나만 몰랐던

W3C HTML5 Conference 2019

2019.10.11

김형규

대덕소프트웨어마이스터고등학교

목차

- 발표자 소개
- 주제를 선정하게 된 계기
- V8이란?
- V8의 역사
  - 기존 V8 엔진들 (2015 ~ 2016)
  - 현재 V8 (2017 ~)
- 현재 V8의 동작
  - Ignition 인터프리팅 과정.

발표자 소개

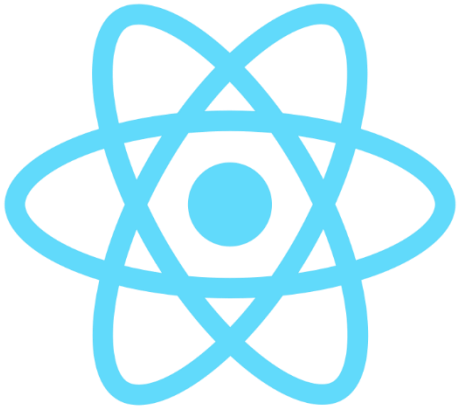
김형규 / Clickim

Github: https://github.com/khg0712

Blog: https://velog.io/@k7120792

Email: k7120792@gmail.com

소속: 대덕소프트웨어마이스터고등학교 / MIDAS IT



주제 선정 이유

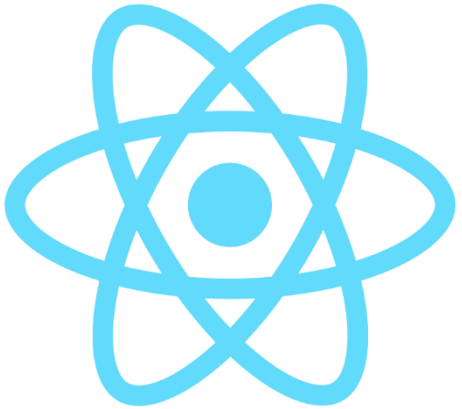
주제 선정 이유

Is Javascript a compiled language?

Fabio Russo May 16 '18 · 2 min read

#discuss #javascript #compile

주제 선정 이유



주제 선정 이유

V8이란?

V8이란?

C++로 작성된 구글의 오픈소스
ECMAScript, WebAssembly 엔진.

V8이란?

즉, JS와 WASM을 실행할 수 있는 환경

V8이 하는 일

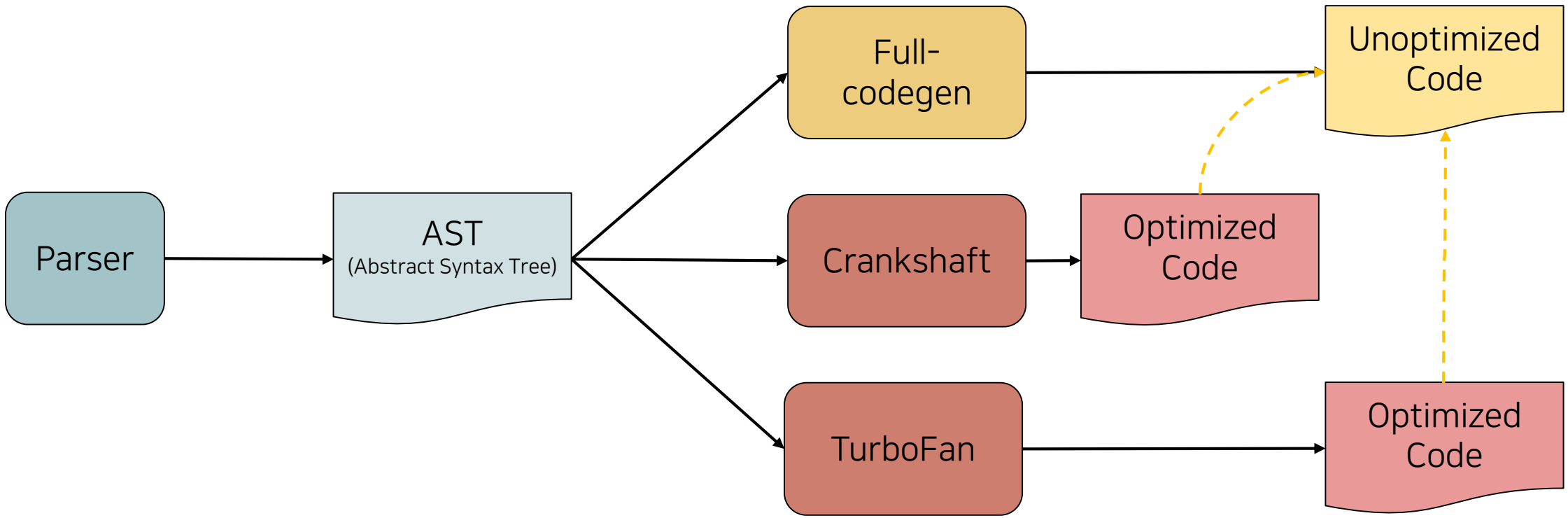
V8이 하는 일

1. JS, WASM 코드 컴파일, 실행.
2. 콜 스택 처리 (함수 실행).
3. 메모리 할당
4. 가비지 컬렉션

기존 V8의 구성 요소 (2015)

- FullCode Generator
- Crankshaft
- TurboFan

기존 V8의 실행 과정 (2015)

