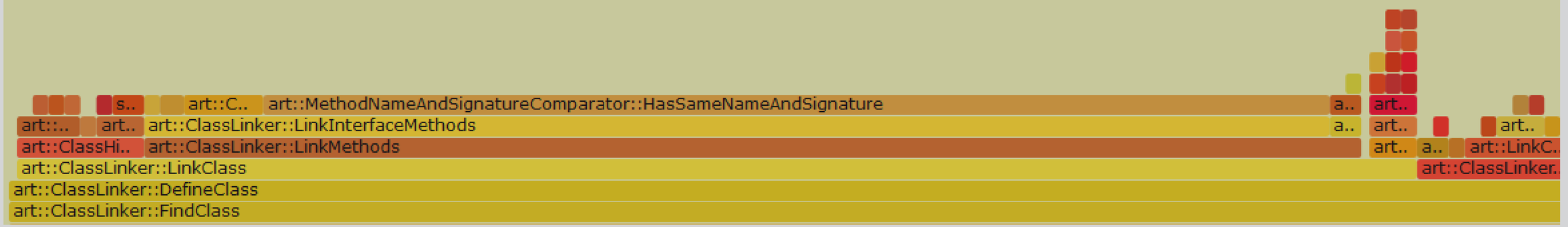
Tracing 에 도움이 되는 요소들

DEVIEW 2019

Tracing 시각화의 유용성

Class Path

art::ClassLinker::FindClass
└ art::ClassLinker::DefineClass
└ art::ClassLinker::LinkClass
└ art::ClassLinker::LinkMethods
└ art::ClassLinker::LinkInterfaceMethods



DEVIEW
2019

Tracing
오픈소스 접근성을 올려주는 요소

개발자가 남긴 메시지는 매우 중요!!

DEVIEW
2019

```
;; Function get_msg (get_msg, funcdef_no=0, decl_uid=1792, cgraph_uid=0, symbol_order=0)
Deleted dead store: str = "Hello world₩n";
__attribute__((always_inline))  get_msg ()
{
    char str[64];
    <bb 2> [0.00%]:
    str ={v} {CLOBBER};
    return &str;
}
```

코드는 접근성이 매우 떨어집니다.


```
;; Function get_msg (get_msg, funcdef_no=0, decl_uid=1792, cgraph_uid=0, symbol_order=0)
De
__
{
  str ={v} {CLOBBER};
  return &str;
}
```

```
(gdb) p *gsi->ptr
$4 = {code = GIMPLE_ASSIGN, no_warning = 0, visited = 0,
nontemporal_move = 0, plf = 1, modified = 0, has_volatile_ops
= 0, pad = 0, subcode = 32, uid = 0,
    location = 2147483655, num_ops = 2, bb = 0x7ffff6ede478,
next = 0x7ffff702d190, prev = 0x7ffff702d140}
```

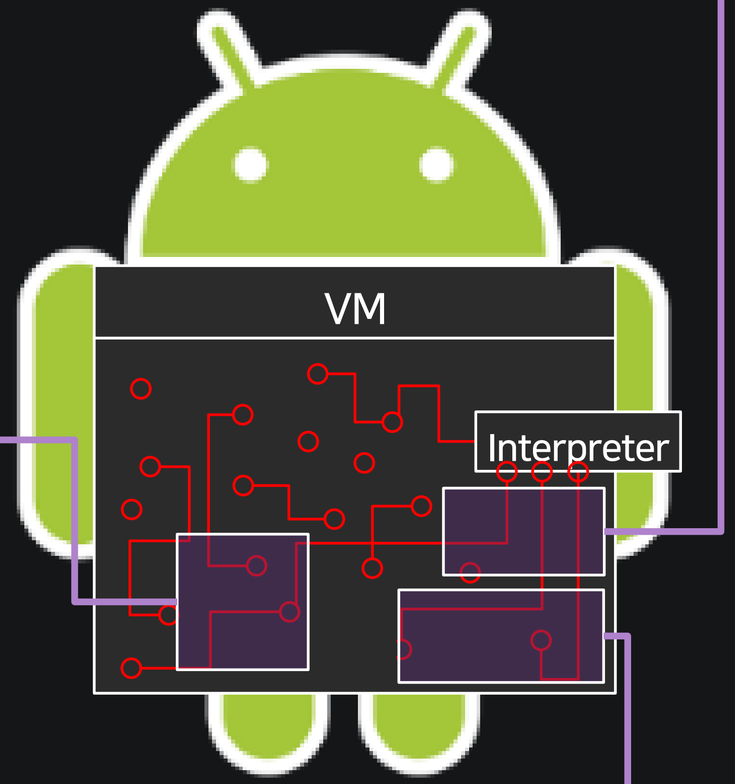
Tracing에는 배경지식이 매우 중요합니다.

