



Guiding the Design and Implementation of JavaScript using Mechanized Specification

고려대학교 KU-KIST융합대학원 2023학년도 저널클럽

박지혁

고려대학교 컴퓨터학과
프로그래밍 언어 연구실

2023. 04. 20

소개

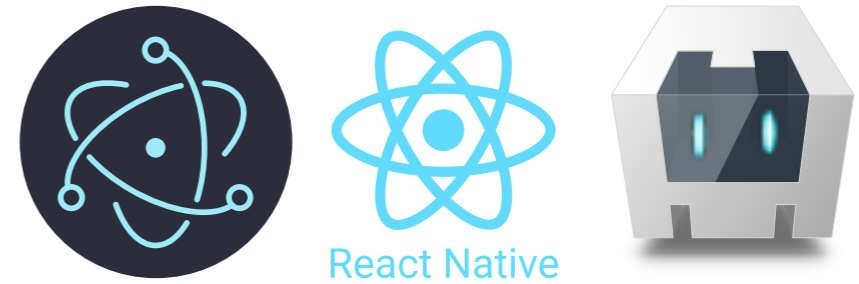
- 이름: 박지혁 (Jihyeok Park)
- 소속: 고려대학교 정보대학 컴퓨터학과 조교수
- 연구 분야: 프로그래밍 언어, 소프트웨어 분석, 소프트웨어 보안
 - ASE (2020, 2021), ICSE 2021, ESEC/FSE (2021, 2022), PLDI (2023)
- 웹사이트: <https://plrg.korea.ac.kr/>



자바스크립트는 어디에나 있다



클라이언트-사이드 프로그래밍

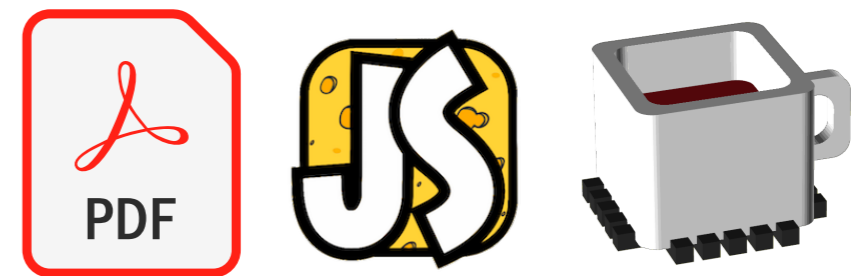


모바일/데스크톱 어플리케이션

JS



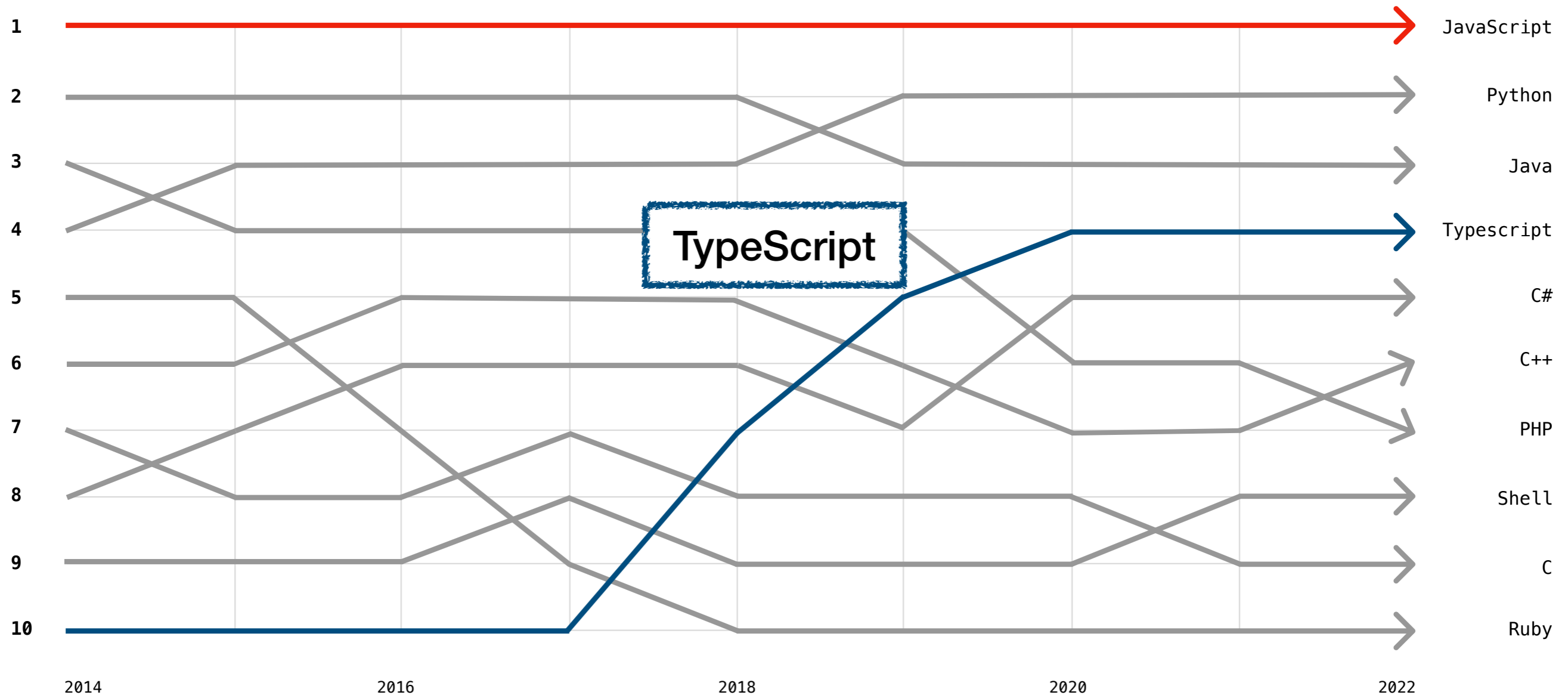
서버-사이드 프로그래밍



그 외 (PDF, 사물 인터넷, 마이크로컨트롤러, etc.)

자바스크립트는 어디에나 있다

JavaScript



<https://octoverse.github.com/>

하지만, 자바스크립트는 복잡하다..

```
function f(x) { return x == !x; }
```

언제나 **false**를 반환할까?

하지만, 자바스크립트는 복잡하다..

```
function f(x) { return x == !x; }
```

언제나 **false**를 반환할까?

NO!!

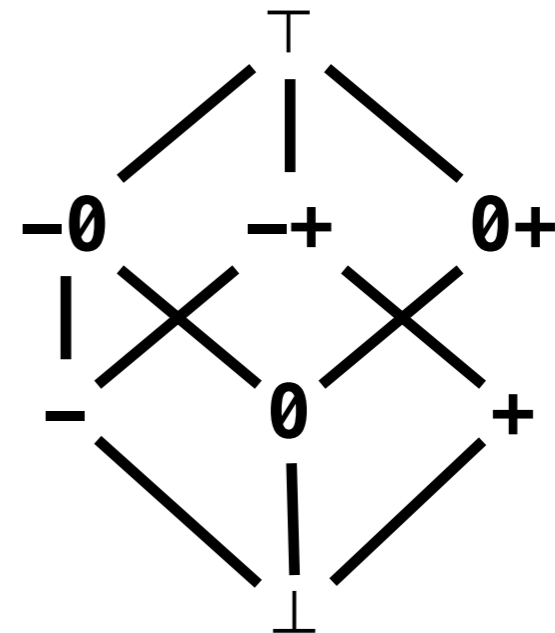
```
f( []) -> [] == ![]  
        -> [] == false  
        -> +[] == +false  
        -> 0 == 0  
        -> true
```

정적 분석 (Static Analysis)

- 주어진 프로그램을 **실행하지 않고 (정적으로)** 분석하는 기법
- 프로그램의 행동을 **요약해서 해석**

예를 들어, 정수를 요약해보자

125 4 12252
0 -42 -1
1024 5
-10



정적 분석 (Static Analysis)

```
function f(x) {  
    if (x == 0) {  
        return 0;  
    } else if (x < 0) {  
        return -x;  
    } else {  
        return x;  
    }  
}
```


정적 분석 (Static Analysis)

```
function f(x) {  
  if (x == 0) {  
    return 0;  
  } else if (x < 0) {  
    return -x;  
  } else {  
    return x;  
  }  
}
```

실제 실행

$f(-4) = 4$ $f(0) = 0$
 . . . $f(-42) = 42$
 $f(5) = 5$. . .

정적 분석 (Static Analysis)

```
function f(x) {  
  // x == T  
  if (x == 0) {  
    return 0;  
  } else if (x < 0) {  
    return -x;  
  } else {  
    return x;  
  }  
}
```

실제 실행

$f(-4) = 4$ $f(0) = 0$
 . . . $f(-42) = 42$
 $f(5) = 5$. . .

정적 분석 (Static Analysis)

```
function f(x) {  
  // x == T  
  if (x == 0) {  
    // x == 0  
    return 0;  
  } else if (x < 0) {  
    return -x;  
  } else {  
    return x;  
  }  
}
```

실제 실행

$f(-4) = 4$ $f(0) = 0$
 . . . $f(-42) = 42$
 $f(5) = 5$. . .

정적 분석 (Static Analysis)

```
function f(x) {  
  // x == T  
  if (x == 0) {  
    // x == 0  
    return 0;  
    // [RETURN] 0  
  } else if (x < 0) {  
  
    return -x;  
  
  } else {  
  
    return x;  
  
  }  
}
```

실제 실행

$f(-4) = 4$ $f(0) = 0$
 . . . $f(-42) = 42$
 $f(5) = 5$. . .

정적 분석 (Static Analysis)

```
function f(x) {  
  // x == T  
  if (x == 0) {  
    // x == 0  
    return 0;  
    // [RETURN] 0  
  } else if (x < 0) {  
    // x == -  
    return -x;  
  } else {  
    return x;  
  }  
}
```

실제 실행

$f(-4) = 4$ $f(0) = 0$
 . . . $f(-42) = 42$
 $f(5) = 5$. . .

정적 분석 (Static Analysis)

```
function f(x) {  
  // x == T  
  if (x == 0) {  
    // x == 0  
    return 0;  
    // [RETURN] 0  
  } else if (x < 0) {  
    // x == -  
    return -x;  
    // [RETURN] +  
  } else {  
  
    return x;  
  
  }  
}
```

실제 실행

$f(-4) = 4$ $f(0) = 0$
 . . . $f(-42) = 42$
 $f(5) = 5$. . .

정적 분석 (Static Analysis)

```
function f(x) {  
  // x == T  
  if (x == 0) {  
    // x == 0  
    return 0;  
    // [RETURN] 0  
  } else if (x < 0) {  
    // x == -  
    return -x;  
    // [RETURN] +  
  } else {  
    // x == +  
    return x;  
  }  
}
```

실제 실행

$f(-4) = 4$ $f(0) = 0$
 . . . $f(-42) = 42$
 $f(5) = 5$. . .

정적 분석 (Static Analysis)

```
function f(x) {  
  // x == T  
  if (x == 0) {  
    // x == 0  
    return 0;  
    // [RETURN] 0  
  } else if (x < 0) {  
    // x == -  
    return -x;  
    // [RETURN] +  
  } else {  
    // x == +  
    return x;  
    // [RETURN] +  
  }  
}
```

실제 실행

$f(-4) = 4$ $f(0) = 0$
 . . . $f(-42) = 42$
 $f(5) = 5$. . .

정적 분석 (Static Analysis)

```
function f(x) {  
  // x == T  
  if (x == 0) {  
    // x == 0  
    return 0;  
    // [RETURN] 0  
  } else if (x < 0) {  
    // x == -  
    return -x;  
    // [RETURN] +  
  } else {  
    // x == +  
    return x;  
    // [RETURN] +  
  }  
} // [RETURN] 0+
```

실제 실행

$f(-4) = 4$ $f(0) = 0$
 . . . $f(-42) = 42$
 $f(5) = 5$. . .

정적 분석 (Static Analysis)

```
function f(x) {  
  // x == T  
  if (x == 0) {  
    // x == 0  
    return 0;  
    // [RETURN] 0  
  } else if (x < 0) {  
    // x == -  
    return -x;  
    // [RETURN] +  
  } else {  
    // x == +  
    return x;  
    // [RETURN] +  
  }  
} // [RETURN] 0+
```

실제 실행

$f(-4) = 4$ $f(0) = 0$
 . . . $f(-42) = 42$
 $f(5) = 5$. . .

요약 해석

$f(T) = 0+$

정적 분석 (Static Analysis)

```
function f(x) {  
  // x == T  
  if (x == 0) {  
    // x == 0  
    return 0;  
    // [RETURN] 0  
  } else if (x < 0) {  
    // x == -  
    return -x;  
    // [RETURN] +  
  } else {  
    // x == +  
    return x;  
    // [RETURN] +  
  }  
} // [RETURN] 0+
```

실제 실행

$f(-4) = 4$ $f(0) = 0$
 . . . $f(-42) = 42$
 $f(5) = 5$. . .