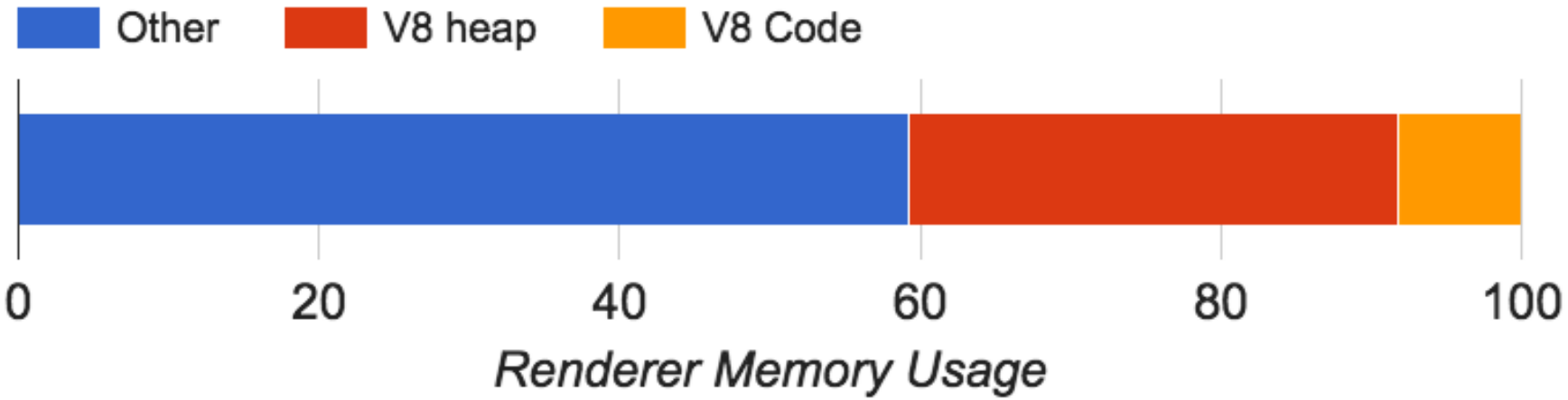


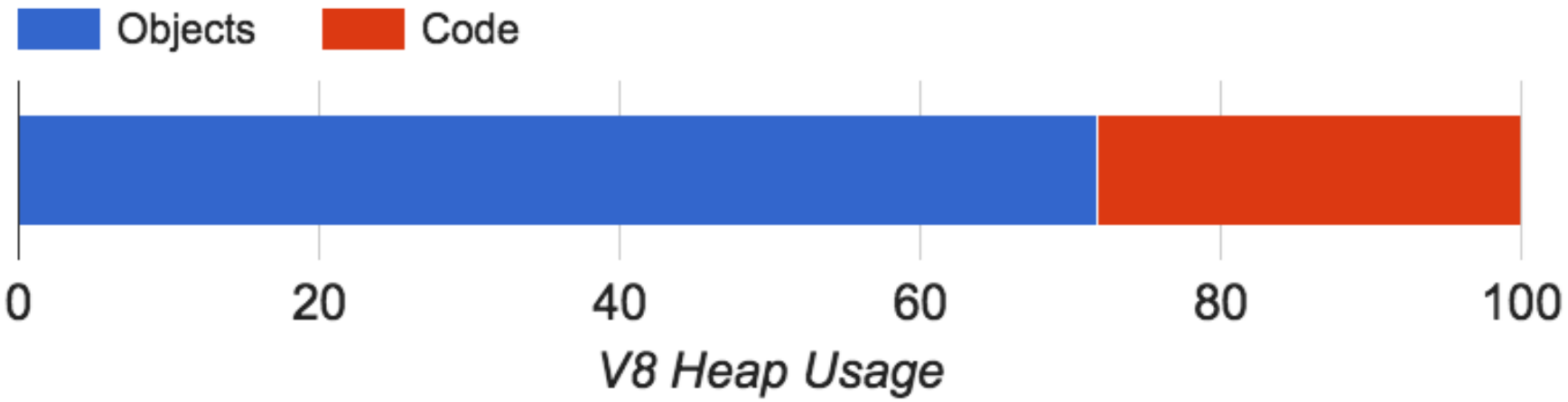
기존 구조의 문제

- 메모리
- 속도
- 복잡성

기존 구조의 문제

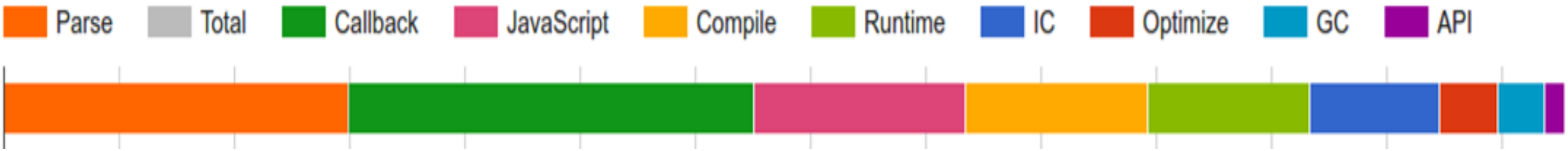
- 메모리
- 속도
- 복잡성





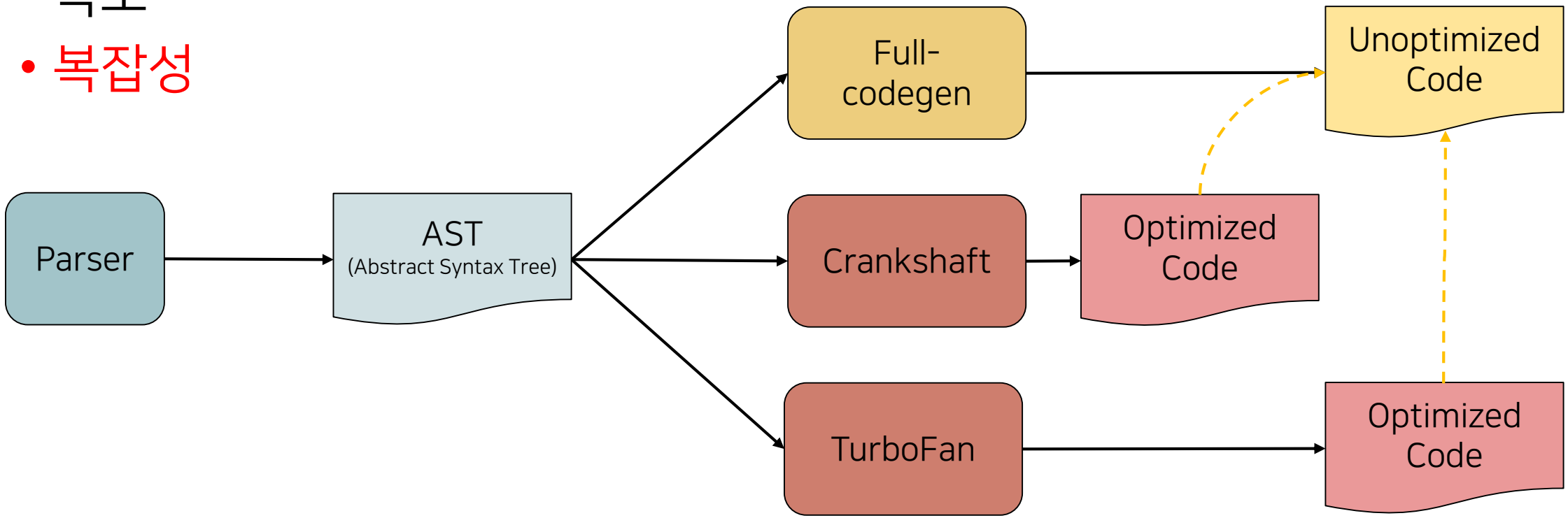
