KOREA UNIVERSITY

기계화 명세를 이용한 자바스크립트 언어의 설계와 구현

컴퓨터학과 2025년 봄학기 콜로퀴움

박지혁

고려대학교 컴퓨터학과
프로그래밍 언어 연구실

2025. 05. 14

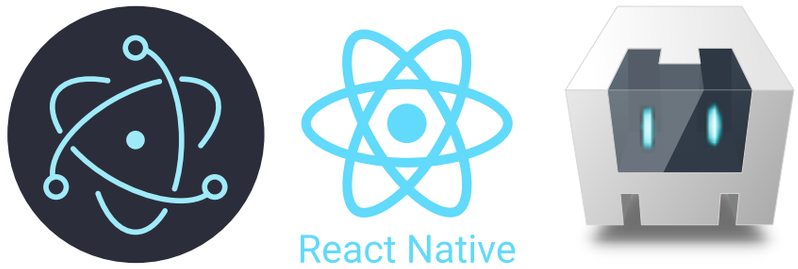
프로그래밍 언어 연구실 PLRG

- 연구 분야: **프로그래밍 언어** 및 **소프트웨어 공학**
  - 프로그램 분석(Program Analysis)
  - 기계화 언어 명세(Mechanized Language Specification)
  - 테스트 자동화(Automated Testing)
  - 프로그램 합성 및 수정(Program Synthesis & Repair)

- 주요 실적
  - PLDI (2023, 2024)
  - ICSE (2017-Demo, 2021)
  - FSE (2021, 2022, 2025-Demo)
  - ASE (2020, 2021)
  - CACM (2024), CSUR (2021)

어디에나 있는 JavaScript

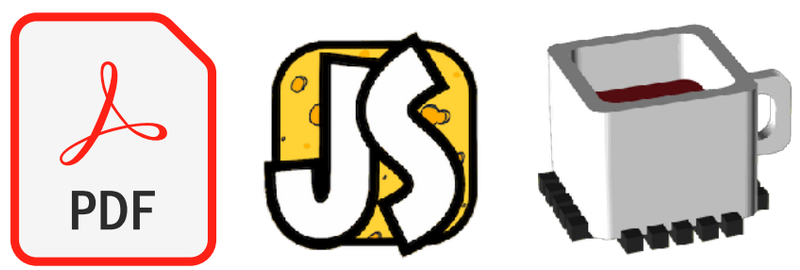
클라이언트-사이드 프로그래밍



모바일/데스크톱 어플리케이션

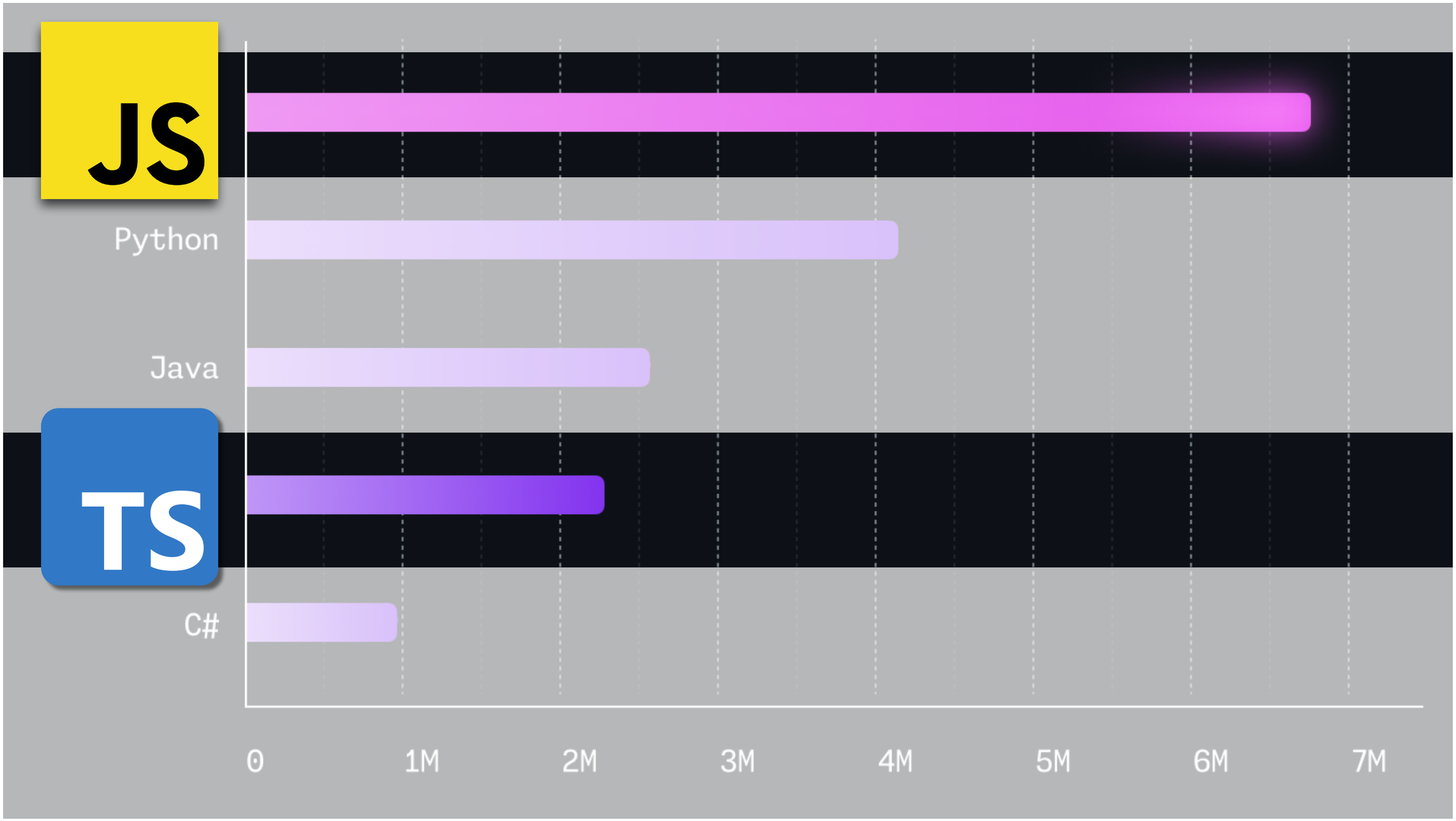
JS

서버-사이드 프로그래밍



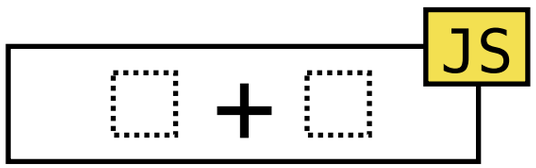
그 외 (PDF, 사물 인터넷, 마이크로컨트롤러, etc.)

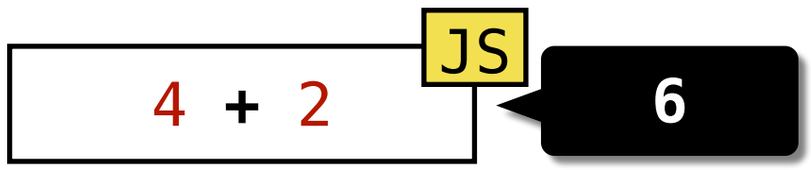
어디에나 있는 JavaScript

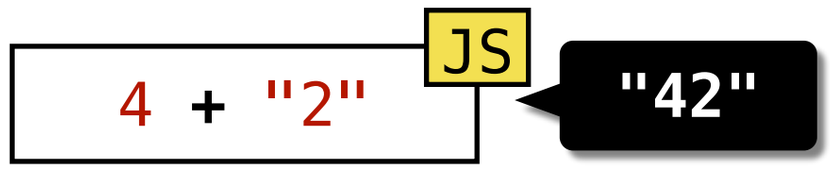


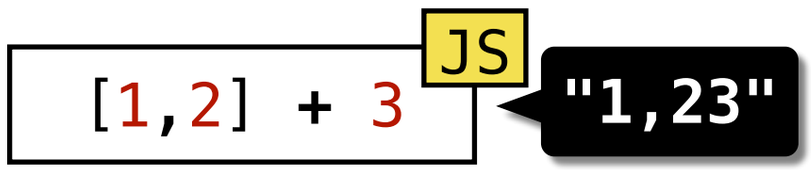
https://github.blog/news-insights/octoverse/octoverse-2024/

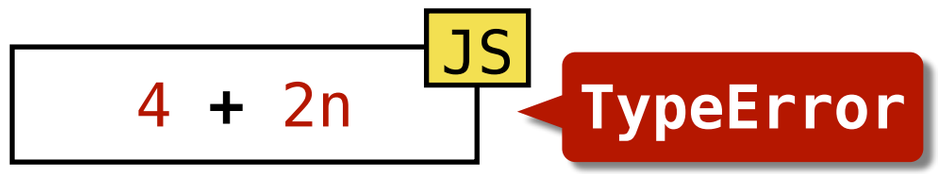
하지만, 복잡한 JavaScript












```
(![]+[])[+[]]                    +       // "f"
(![]+[])[+!+[]]                  +       // "a"
([![]]+[][[]])[+!+[]+[+[]]] +            // "i"
(![]+[])[!+[]+!+[]]                      // "l"
```

JS

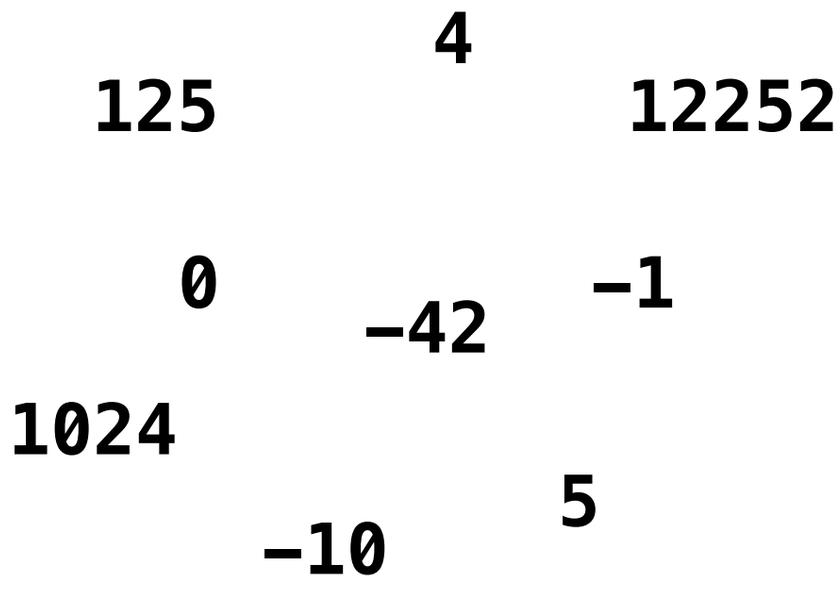
"fail"

정적 분석 (Static Analysis)