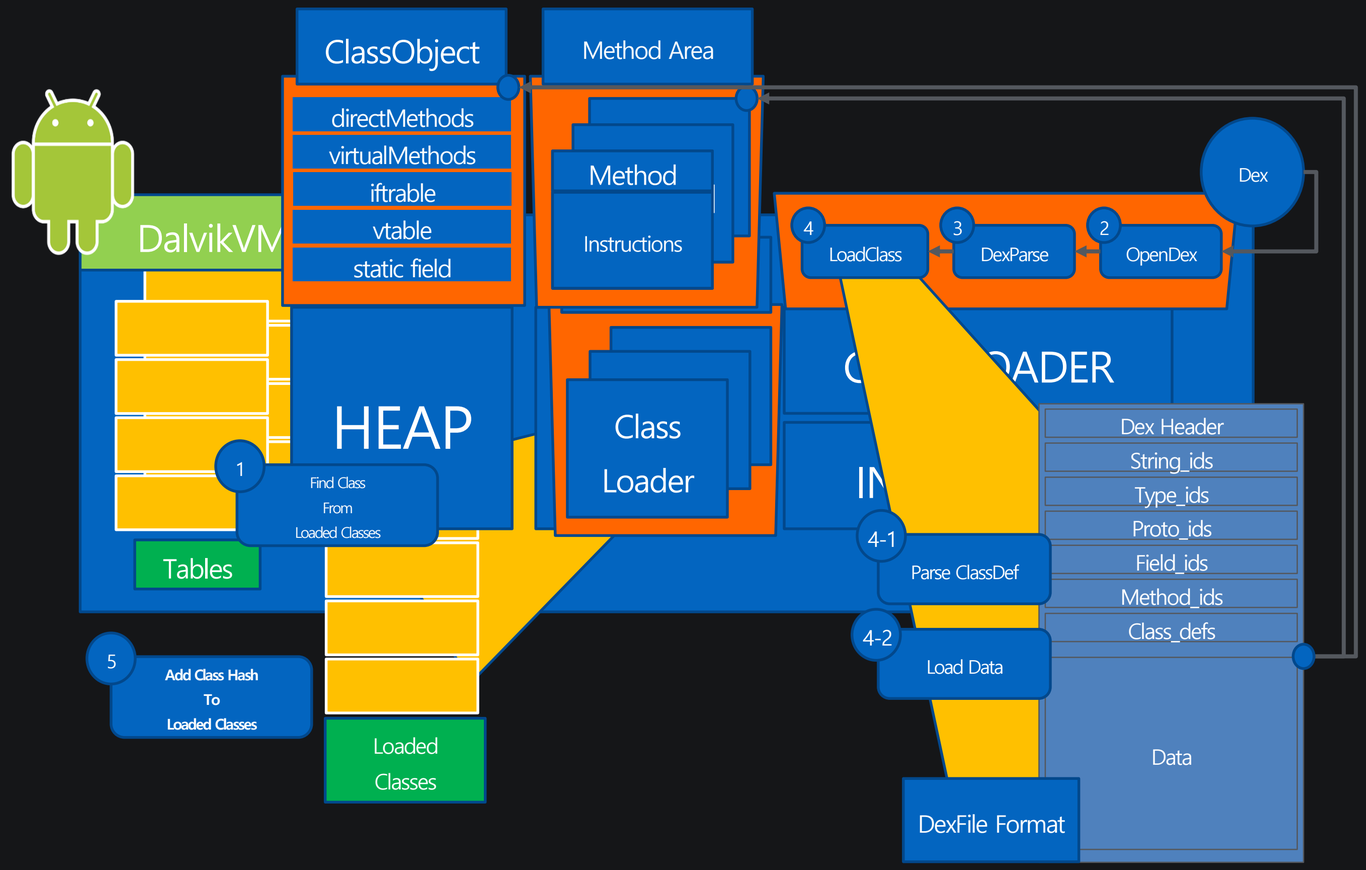
DEVIEW 2019

Bytecode
동작 방식에
대한 이해

DEX 파일
포맷에
대한 지식

Bytecode에
대한 지식

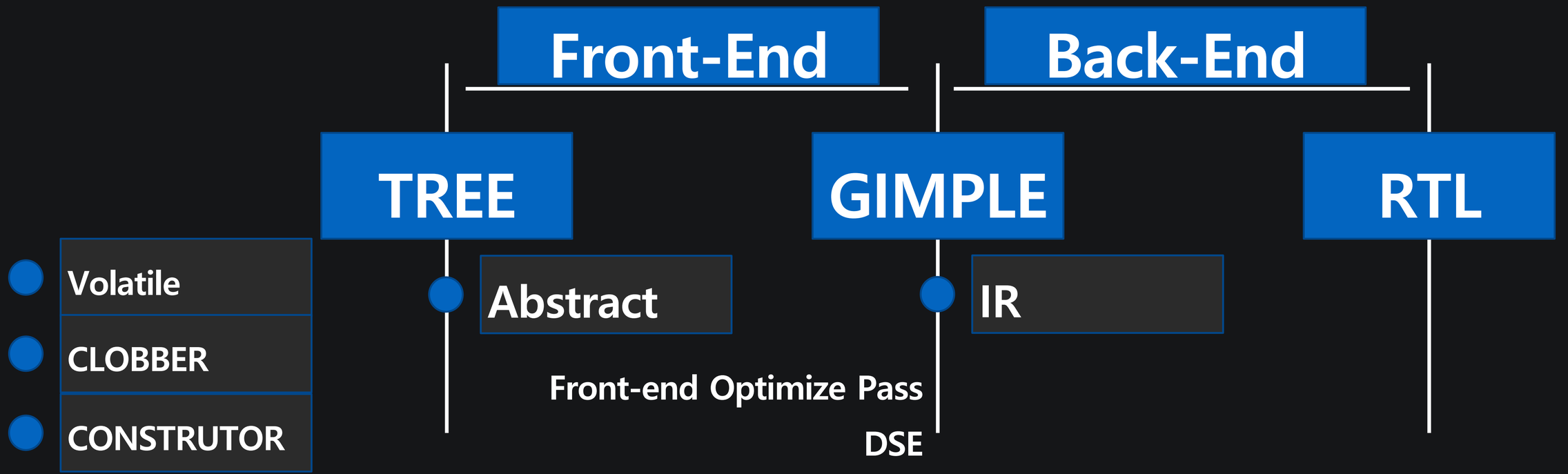




부족한 배경지식은 강제로 배우게 되니까요

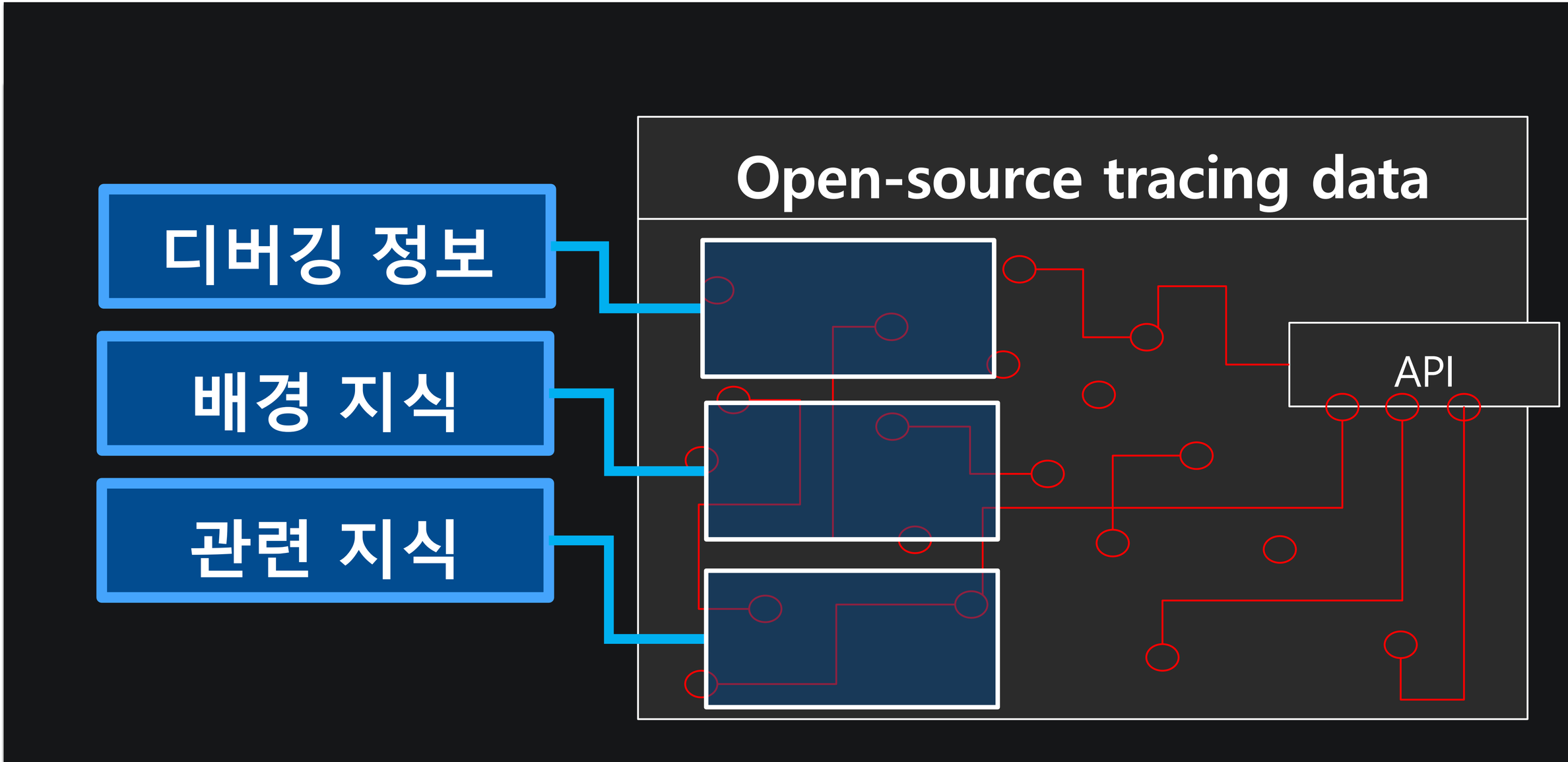
DEVIEW 2019

```
else if (DECL_DECLARED_INLINE_P(current_function_decl))
    inner->base.volatile_flag = true;
```



누군가 알려줬다면 훨씬 쉬웠을텐데…

DEVIEW 2019



DEVIEW
2019

Tracing

배경지식과 개발자 메시지 입히기

(feat, NodeJS/Pinpoint/v8)

NodeJS 개발자들이 남긴 메시지

DEVIEW
2019

```
function find_me_a() {
  console.log("hellworld");
}
```

```
$ nodejs --trace hellworld.js
1: ~+0(this=0x355aaee01521 <JSGlobal Object>) {
2: ~find_me_a+0(this=0x355aaee01521 <JSGlobal Object>) {
hellworld
2: } -> 0x1aa1629004d1 <undefined>
1: } -> 0x1aa1629004d1 <undefined>
```

또 다른 메시지 —print-bytecode

DEVIEW 2019

--trace 옵션 disable 시 생성되는 bytecode

```
[generated bytecode for function: find_me_a]
Parameter count 1
Register count 3
Frame size 24
   18 E> 0x1ce45b01eb1f @    5 : a5                StackCheck
   27 S> 0x1ce45b01eb20 @    6 : 13 00 00          LdaGlobal [0], [0]
         0x1ce45b01eb23 @    9 : 26 fa             Star r1
   35 E> 0x1ce45b01eb25 @   11 : 28 fa 01 02       LdaNamedProperty r1, [1], [2]
         0x1ce45b01eb29 @   15 : 26 fb             Star r0
         0x1ce45b01eb2b @   17 : 12 02             LdaConstant [2]
         0x1ce45b01eb2d @   19 : 26 f9             Star r2
   35 E> 0x1ce45b01eb2f @   21 : 59 fb fa f9 04    CallProperty1 r0, r1, r2, [4]
         0x1ce45b01eb34 @   26 : 0d                LdaUndefined
         0x1ce45b01eb35 @   27 : 26 fb             Star r0
   53 S> 0x1ce45b01eb3c @   34 : a9                Return
```

--trace 옵션 enable 시 생성되는 bytecode

```
[generated bytecode for function: find_me_a]
Parameter count 1
Register count 3
```

CallRuntime [TraceEnter], r0-r0

```
   18 E> 0x1ce45b01eb1f @    5 : a5                StackCheck
   27 S> 0x1ce45b01eb20 @    6 : 13 00 00          LdaGlobal [0], [0]
         0x1ce45b01eb23 @    9 : 26 fa             Star r1
   35 E> 0x1ce45b01eb25 @   11 : 28 fa 01 02       LdaNamedProperty r1, [1], [2]
         0x1ce45b01eb29 @   15 : 26 fb             Star r0
         0x1ce45b01eb2b @   17 : 12 02             LdaConstant [2]
         0x1ce45b01eb2d @   19 : 26 f9             Star r2
   35 E> 0x1ce45b01eb2f @   21 : 59 fb fa f9 04    CallProperty1 r0, r1, r2, [4]
         0x1ce45b01eb34 @   26 : 0d                LdaUndefined
```

CallRuntime [TraceExit], r0-r0

```
   53 S> 0x1ce45b01eb3c @   34 : a9                Return
```