기존 구조의 문제

- 메모리
- 속도
- 복잡성



기존 구조의 문제

- 메모리
- 속도
- 복잡성

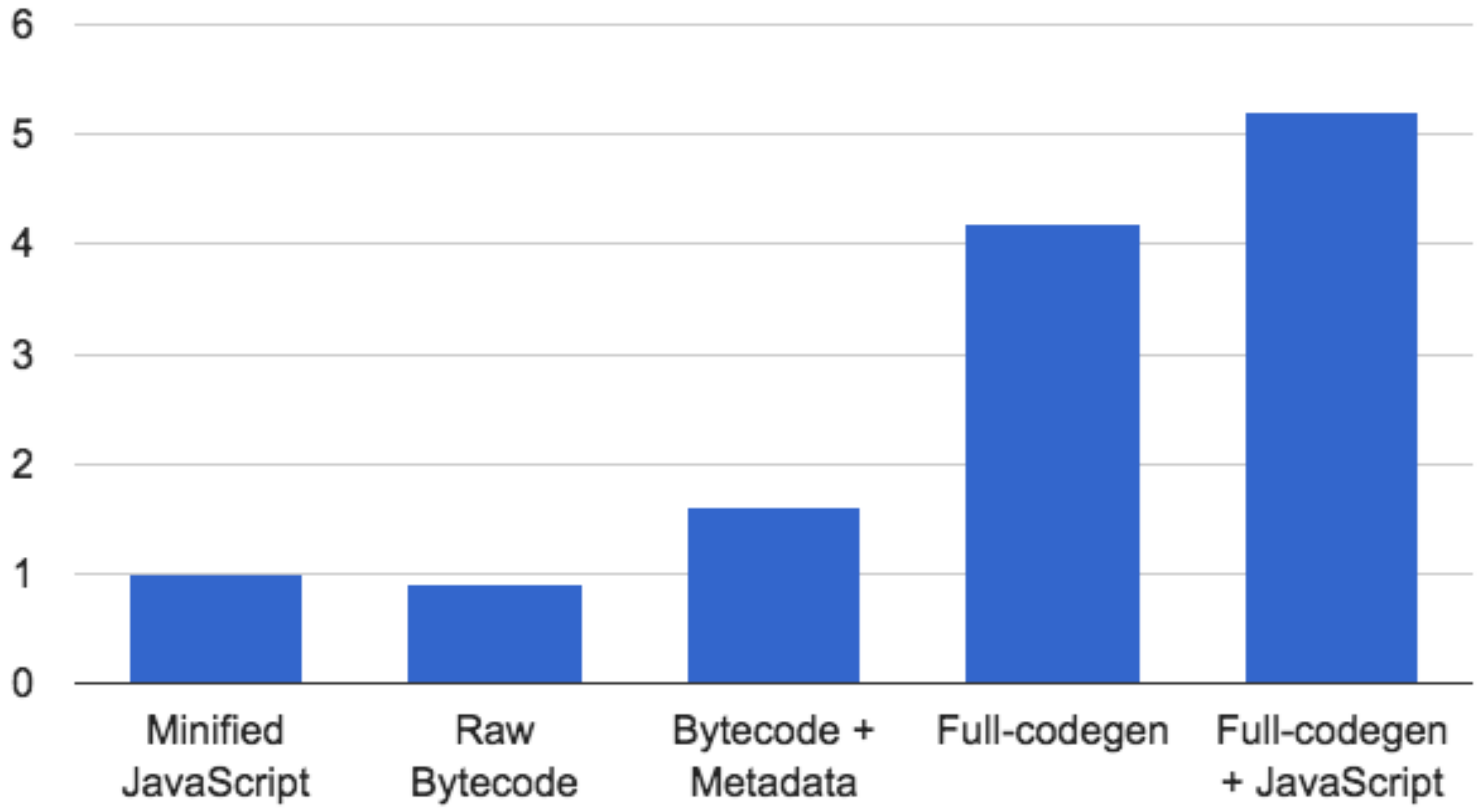


인터프리팅으로의 전환

- 메모리 사용량 감소
  - 기계어가 아닌 바이트 코드로의 변환
- 파싱 오버헤드 감소
- 컴파일러 복잡도 감소
  - 바이트 코드를 source of truth로!!!