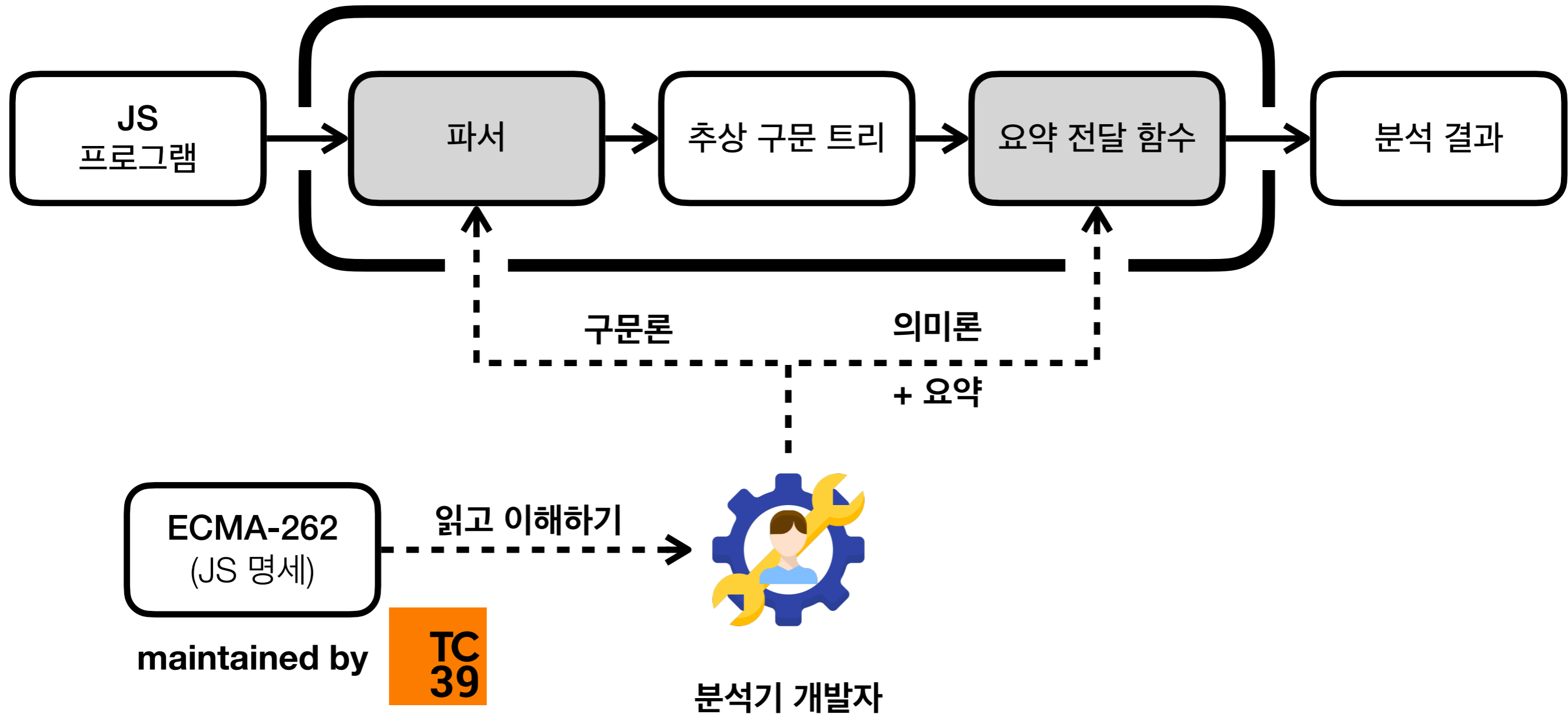
요약 해석

$f(T) = 0+$

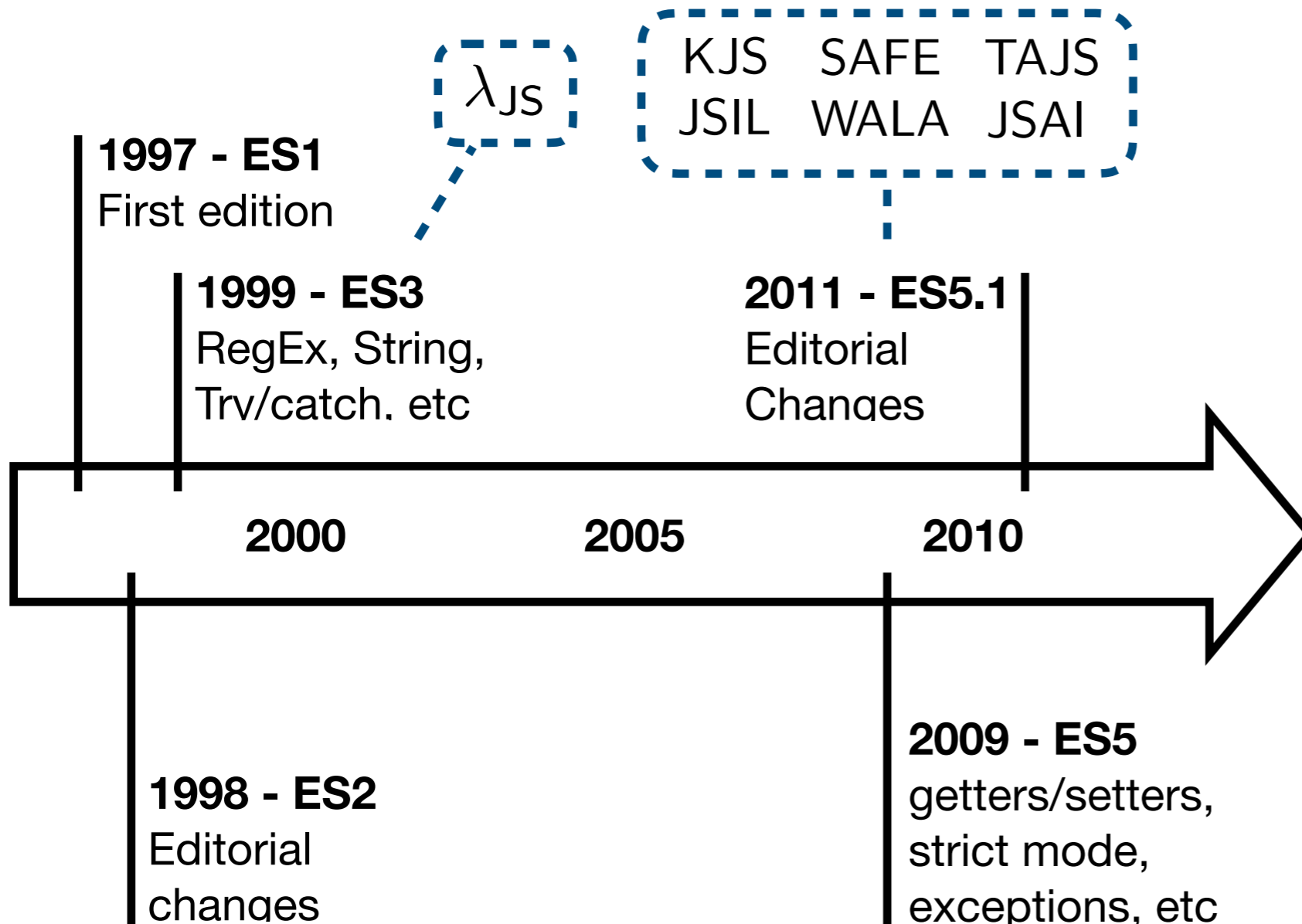
타입 추론 (분석) \subseteq

자바스크립트를 위한 정적 분석

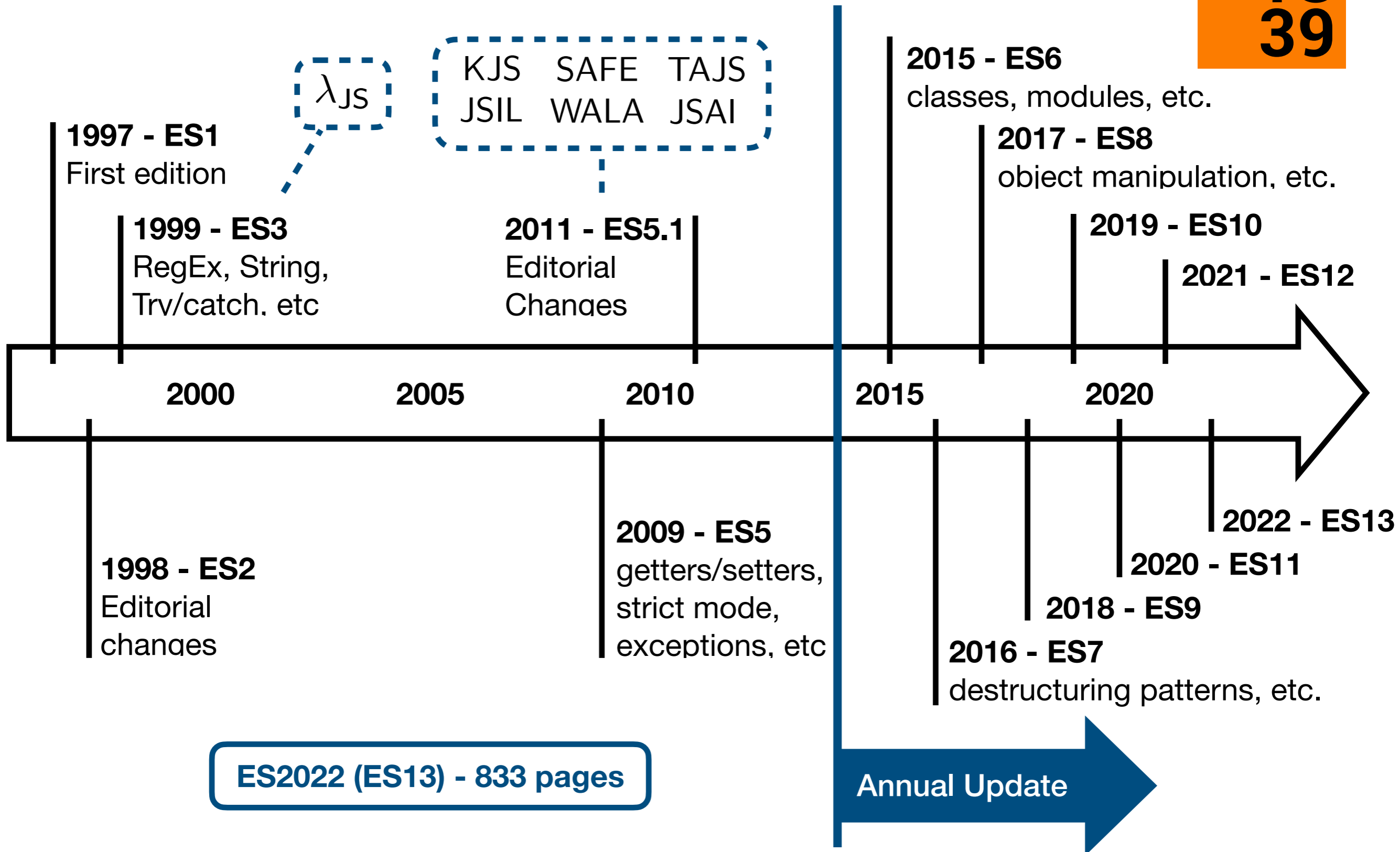
자바스크립트 정적 분석기



문제: 빠르게 성장하는 자바스크립트

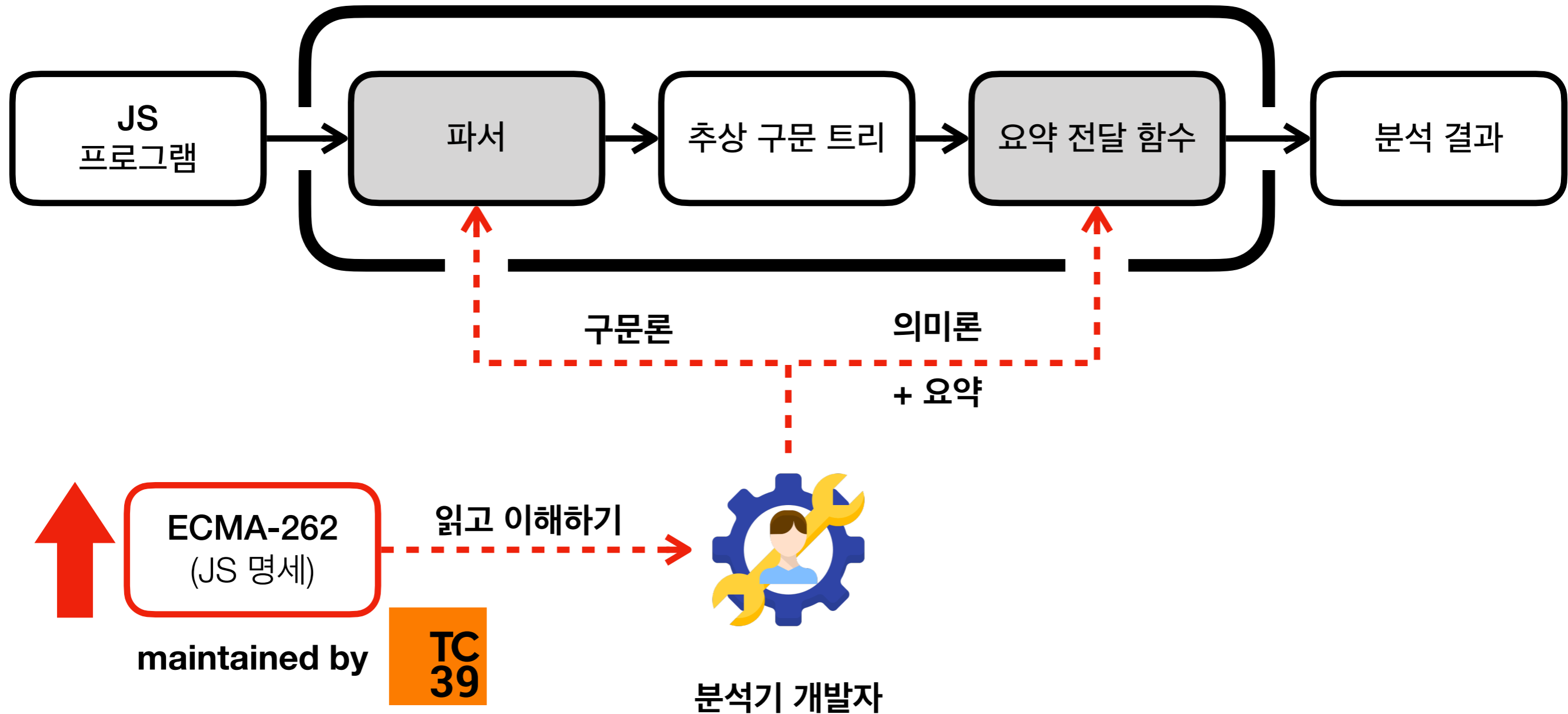


문제: 빠르게 성장하는 자바스크립트

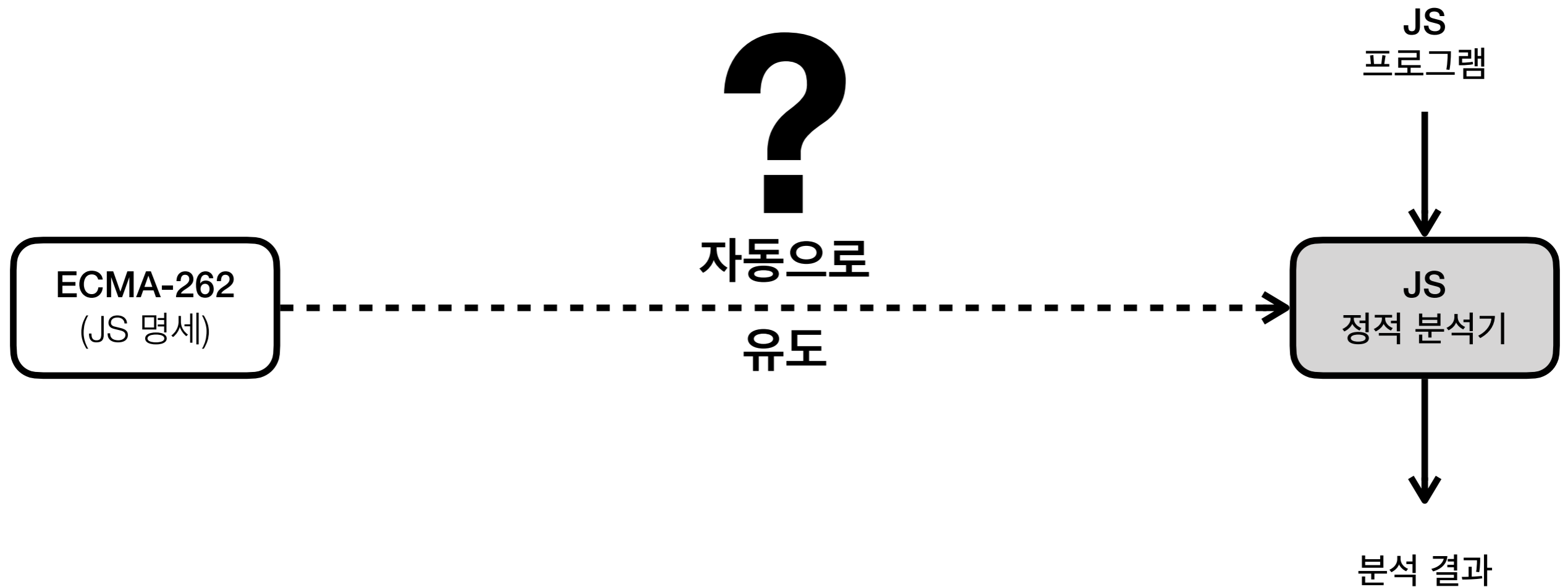


문제: 수동으로 분석기 갱신

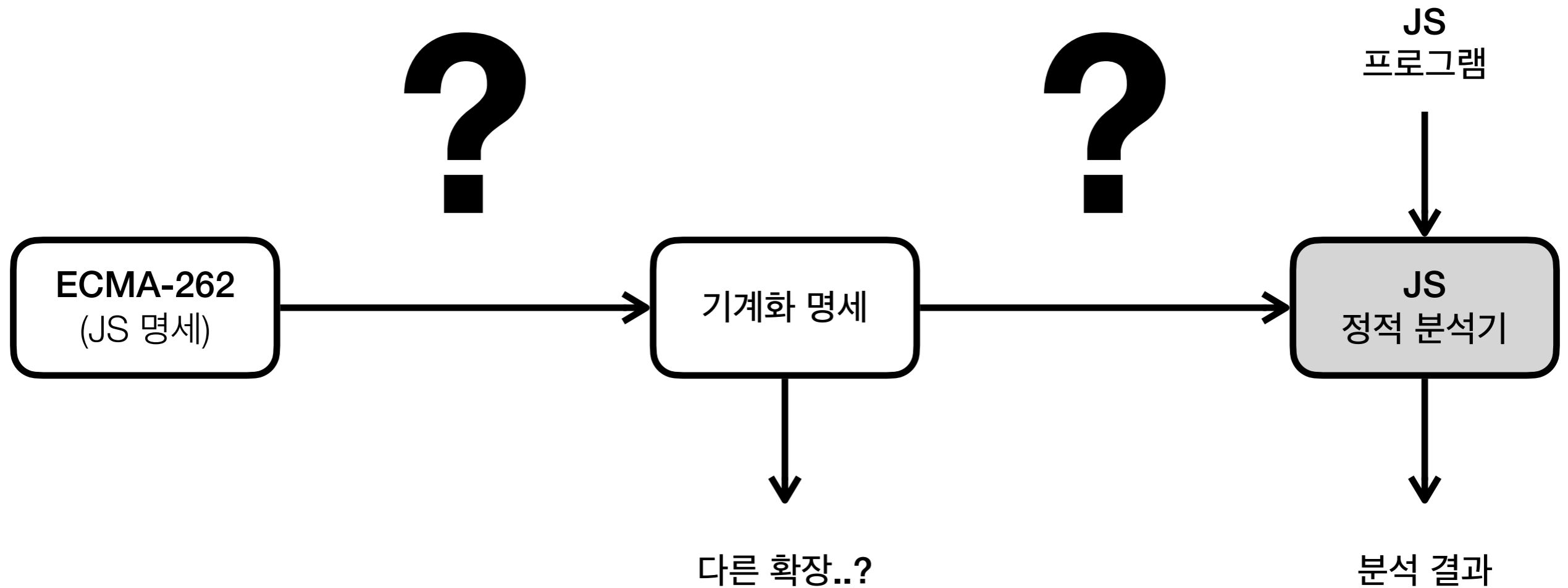
자바스크립트 정적 분석기

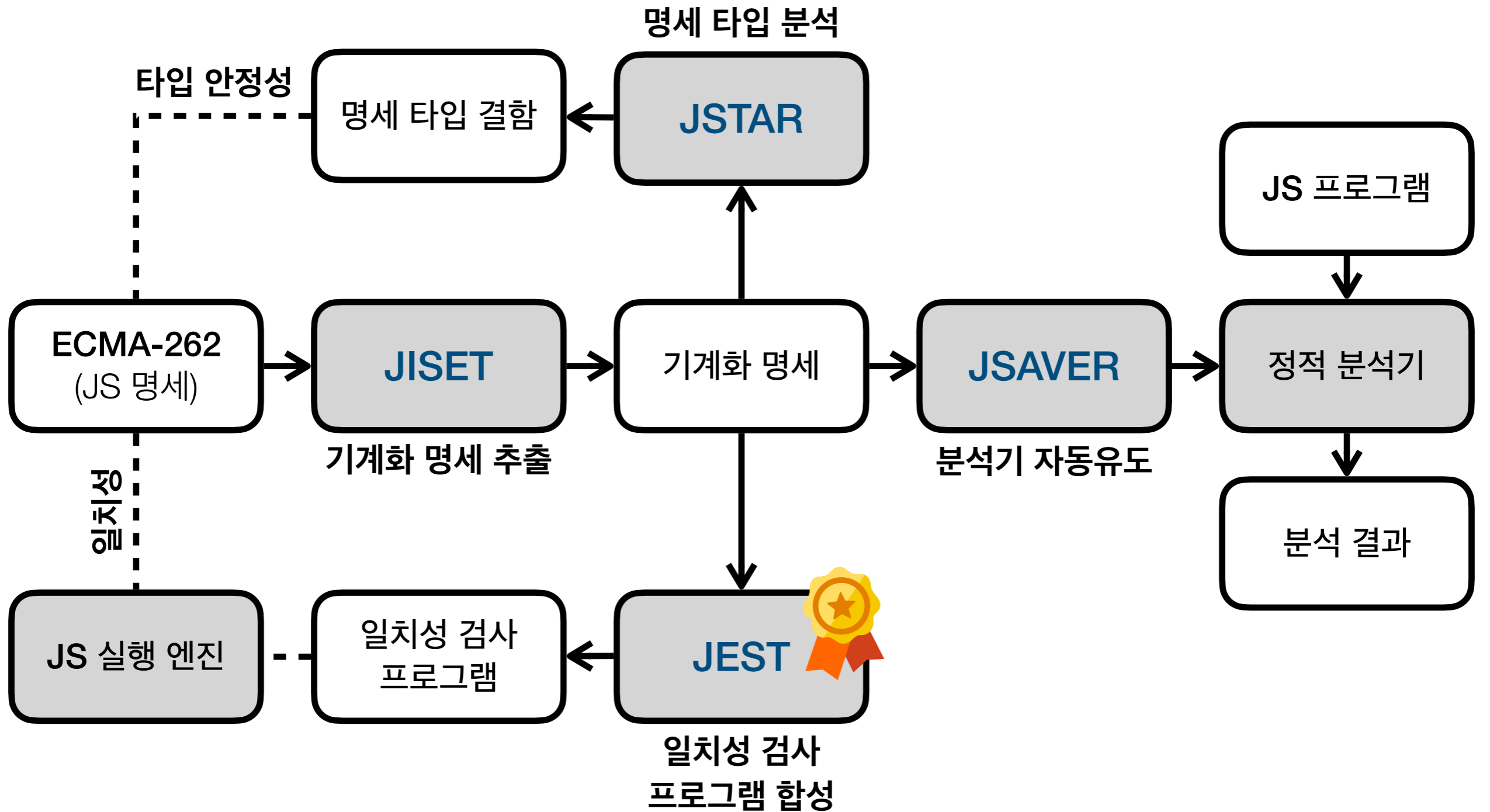


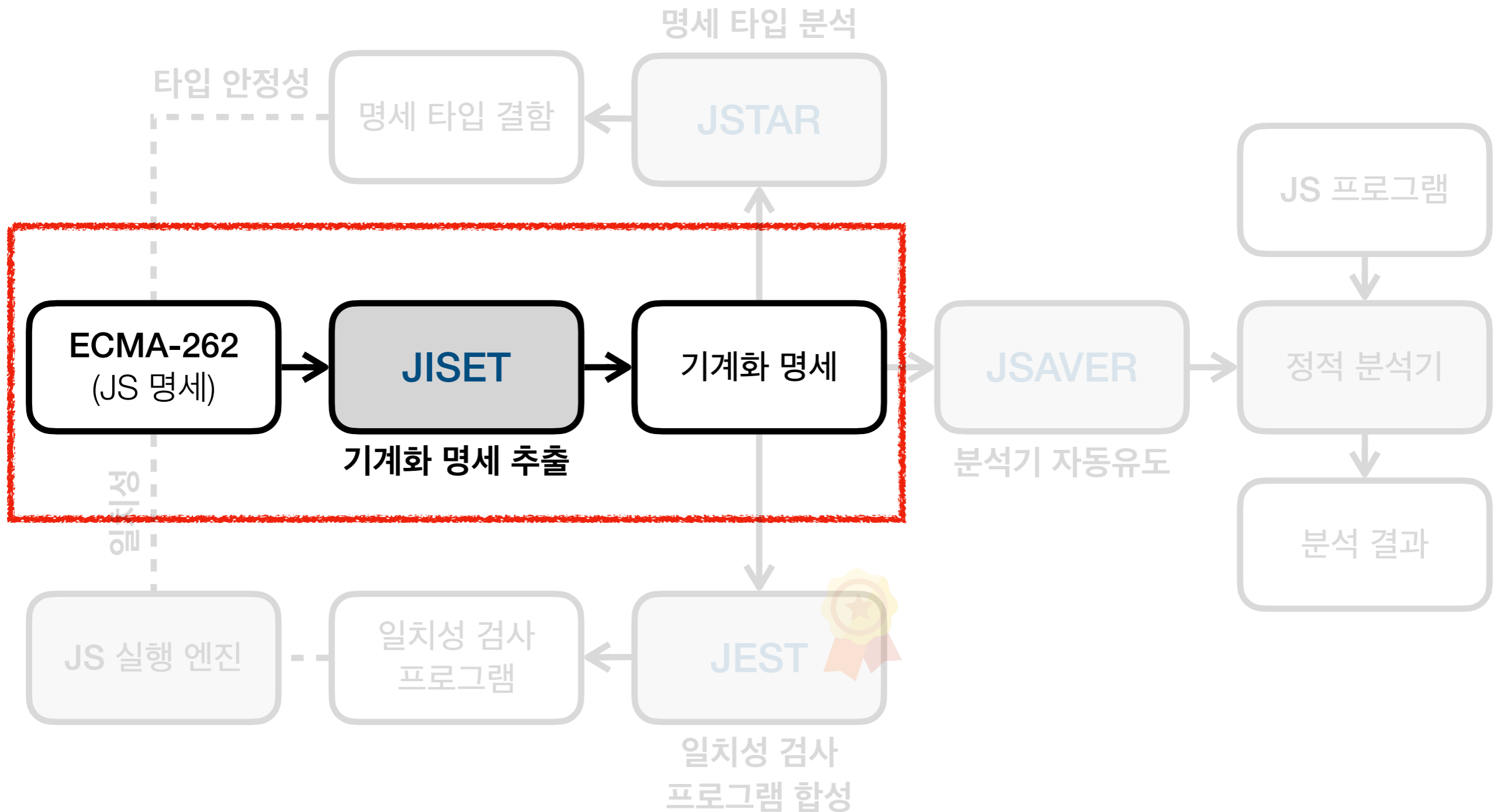
분석기를 자동으로 유도하기?