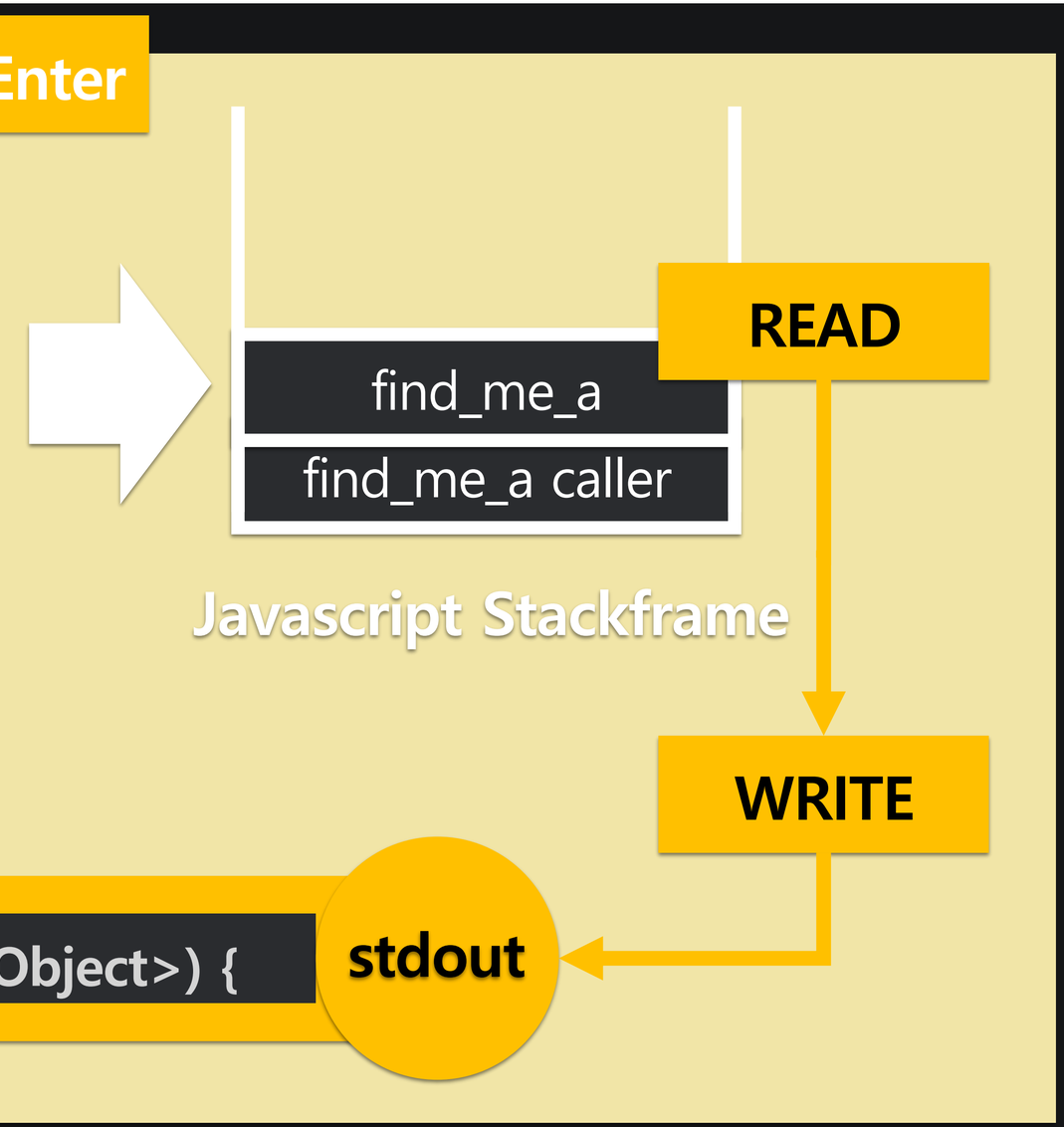
TraceEnter/Exit 가 추가되는 것을 확인


```
function find_me_a() {
    console.log("hellworld");
}
```

```
function find_me_a() {
    TraceEnter();
    console.log("hellworld");
    TraceExit();
}
```

TraceEnter 의도된 작업

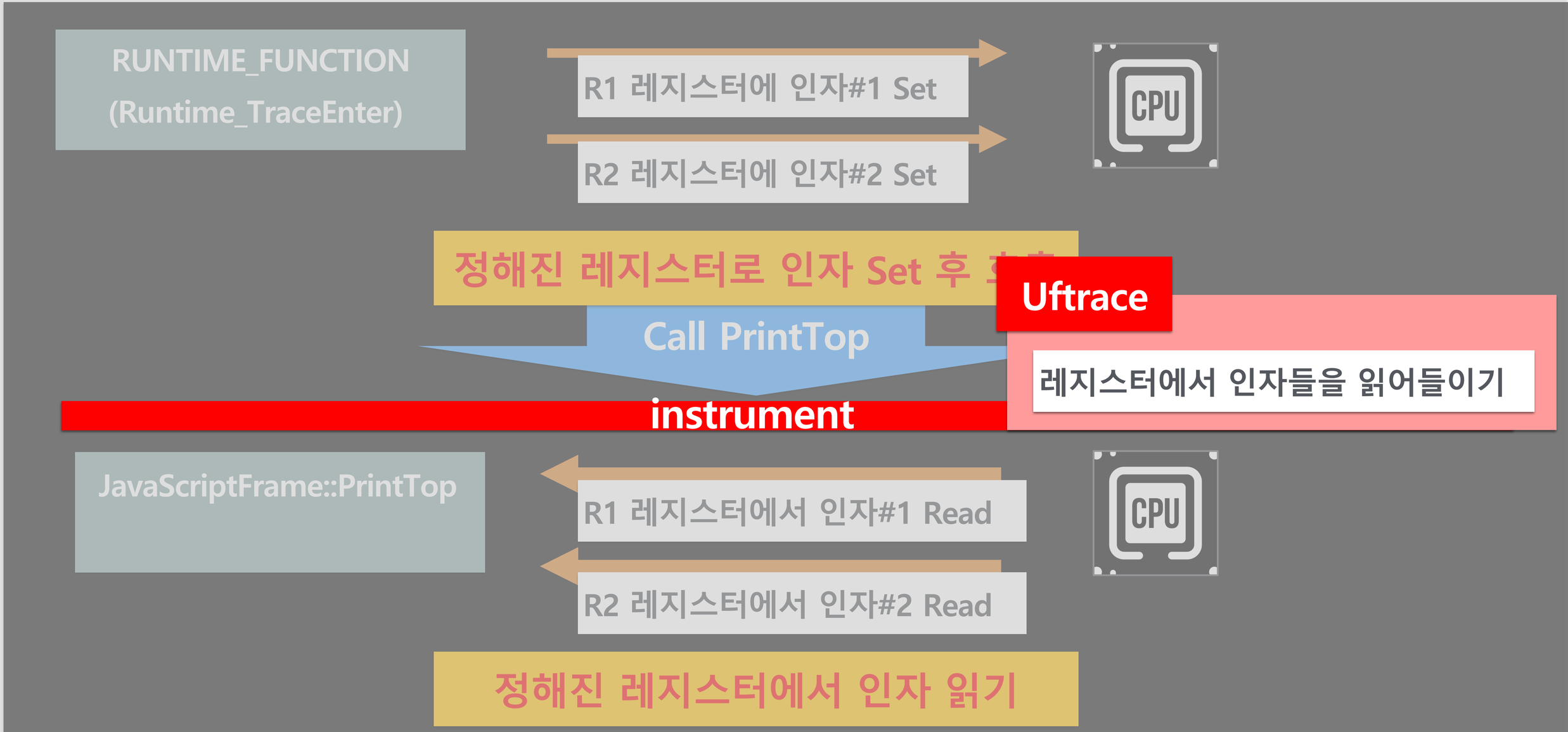

```
function find_me_a() {
    TraceEnter();
    console.log("hellworld");
    TraceExit();
}
```




```
2: ~find_me_a+0(this=0x355aaee01521 <JSGlobal Object>) {
```

Calling Convention – 인자 값 전달 방식

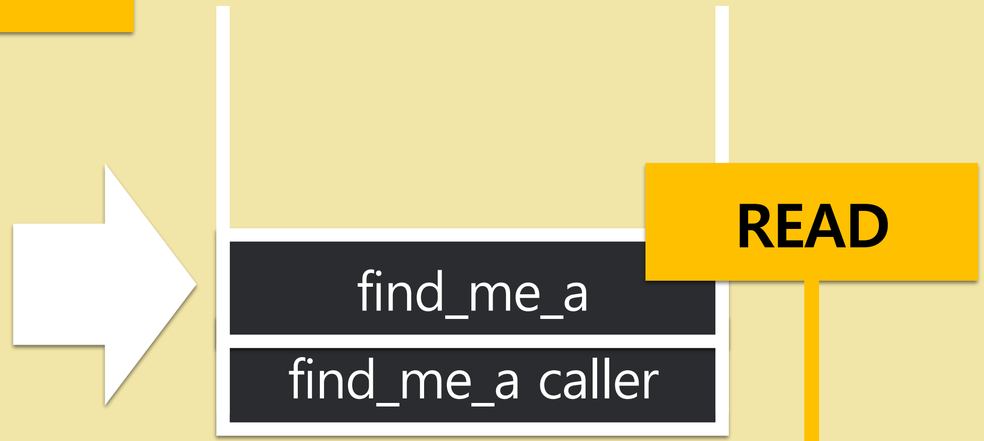
Uftrace에서 활용하는 방식



Uftrace로 출력 Stream 가로채기


```
function find_me_a() {
    TraceEnter();
    console.log("hellworld");
    TraceExit();
}
```