인터프리팅으로의 전환

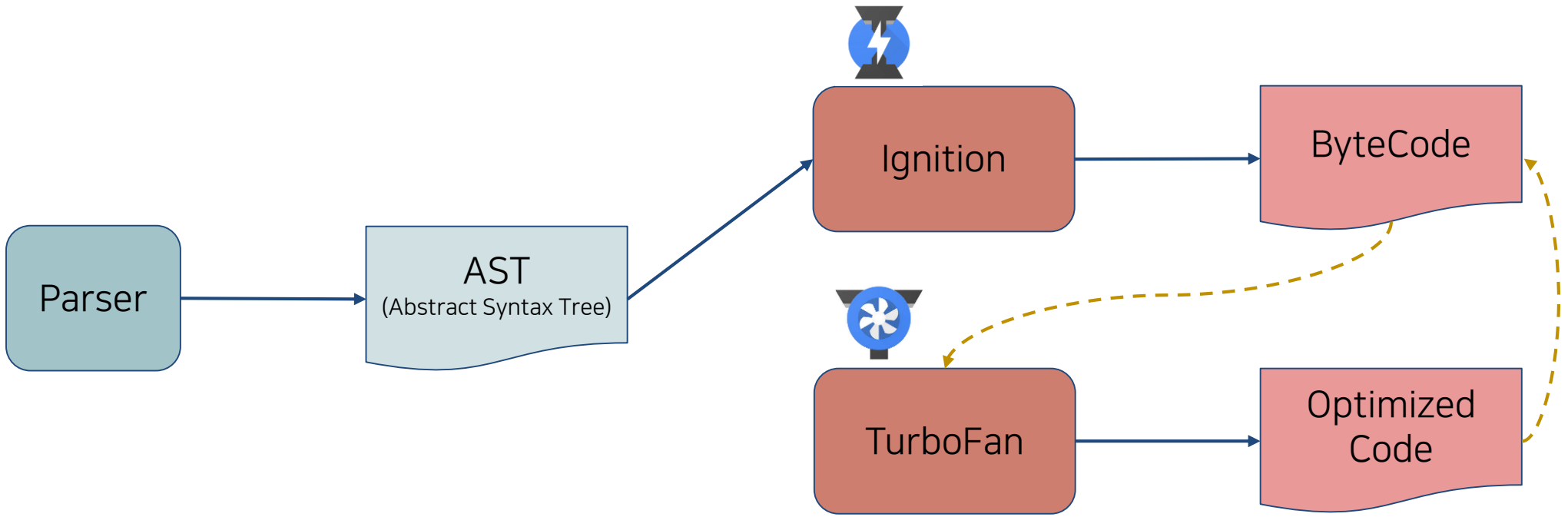
- 메모리 사용량 감소
  - 기계어가 아닌
- 파싱 오버헤드 감
- 컴파일러 복잡도
  - 바이트 코드를



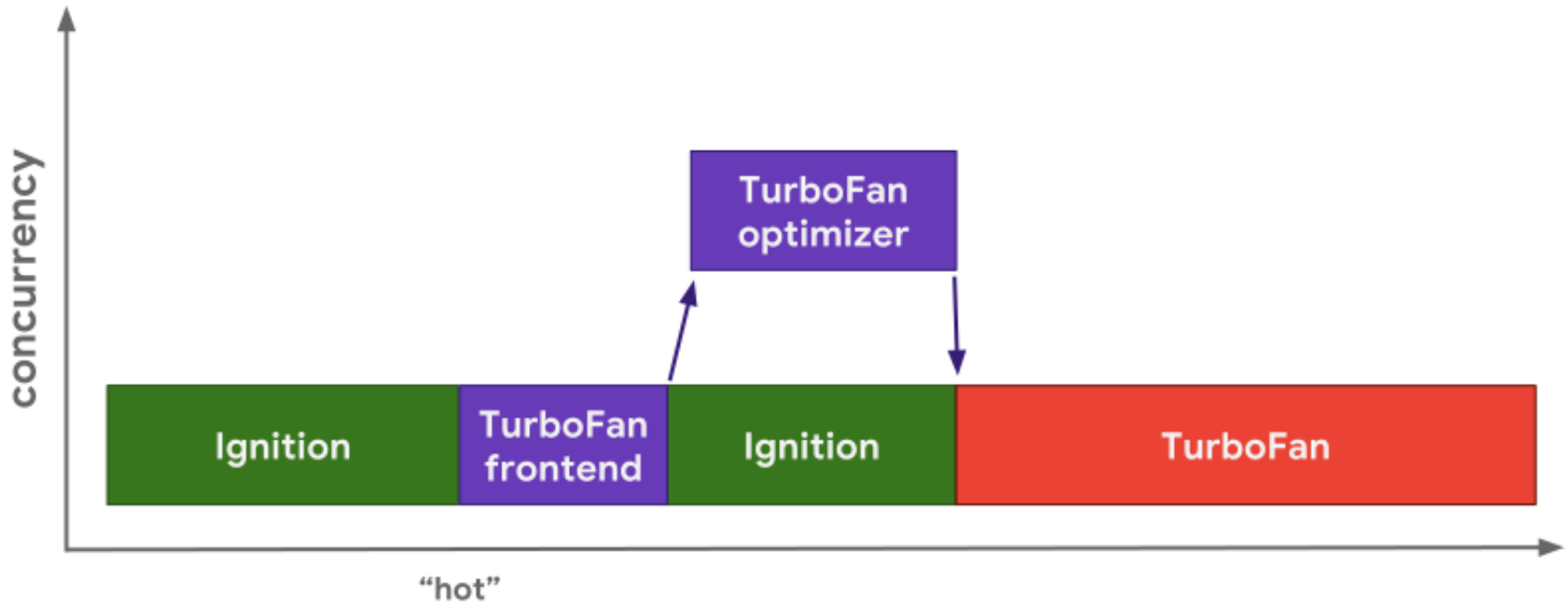
현재 V8의 구성 요소

- Ignition
- TurboFan

현재 V8의 실행 과정


Parser
AST
(Abstract Syntax Tree)
Ignition
ByteCode
TurboFan
Optimized
Code

현재 V8의 실행 과정


concurrency
TurboFan optimizer
Ignition
TurboFan frontend
Ignition
TurboFan
"hot"

코드를 어떻게 최적화할까?