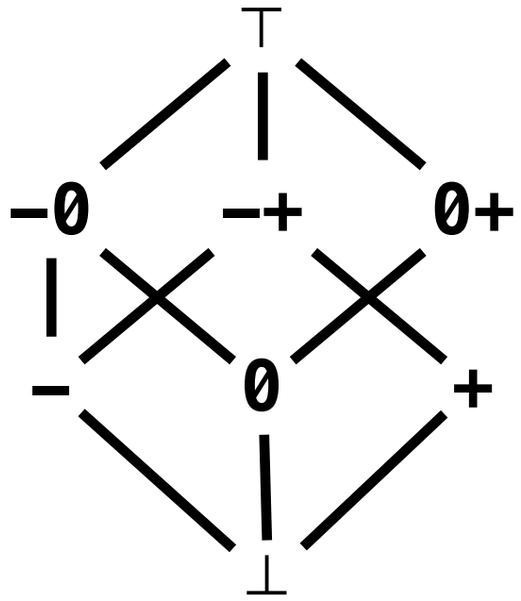
- 주어진 프로그램을 **실행하지 않고 (정적으로)** 분석하는 기법
- 프로그램의 행동을 **요약해서 해석**

  예를 들어, 정수를 요약해보자

```
{ }                  -> ⊥
{ 4, 5 }             -> +
{ -1, -42, -10 }     -> -
{ 0, 125 }           -> 0+
{ -1, 0, 5 }         -> ⊤
...
```