TraceEnter Enter



READ

Javascript Stackframe

WRITE

Uftrace

```
2: ~find_me_a+0(this=0x355aaee01521 <JSGlobal Object>) {
```

UFTRACE
Stream

Tracing Data에 기록한 결과

JavaScript 호출

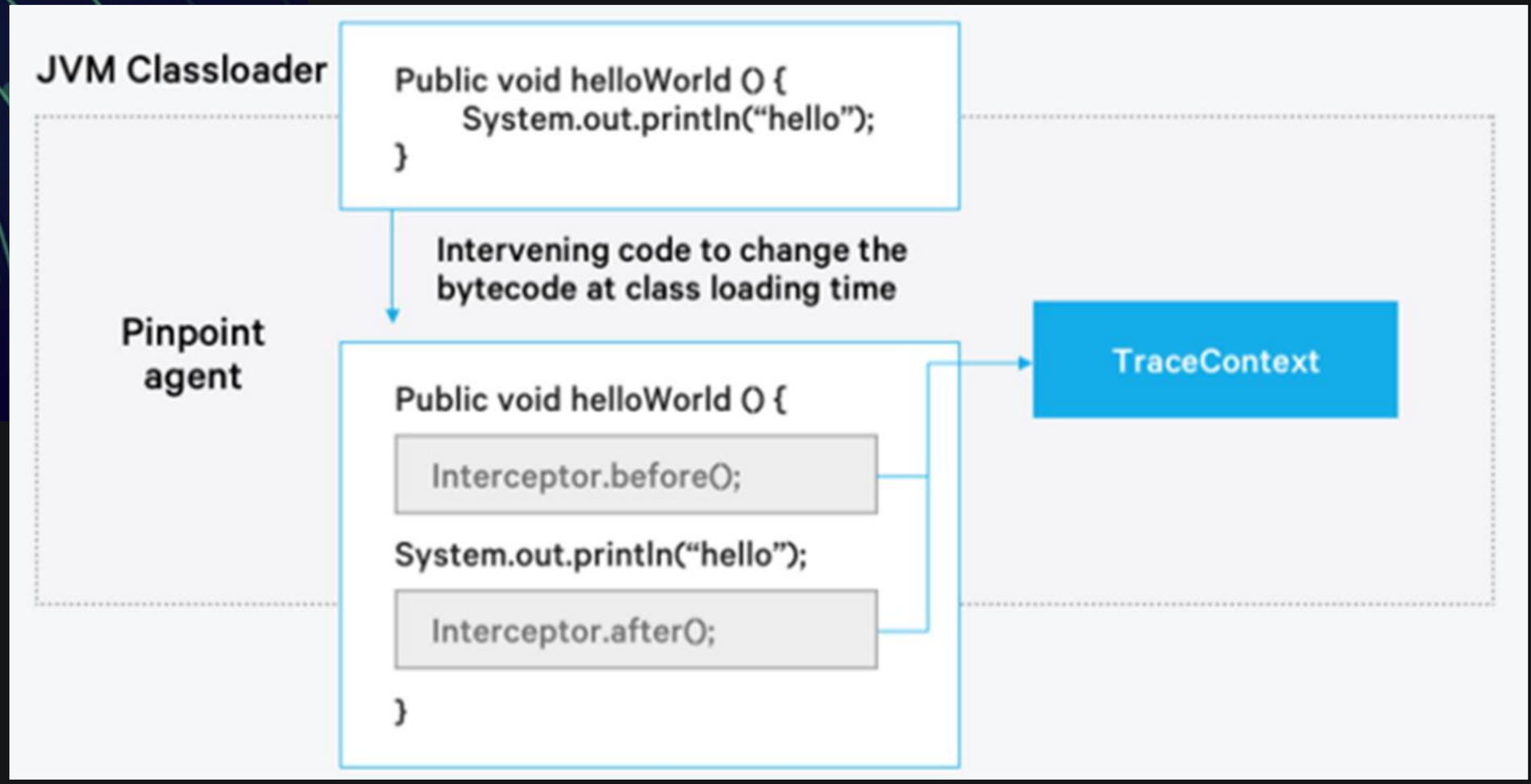
```
[ 30053] |  ~find_me_a+0(this=0x2c3bb9284de9 <JSGlobal Object>, 0x081d3615ead1 ...)
```

JavaScript 호출
처리 동작

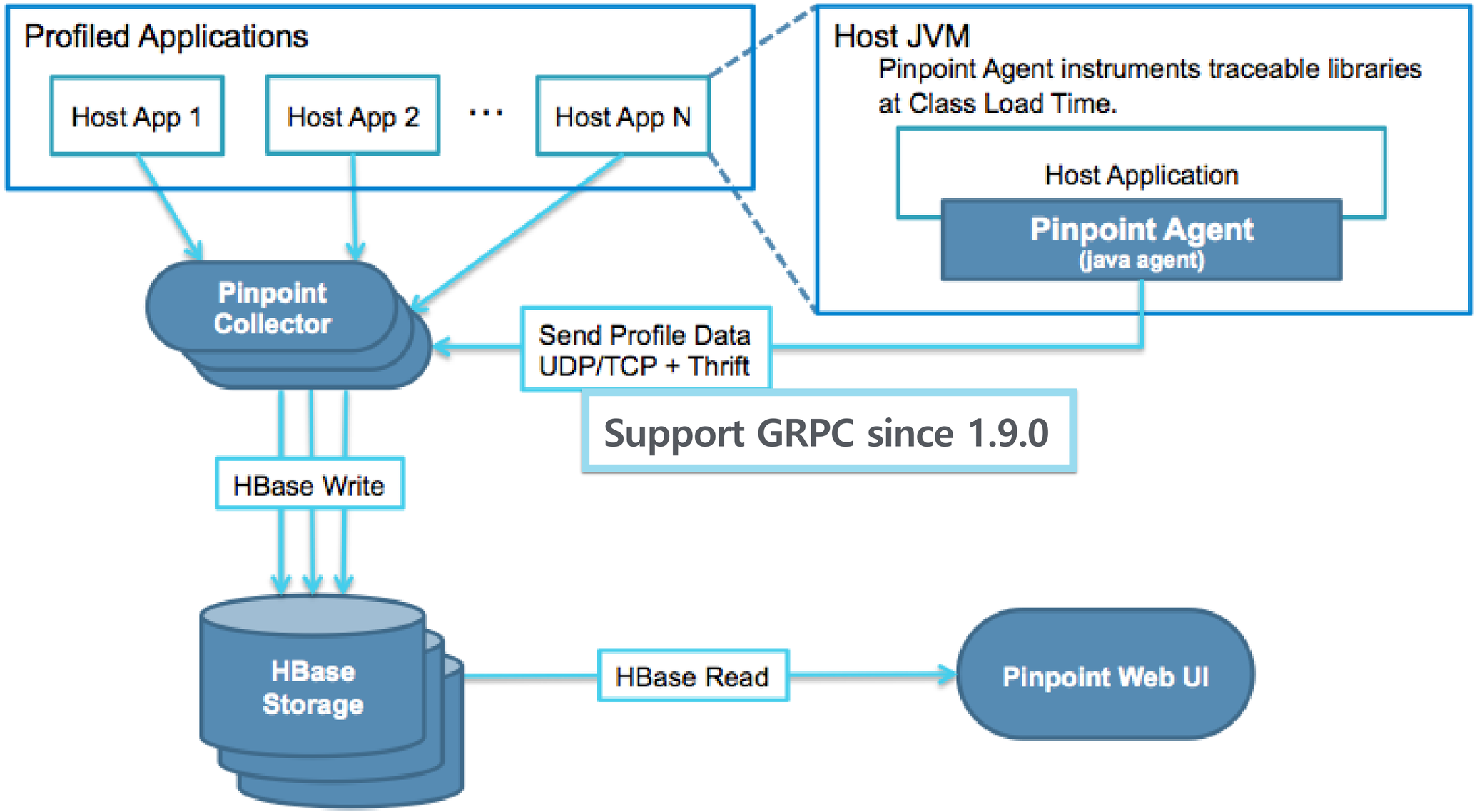
```
[ 30053] |      v8::internal::Runtime_CompileLazy() {
[ 30053] |        v8::internal::StackLimitCheck::JsHasOverflowed() {
0.145 us [ 30053] |          v8::internal::GetCurrentStackPosition();
1.453 us [ 30053] |        } /* v8::internal::StackLimitCheck::JsHasOverflowed */
[ 30053] |        v8::internal::Compiler::Compile() {
[ 30053] |          v8::internal::Compiler::Compile() {
```