- SSA(Static Single Assignment form)을 활용한 최적화
  - Loop-invariant code motion
  - Linear-scan register allocation
  - Inlining

Ignition의 작동 과정

```
function f(a, b) {
    let result = a + b;
    if (a > b) {
        result = a - b;
    }
    return result;
}


f(4, 10);
```

```
00 : a1          StackCheck
01 : 25 02       Ldar a1
03 : 32 03 00    Add a0, [0]
06 : 26 fb       Star r0
08 : 25 02       Ldar a1
10 : 67 03 01    TestGreaterThan a0, [1]
13 : 95 09       JumpIfFalse [9] (@ 22)
15 : 25 02       Ldar a1
17 : 33 03 02    Sub a0, [2]
20 : 26 fb       Star r0
22 : 25 fb       Ldar r0
24 : a5          Return
```

Ignition의 작동 과정

f(4, 10);

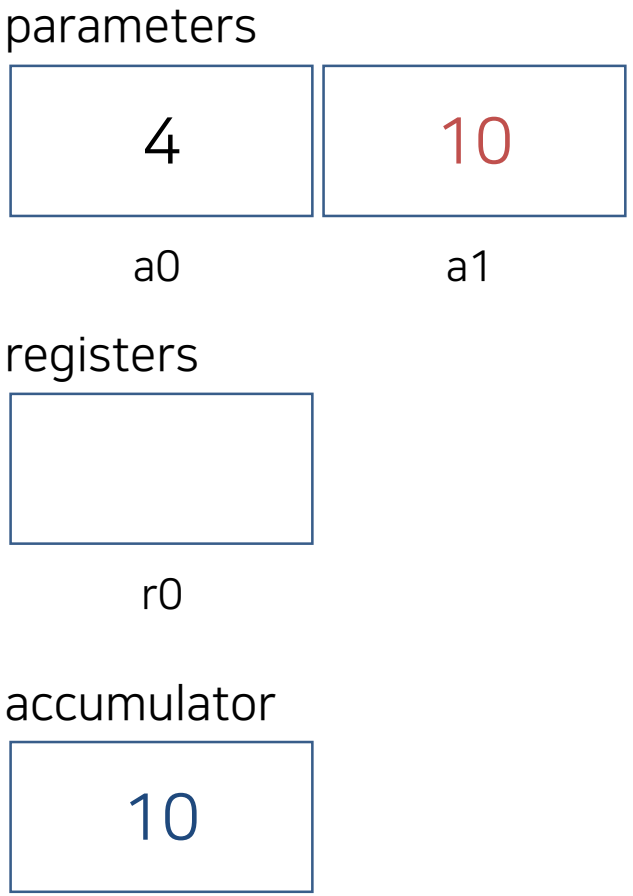
```
00 : a1          StackCheck
01 : 25 02       Ldar a1
03 : 32 03 00    Add a0, [0]
06 : 26 fb       Star r0
08 : 25 02       Ldar a1
10 : 67 03 01    TestGreaterThan a0, [1]
13 : 95 09       JumpIfFalse [9] (@ 22)
15 : 25 02       Ldar a1
17 : 33 03 02    Sub a0, [2]
20 : 26 fb       Star r0
22 : 25 fb       Ldar r0
24 : a5          Return
```

버퍼 오버플로우를 막기 위한
스택 가드 체크

Ignition의 작동 과정

f(4, 10);

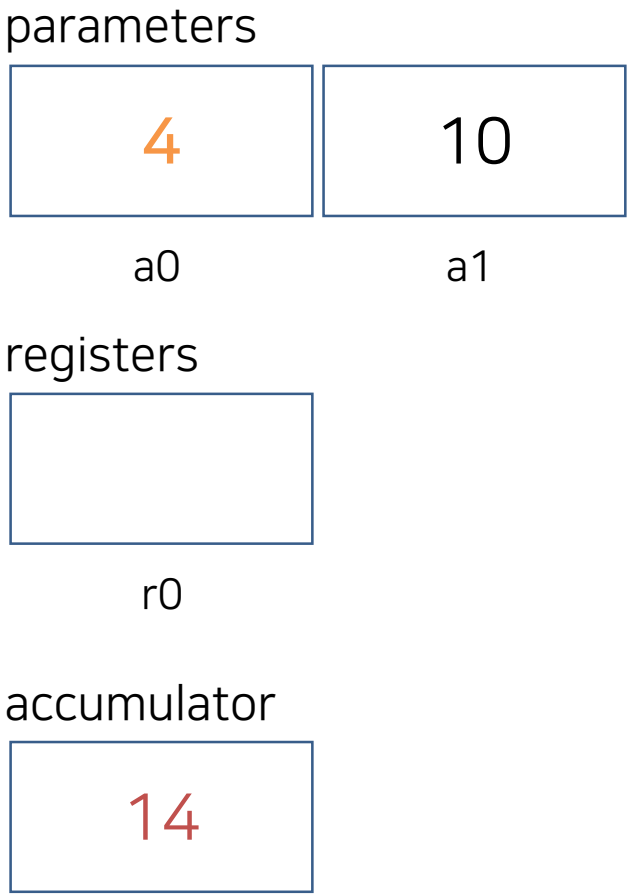
```
00 : a1          StackCheck
01 : 25 02       Ldar a1
03 : 32 03 00    Add a0, [0]
06 : 26 fb       Star r0
08 : 25 02       Ldar a1
10 : 67 03 01    TestGreaterThan a0, [1]
13 : 95 09       JumpIfFalse [9] (@ 22)
15 : 25 02       Ldar a1
17 : 33 03 02    Sub a0, [2]
20 : 26 fb       Star r0
22 : 25 fb       Ldar r0
24 : a5          Return
```



Ignition의 작동 과정

f(4, 10);