요약



정적 분석 (Static Analysis)

```
function f(x) {
  // x == ⊤
  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**

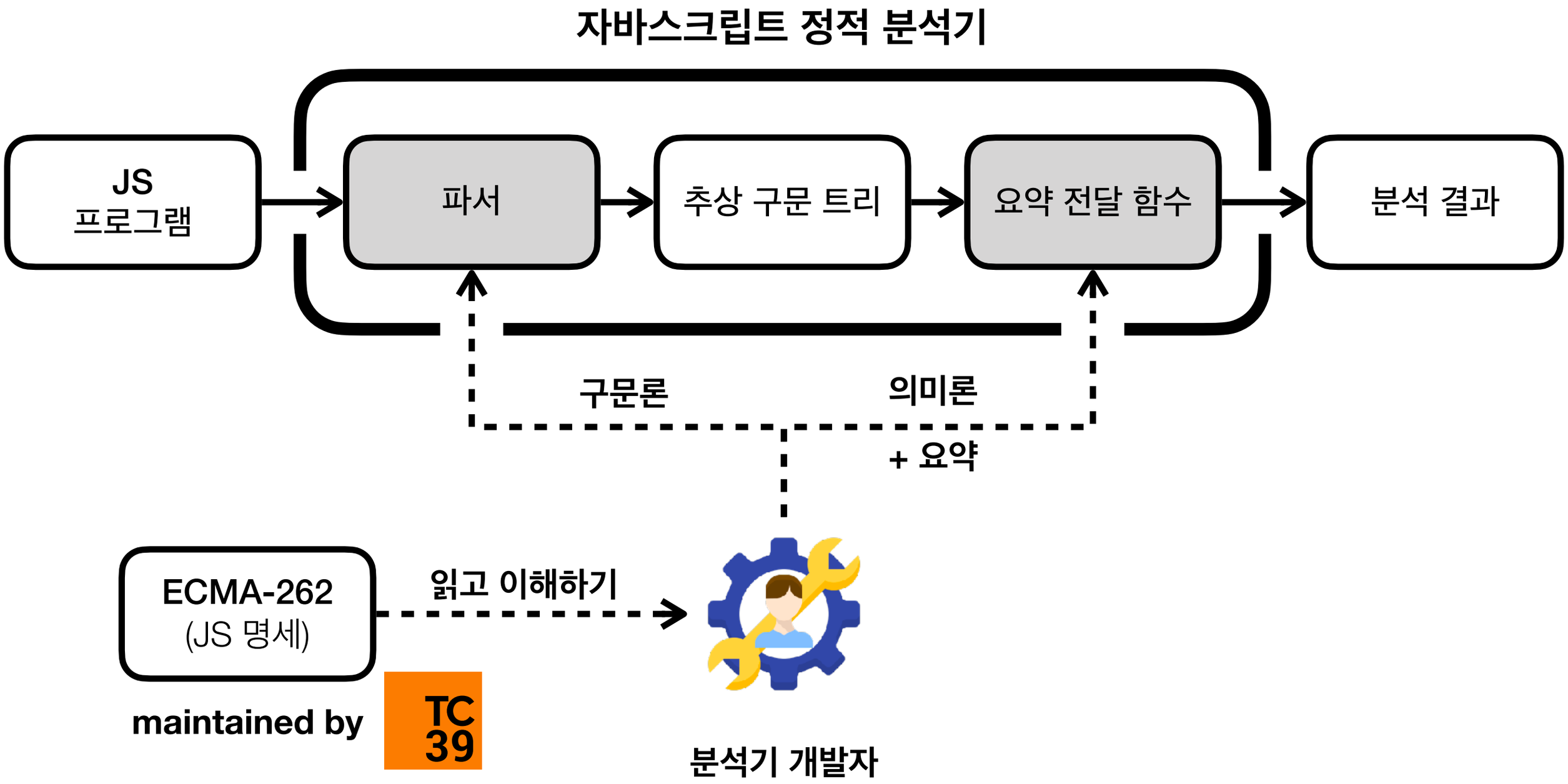
f(5) = 5 **. . .**

요약 해석

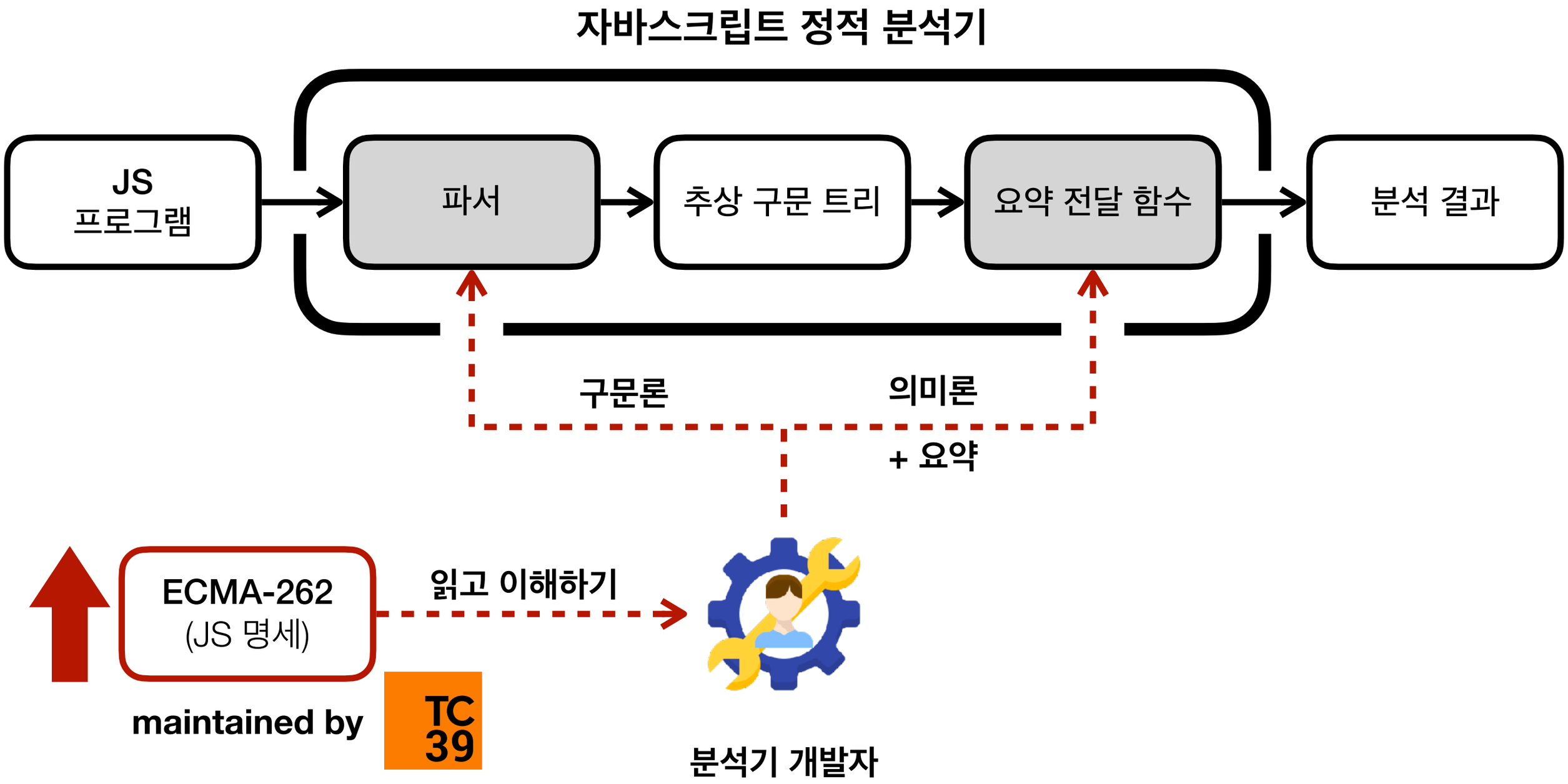
f(⊤) = 0+

타입 추론 (분석) ⊆

JavaScript를 위한 정적 분석기



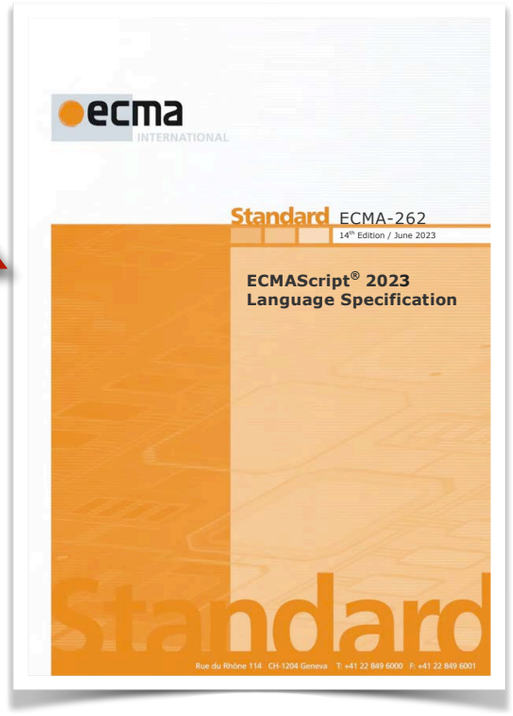
문제: 수동으로 분석기 갱신



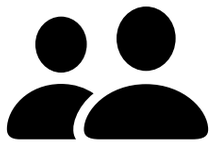
JavaScript 언어의 설계 및 구현

수동

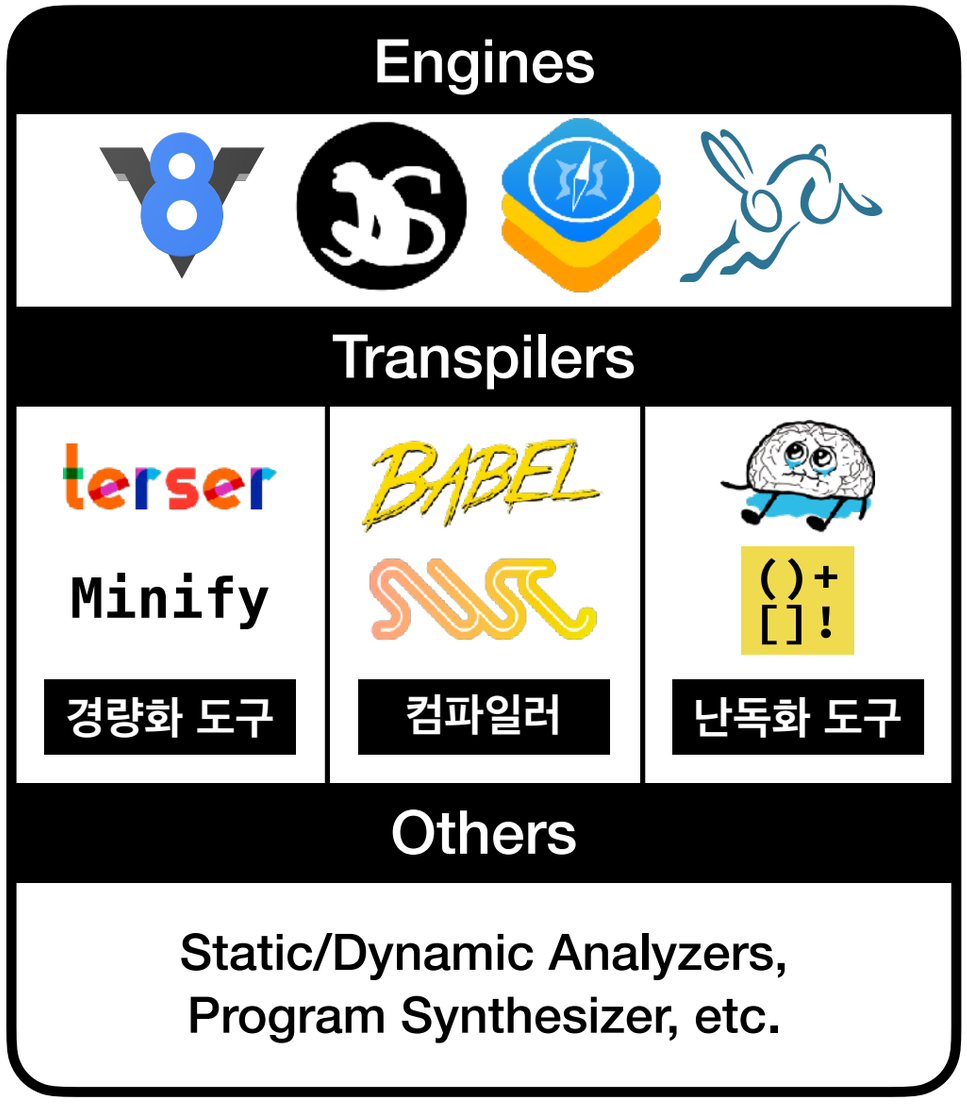
갱신



ECMA-262
(자바스크립트 명세)



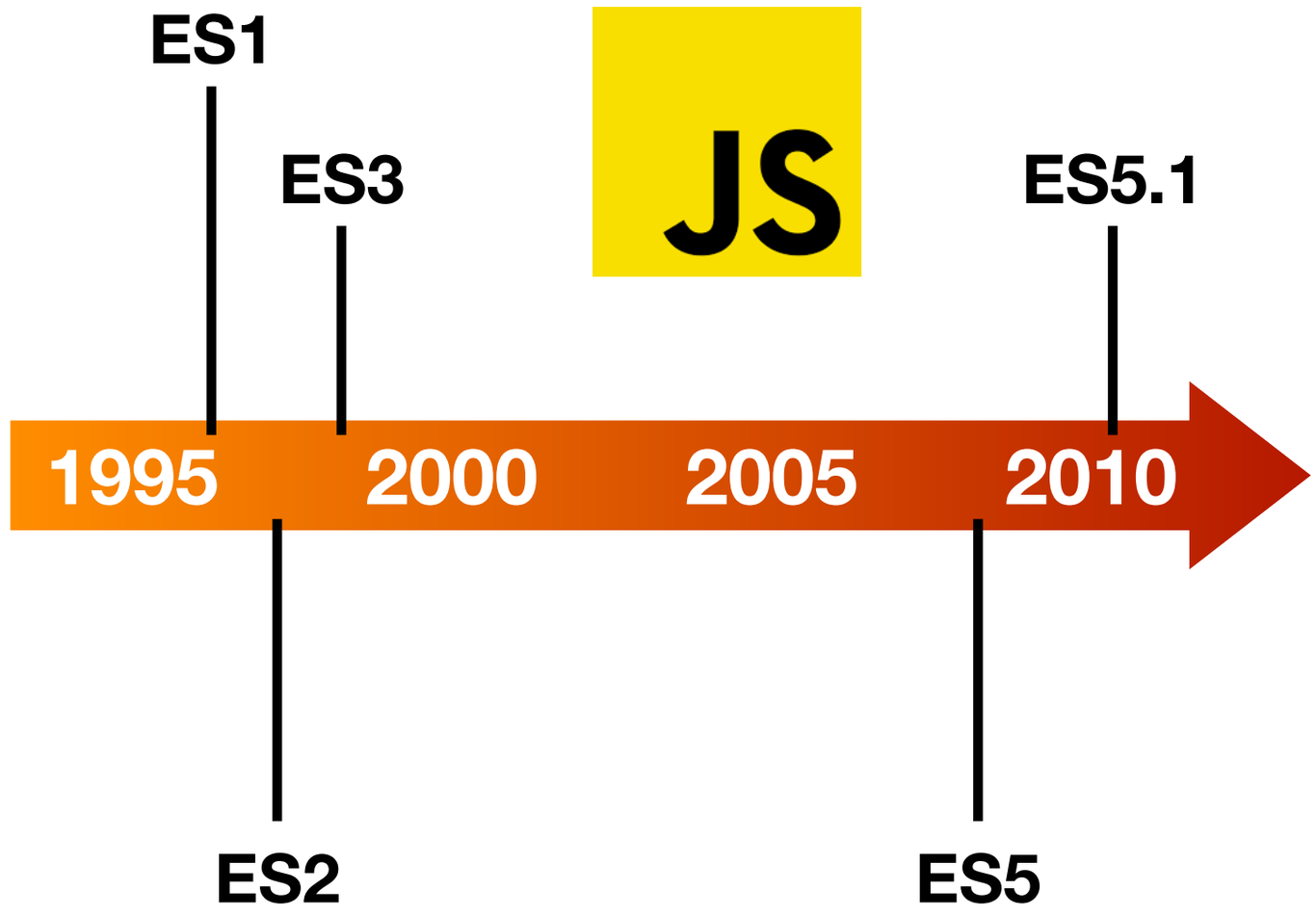
일치성 검사



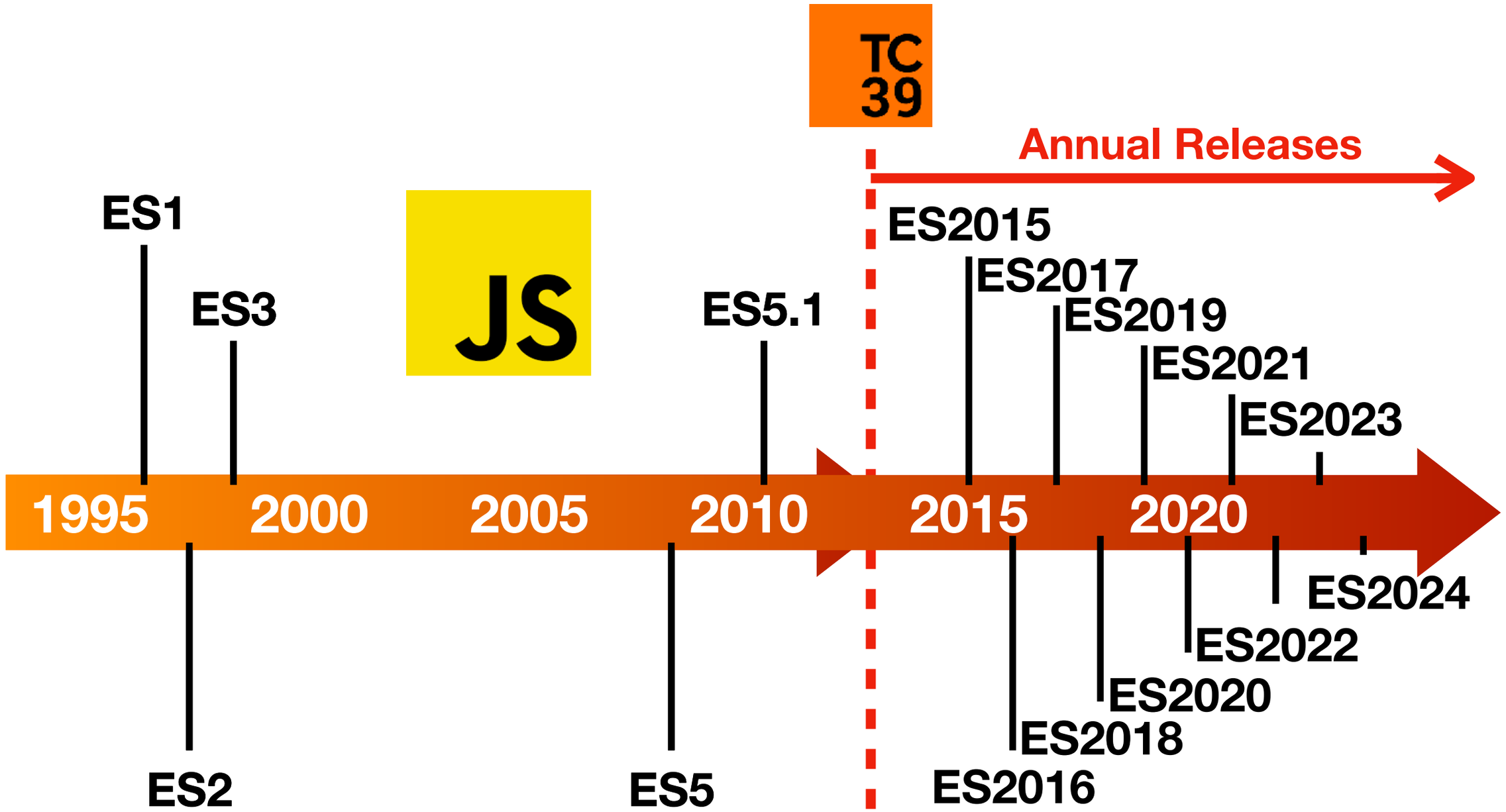