아이디어: 기계화 명세







ECMA-262 (자바스크립트 명세)

```
ArrayLiteral[Yield, Await] :  
  [ Elisionopt ]  
  [ ElementList[?Yield, ?Await] ]  
  [ ElementList[?Yield, ?Await] , Elisionopt ]
```

The production of *ArrayLiteral* in ES13

구문론

TC
39

ECMA-262 (자바스크립트 명세)

```
ArrayLiteral[Yield, Await] :  
  [ Elisionopt ]  
  [ ElementList[?Yield, ?Await] ]  
  [ ElementList[?Yield, ?Await] , Elisionopt ]
```

The production of *ArrayLiteral* in ES13

구문론

TC
39

의미론

13.2.5.2 Runtime Semantics: Evaluation

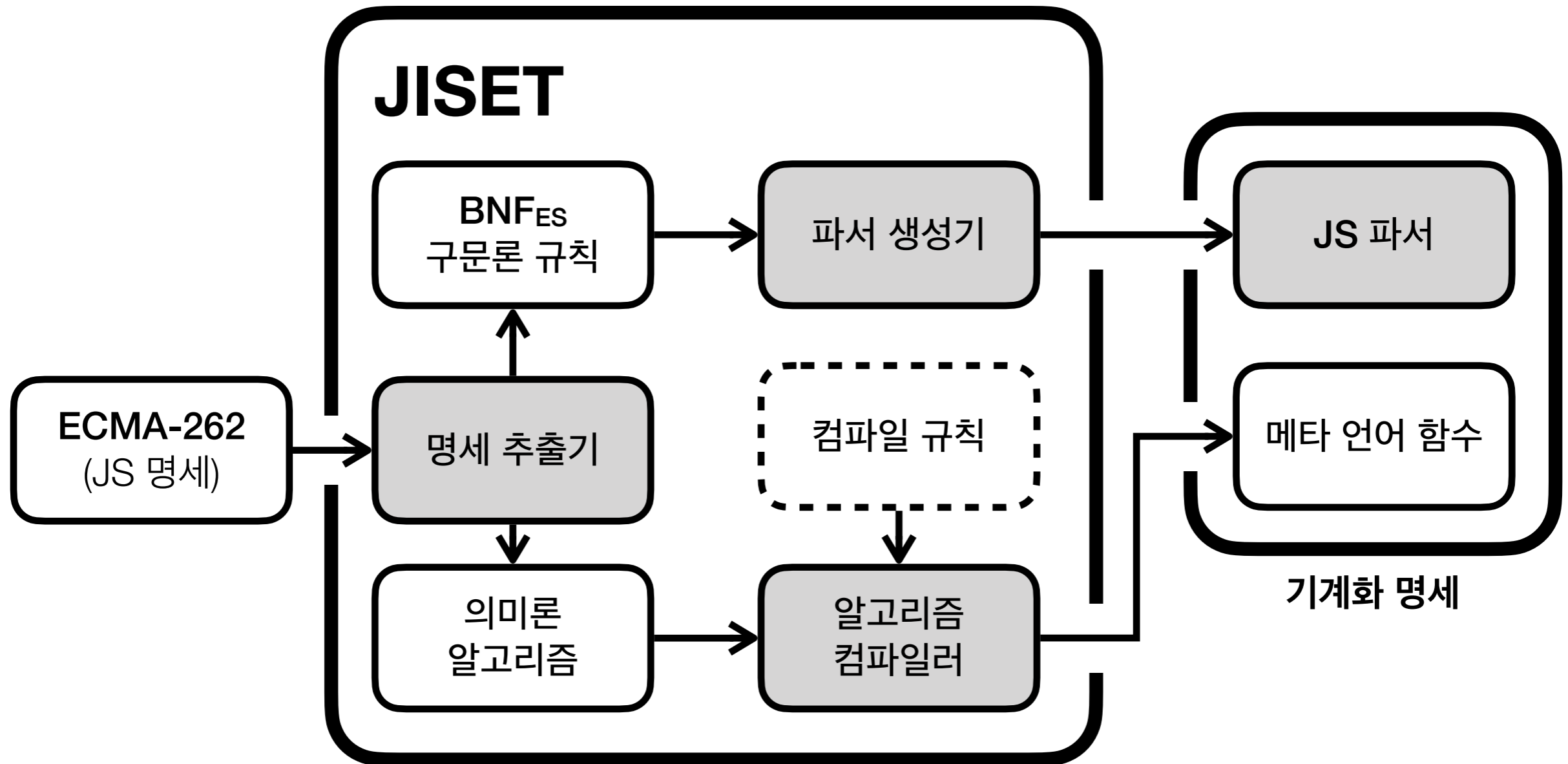
ArrayLiteral : [*ElementList* , *Elision*_{opt}]

1. Let *array* be ! *ArrayCreate*(0).
2. Let *nextIndex* be the result of performing *ArrayAccumulation* for *ElementList* with arguments *array* and 0.
3. *ReturnIfAbrupt*(*nextIndex*).
4. If *Elision* is present, then
 - a. Let *len* be the result of performing *ArrayAccumulation* for *Elision* with arguments *array* and *nextIndex*.
 - b. *ReturnIfAbrupt*(*len*).
5. Return *array*.

The Evaluation algorithm for the third alternative of *ArrayLiteral* in ES13

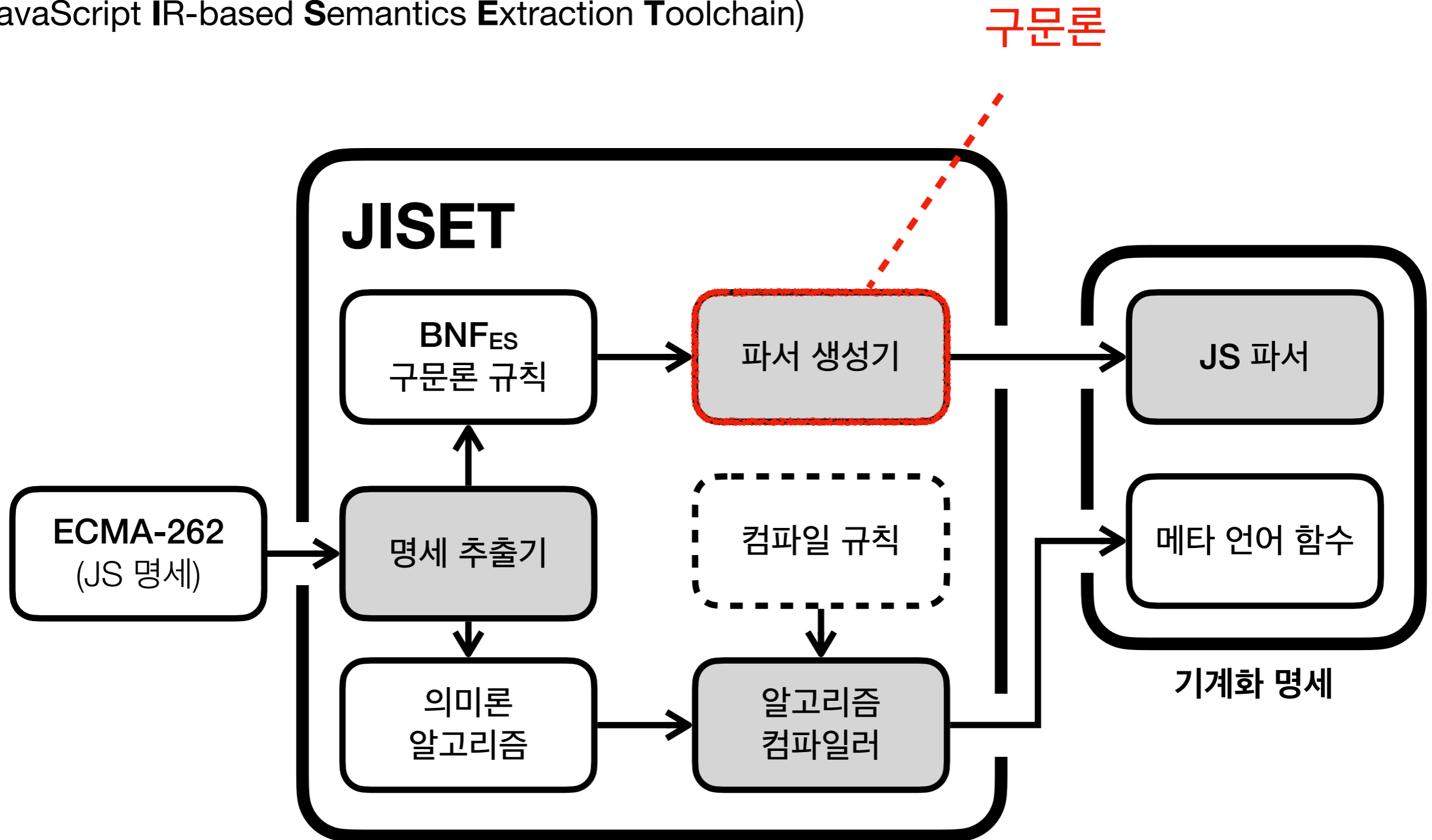
JISSET - ASE'20

(JavaScript IR-based Semantics Extraction Toolchain)



JISSET - ASE'20

(JavaScript IR-based Semantics Extraction Toolchain)



JISET - 자동 파서 생성 (구문론)

```
ArrayLiteral[Yield, Await] :  
  [ Elisionopt ]  
  [ ElementList[?Yield, ?Await] ]  
  [ ElementList[?Yield, ?Await] , Elisionopt ]
```

**Parsing Expression Grammar
(PEG)**

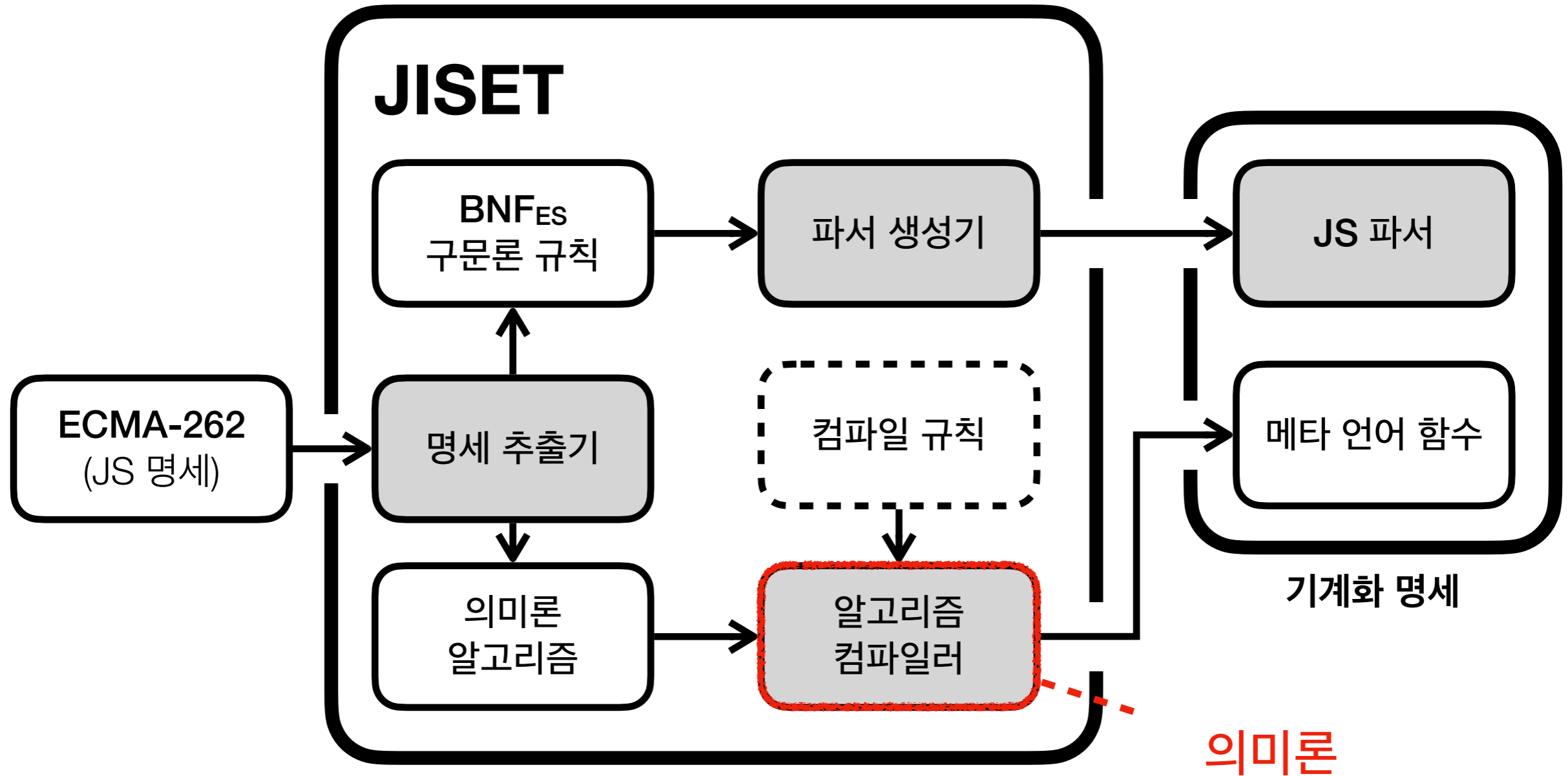


```
val ArrayLiteral: List[Boolean] => LAParser[T] = memo {  
  case List(Yield, Await) =>  
    "[" ~ opt(Elision) ~ "]"          ^^ ArrayLiteral0 |  
    "[" ~ ElementList(Yield, Await) ~ "]" ^^ ArrayLiteral1 |  
    "[" ~ ElementList(Yield, Await) ~ "," ~  
      ~ opt(Elision) ~ "]"          ^^ ArrayLiteral2  
}
```

[POPL'04] B. Ford, "Parsing Expression Grammars: A Recognition-based Syntactic Foundation"

JISET - ASE'20

(JavaScript IR-based Semantics Extraction Toolchain)



JISSET - ECMA-262를 위한 메타 언어

- ECMA-262의 추상 알고리즘을 표현하기 위한 **메타 언어**를 정의

Programs	$\mathfrak{P} \ni P ::= f^*$
Functions	$\mathcal{F} \ni f ::= \text{syntax}^? \text{ def } x(x^*) \{ [l : i]^* \}$
Variables	$\mathcal{X} \ni x$
Labels	$\mathcal{L} \ni l$
Instructions	$\mathcal{I} \ni i ::= r := e \mid x := \{ \} \mid x := e(e^*)$ $\mid \text{if } e \text{ } l \text{ } l \mid \text{return } e$
Expressions	$\mathcal{E} \ni e ::= v^p \mid \text{op}(e^*) \mid r$
References	$\mathcal{R} \ni r ::= x \mid e[e] \mid e[e]_{\text{js}}$

• • •

Values

$$v \in \mathbb{V} = \mathbb{A} \uplus \mathbb{V}^p \uplus \mathbb{T} \uplus \mathcal{F}$$

Primitive Values

$$v^p \in \mathbb{V}^p = \mathbb{V}_{\text{bool}} \uplus \mathbb{V}_{\text{int}} \uplus \mathbb{V}_{\text{str}} \uplus \dots$$

JS ASTs

$$t \in \mathbb{T}$$

• • •

JISET - ECMA-262를 위한 메타 언어

- ECMA-262의 추상 알고리즘을 표현하기 위한 **메타 언어**를 정의

Programs	$\mathfrak{P} \ni P ::= f^*$
Functions	$\mathcal{F} \ni f ::= \text{syntax}^? \text{ def } x(x^*) \{ [l : i]^* \}$
Variables	$\mathcal{X} \ni x$
Labels	$\mathcal{L} \ni l$
Instructions	$\mathcal{I} \ni i ::= r := e \mid x := \{ \} \mid x := e(e^*)$ $\mid \text{if } e \text{ } l \text{ } l \mid \text{return } e$
Expressions	$\mathcal{E} \ni e ::= v^p \mid \text{op}(e^*) \mid r$
References	$\mathcal{R} \ni r ::= x \mid e[e] \mid e[e]_{js}$

Values

$$v \in \mathbb{V} = \mathbb{A} \uplus \mathbb{V}^p \uplus \mathbb{T} \uplus \mathcal{F}$$

Primitive Values

$$v^p \in \mathbb{V}^p = \mathbb{V}_{\text{bool}} \uplus \mathbb{V}_{\text{int}} \uplus \mathbb{V}_{\text{str}} \uplus \dots$$