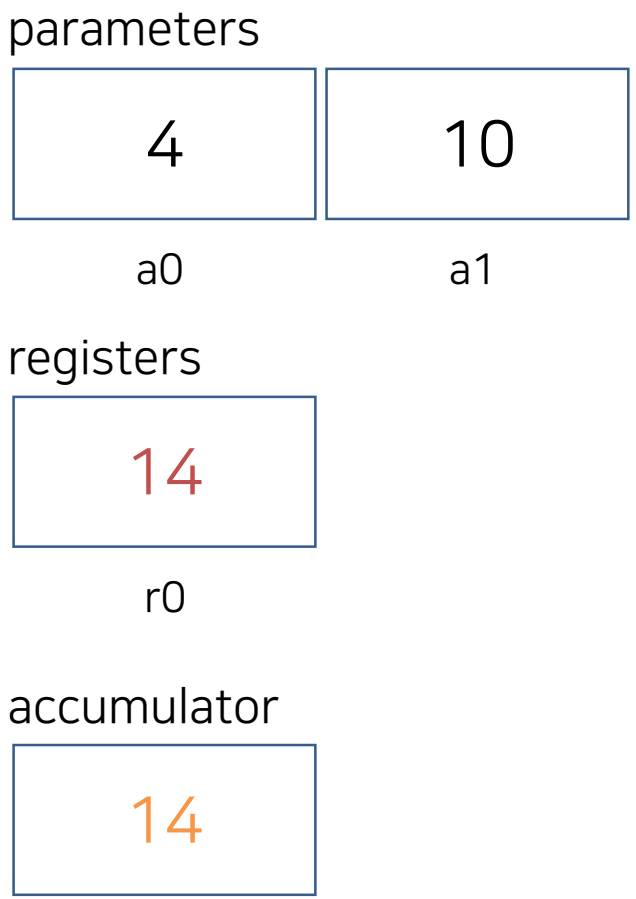
```
00 : a1          StackCheck
01 : 25 02       Ldar a1
03 : 32 03 00    Add a0, [0]
06 : 26 fb       Star r0
08 : 25 02       Ldar a1
10 : 67 03 01    TestGreaterThan a0, [1]
13 : 95 09       JumpIfFalse [9] (@ 22)
15 : 25 02       Ldar a1
17 : 33 03 02    Sub a0, [2]
20 : 26 fb       Star r0
22 : 25 fb       Ldar r0
24 : a5          Return
```



Ignition의 작동 과정

f(4, 10);

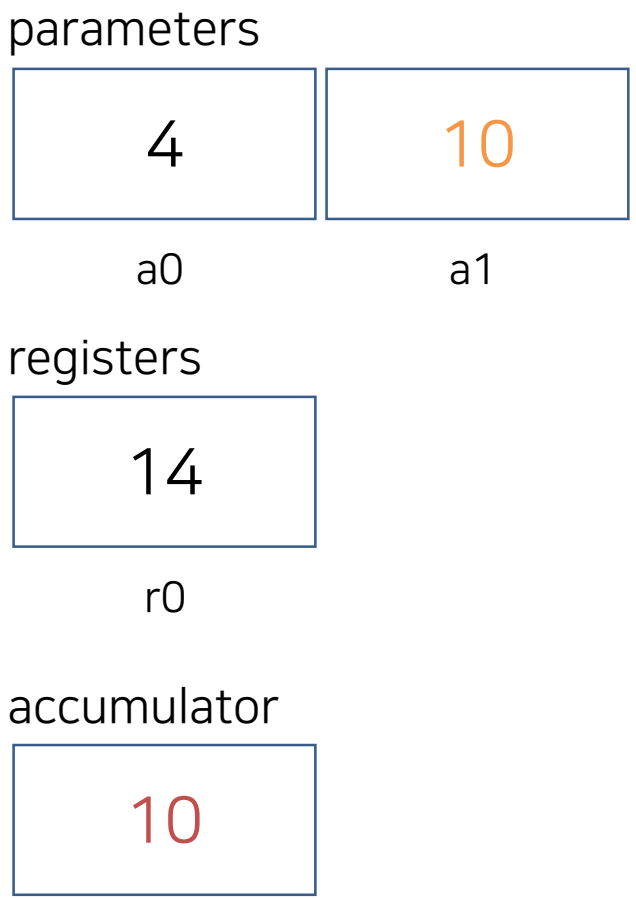
```
00 : a1          StackCheck
01 : 25 02       Ldar a1
03 : 32 03 00    Add a0, [0]
06 : 26 fb       Star r0
08 : 25 02       Ldar a1
10 : 67 03 01    TestGreaterThan a0, [1]
13 : 95 09       JumpIfFalse [9] (@ 22)
15 : 25 02       Ldar a1
17 : 33 03 02    Sub a0, [2]
20 : 26 fb       Star r0
22 : 25 fb       Ldar r0
24 : a5          Return
```



Ignition의 작동 과정

f(4, 10);

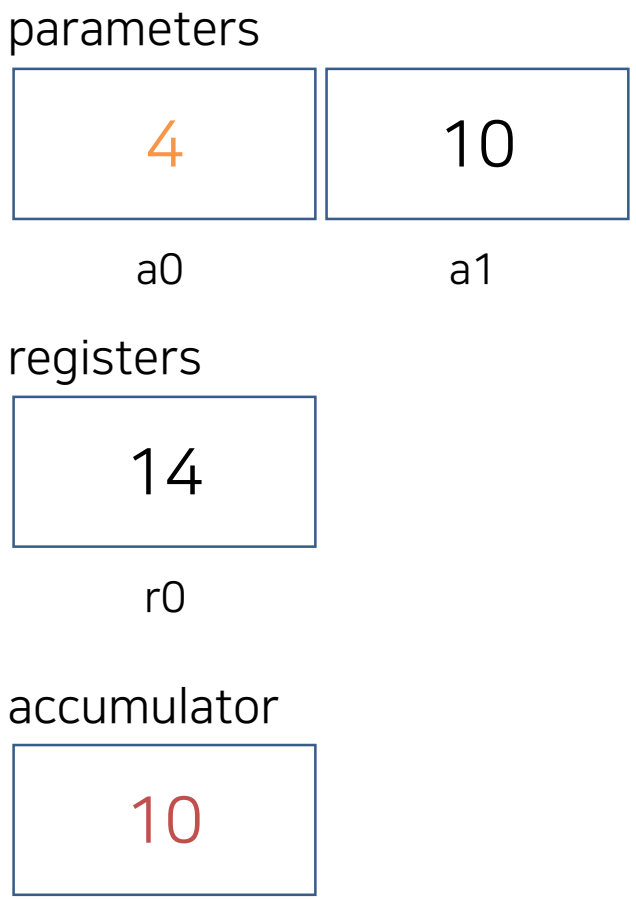
```
00 : a1          StackCheck
01 : 25 02       Ldar a1
03 : 32 03 00    Add a0, [0]
06 : 26 fb       Star r0
08 : 25 02       Ldar a1
10 : 67 03 01    TestGreaterThan a0, [1]
13 : 95 09       JumpIfFalse [9] (@ 22)
15 : 25 02       Ldar a1
17 : 33 03 02    Sub a0, [2]
20 : 26 fb       Star r0
22 : 25 fb       Ldar r0
24 : a5          Return
```