JavaScript 구현체

갱신

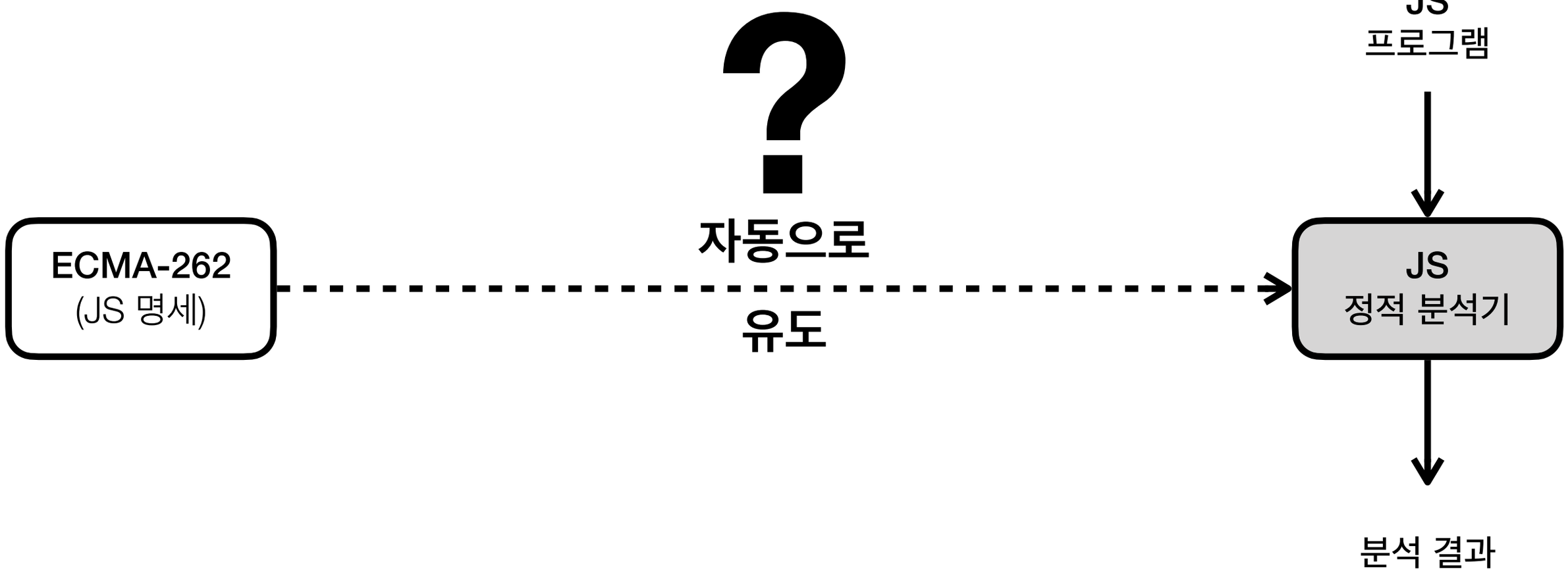
문제 - 빠르게 성장하는 JavaScript



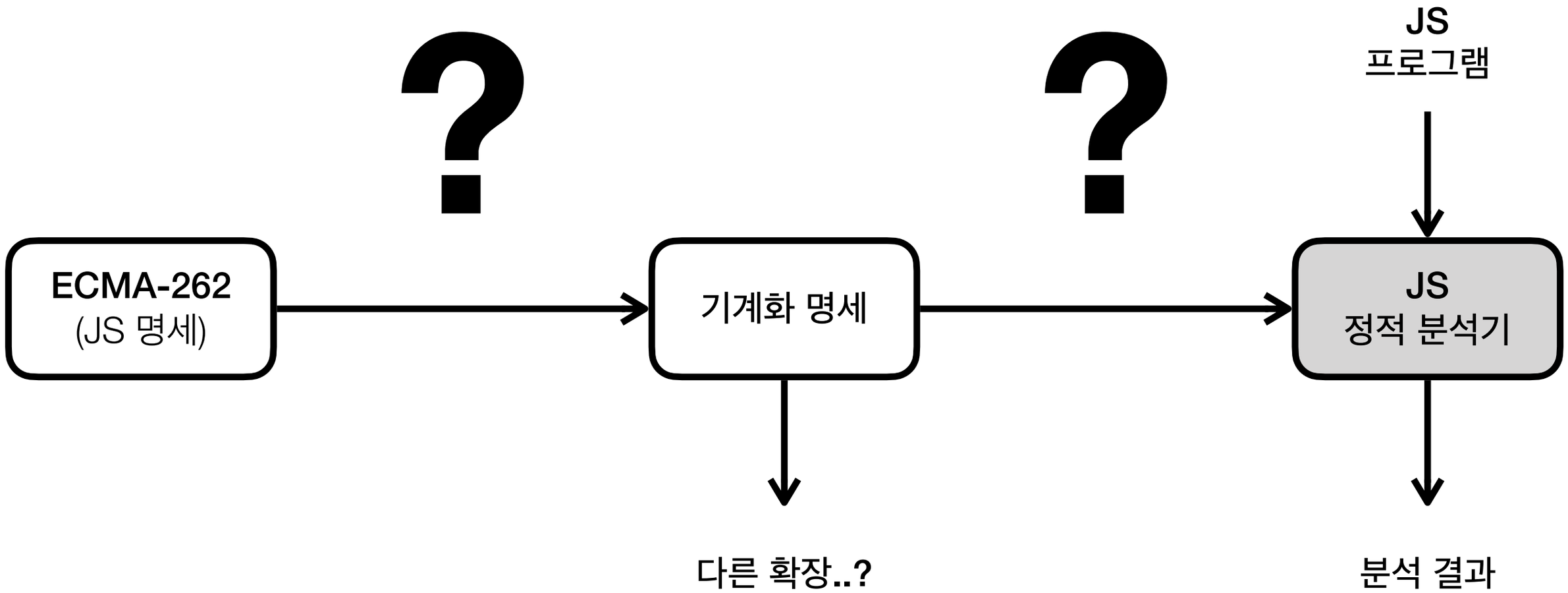
문제 - 빠르게 성장하는 JavaScript

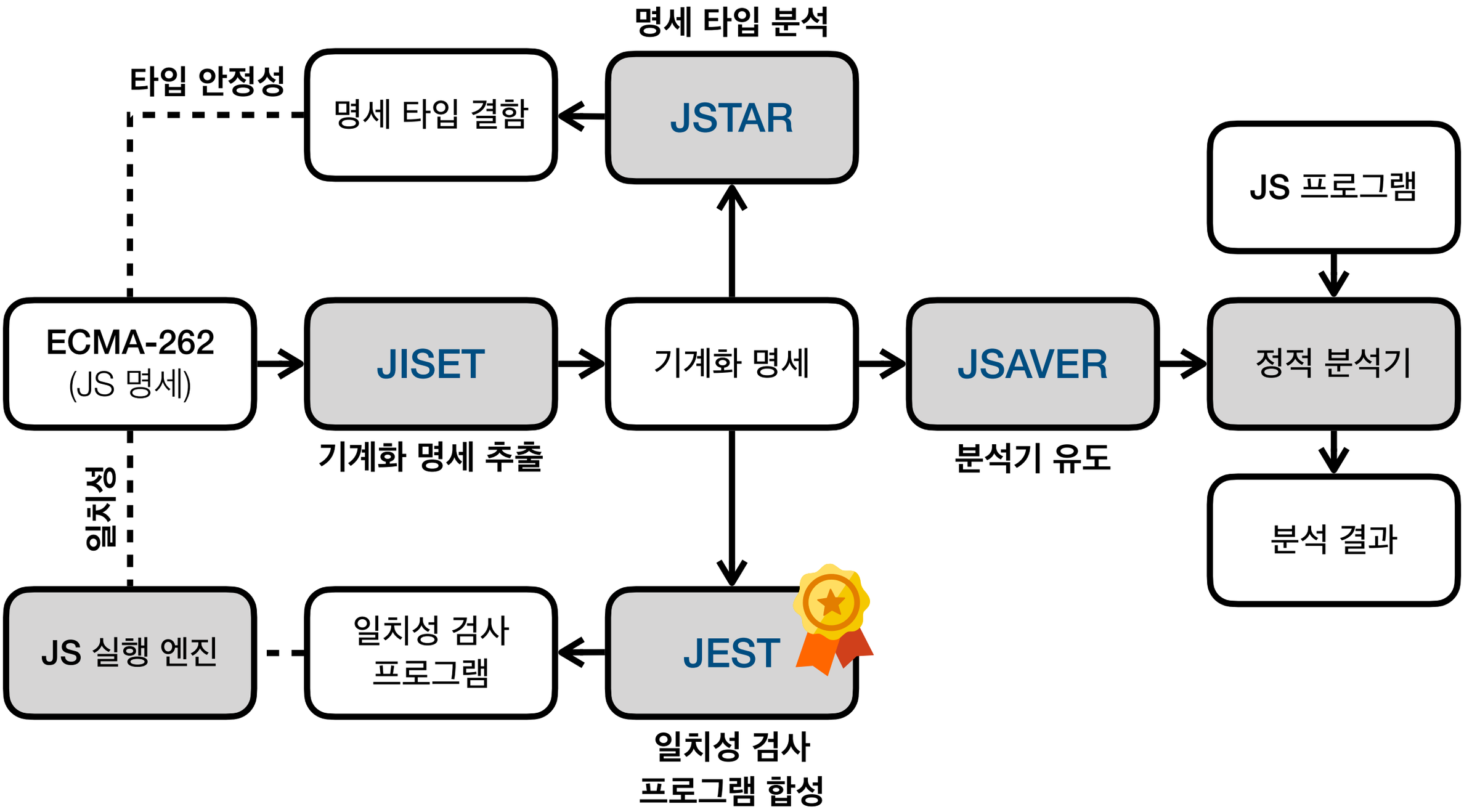


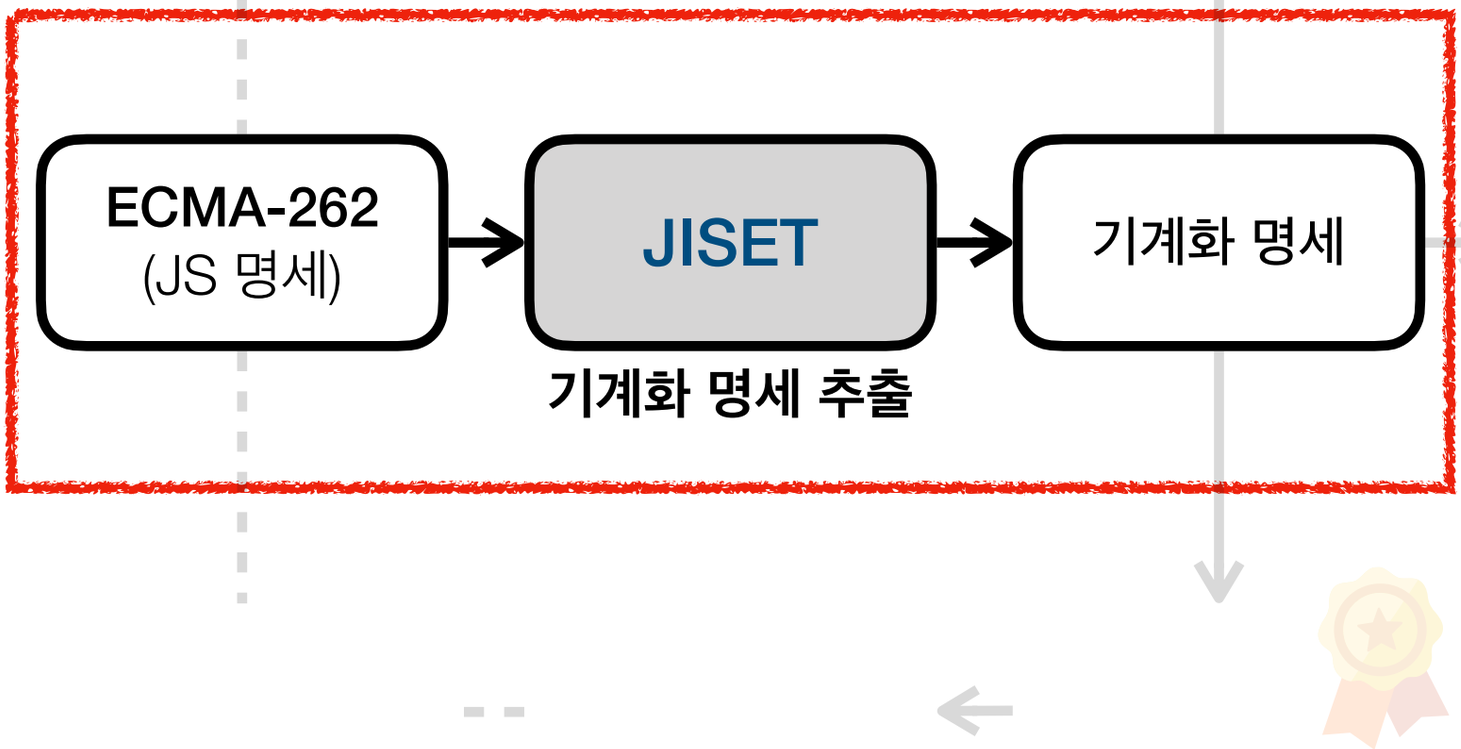
분석기를 자동으로 유도하기?



아이디어 - 명세를 실행가능한 형태로 변환

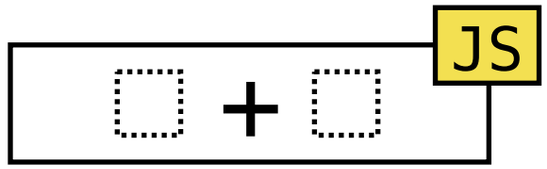


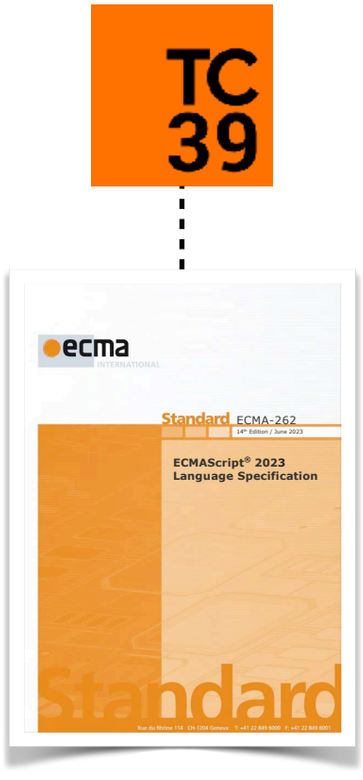

명세 타입 분석
타입 안정성
명세 타입 결함
JSTAR
JS 프로그램
ECMA-262
(JS 명세)
JISET
기계화 명세
JSAVER
정적 분석기
기계화 명세 추출
분석기 유도
일치성
분석 결과
JS 실행 엔진
일치성 검사
프로그램
JEST
일치성 검사
프로그램 합성



[ASE'20] J. Park, et al. "JISET: JavaScript IR-based Semantics Extraction Toolchain"

ECMA-262 - 자바스크립트 공식 언어 명세





ECMA-262
(JavaScript Spec.)

Syntax

AdditiveExpression`[Yield, Await]` :
MultiplicativeExpression`[?Yield, ?Await]`
AdditiveExpression`[?Yield, ?Await]` **+** *MultiplicativeExpression*`[?Yield, ?Await]`
AdditiveExpression`[?Yield, ?Await]` **−** *MultiplicativeExpression*`[?Yield, ?Await]`

AdditiveExpression : *AdditiveExpression* **+** *MultiplicativeExpression*

1. Return ? EvaluateStringOrNumericBinaryExpression(*AdditiveExpression*, **+**, *MultiplicativeExpression*).

Semantics

TypeError

JS