JS ASTs

$$t \in \mathbb{T}$$

JISET - 알고리즘 컴파일러 (의미론)

13.2.5.2 Runtime Semantics: Evaluation

ArrayLiteral : [*ElementList* , *Elision*_{opt}]

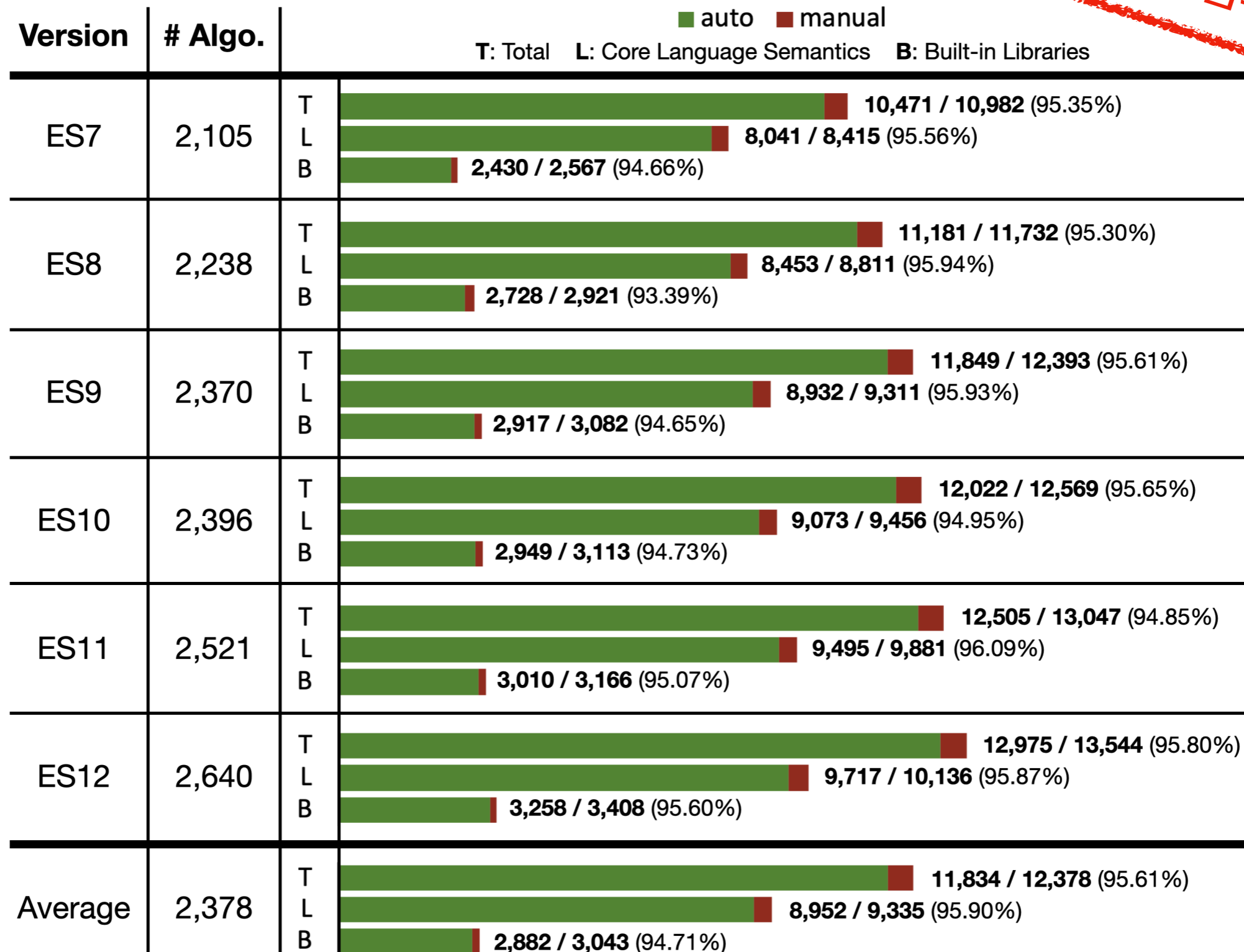
1. Let *array* be ! *ArrayCreate*(0).
2. Let *nextIndex* be the result of performing *ArrayAccumulation* for *ElementList* with arguments *array* and 0.
3. *ReturnIfAbrupt*(*nextIndex*).
4. If *Elision* is present, then
 - a. Let *len* be the result of performing *ArrayAccumulation* for *Elision* with arguments *array* and *nextIndex*.
 - b. *ReturnIfAbrupt*(*len*).
5. Return *array*.

추상 알고리즘을 메타 언어로
컴파일하기 위한
118 가지 컴파일 규칙

```
syntax def ArrayLiteral[2].Evaluation(  
  this, ElementList, Elision  
) {  
  let array = [! (ArrayCreate 0)]  
  let nextIndex = (ElementList.ArrayAccumulation array 0)  
  [? nextIndex]  
  if (! (= Elision absent)) {  
    let len = (Elision.ArrayAccumulation array nextIndex)  
    [? len]  
  }  
  return array  
}
```

JISET - Evaluation

≈ 96%
 자동 컴파일



ESMeta

The screenshot shows the GitHub repository page for `es-meta/esmeta`. The repository is public and has 115 stars, 7 watchers, and 9 forks. The main branch is `main`. The repository description is "ECMAScript Specification (ECMA-262) Metalanguage". The repository includes tags for `javascript` and `ecmascript`. The repository has 11 releases, with the latest being `v0.2.0` on Dec 14, 2022. The repository also has 8 issues, 1 pull request, and 0 discussions. The repository is maintained by `jhnaldo`, who updated the version on Dec 14, 2022, with 976 commits. The repository includes a `README`, `BSD-3-Clause license`, and `9 forks`.

es-meta / esmeta Public

Edit Pins Unwatch 7 Fork 9 Starred 115

Code Issues 8 Pull requests 1 Discussions Actions

main Go to file Add file Code

About

ECMAScript Specification (ECMA-262) Metalanguage

javascript ecmascript

Readme

BSD-3-Clause license

115 stars

7 watching

9 forks

Releases 11

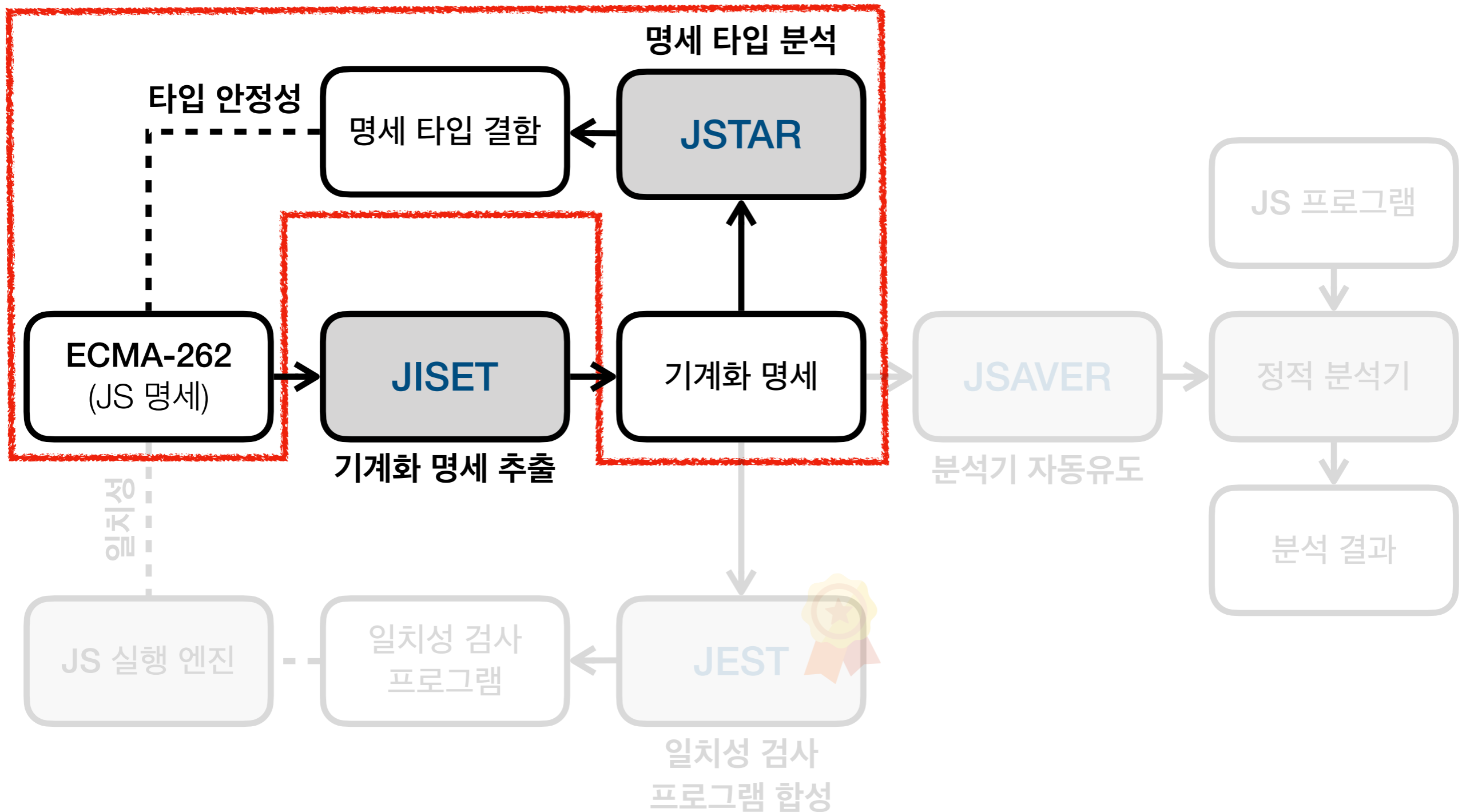
v0.2.0 Latest on Dec 14, 2022

+ 10 releases

jhnaldo Updated version ... on Dec 14, 2022 976

.github/wo...	Updated ci.yml and e2e...	3 months ago
client @ c6...	Updated version of client	3 months ago
ecma262 ...	Remove implicit wrapping/u...	8 months ago
project	Downgraded sbt-assembly f...	4 months ago
src	Updated version	3 months ago
tests	Fixed bugs for Test262 (#118)	3 months ago
.completion	Supported -extract:eval to e...	3 months ago
.gitignore	Updated .gitignore for local ...	6 months ago
.gitmodules	Updated README / Added c...	6 months ago
.jvmopts	Added -XX:ReservedCodeC...	7 months ago

<https://github.com/es-meta/esmeta>



Types in ECMA-262

20.3.2.28 Math.round (x)

1. Let n be ? `ToNumber`(x).
2. If n is an integral Number, return n .
3. If $x < 0.5$ and $x > 0$, return `+0`.
4. If $x < 0$ and $x \geq -0.5$, return `-0`.
- • •

<https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c>

Types in ECMA-262

20.3.2.28 Math.round (x) x : (String v Boolean v Number v Object v ...)

1. Let n be ? **ToNumber**(x).
2. If n is an integral Number, return n .
3. If $x < 0.5$ and $x > 0$, return $+0$.
4. If $x < 0$ and $x \geq -0.5$, return -0 .
- • •

<https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c>

Types in ECMA-262

20.3.2.28 Math.round (*x*) *x*: (String v Boolean v Number v Object v ...)

1. Let *n* be ? **ToNumber(*x*)** ToNumber(*x*): (Number v Exception)
2. If *n* is an integral Number, return *n*.
3. If *x* < 0.5 and *x* > 0, return +0.
4. If *x* < 0 and *x* ≥ -0.5, return -0.
- • •

<https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c>

Types in ECMA-262

20.3.2.28 Math.round (x) x : (String v Boolean v Number v Object v ...)

1. Let n be **? ToNumber(x)** ToNumber(x): (Number v Exception)
2. If n is an integral Number, return n .
3. If $x < 0.5$ and $x > 0$, return $+0$.
4. If $x < 0$ and $x \geq -0.5$, return -0 .
- • •

<https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c>

Types in ECMA-262

20.3.2.28 Math.round (x) x : (String v Boolean v Number v Object v ...)

1. Let n be ? **ToNumber(x)** ToNumber(x): (Number v Exception) \wedge n : (Number)
2. If n is an integral Number, return n .
3. If $x < 0.5$ and $x > 0$, return $+0$.
4. If $x < 0$ and $x \geq -0.5$, return -0 .
- • •

<https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c>

Types in ECMA-262

20.3.2.28 Math.round (x) x : (String v Boolean v Number v Object v ...)

1. Let n be ? **ToNumber(x)** ToNumber(x): (Number v Exception) \wedge n : (Number)

2. If n is an integral Number, return n .

3. If $x < 0.5$ and $x > 0$ return +0.

4. If $x < 0$ and $x \geq -0.5$ return -0.

• • •

`>` 연산 타입 불일치
(숫자만 가능)

<https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c>

Types in ECMA-262

20.3.2.28 Math.round (x) x : (String v Boolean v Number v Object v ...)

1. Let n be ? **ToNumber(x)** ToNumber(x): (Number v Exception) \wedge n : (Number)
2. If n is an integral Number, return n .

3. If $x < 0.5$ and $x > 0$ return +0.
4. If $x < 0$ and $x \geq -0.5$ return -0.

• • •

`>` 연산 타입 불일치
(숫자만 가능)

Math.round(true) = ?
Math.round(false) = ?

<https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c>

Types in ECMA-262

20.3.2.28 Math.round (x) x : (String v Boolean v Number v Object v ...)

1. Let n be ? **ToNumber(x)** ToNumber(x): (Number v Exception) \wedge n : (Number)
2. If n is an integral Number, return n .

3. If $x < 0.5$ and $x > 0$ return +0.
4. If $x < 0$ and $x \geq -0.5$ return -0.

`>` 연산 타입 불일치
(숫자만 가능)

Math.round(true) = ?
Math.round(false) = ?



3. If $n < 0.5$ and $n > 0$, return +0.
4. If $n < 0$ and $n \geq -0.5$, return -0.

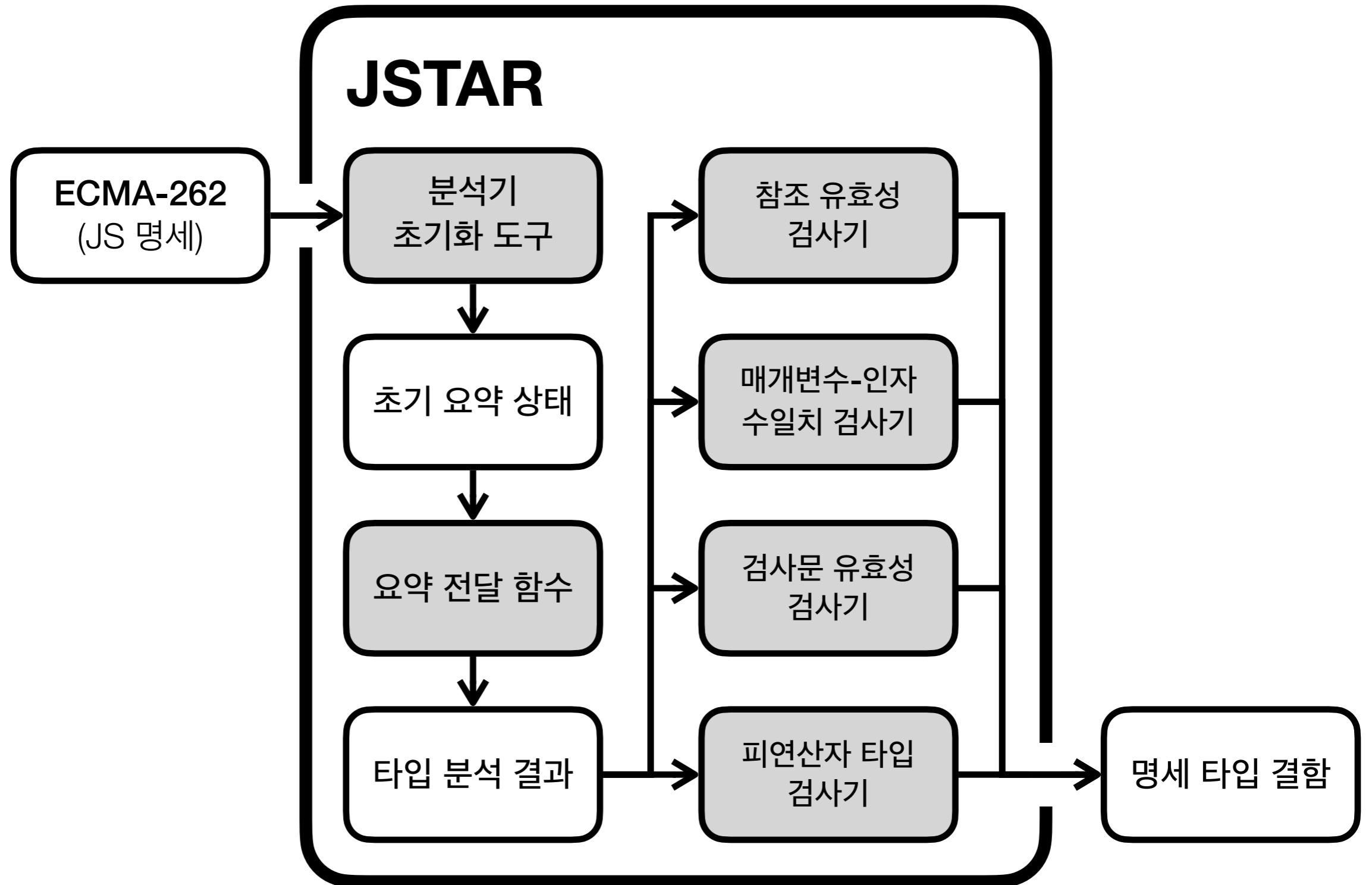


Math.round(true) = 1
Math.round(false) = 0

<https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c>

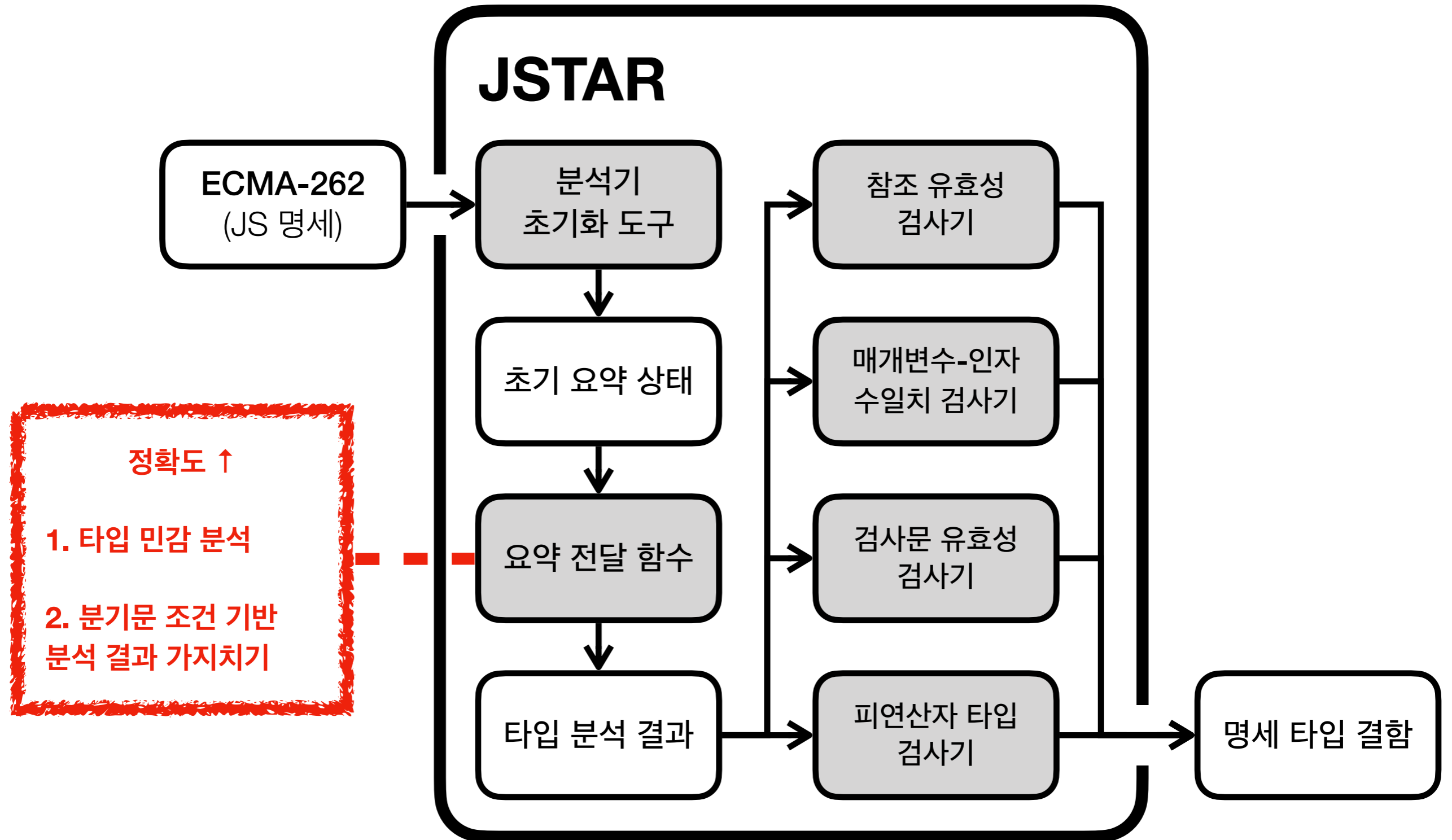
JSTAR - ASE'21

(JavaScript Specification Type Analyzer using Refinement)