Ignition의 작동 과정

f(4, 10);

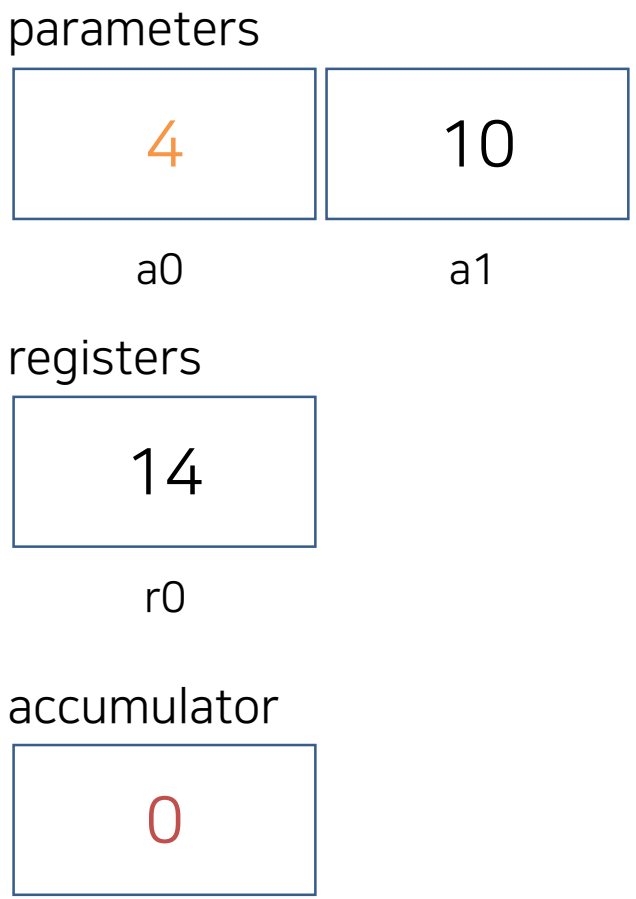
```
00 : a1          StackCheck
01 : 25 02       Ldar a1
03 : 32 03 00    Add a0, [0]
06 : 26 fb       Star r0
08 : 25 02       Ldar a1
10 : 67 03 01    TestGreaterThan a0, [1]
13 : 95 09       JumpIfFalse [9] (@ 22)
15 : 25 02       Ldar a1
17 : 33 03 02    Sub a0, [2]
20 : 26 fb       Star r0
22 : 25 fb       Ldar r0
24 : a5          Return
```



Ignition의 작동 과정

f(4, 10);

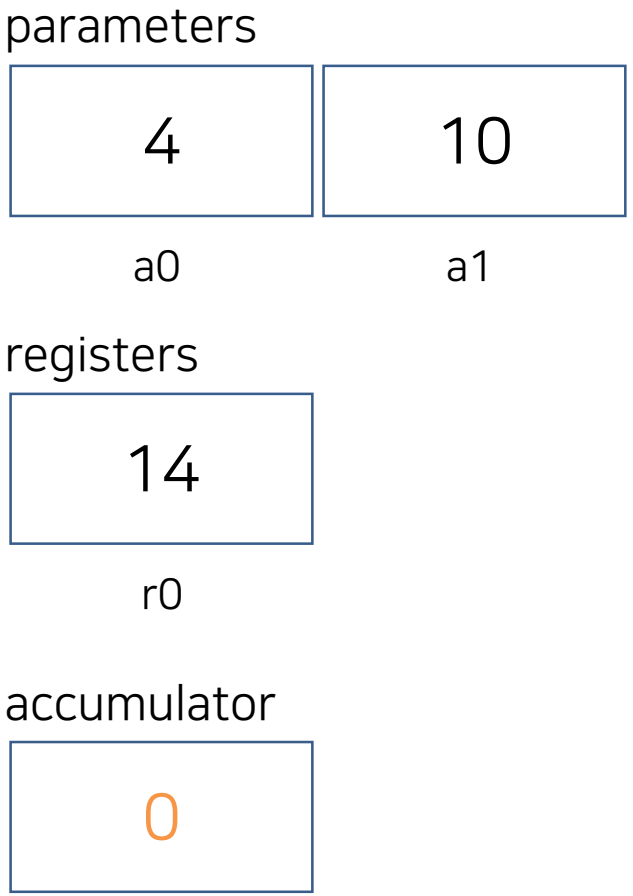
```
00 : a1          StackCheck
01 : 25 02       Ldar a1
03 : 32 03 00    Add a0, [0]
06 : 26 fb       Star r0
08 : 25 02       Ldar a1
10 : 67 03 01    TestGreaterThan a0, [1]
13 : 95 09       JumpIfFalse [9] (@ 22)
15 : 25 02       Ldar a1
17 : 33 03 02    Sub a0, [2]
20 : 26 fb       Star r0
22 : 25 fb       Ldar r0
24 : a5          Return
```



Ignition의 작동 과정

f(4, 10);