Pinpoint 의 instrument

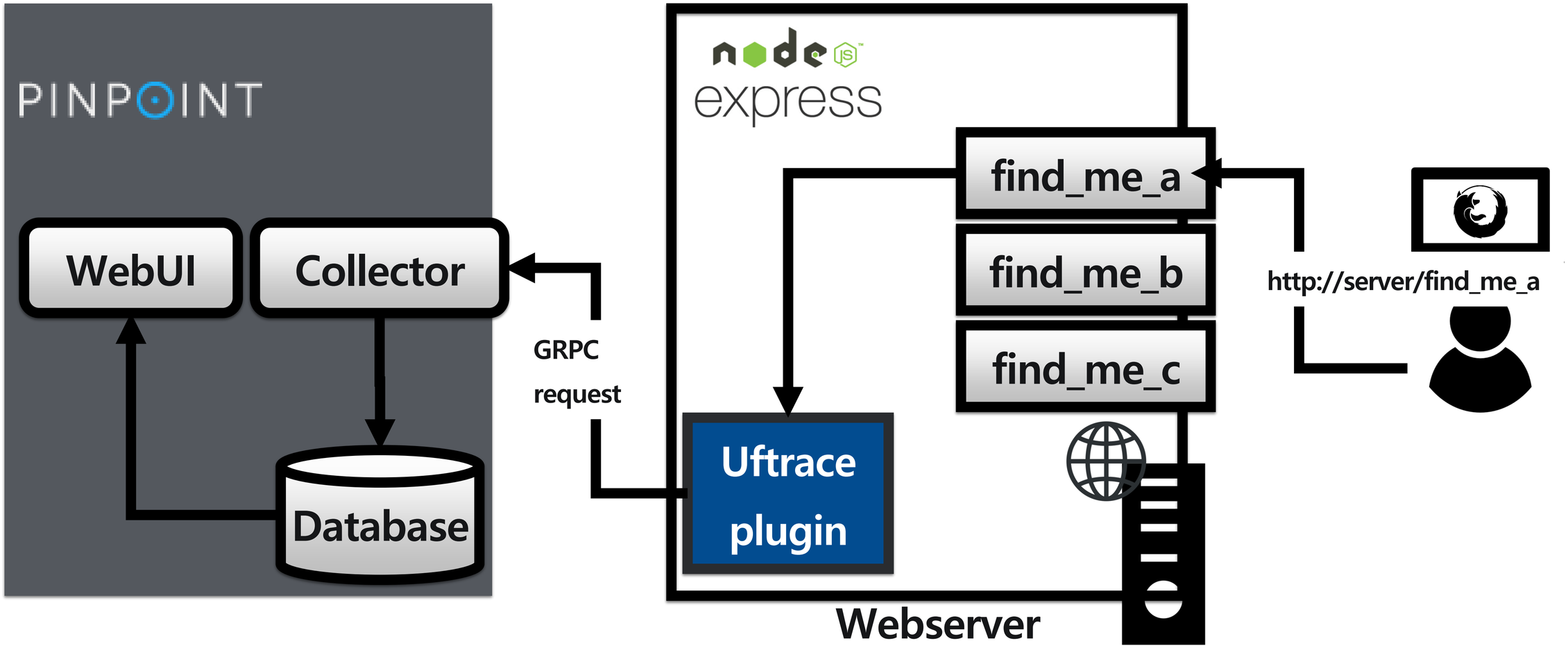
애플리케이션
성능 모니터링
Pinpoint



Pinpoint APM

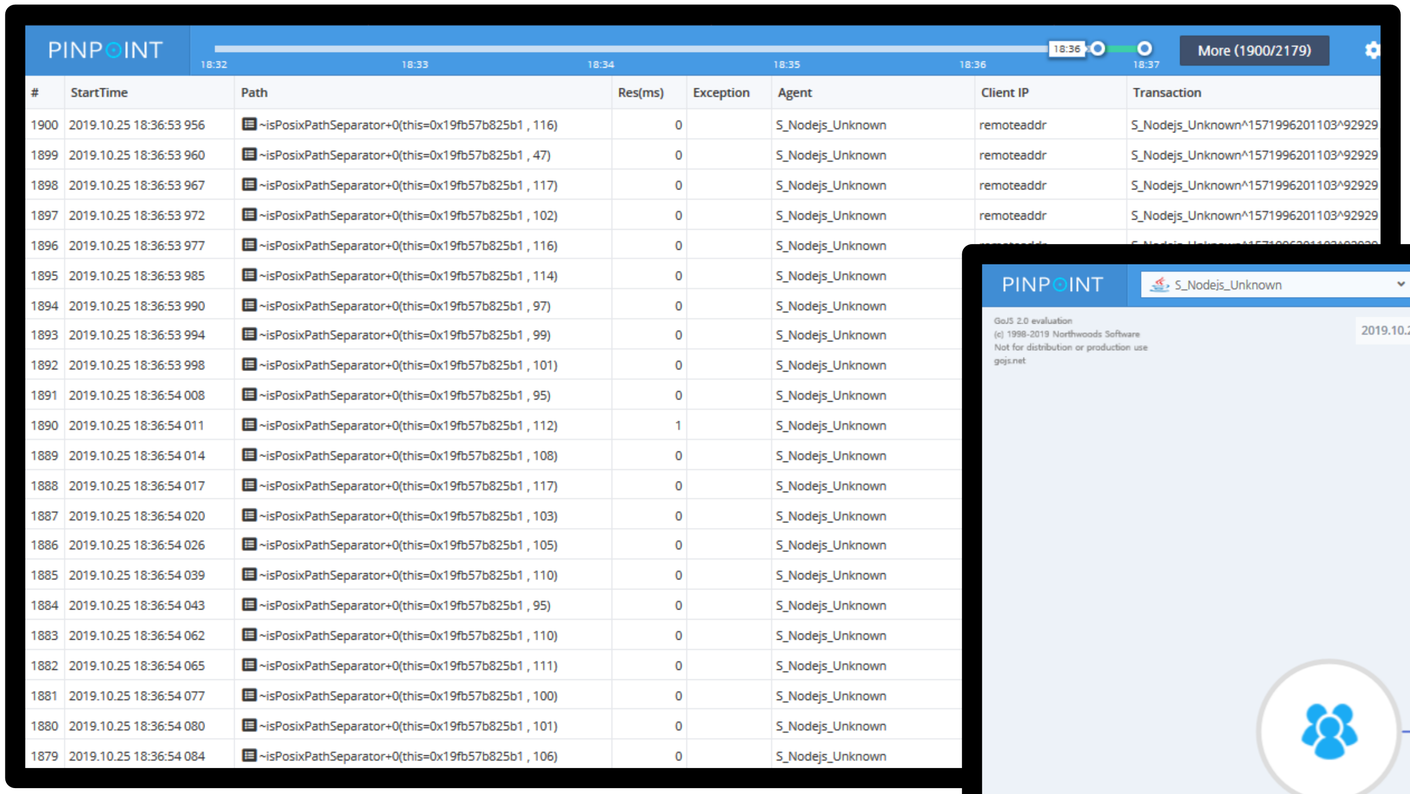


Simple NodeJS APM

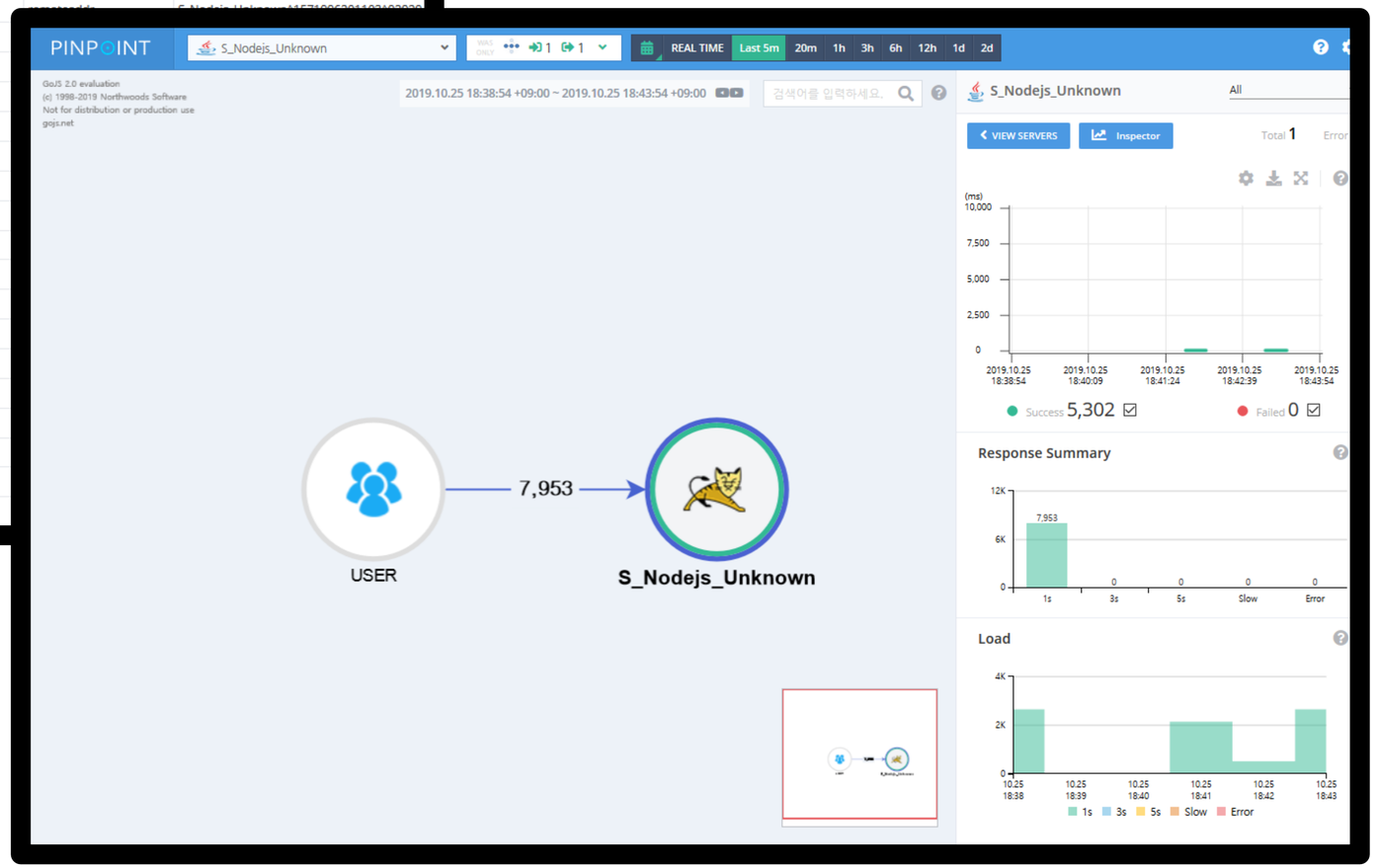


too much information

DEVIEW 2019



#	StartTime	Path	Res(ms)	Exception	Agent	Client IP	Transaction
1900	2019.10.25 18:36:53 956	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 116)	0		S_Nodejs_Unknown	remoteaddr	S_Nodejs_Unknown^1571996201103^92929
1899	2019.10.25 18:36:53 960	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 47)	0		S_Nodejs_Unknown	remoteaddr	S_Nodejs_Unknown^1571996201103^92929
1898	2019.10.25 18:36:53 967	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 117)	0		S_Nodejs_Unknown	remoteaddr	S_Nodejs_Unknown^1571996201103^92929
1897	2019.10.25 18:36:53 972	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 102)	0		S_Nodejs_Unknown	remoteaddr	S_Nodejs_Unknown^1571996201103^92929
1896	2019.10.25 18:36:53 977	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 116)	0		S_Nodejs_Unknown		
1895	2019.10.25 18:36:53 985	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 114)	0		S_Nodejs_Unknown		
1894	2019.10.25 18:36:53 990	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 97)	0		S_Nodejs_Unknown		
1893	2019.10.25 18:36:53 994	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 99)	0		S_Nodejs_Unknown		
1892	2019.10.25 18:36:53 998	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 101)	0		S_Nodejs_Unknown		
1891	2019.10.25 18:36:54 008	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 95)	0		S_Nodejs_Unknown		
1890	2019.10.25 18:36:54 011	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 112)	1		S_Nodejs_Unknown		
1889	2019.10.25 18:36:54 014	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 108)	0		S_Nodejs_Unknown		
1888	2019.10.25 18:36:54 017	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 117)	0		S_Nodejs_Unknown		
1887	2019.10.25 18:36:54 020	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 103)	0		S_Nodejs_Unknown		
1886	2019.10.25 18:36:54 026	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 105)	0		S_Nodejs_Unknown		
1885	2019.10.25 18:36:54 039	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 110)	0		S_Nodejs_Unknown		
1884	2019.10.25 18:36:54 043	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 95)	0		S_Nodejs_Unknown		
1883	2019.10.25 18:36:54 062	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 110)	0		S_Nodejs_Unknown		
1882	2019.10.25 18:36:54 065	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 111)	0		S_Nodejs_Unknown		
1881	2019.10.25 18:36:54 077	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 100)	0		S_Nodejs_Unknown		
1880	2019.10.25 18:36:54 080	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 101)	0		S_Nodejs_Unknown		
1879	2019.10.25 18:36:54 084	~isPosixPathSeparator+0(this=0x19fb57b825b1 , 106)	0		S_Nodejs_Unknown		



Javascript 함수의 Parsing


```
function find_me_a() {
  return find_me_b();
}
function find_me_b() {
  return find_me_c();
}
function find_me_c() {
  return 1;
}
find_me_a();
```

Javascript 코드의 실행 과정

DEVIEW 2019

START v8::internal::_GLOBAL__N_1::CompileToplevel()

Enter Func 0

Call find_me_a()

v8::internal::Execution::Call

v8::internal::Execution::(anonymous

v8::internal::Runtime_CompileLazy

v8::internal:__RT_imple_Runtime_Co

v8::internal::Compiler::Compile

Enter find_me_a

Call find_me_b()

v8::internal::Execution::Call

v8::internal::Execution::(anonymous

v8::internal::Runtime_CompileLazy

v8::internal:__RT_imple_Runtime_Co

v8::internal::Compiler::Compile

Enter find_me_b

Call find_me_c()