JSTAR - ASE'21

(JavaScript Specification Type Analyzer using Refinement)



JSTAR - 실험 결과

59.2% 정확도
타입 결함 93개 검출

- 검사 대상: 3년 동안 864개의 다른 ECMA-262 버전

Checker	Bug Kind	Precision = (# True Bugs) / (# Detected Bugs)					
		no-refine		refine		Δ	
Reference	UnknownVar	62 / 106	17 / 60	63 / 78	17 / 31	+1 / -28	/ -29
	DuplicatedVar		45 / 46		46 / 47		+1 / +1
Arity	MissingParam	4 / 4	4 / 4	4 / 4	4 / 4	/	/
Assertion	Assertion	4 / 56	4 / 56	4 / 31	4 / 31	/ -25	/ -25
Operand	NoNumber	22 / 113	2 / 65	22 / 44	2 / 6	/ -69	/ -59
	Abrupt		20 / 48		20 / 38		/ -10
Total		92 / 279 (33.0%)		93 / 157 (59.2%)		+1 / -122 (+26.3%)	

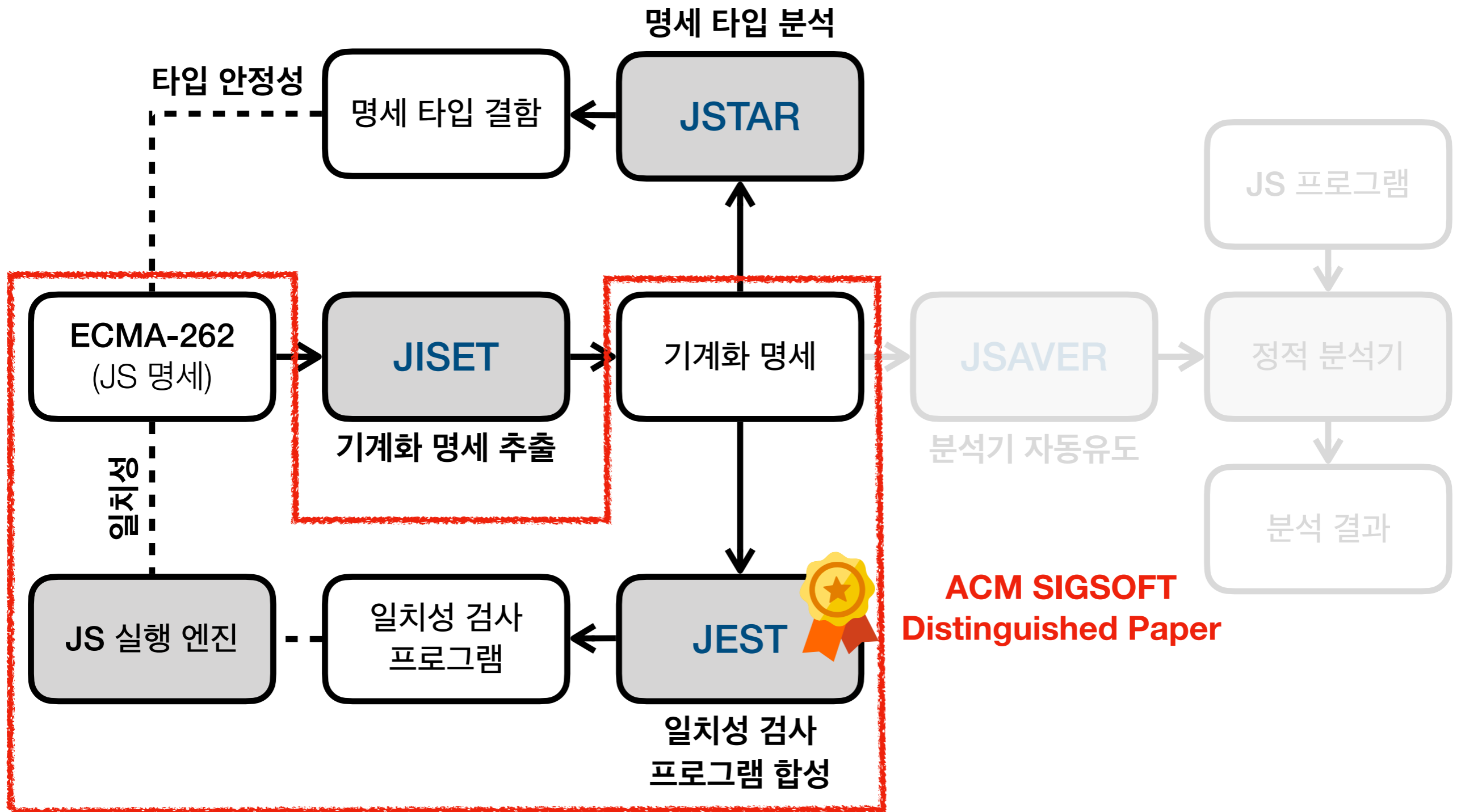
Name	Feature	#	Checker	Created	Life Span
ES12-1	Switch	3	Reference	2015-09-22	1,996 days
ES12-2	Try	3	Reference	2015-09-22	1,996 days
ES12-3	Arguments	1	Reference	2015-09-22	1,996 days
ES12-4	Array	2	Reference	2015-09-22	1,996 days
ES12-5	Async	1	Reference	2015-09-22	1,996 days
ES12-6	Class	1	Reference	2015-09-22	1,996 days
ES12-7	Branch	1	Reference	2015-09-22	1,996 days
ES12-8	Arguments	2	Operand	2015-12-16	1,910 days

ES12에서 결함 14개

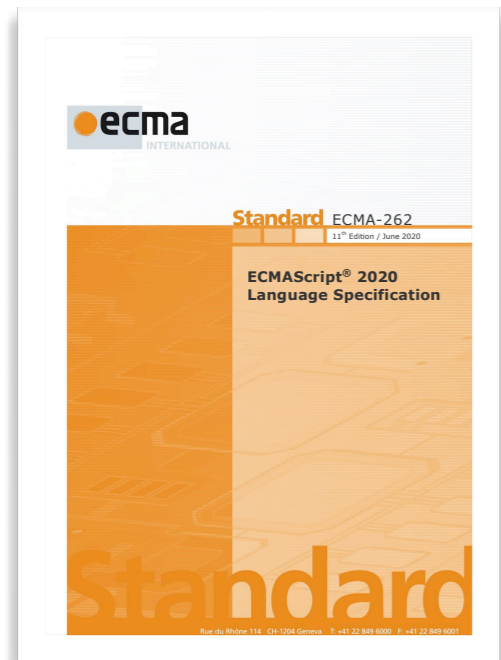
ECMA-262 공식 CI 시스템으로 선정

The screenshot shows the GitHub Actions interface for the repository `tc39/ecma262`. The `Actions` tab is selected, displaying a list of workflow runs for the `esmeta typecheck` workflow. The workflow file is `esmeta-typecheck.yml`. The interface includes a search bar for workflow runs and a table of recent runs.

Event	Status	Branch	Actor	Time
Editorial: Split identity...	Success	syg:stratified-identity	syg	yesterday ... 2m 31s
[Stage 4] Normati...	Success	acutmore:change-array-by-...	acutmore	yesterday ... 2m 42s
Add Class and Class Elem...	Failure	nzurag:decorators	nzurag	2 days ago ...



JS 명세와 실행 엔진 간의 일치성



ECMA-262
(JS 명세)

?



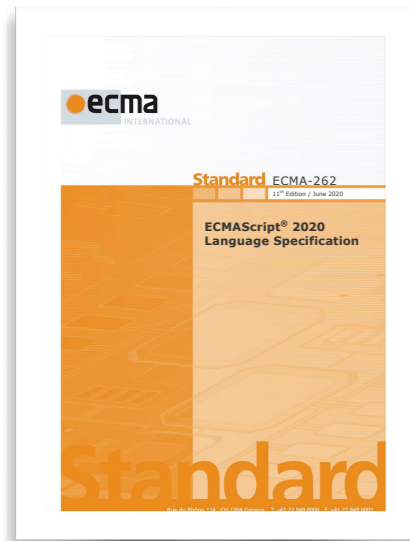
GraalVM™

QuickJS



JS 실행 엔진

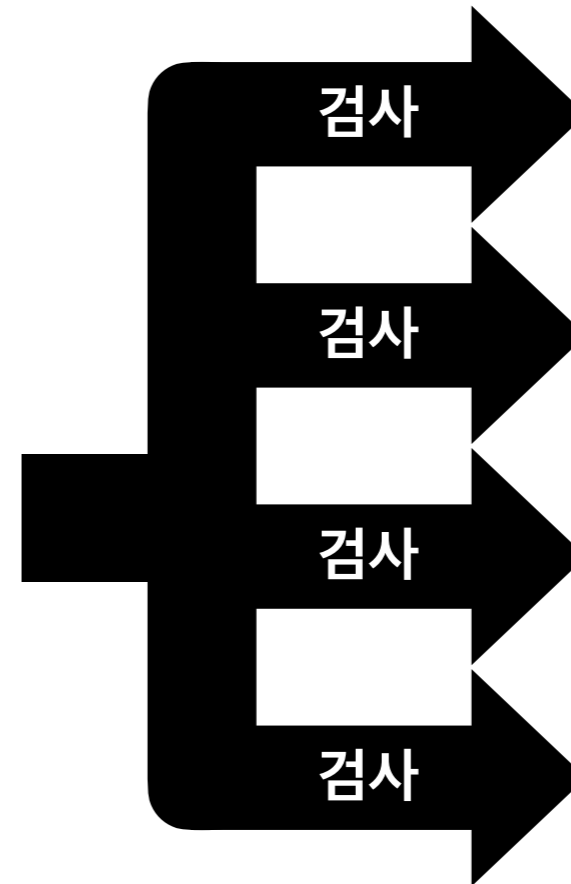
아이디어: N+1-버전 차분 테스트



ECMA-262
(JS 명세)



일치성 검사
프로그램



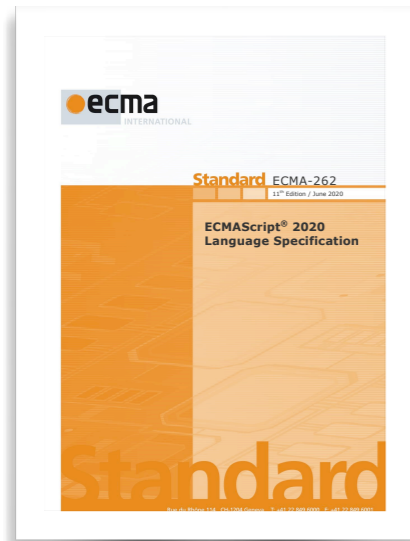
GraalVM™

QuickJS



JS 실행 엔진

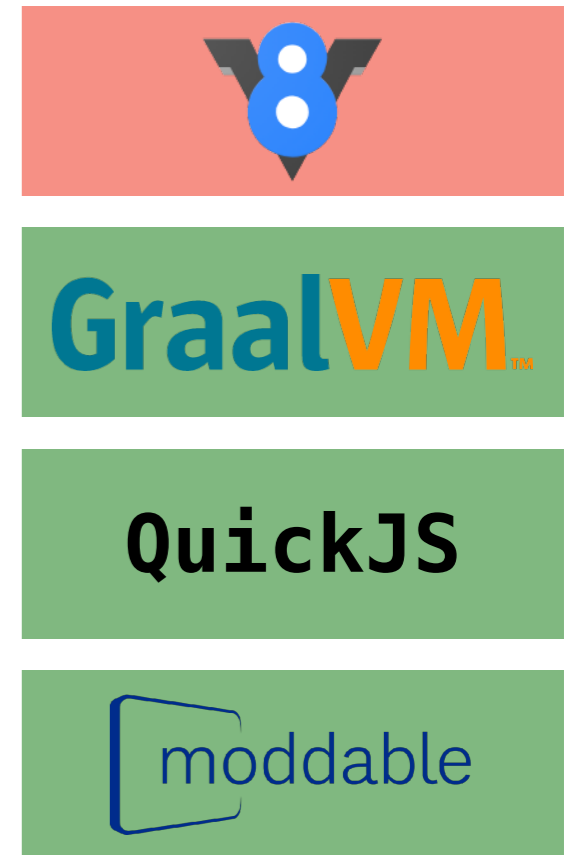
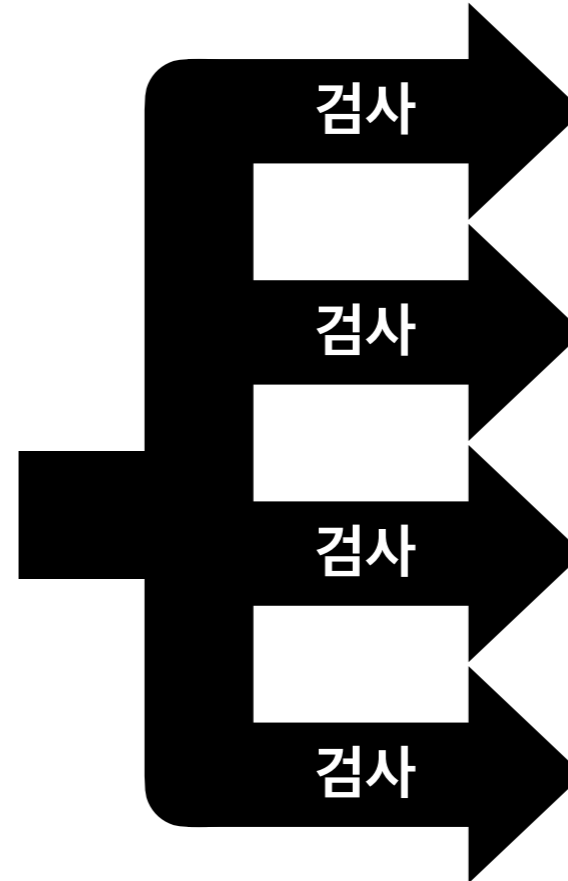
아이디어: N+1-버전 차분 테스트



ECMA-262
(JS 명세)

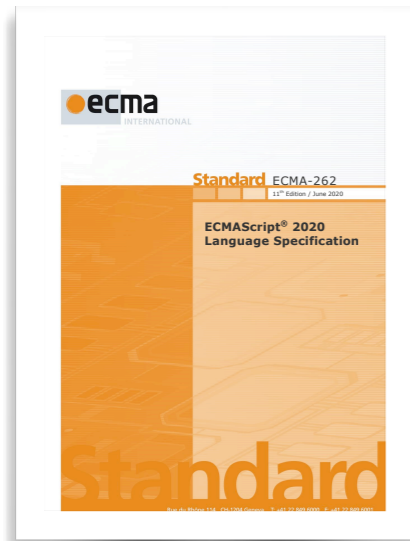


일치성 검사
프로그램



JS 실행 엔진

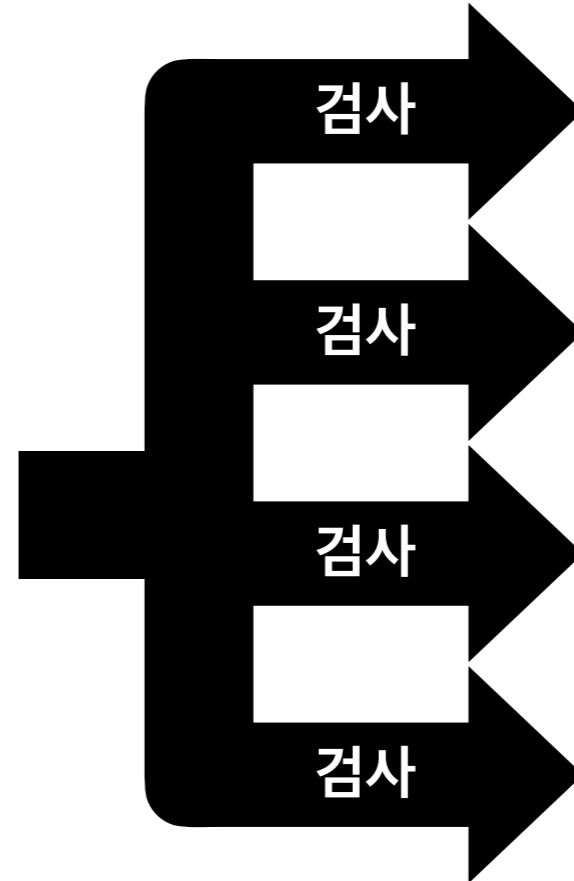
아이디어: N+1-버전 차분 테스트



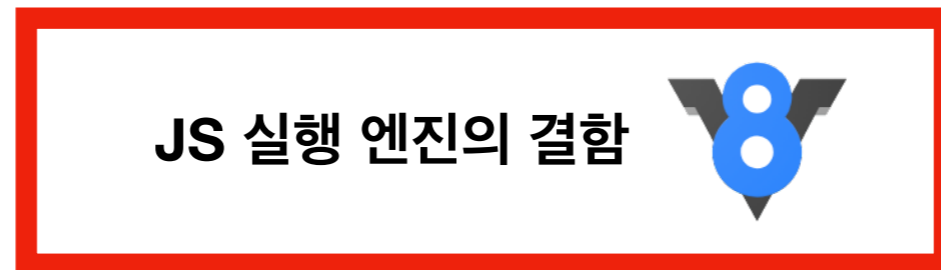
ECMA-262
(JS 명세)