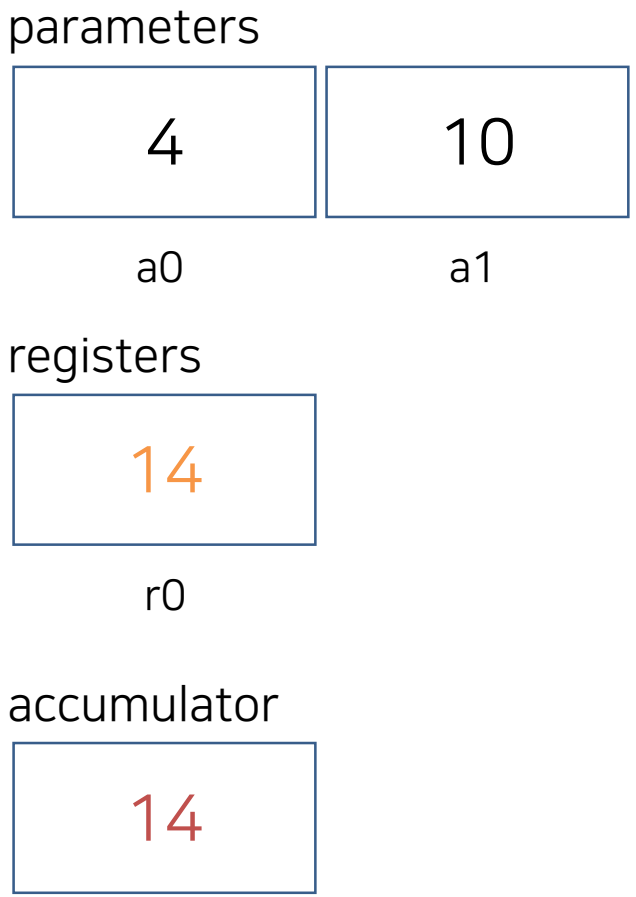
```
00 : a1          StackCheck
01 : 25 02       Ldar a1
03 : 32 03 00    Add a0, [0]
06 : 26 fb       Star r0
08 : 25 02       Ldar a1
10 : 67 03 01    TestGreaterThan a0, [1]
13 : 95 09       JumpIfFalse [9] (@ 22)
15 : 25 02       Ldar a1
17 : 33 03 02    Sub a0, [2]
20 : 26 fb       Star r0
22 : 25 fb       Ldar r0
24 : a5          Return
```



Ignition의 작동 과정

f(4, 10);

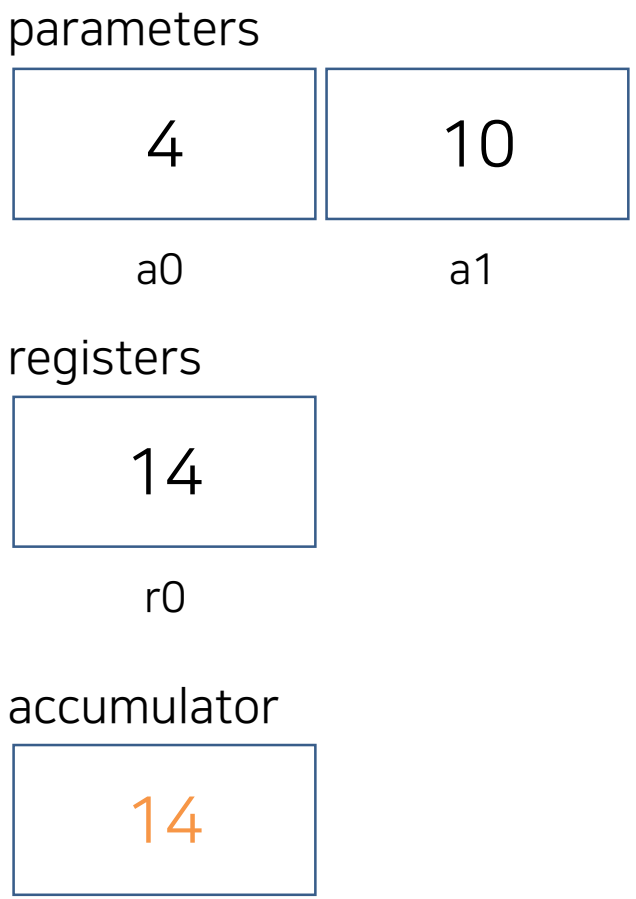
```
00 : a1          StackCheck
01 : 25 02       Ldar a1
03 : 32 03 00    Add a0, [0]
06 : 26 fb       Star r0
08 : 25 02       Ldar a1
10 : 67 03 01    TestGreaterThan a0, [1]
13 : 95 09       JumpIfFalse [9] (@ 22)
15 : 25 02       Ldar a1
17 : 33 03 02    Sub a0, [2]
20 : 26 fb       Star r0
22 : 25 fb       Ldar r0
24 : a5          Return
```



Ignition의 작동 과정

f(4, 10);

```
00 : a1          StackCheck
01 : 25 02       Ldar a1
03 : 32 03 00    Add a0, [0]
06 : 26 fb       Star r0
08 : 25 02       Ldar a1
10 : 67 03 01    TestGreaterThan a0, [1]
13 : 95 09       JumpIfFalse [9] (@ 22)
15 : 25 02       Ldar a1
17 : 33 03 02    Sub a0, [2]
20 : 26 fb       Star r0
22 : 25 fb       Ldar r0
24 : a5          Return
```



Q & A

참고 자료

- https://mathiasbynens.be/notes/prototypes
- https://draft.li/blog/2016/01/22/chromium-chrome-v8-crankshaft-bailout-reasons/
- https://v8.dev/blog/launching-ignition-and-turbofan
- https://v8.dev/blog/ignition-interpreter
- https://github.com/v8/v8/blob/master/src/interpreter/bytecodes.h
- https://nodesource.com/blog/why-the-new-v8-is-so-damn-fast/
- https://medium.com/dailyjs/understanding-v8s-bytecode-317d46c94775
- https://hackernoon.com/javascript-v8-engine-explained-3f940148d4ef
- https://benediktmeurer.de/2016/11/25/v8-behind-the-scenes-november-edition
- https://v8.dev/docs/ignition
- https://v8.dev/docs/turbofan

감사합니다