v8::internal::Execution::Call

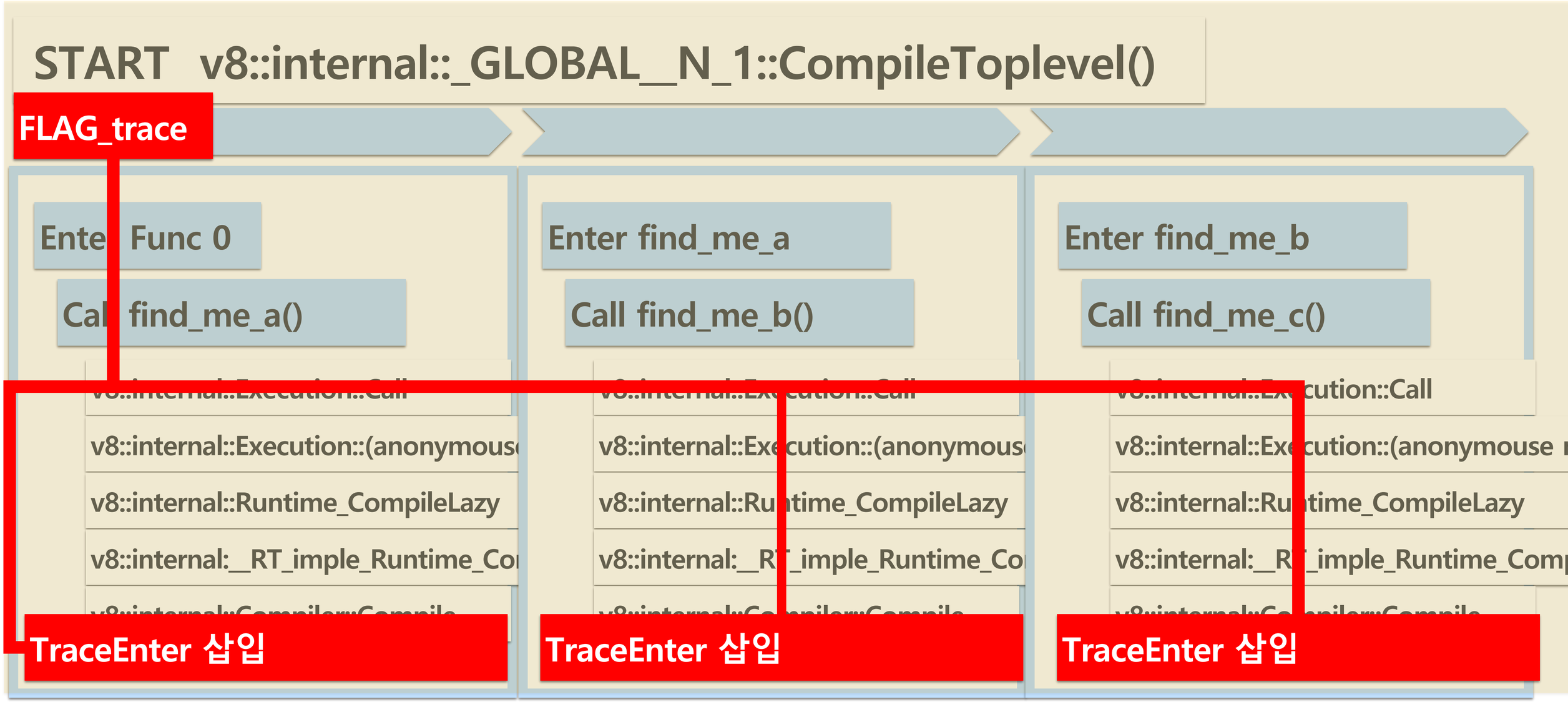
v8::internal::Execution::(anonymouse na

v8::internal::Runtime_CompileLazy

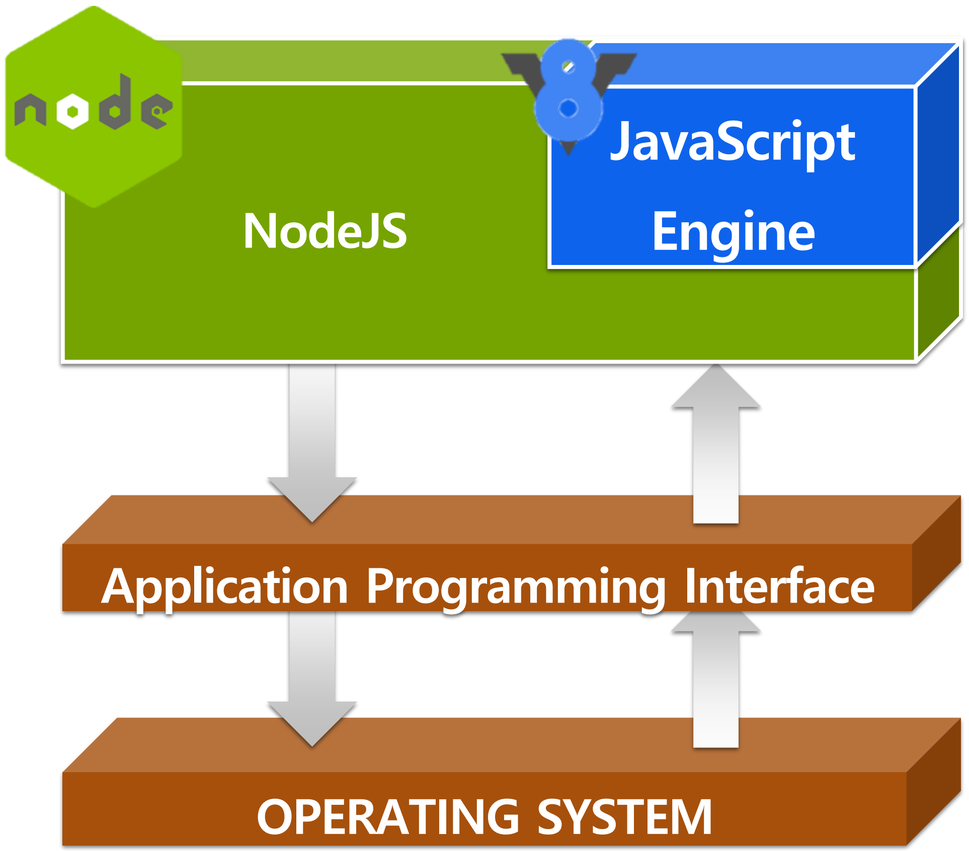
v8::internal:__RT_imple_Runtime_Compi

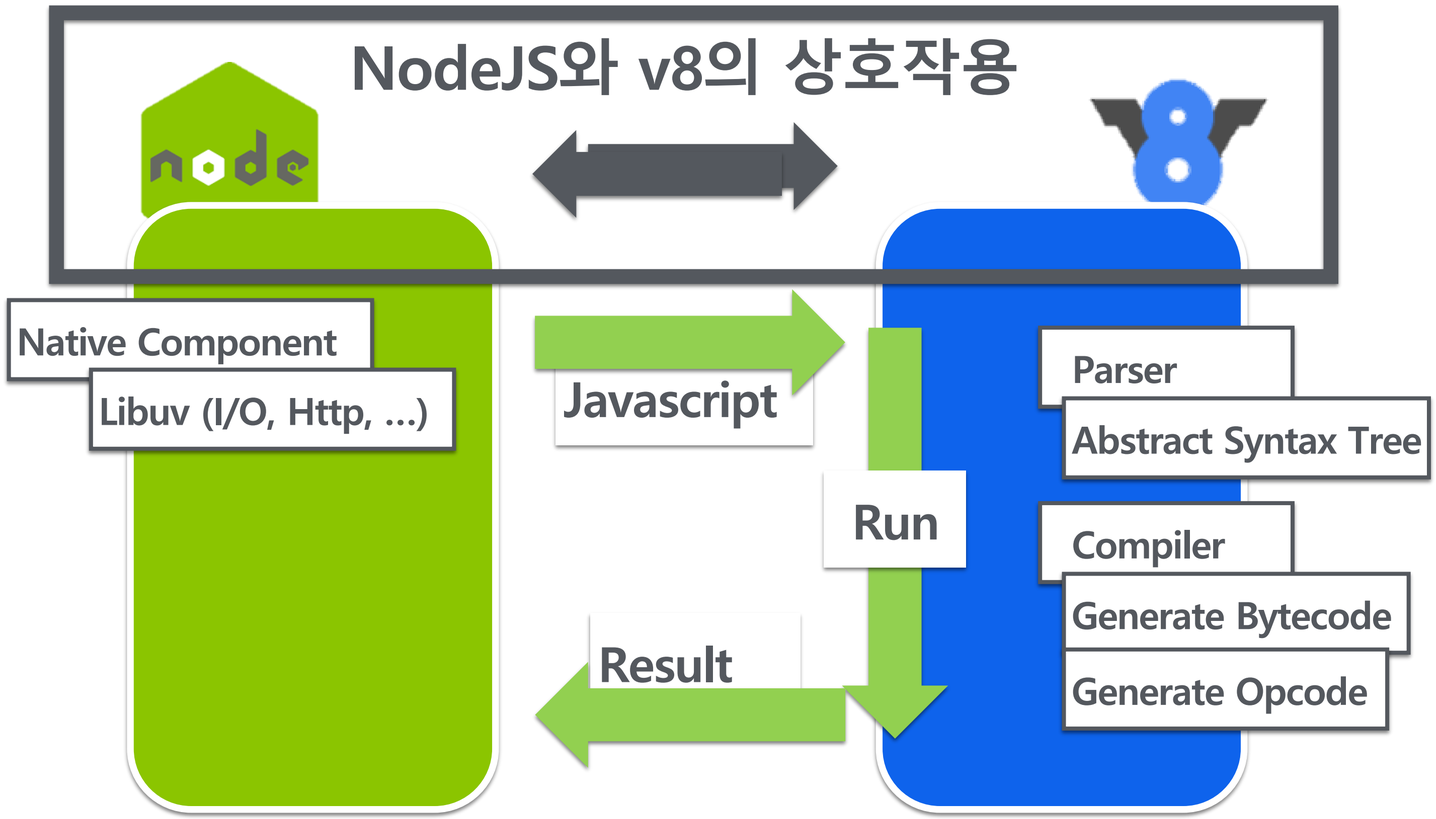
v8::internal::Compiler::Compile

TraceEnter 삽입 지점



NodeJS에 대한 배경지식

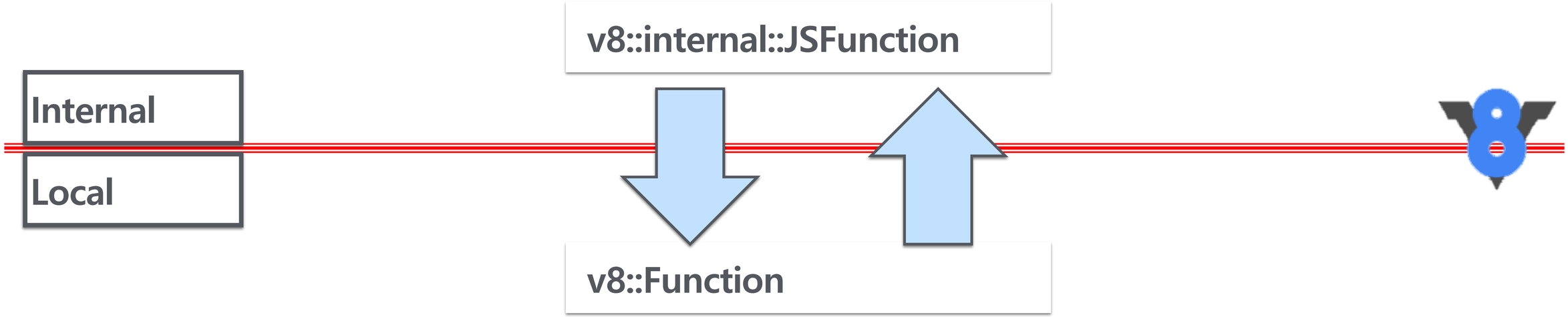





```
v8::Local<v8::Script> script = v8::Script::Compile(context, source);