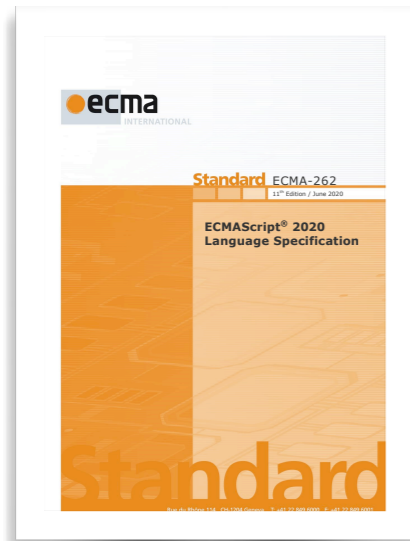
일치성 검사
프로그램



JS 실행 엔진



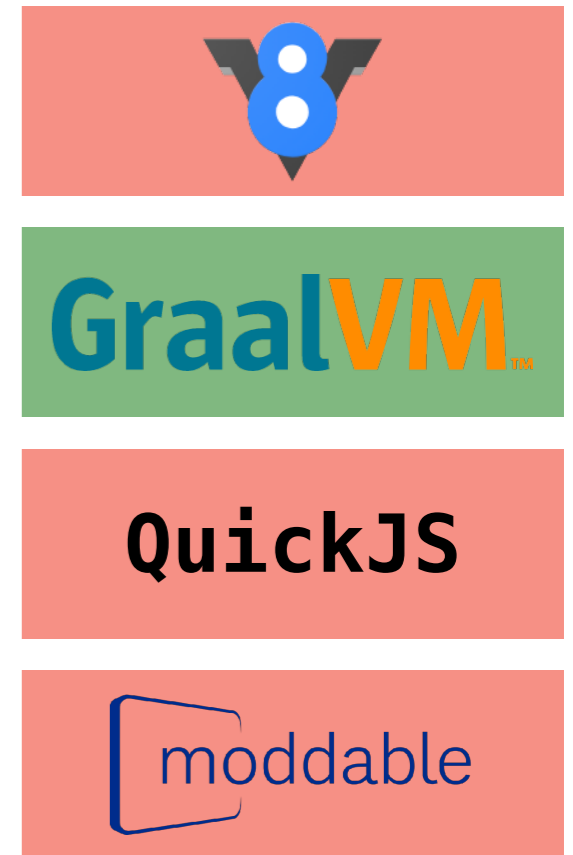
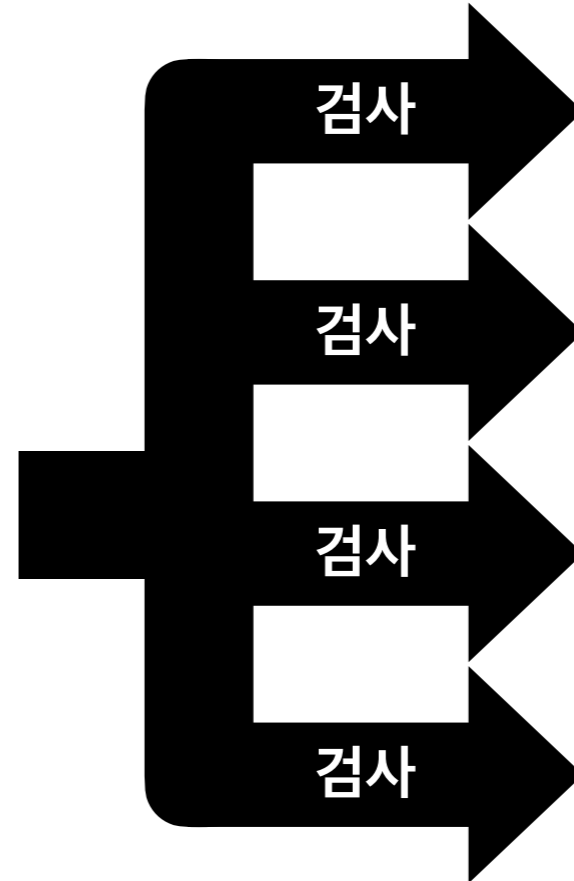
아이디어: N+1-버전 차분 테스트



ECMA-262
(JS 명세)

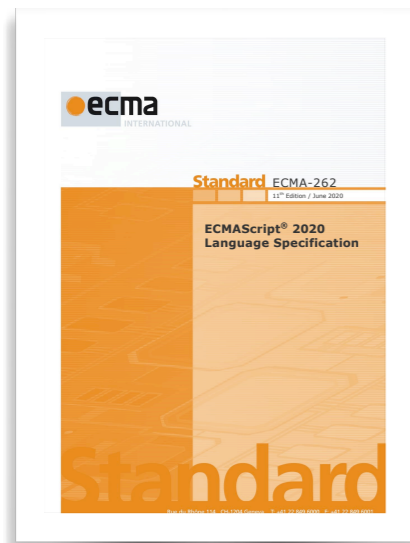


일치성 검사
프로그램



JS 실행 엔진

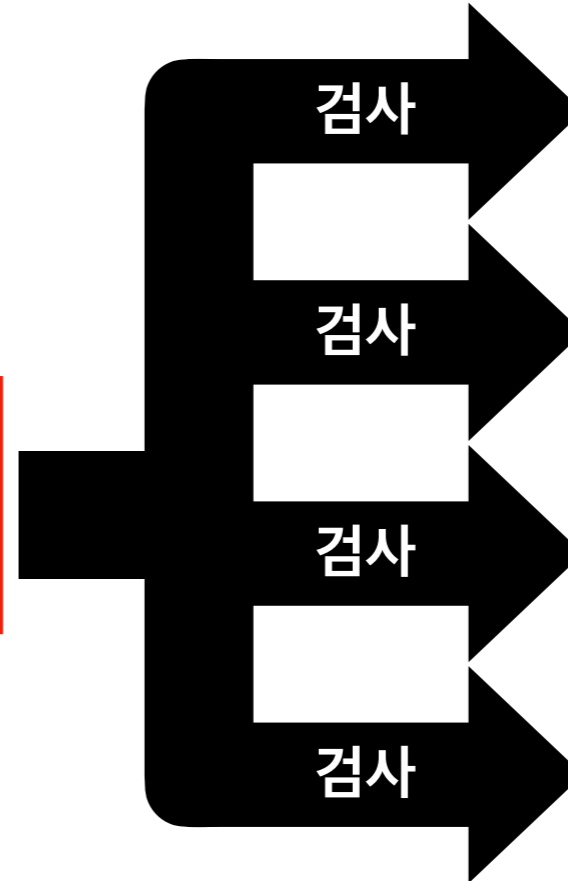
아이디어: N+1-버전 차분 테스트



ECMA-262
(JS 명세)

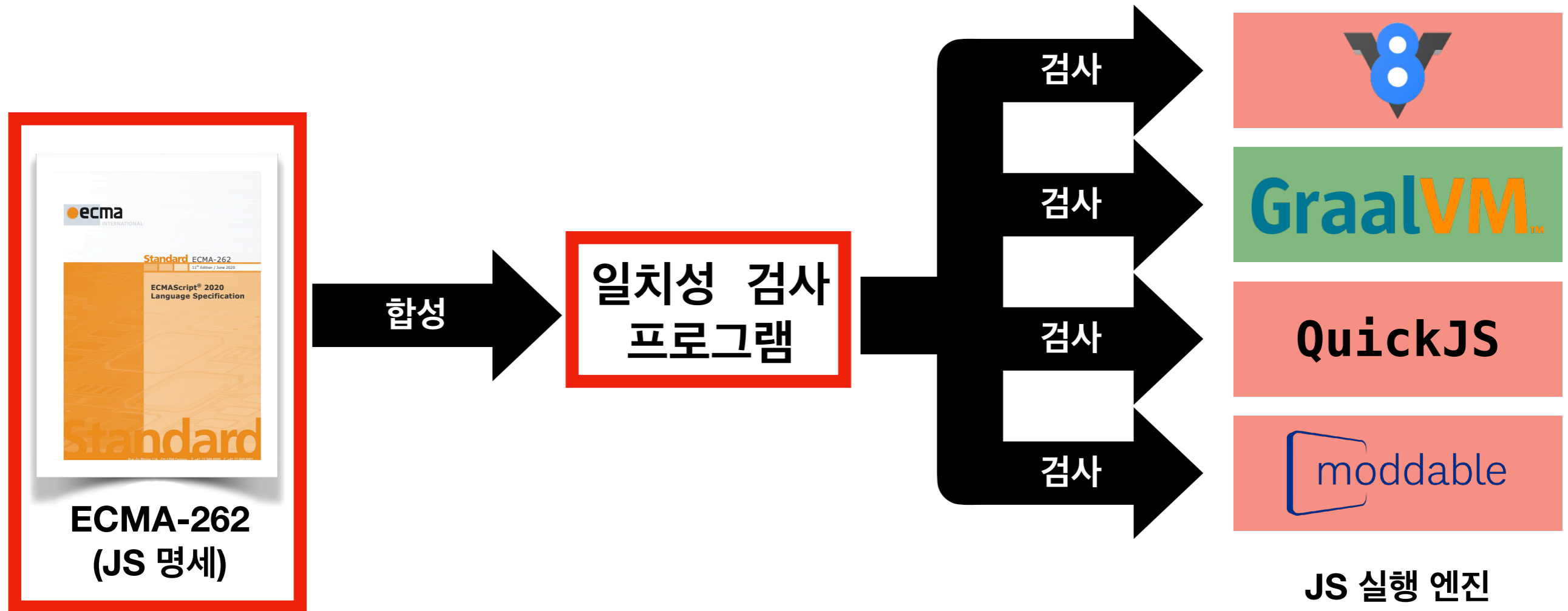


일치성 검사
프로그램

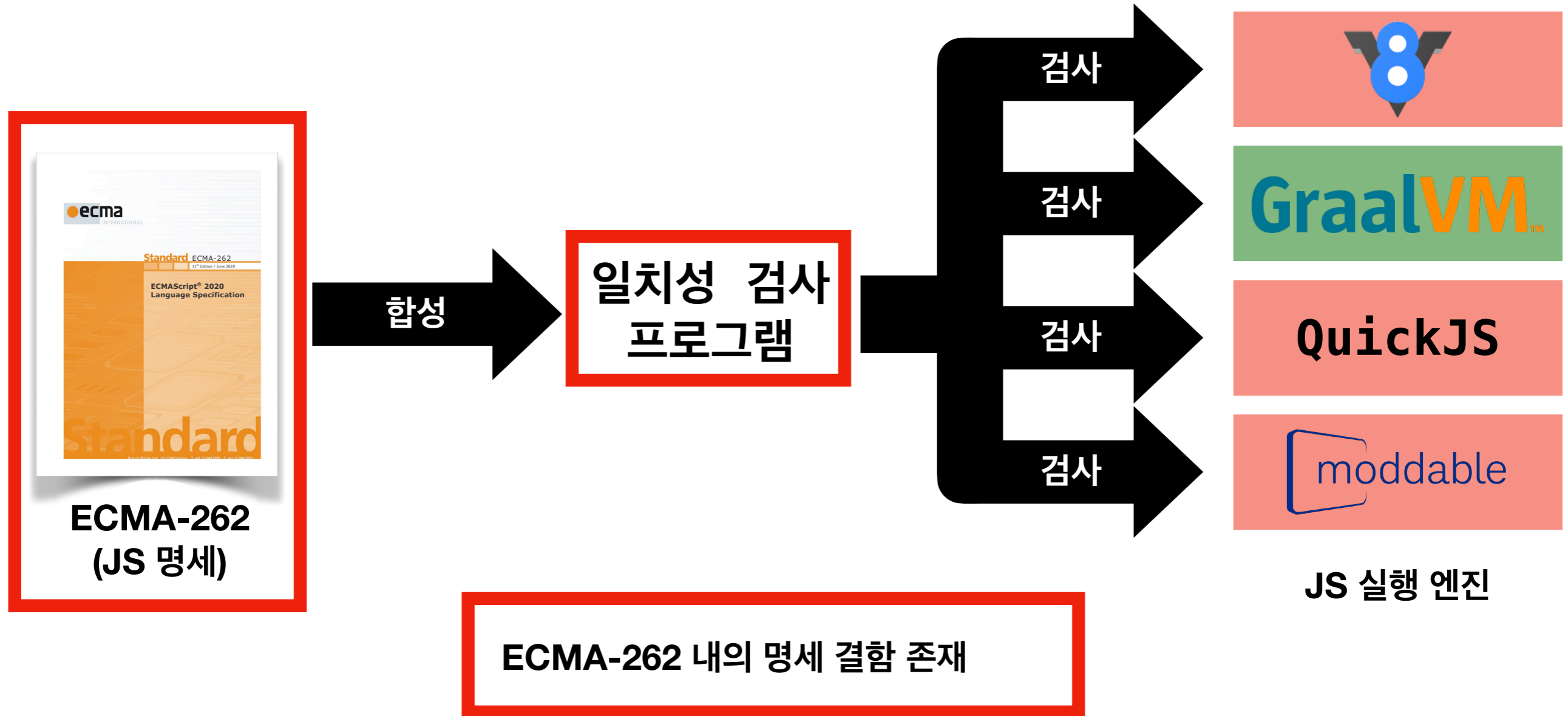


JS 실행 엔진

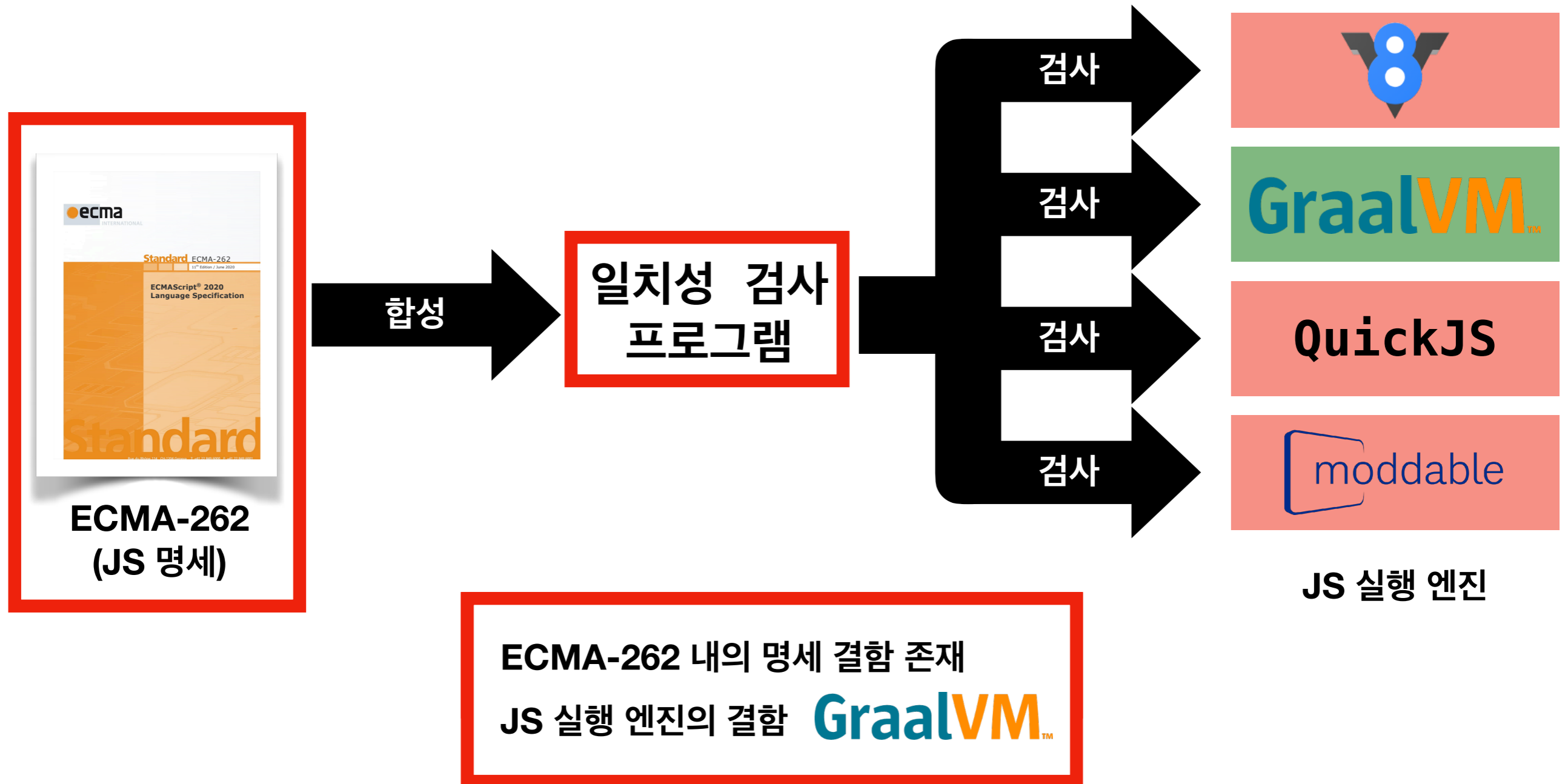
아이디어: N+1-버전 차분 테스트



아이디어: N+1-버전 차분 테스트



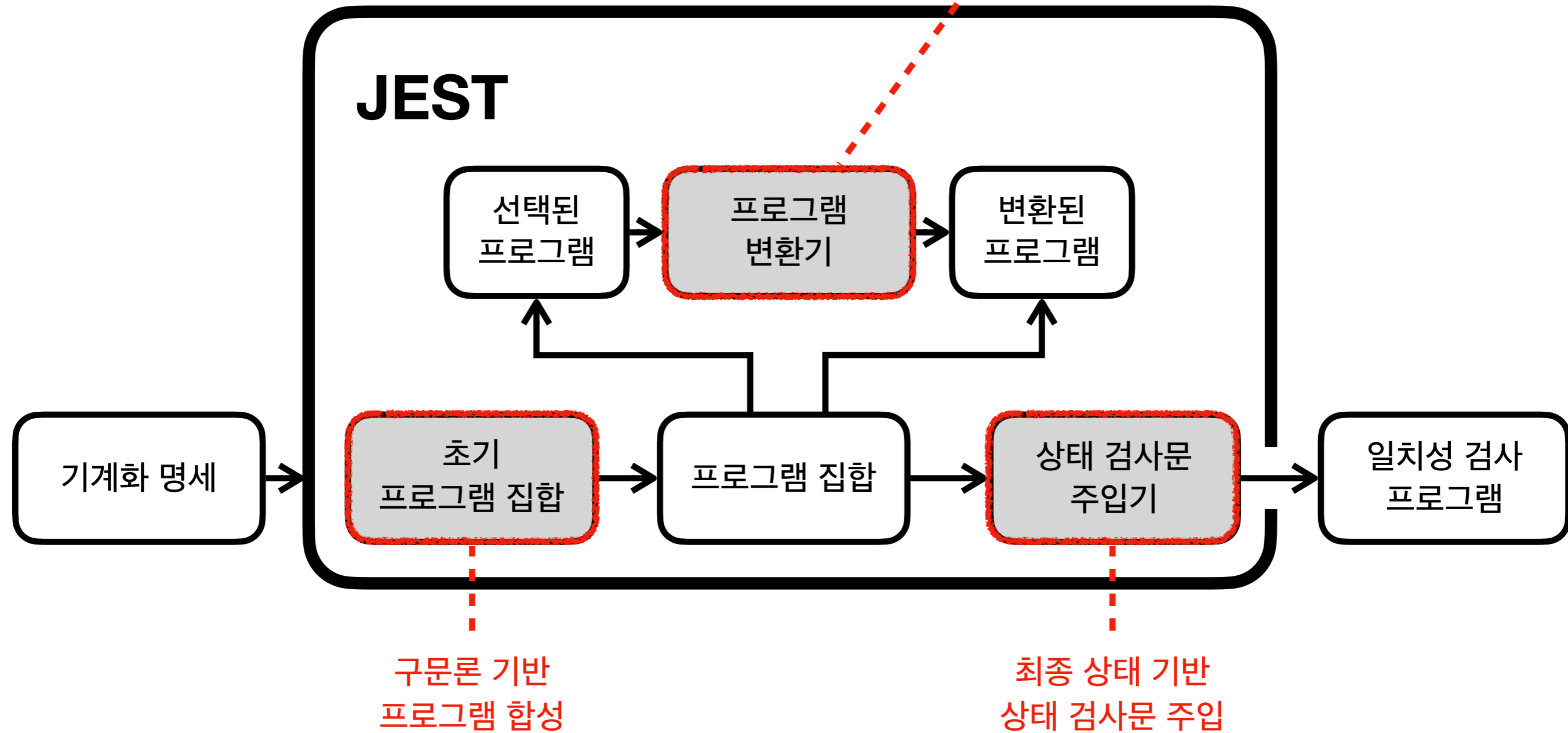
아이디어: N+1-버전 차분 테스트



JEST - ICSE'21

(JavaScript Engines and Specification Tester)

명세의 커버리지
기반 퍼징



JEST - 명세의 커버리지 기반 퍼징

```
0 + { valueOf() { return 1; } }
```

7.1.3 ToNumeric (*value*)

1. Let *primValue* be ? ToPrimitive(*value*, number)
2. If `Type(primValue)` is `BigInt`, return *primValue*.
3. Return ? `ToNumber(primValue)`.