```
!0 + 2n
```

AdditiveExpression : *AdditiveExpression* **+** *MultiplicativeExpression*

1. Return ? EvaluateStringOrNumericBinaryExpression(*AdditiveExpression*, **+**, *MultiplicativeExpression*).

Expr **!0** + Expr **2n**

EvaluateStringOrNumericBinaryExpression (*leftOperand*, *opText*, *rightOperand*)

1. Let *lref* be ? Evaluation of *leftOperand*.
2. Let *lval* be ? GetValue(*lref*).

Evaluate Left (lval = **true**)

3. Let *rref* be ? Evaluation of *rightOperand*.
4. Let *rval* be ? GetValue(*rref*).

Evaluate Right (rval = **2n**)

5. Return ? ApplyStringOrNumericBinaryOperator(*lval*, *opText*, *rval*).

Boolean **true** + BigInt **2n**

ApplyStringOrNumericBinaryOperator (*lval*, *opText*, *rval*)

1. If *opText* is **+**, then
   a. Let *lprim* be ? ToPrimitive(*lval*).
   b. Let *rprim* be ? ToPrimitive(*rval*).
   c. If *lprim* is a String or *rprim* is a String, then ...
   d. Set *lval* to *lprim*.
   e. Set *rval* to *rprim*.
2. NOTE: At this point, it must be a numeri
3. Let *lnum* be ? ToNumeric(*lval*).
4. Let *rnum* be ? ToNumeric(*rval*).
5. If Type(*lnum*) is not Type(*rnum*), throw a **TypeError** exception.

...

Conversion to Primitive (lprim = **true** and rprim = **2n**)

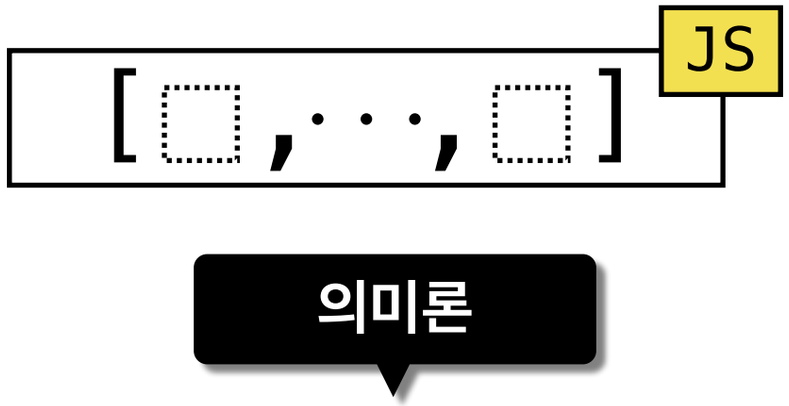
Conversion to Numeric (lnum = **1** and rnum = **2n**)

TypeError

Number

BigInt

JISET - 명세 알고리즘 속 패턴



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

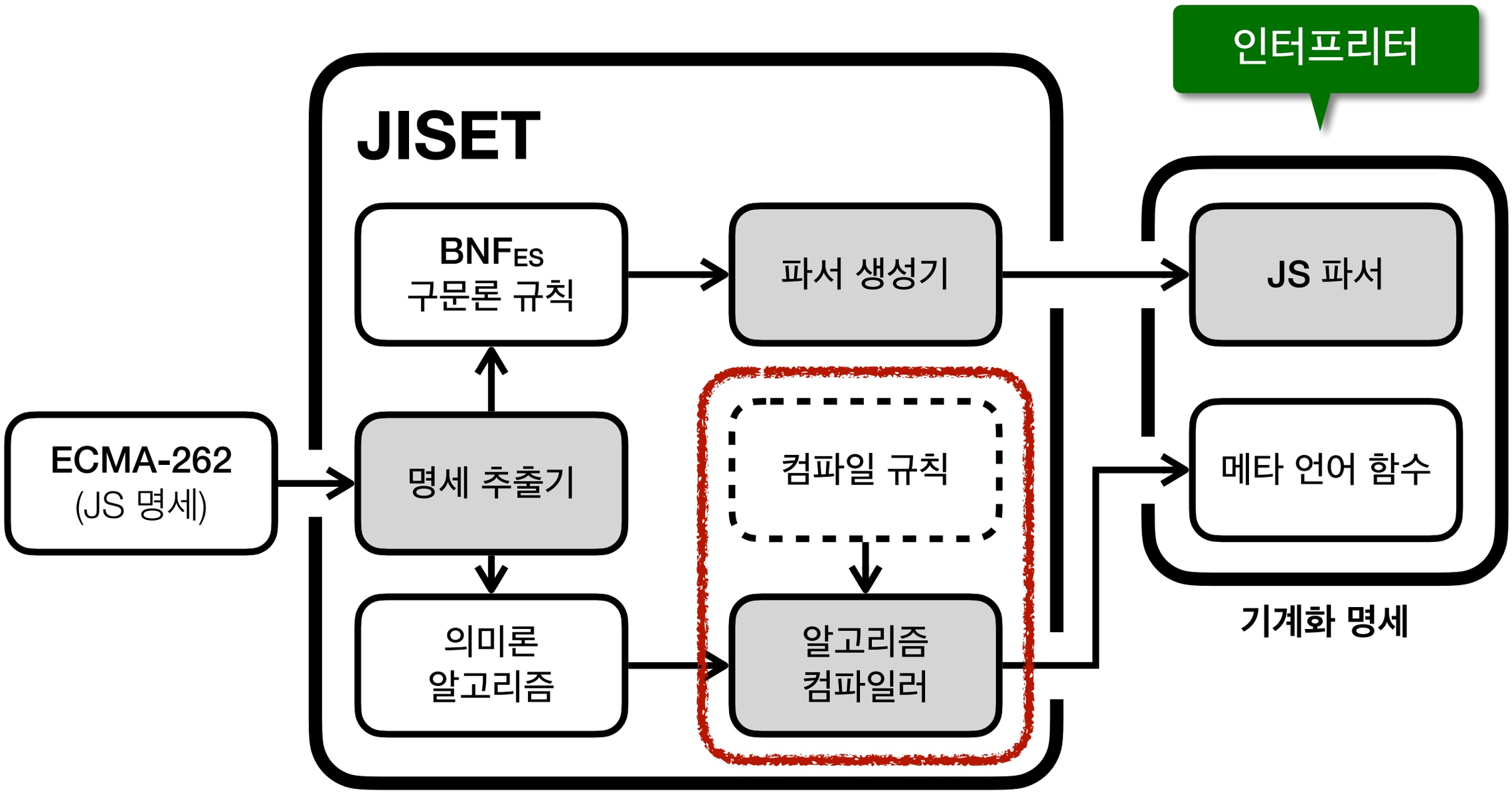
1. Let *array* be ! ArrayCreate(0).
2. Let *nextIndex* be ? ArrayAccumulation of *ElementList* with arguments *array* and 0.
3. If *Elision* is present, then
    a. Perform ? ArrayAccumulation of *Elision* with arguments *array* and *nextIndex*.
4. Return *array*.

JISET - ECMA-262를 위한 메타 언어

$$
\begin{array}{lll}
\text{Programs} & \mathfrak{P} \ni P ::= f^* & \\
\text{Functions} & \mathcal{F} \ni f ::= \texttt{syntax}^? \ \texttt{def x(x}^*\texttt{) \{}[\ell : i]^*\texttt{\}} & \\
\text{Variables} & \mathcal{X} \ni \texttt{x} & \\
\text{Labels} & \mathcal{L} \ni \ell & \\
\text{Instructions} & \mathcal{I} \ni i ::= r := e \mid \texttt{x} := \texttt{\{\}} \mid \texttt{x} := e(e^*) & \\
 & \qquad \mid \ \texttt{if}\ e\ \ell\ \ell \mid \texttt{return}\ e & \\
\text{Expressions} & \mathcal{E} \ni e ::= v^{\text{p}} \mid \text{op}(e^*) \mid r & \\
\text{References} & \mathcal{R} \ni r ::= \texttt{x} \mid e[e] \mid e[e]_{\text{js}} & \\
 & \vdots & \\
\text{Values} & v \in \mathbb{V} & = \mathbb{A} \uplus \mathbb{V}^{\text{p}} \uplus \mathbb{T} \uplus \mathcal{F} \\
\text{Primitive Values} & v^{\text{p}} \in \mathbb{V}^{\text{p}} & = \mathbb{V}_{\text{bool}} \uplus \mathbb{V}_{\text{int}} \uplus \mathbb{V}_{\text{str}} \uplus \cdots \\
\text{JS ASTs} & t \in \mathbb{T} &
\end{array}
$$

JISET

(**J**avaScript **I**R-based **S**emantics **E**xtraction **T**oolchain)



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

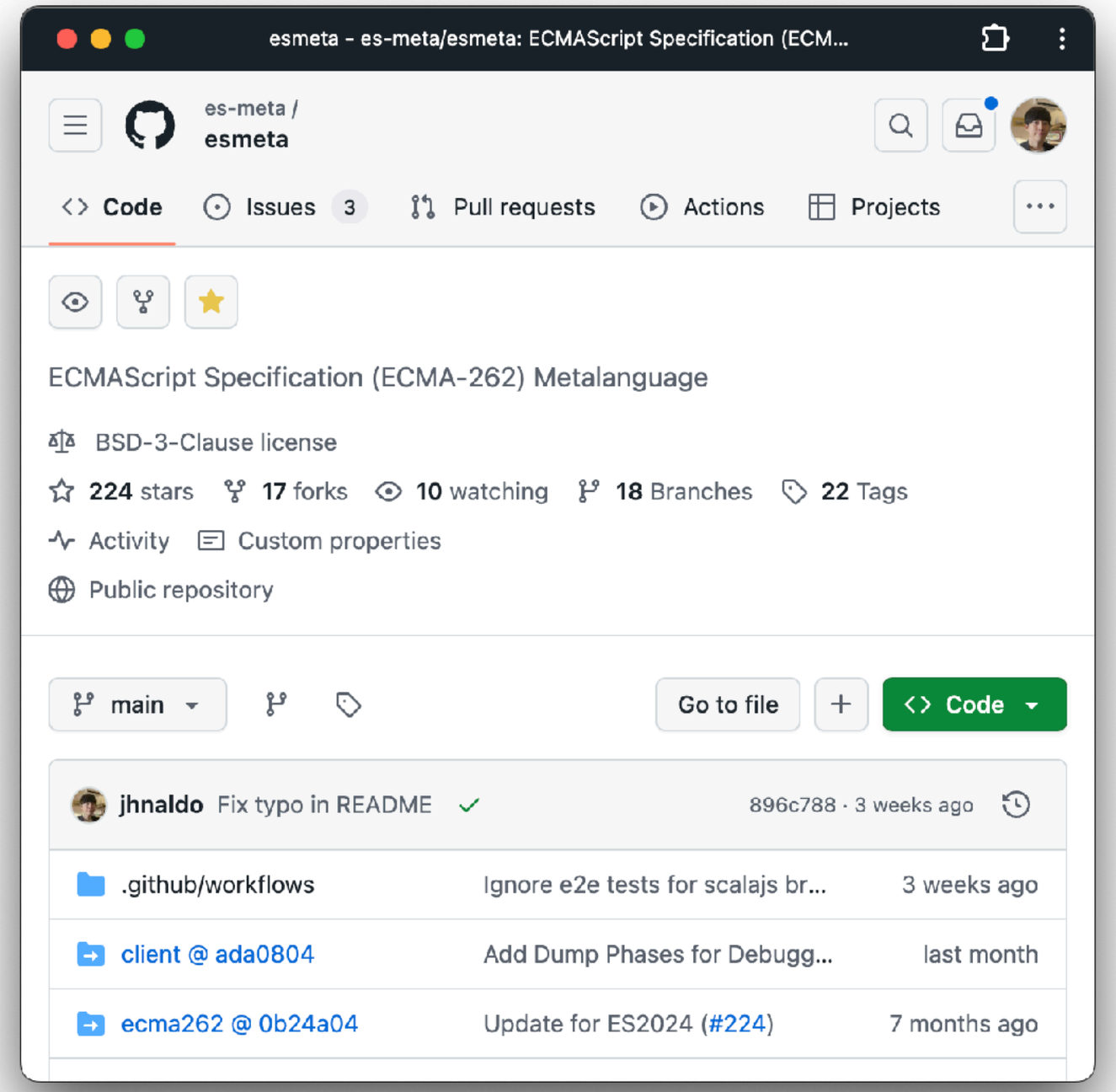
≈ 96%
자동 컴파일

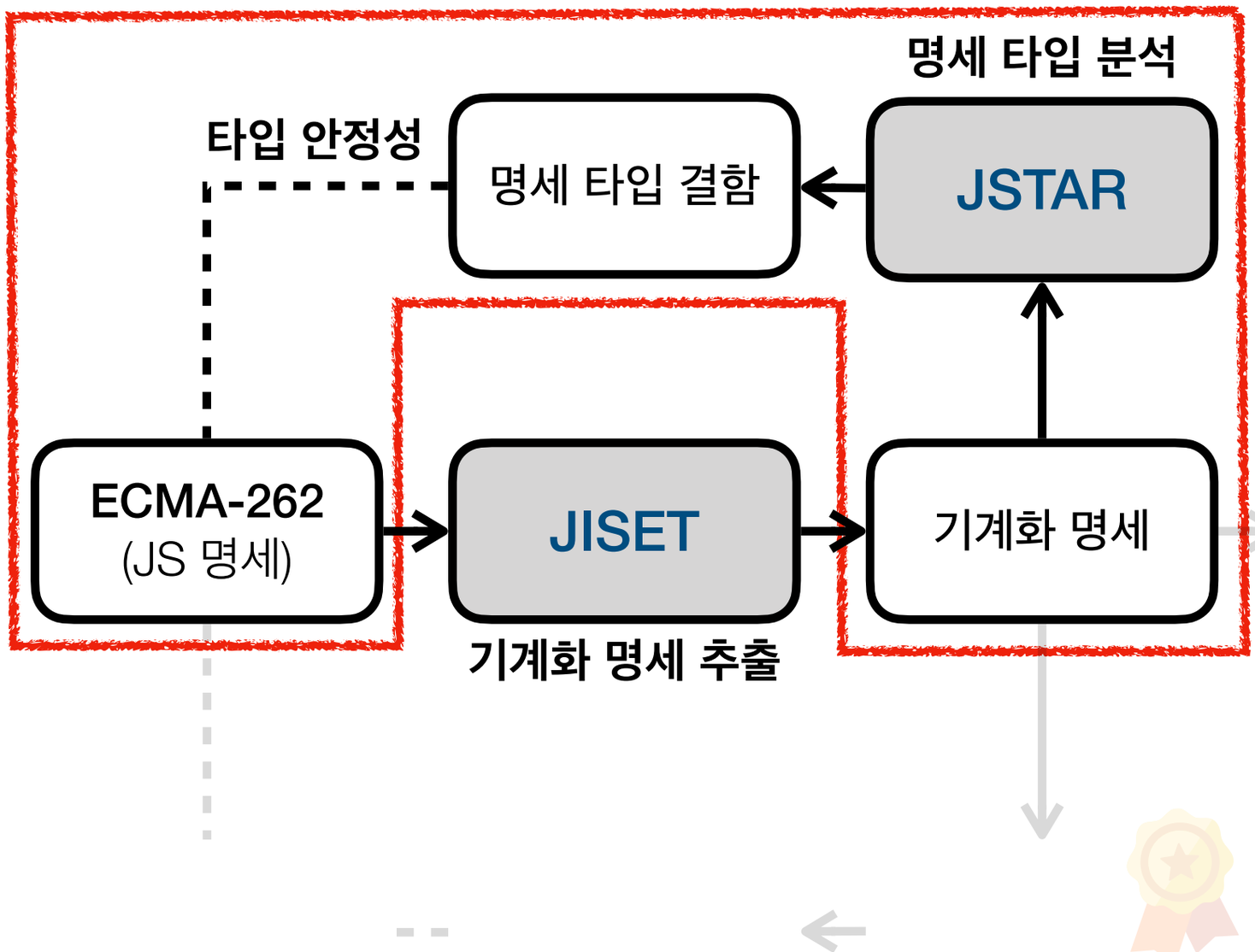
auto manual

T: Total **L**: Core Language Semantics **B**: Built-in Libraries

Version	# Algo.		
ES7	2,105	T	**10,471 / 10,982** (95.35%)
		L	**8,041 / 8,415** (95.56%)
		B	**2,430 / 2,567** (94.66%)
ES8	2,238	T	**11,181 / 11,732** (95.30%)
		L	**8,453 / 8,811** (95.94%)
		B	**2,728 / 2,921** (93.39%)
ES9	2,370	T	**11,849 / 12,393** (95.61%)
		L	**8,932 / 9,311** (95.93%)
		B	**2,917 / 3,082** (94.65%)
ES10	2,396	T	**12,022 / 12,569** (95.65%)
		L	**9,073 / 9,456** (94.95%)
		B	**2,949 / 3,113** (94.73%)
ES11	2,521	T	**12,505 / 13,047** (94.85%)
		L	**9,495 / 9,881** (96.09%)
		B	**3,010 / 3,166** (95.07%)
ES12	2,640	T	**12,975 / 13,544** (95.80%)
		L	**9,717 / 10,136** (95.87%)
		B	**3,258 / 3,408** (95.60%)
Average	2,378	T	**11,834 / 12,378** (95.61%)
		L	**8,952 / 9,335** (95.90%)
		B	**2,882 / 3,043** (94.71%)

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





[ASE'21] J. Park, et al. "JSTAR: JavaScript Specification Type Analyzer using Refinement"

JSTAR - 명세 타입 분석

(**J**avaScript **S**pecification **T**ype **A**nalyzer using **R**efinement)

20.3.2.28 Math.round (x)

x : String | Boolean | Number | Object | ...

n : Number

Number | Exception

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.

. . .

Type Error:
`<`, `>`, and `>=`
are numeric operators