v8::Local<v8::Value> result = script->Run(context);
```

Javascript Compile 함수

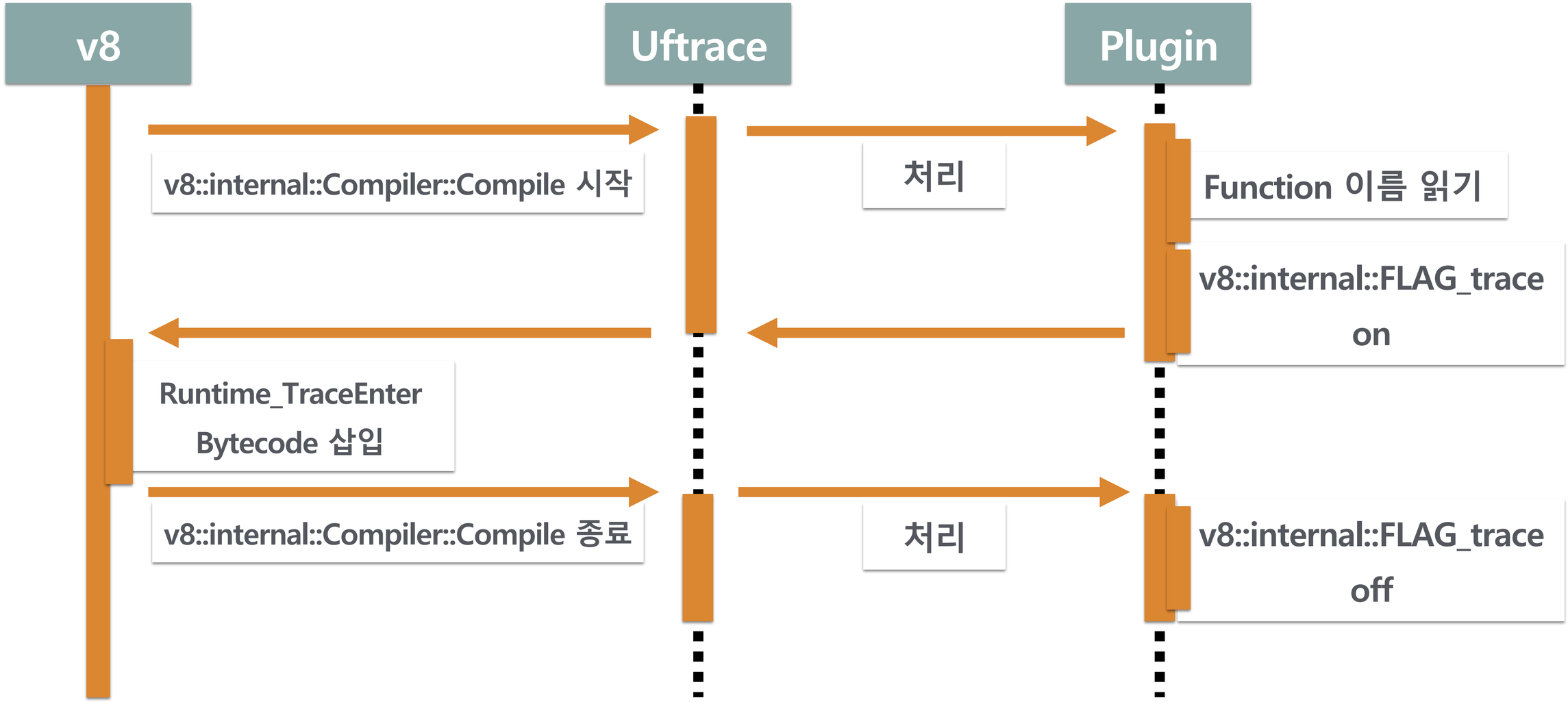

```
v8::internal::Compiler::Compile(Handle<JSFunction> function, ClearExceptionFlag, IsCompiledScope*)
```




```
v8::Function *func = reinterpret_cast<v8::Function*>(ARG1(regs));
v8::String::Utf8Value name(func->CreationContext()->GetIsolate(), func->GetDebugName());
```