JEST - 명세의 커버리지 기반 퍼징

```
0 + { valueOf() { return 1; } }
```

7.1.3 ToNumeric (*value*)

1. Let *primValue* be ? ToPrimitive(*value*, number)
2. If `Type(primValue)` is `BigInt`, return *primValue*.
3. Return ? `ToNumber(primValue)`.

```
0 + { valueOf() { throw 42; } }
```


JEST - 최종 상태 기반 상태 검사문 주입

```
function f() {}
```

JEST - 최종 상태 기반 상태 검사문 주입

```
function f() {}
```

```
+ $assert.sameValue(Object.getPrototypeOf(f),  
+     Function.prototype);  
+ $assert.sameValue(Object.isExtensible(x), true);  
+ $assert.callable(f);  
+ $assert.constructable(f);
```

JEST - 실험 결과

- JEST가 성공적으로 ES11에서 1,700 일치성 테스트를 합성

TABLE II: The number of engine bugs detected by JEST

Engines	Exc	Abort	Var	Obj	Desc	Key	In	Total
V8	0	0	0	0	0	2	0	2
GraalVM	6	0	0	0	2	8	0	16
QuickJS	3	0	1	0	0	2	0	6
Moddable XS	12	0	0	0	3	5	0	20
Total	21	0	1	0	5	17	0	44

실행 엔진
결함 44개

명세 결함
27개

TABLE III: Specification bugs in ECMAScript 2020 (ES11) detected by JEST

Name	Feature	#	Assertion	Known	Created	Resolved	Existed
ES11-1	Function	12	Key	O	2019-02-07	2020-04-11	429 days
ES11-2	Function	8	Key	O	2015-06-01	2020-04-11	1,776 days
ES11-3	Loop	1	Exc	O	2017-10-17	2020-04-30	926 days
ES11-4	Expression	4	Abort	O	2019-09-27	2020-04-23	209 days
ES11-5	Expression	1	Exc	O	2015-06-01	2020-04-28	1,793 days
ES11-6	Object	1	Exc	X	2019-02-07	2020-11-05	637 days

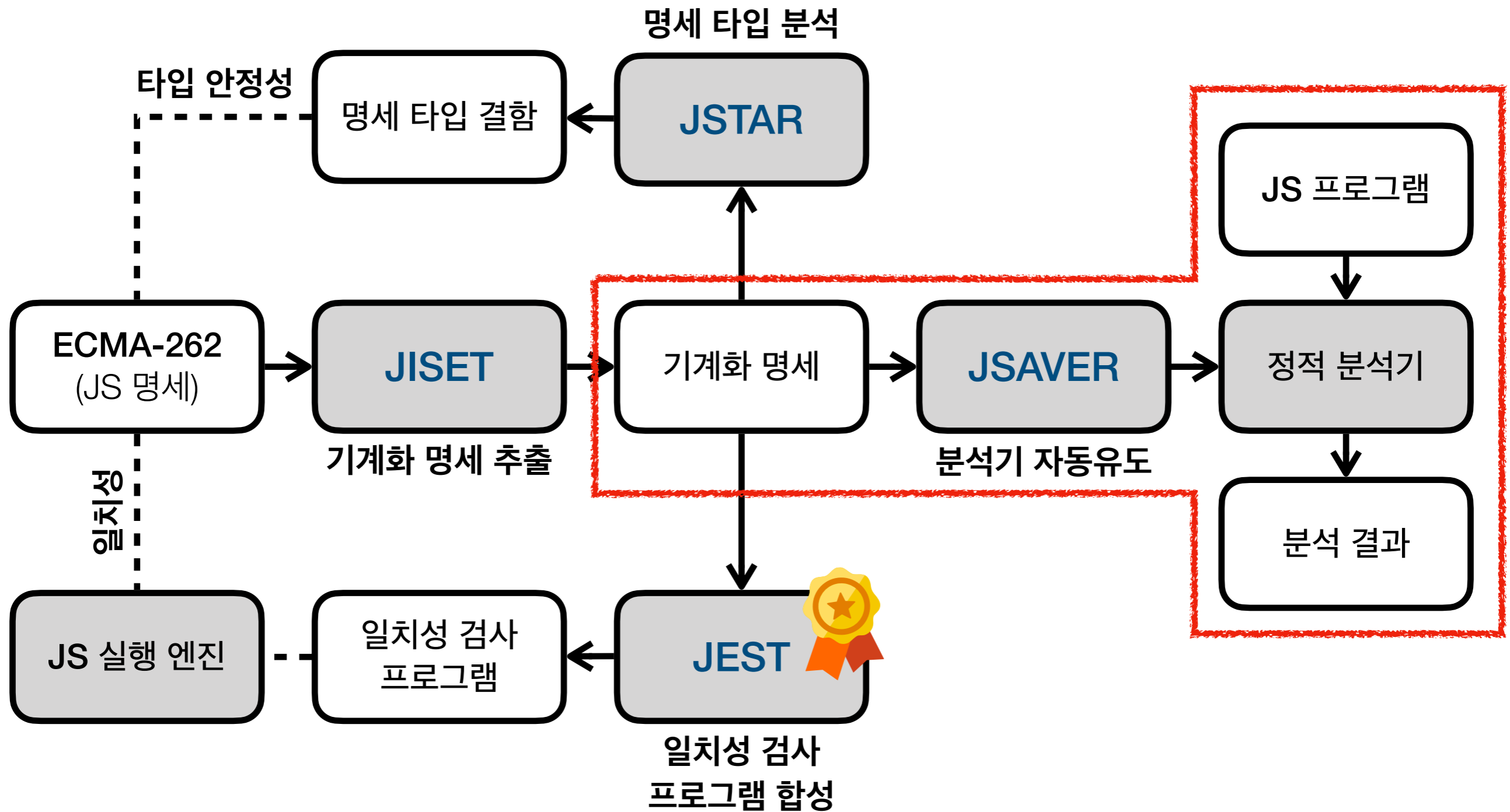
JEST - Example in GraalVM™



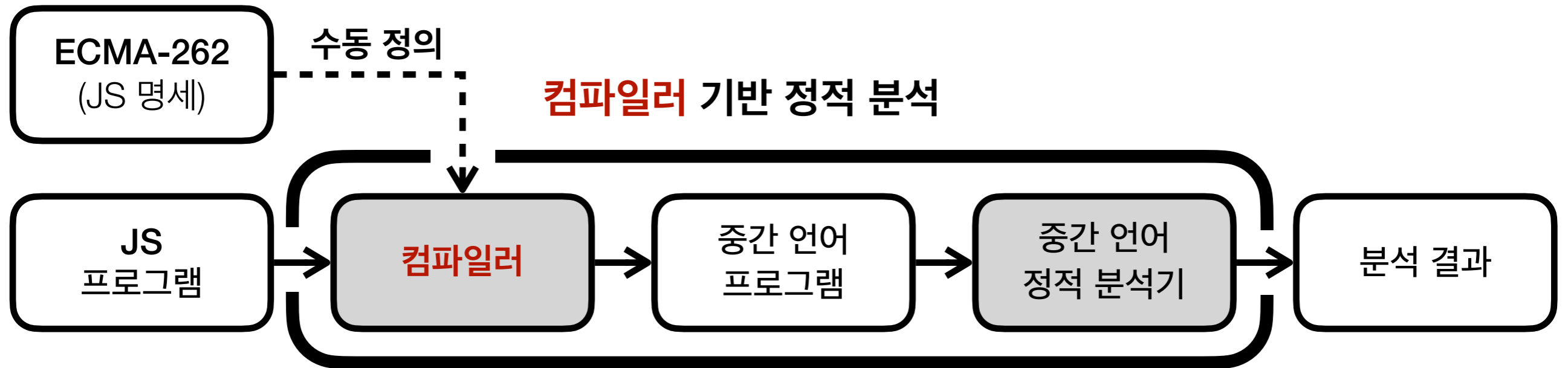
```
try { ++undefined; } catch (e) { }
```

*“Right now, we are running Test262 and the V8 and Nashorn unit test suites in our CI for every change, it might make sense to **add your suite as well.**”*

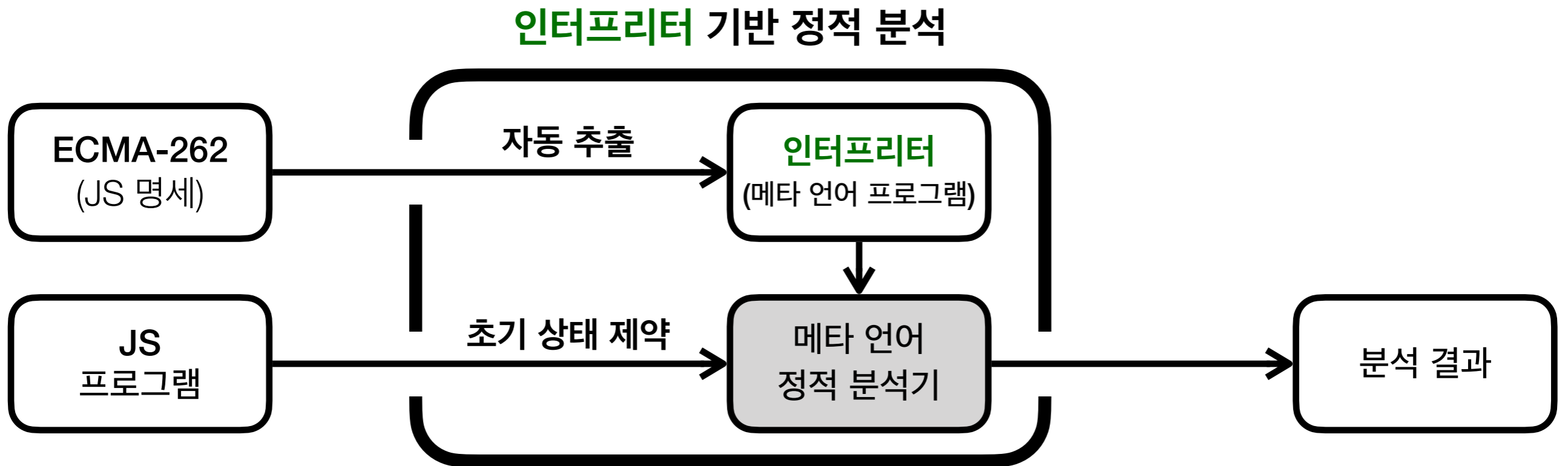
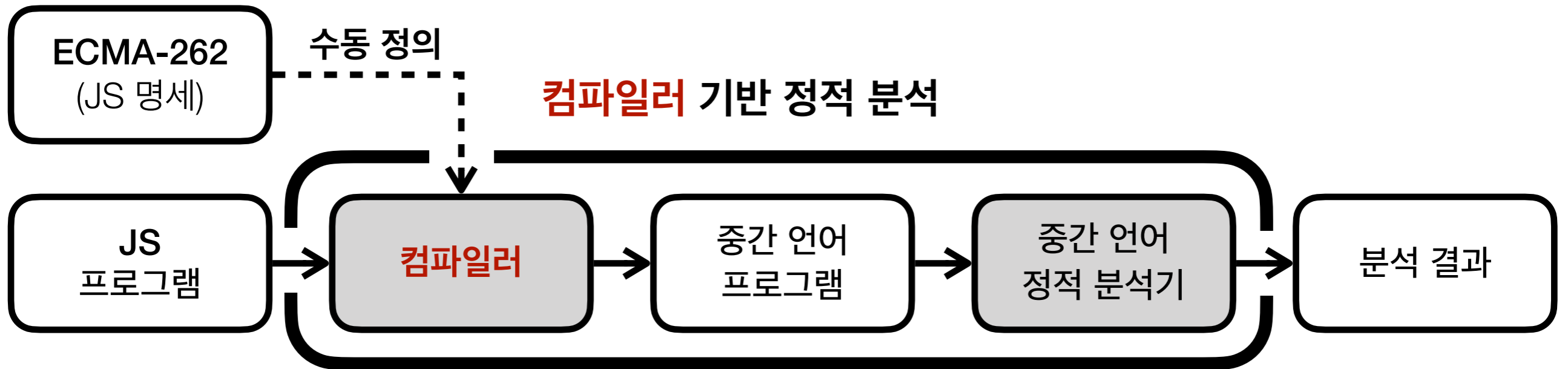
- A Developer of GraalVM