```
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.

Fixed

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

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

JSTAR - 실험 결과

- **검사 대상**: 3년 동안 864개의 다른 ECMA-262 버전

59.2% Precision

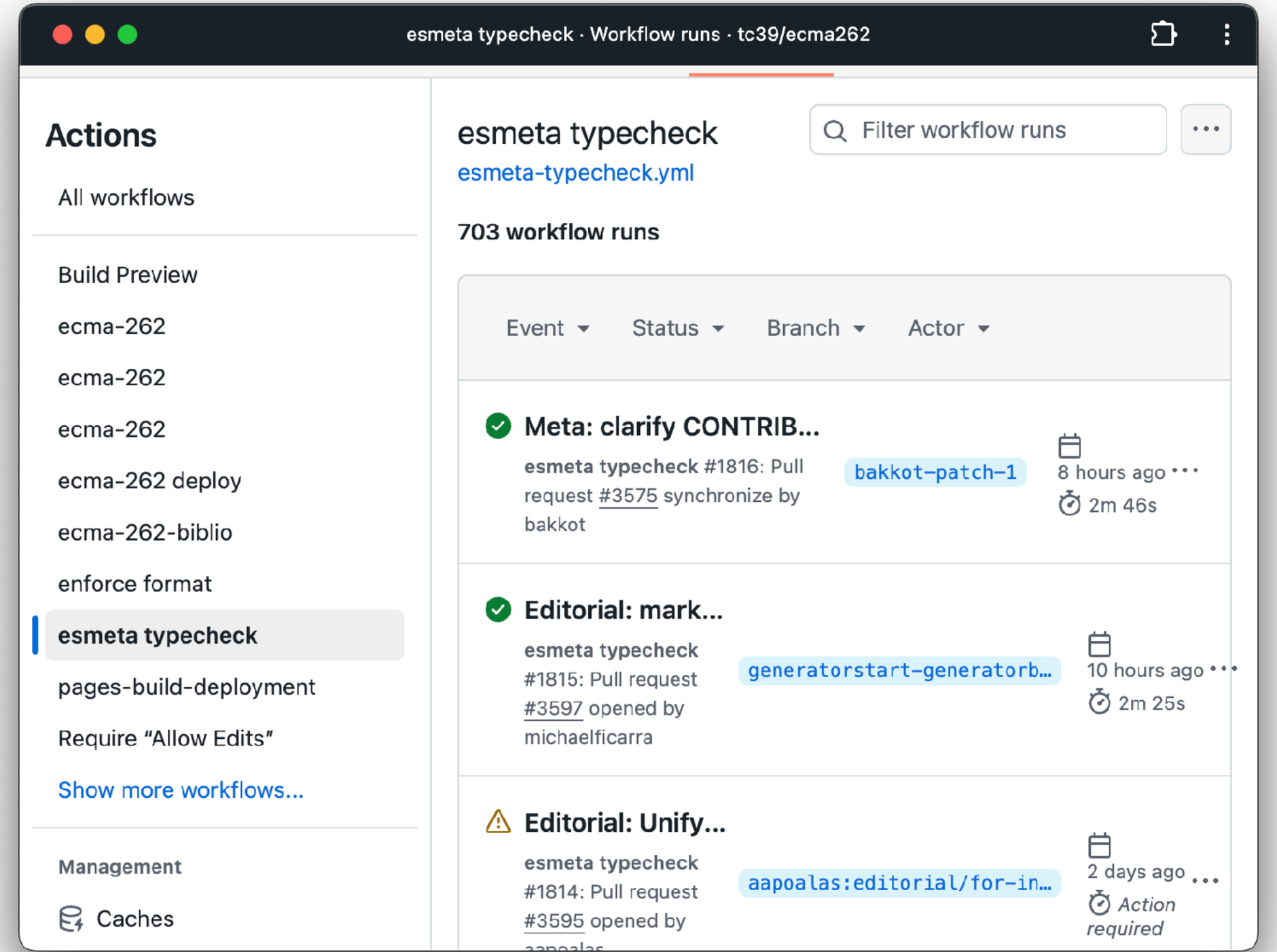
93 Errors Detected

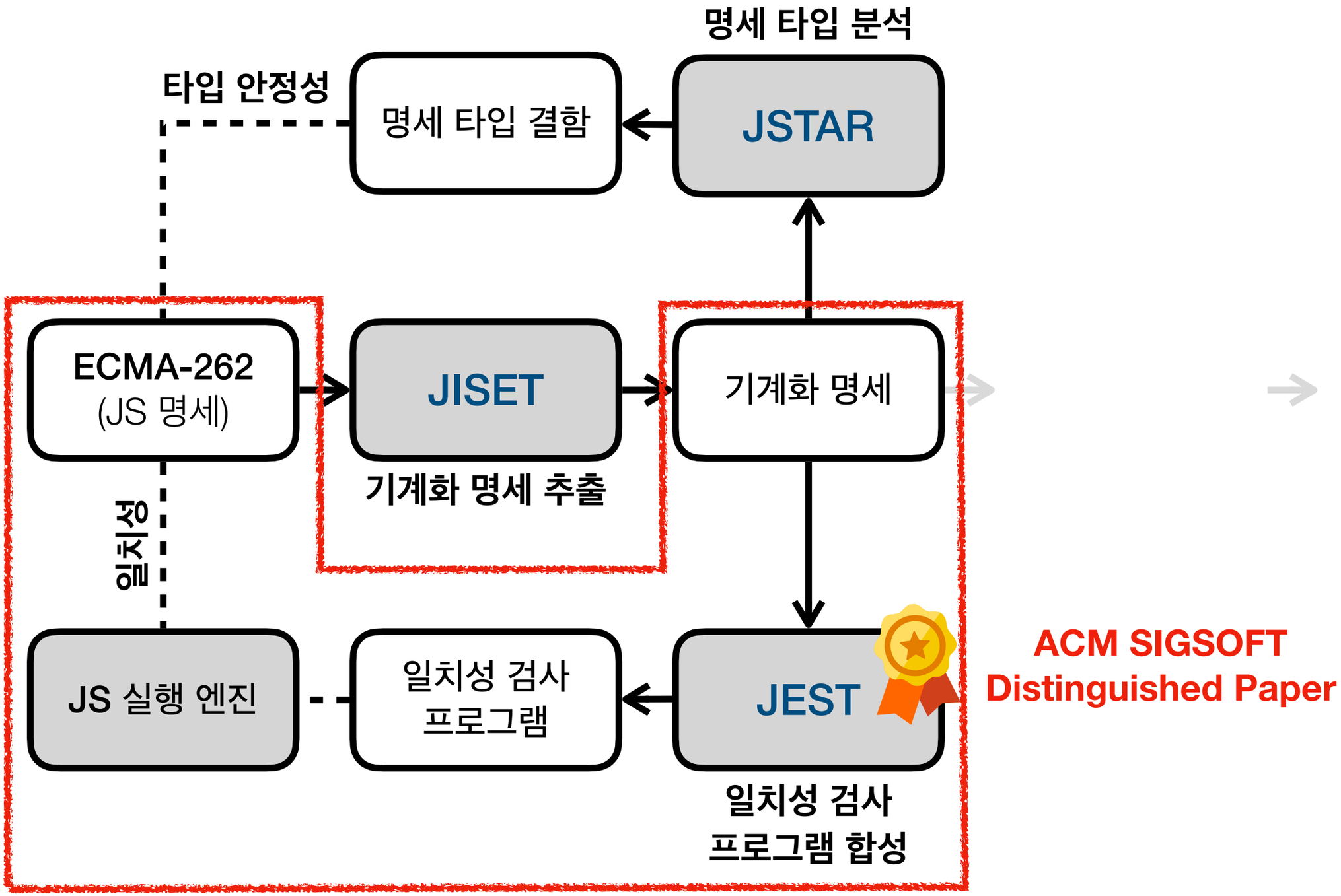
Checker	Bug Kind	Precision = (# True Bugs) / (# Detected Bugs)					
		no-refine		refine		Δ	
Reference	UnknownVar	62 / 106	17 / 60	63 / 78	17 / 31	+1 / -28	/ -29
	DuplicatedVar		45 / 46		46 / 47		+1 / +1
Arity	MissingParam	4 / 4	4 / 4	4 / 4	4 / 4	/	/
Assertion	Assertion	4 / 56	4 / 56	4 / 31	4 / 31	/ -25	/ -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

14 New Bugs In ES2021

JS 언어 명세 공식 안정성 검사 도구로 채택



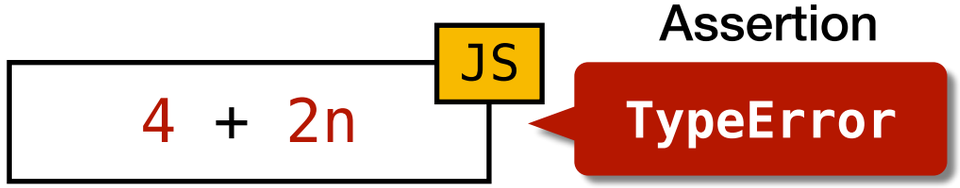


ACM SIGSOFT
Distinguished Paper

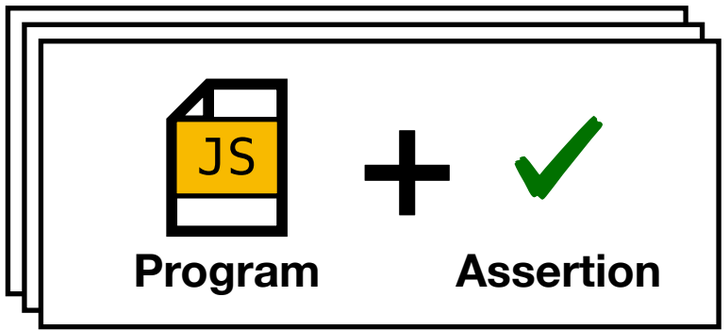
[ICSE'21] J. Park, et al. "JEST: N+1-version Differential Testing of Both JavaScript Engines"
[PLDI'23] J. Park, et al. "Feature-Sensitive Coverage for Conformance Testing of Programming Language Implementations"

JavaScript 일치성 검사

Example:



ECMA-262
(JavaScript Spec.)