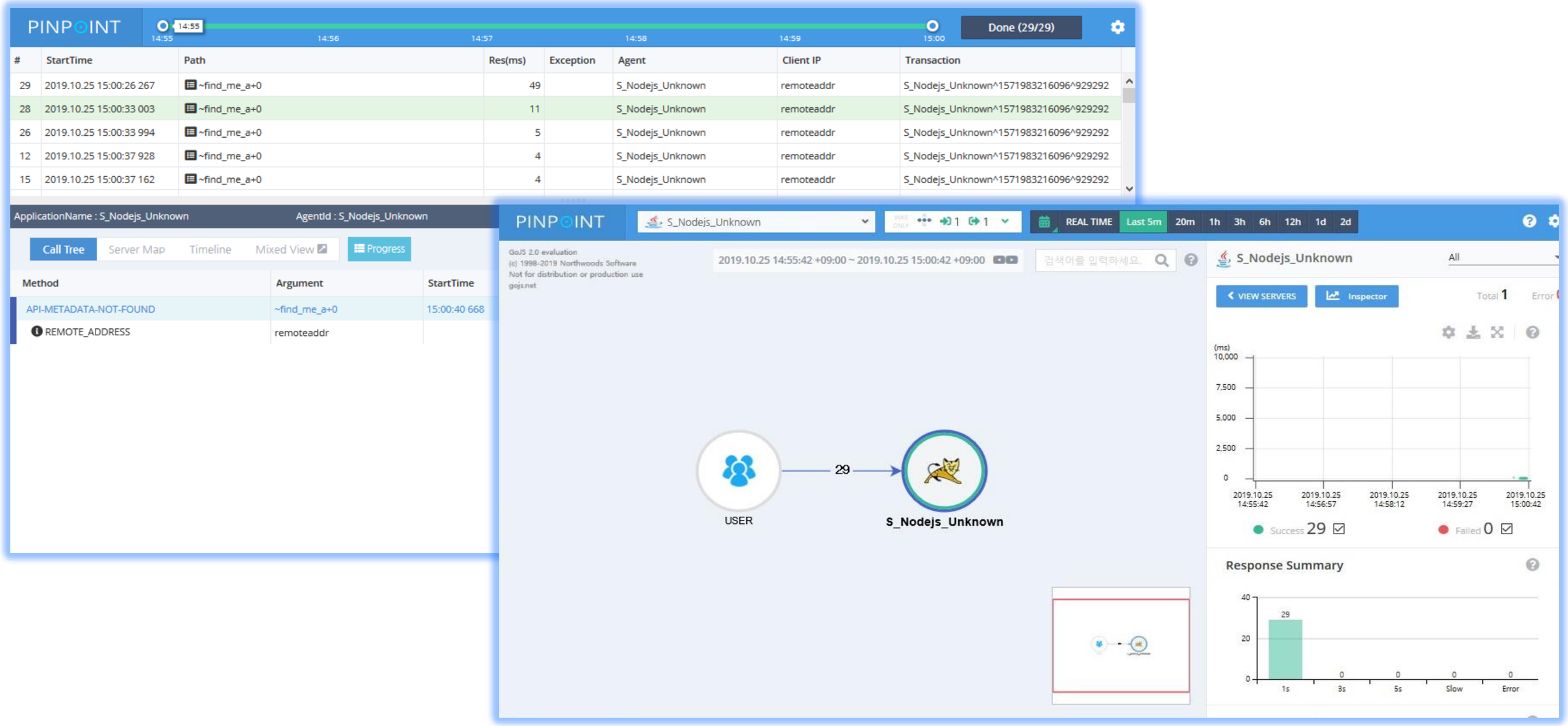
배경지식 입히기

특정 JavaScript Function만 기록되게 만드는 로직



필요 함수만을 선별 기록

DEVIEW 2019



DEVIEW
2019

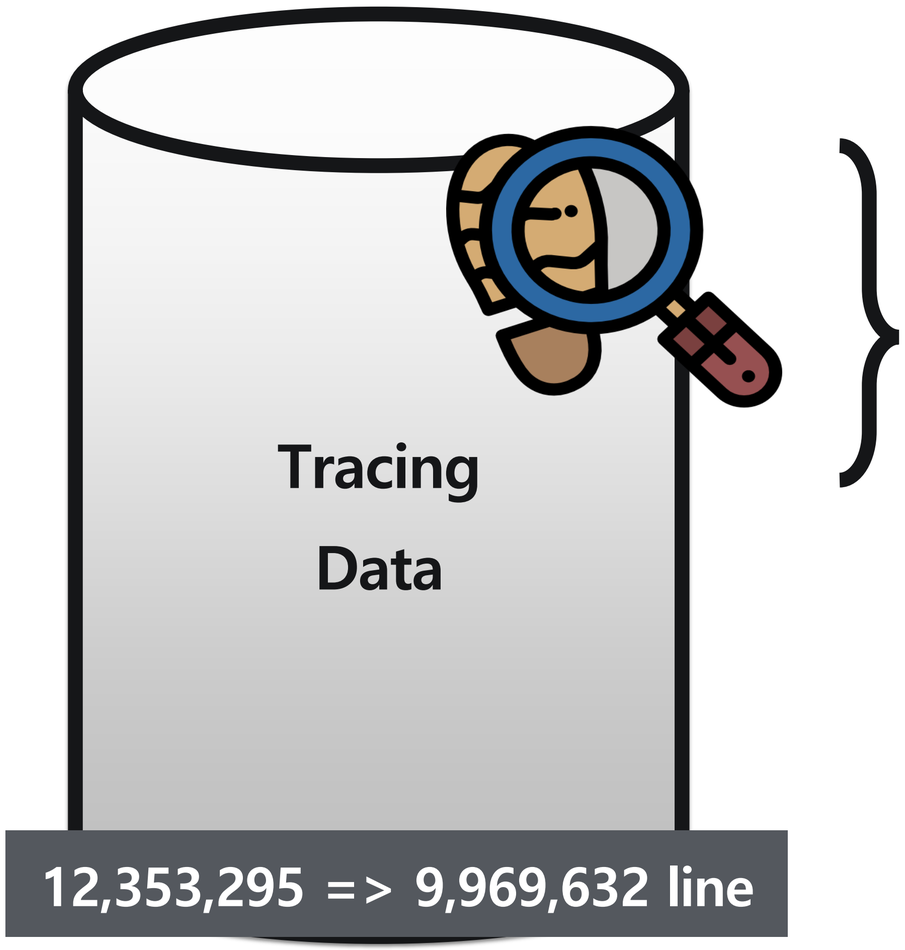
눈발자국

뒷사람의 이정표가 된다


KBS 광주(뉴)
'눈 발자국'으로 20대 강도 검거

Plugin 제작을 통한 소기 목적 달성

DEVIEW 2019



Javascript 함수를 컴파일하는 CallTree

v8::internal::Runtime_CompileLazy

v8::internal:__RT_imple_Runtime_CompileLazy

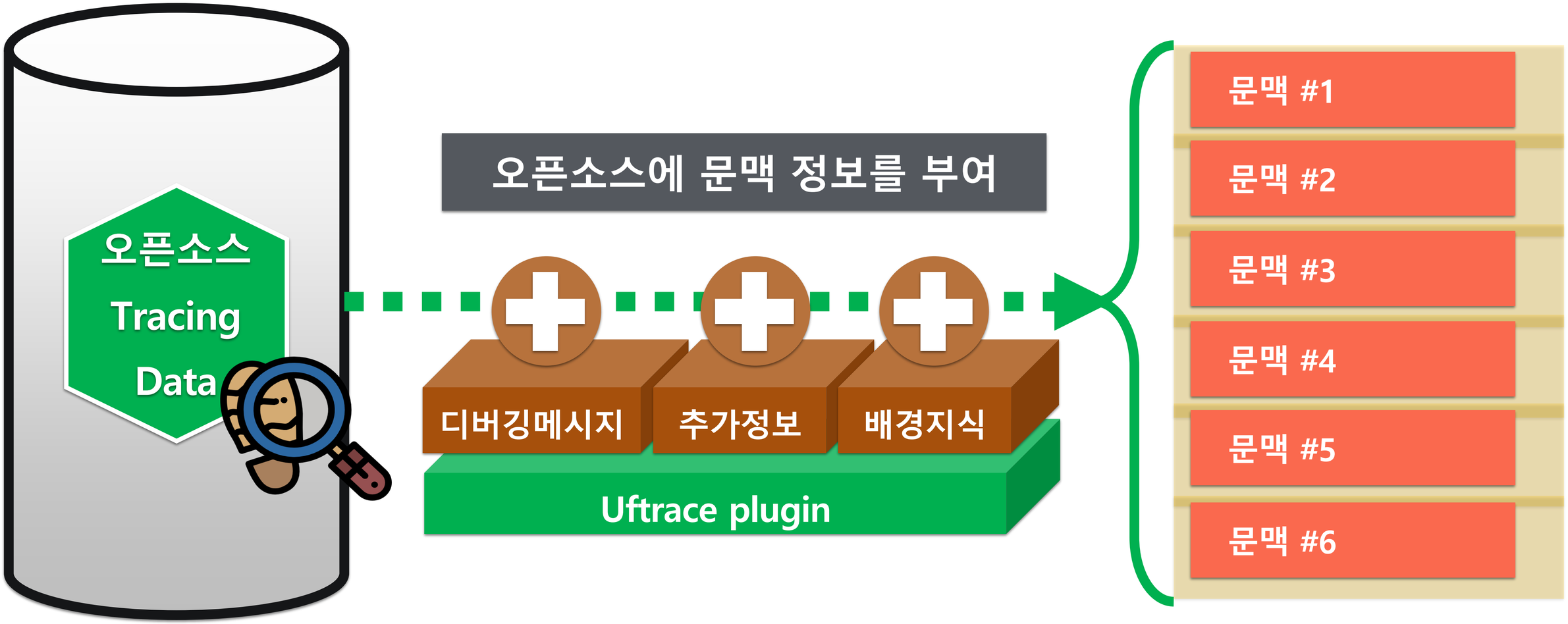
v8::internal::Compiler::Compile

치환

Javascript 함수가 컴파일된다는 문맥

<Function name> Lazy Compile

오픈소스 접근성 높히기


오픈소스
Tracing
Data
오픈소스에 문맥 정보를 부여
디버깅메시지
추가정보
배경지식
Uftrace plugin
문맥 #1
문맥 #2
문맥 #3
문맥 #4
문맥 #5
문맥 #6

DEVIEW
2019

Q & A

DEVIEW
2019

Thank You