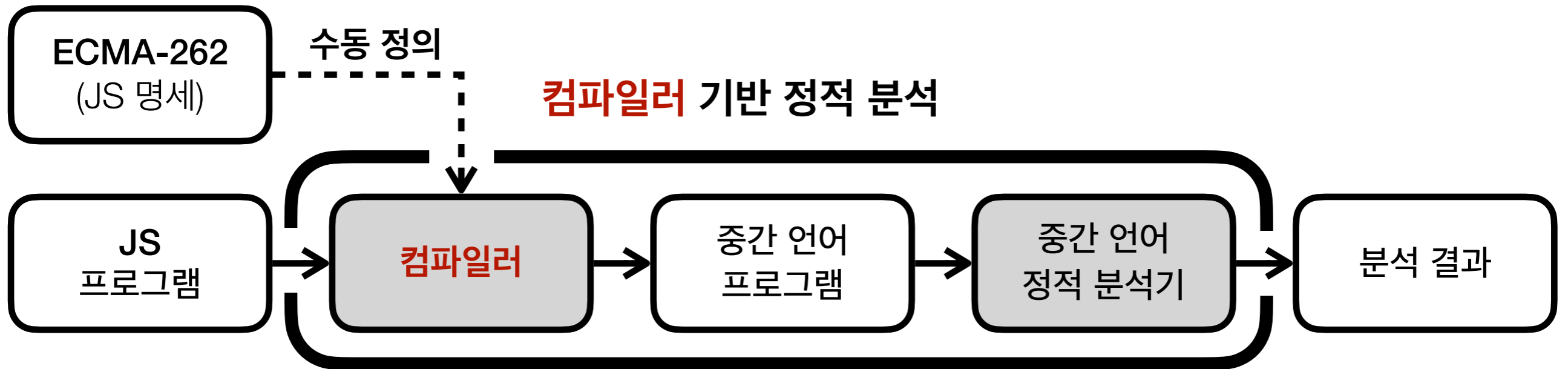
아이디어: 인터프리터 기반 정적 분석



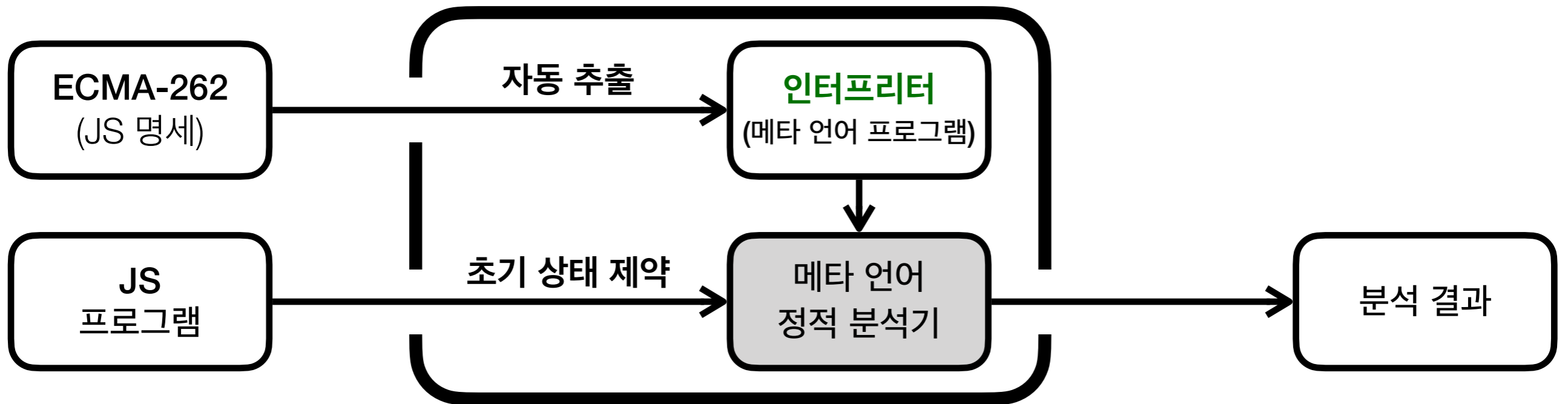
아이디어: 인터프리터 기반 정적 분석



아이디어: 인터프리터 기반 정적 분석

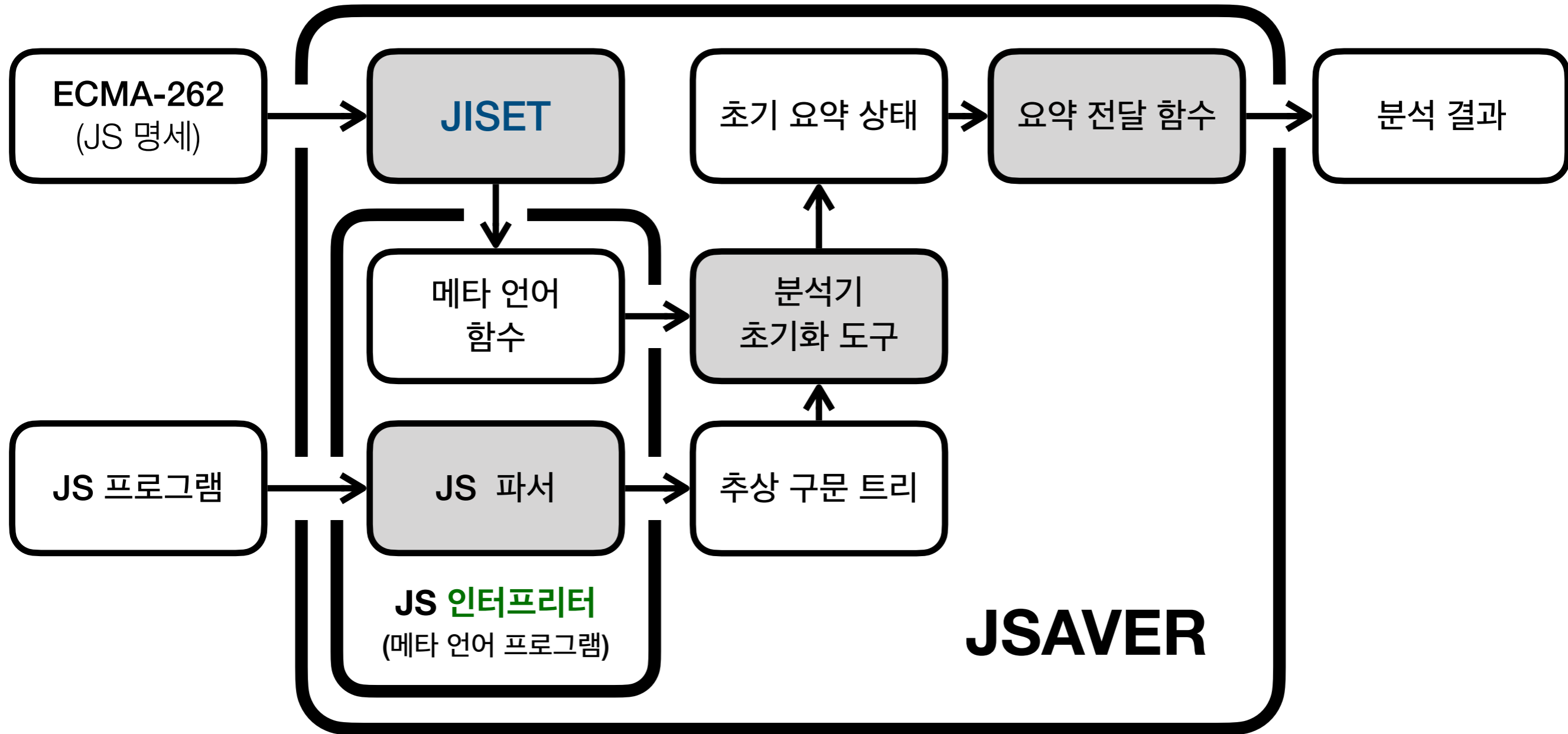


인터프리터 기반 정적 분석 = 메타 정적 분석

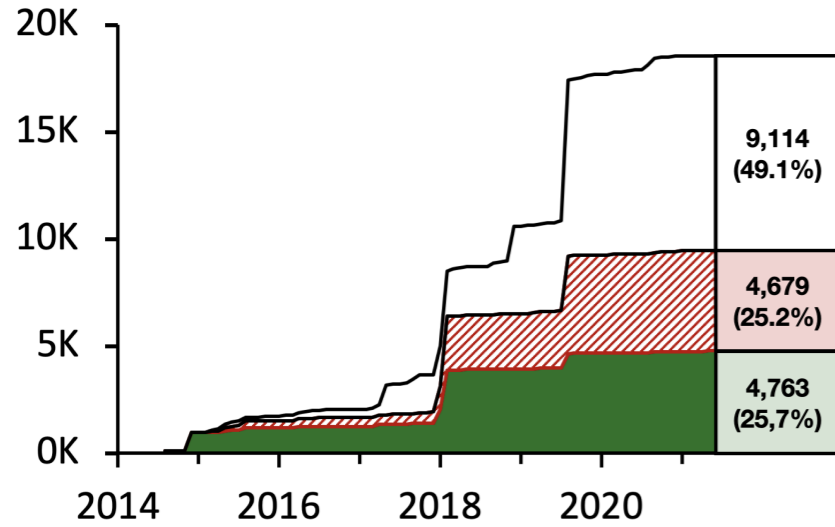


JSAVER - FSE'22

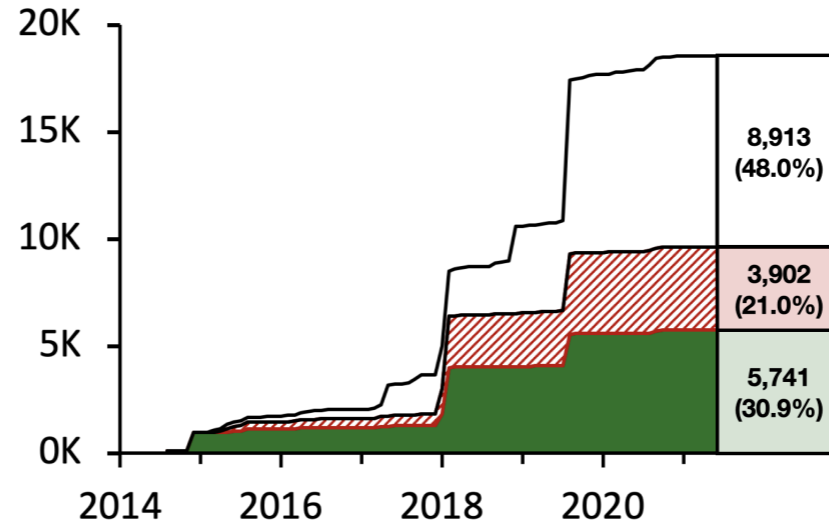
(JavaScript Static Analyzer via ECMAScript Representation)



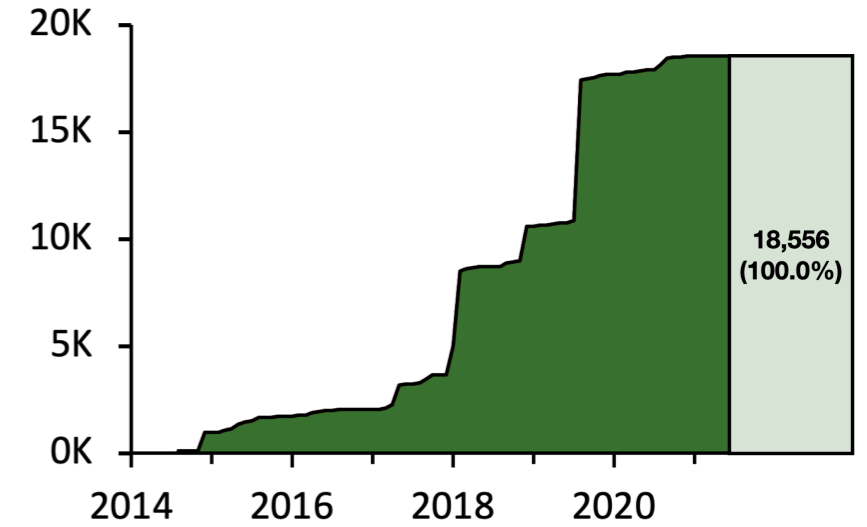
JSAVER - 평가 (RQ1: 올바른 분석)



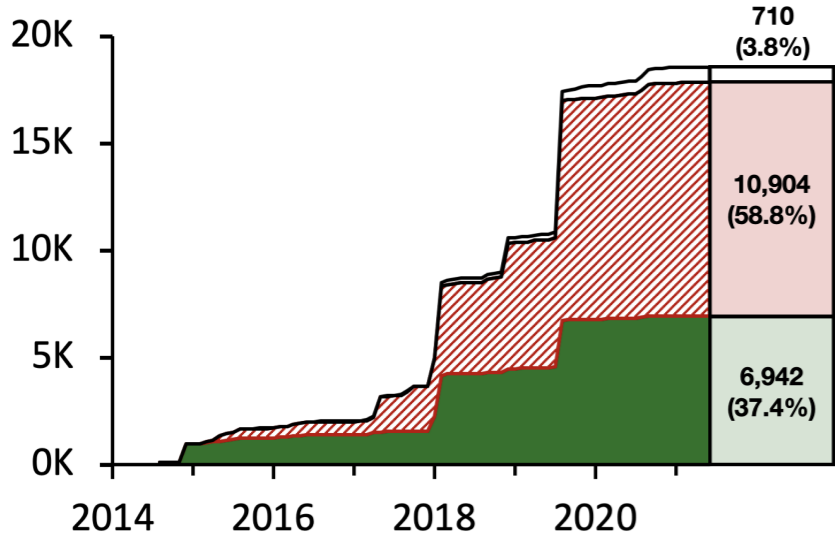
(a) Analysis results of TAJs



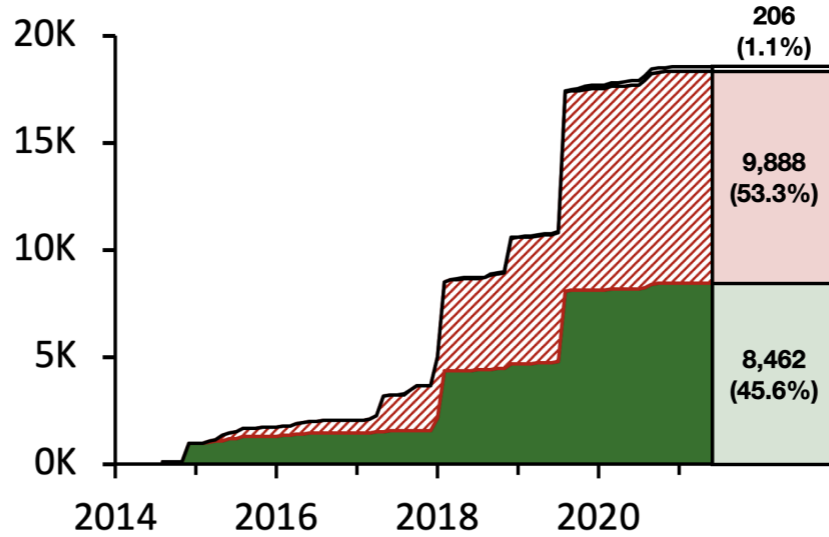
(b) Analysis results of SAFE



(c) Analysis results of JSA_{ES12}



(d) Analysis results of TAJs with Babel

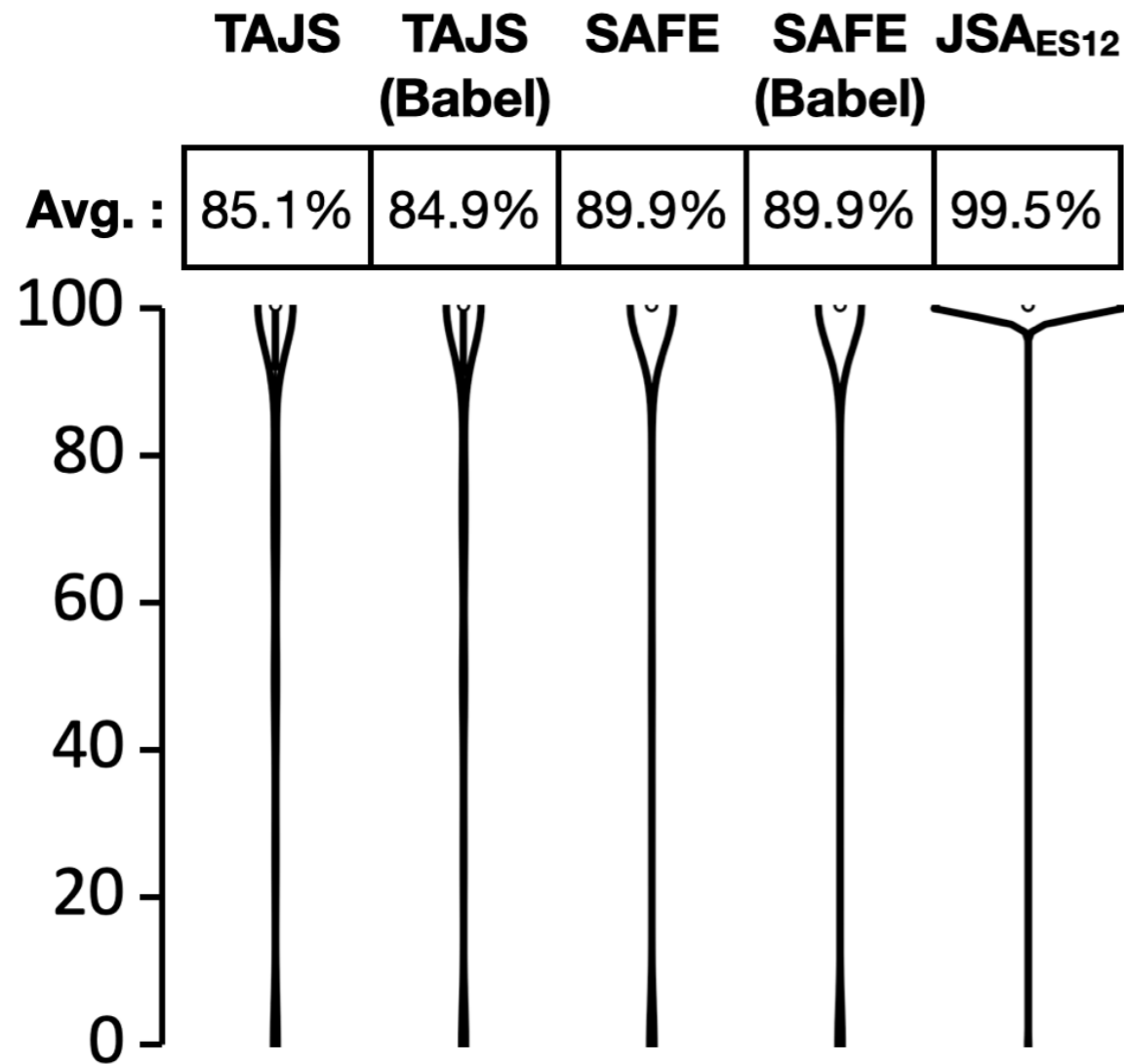


(e) Analysis results of SAFE with Babel

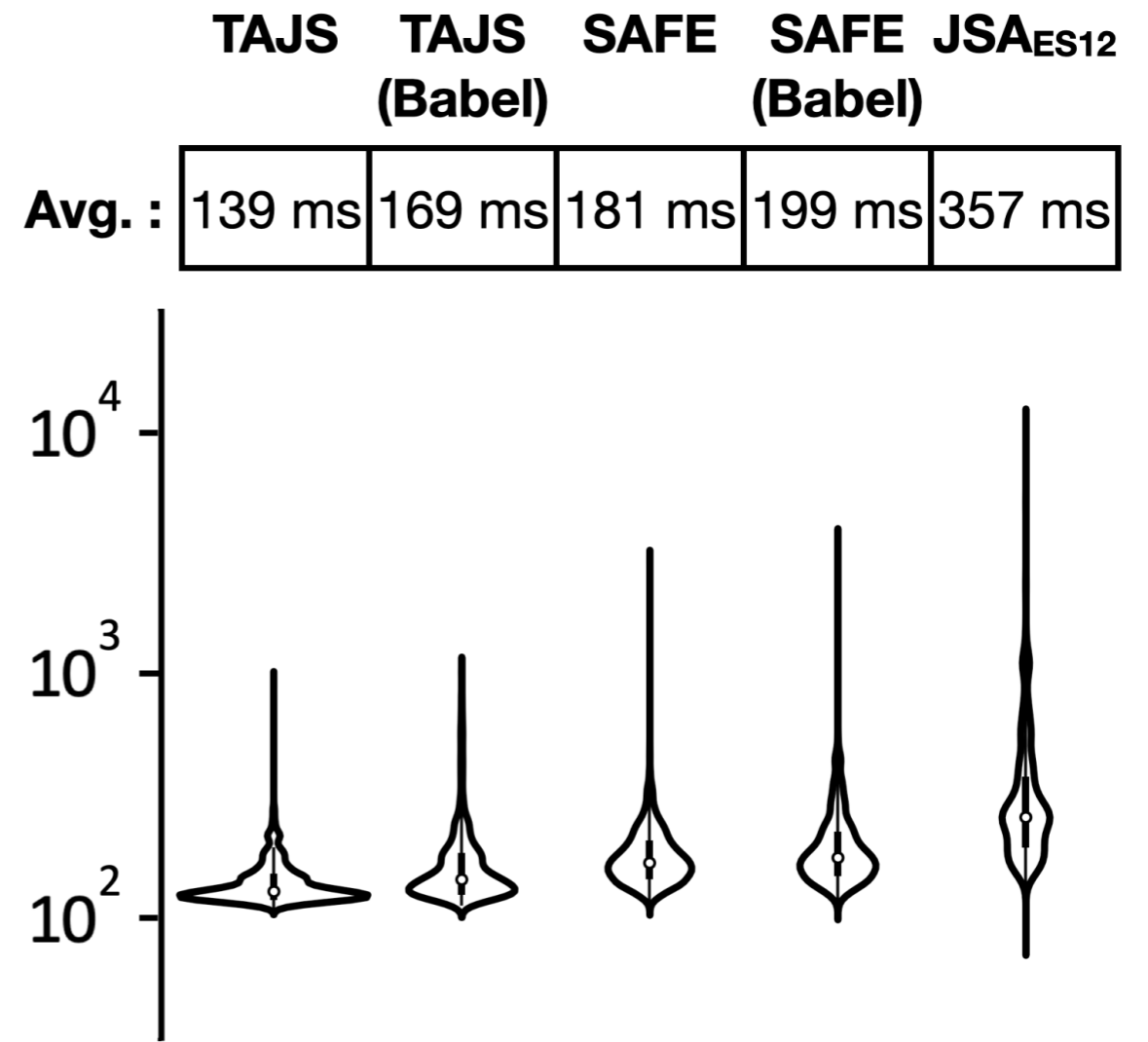
legend :
 □ error
 ▨ unsound
 ■ sound

x-axis : creation time (year)
 y-axis : # tests

JSAVER - 평가 (RQ2: 정확도 / 속도)



(a) The analysis precision

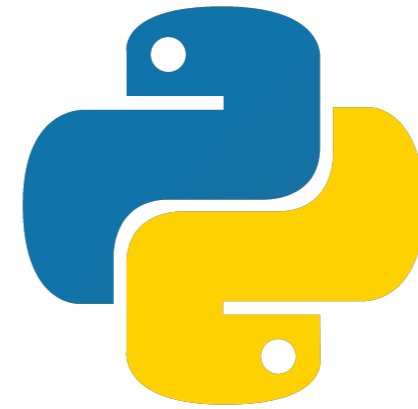


(b) The analysis performance

추후 연구 - 다른 프로그래밍 언어로 확장



WEBASSEMBLY



python™



Rust



EXPLORING VERSE
A New Metaverse
Programming
Language from
Epic Games