일치성 검사 프로그램

일치성 검사 필요

GraalVM

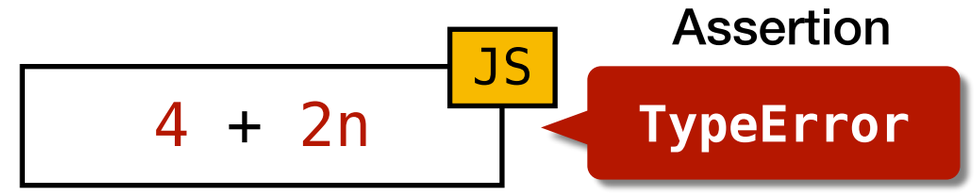
QuickJS

moddable

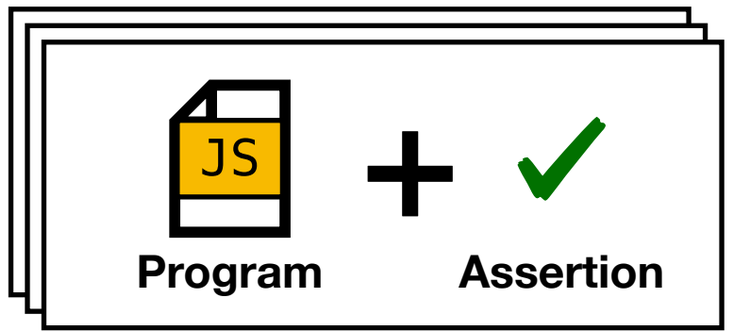
JavaScript
Engines

문제 - 일치성 검사 프로그램 수동 작성

Example:



ECMA-262
(JavaScript Spec.)



일치성 검사 프로그램

일치성 검사 필요

Test262
(공식 테스트 집합)

수동 작성

GraalVM

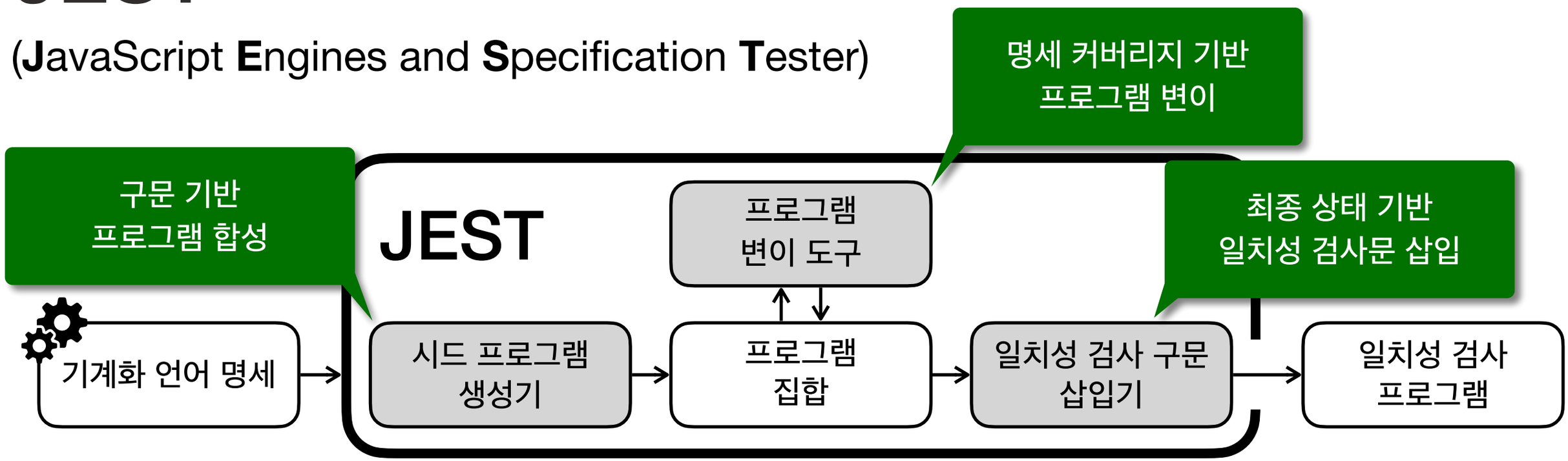
QuickJS

moddable

JavaScript
Engines

JEST

(**J**avaScript **E**ngines and **S**pecification **T**ester)



프로그램 집합

```
let x = 1 + 2;
assert(x == 3);
```

```
let x = 42;
assert(x == 42);
```

```
let x = ![];
assert(x == false);
```

JEST - 명세 커버리지 기반 프로그램 생성

ApplyStringOrNumericBinaryOperator (*lval*, *opText*, *rval*)

...

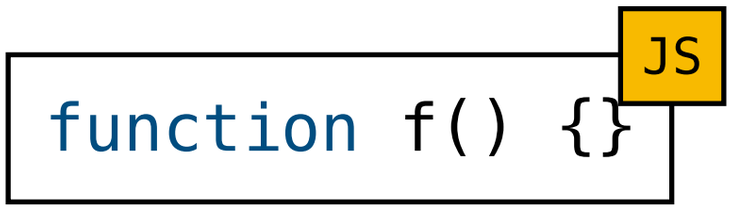
3. Let *lnum* be ? ToNumeric(*lval*).
4. Let *rnum* be ? ToNumeric(*rval*).
5. If Type(*lnum*) is not Type(*rnum*), throw a **TypeError** exception.
6. If *lnum* is a BigInt, then
   ...
7. Else,
   ...

```JS
3 + 2
```

```JS
1n + 2n
```

```JS
4 + 2n
```

JEST - 최종 상태 기반 상태 검사문 삽입




```
  function f() {}
+ $assert.equal(Object.getPrototypeOf(f), Function.prototype);
+ $assert.verifyProperty(f, "prototype", {
+   writable: true,
+   enumerable: false,
+   configurable: false,
+ });

+ $assert.compare(Reflect.ownKeys(f), ['length', 'name', 'prototype'], f);

+ ...
```

Prototype Chain

Property Descriptor

Property Order

Etc.

JEST - 실험 결과

Kind	Name	Version	Release	# Detected Unique Bugs		
				# New	# Confirmed	# Reported
Engine	V8	v10.8.121	2022.10.06	3	3	4
	JSC	v615.1.10	2022.10.26	26	[illegible]	26
	GraalJS	v22.2.0	2022.07.26	11	[illegible]	11
	SpiderMonkey	v107.0b4	2022.10.24	2	[illegible]	4
	Total			**42**	**44**	**45**
Transpiler	Babel	v7.19.1	2022.09.15	37	37	39
	SWC	v1.3.10	2022.10.21	37	[illegible]	47
	Terser	v5.15.1	2022.10.05	19	[illegible]	19
	Obfuscator	v4.0.0	2022.02.15	1	[illegible]	7
	Total			**94**	**95**	**112**
Total				**136**	**139**	**157**

42 Bugs
In Engines

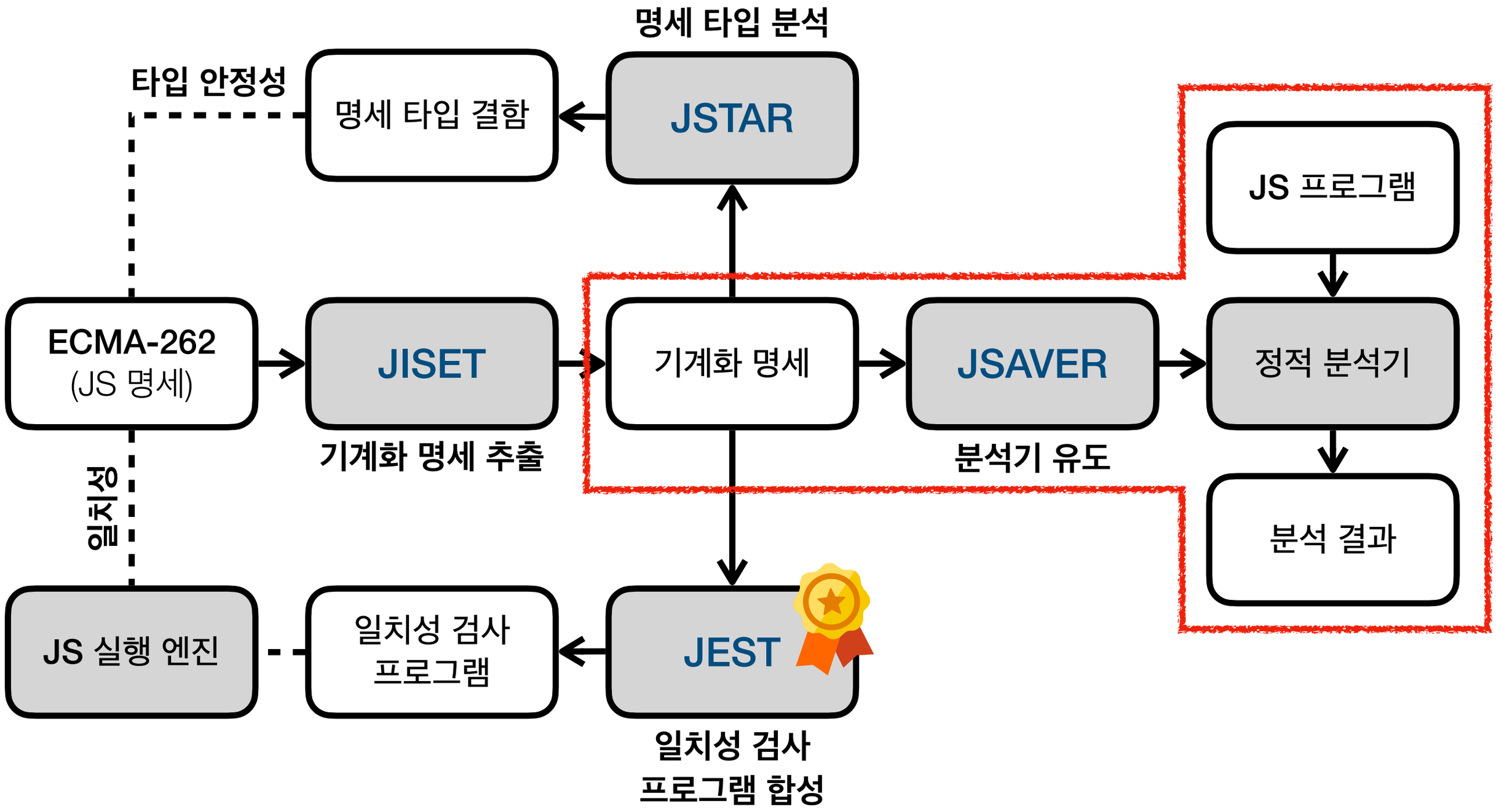
94 Bugs
In Transpilers

JEST - Example in GraalVM


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

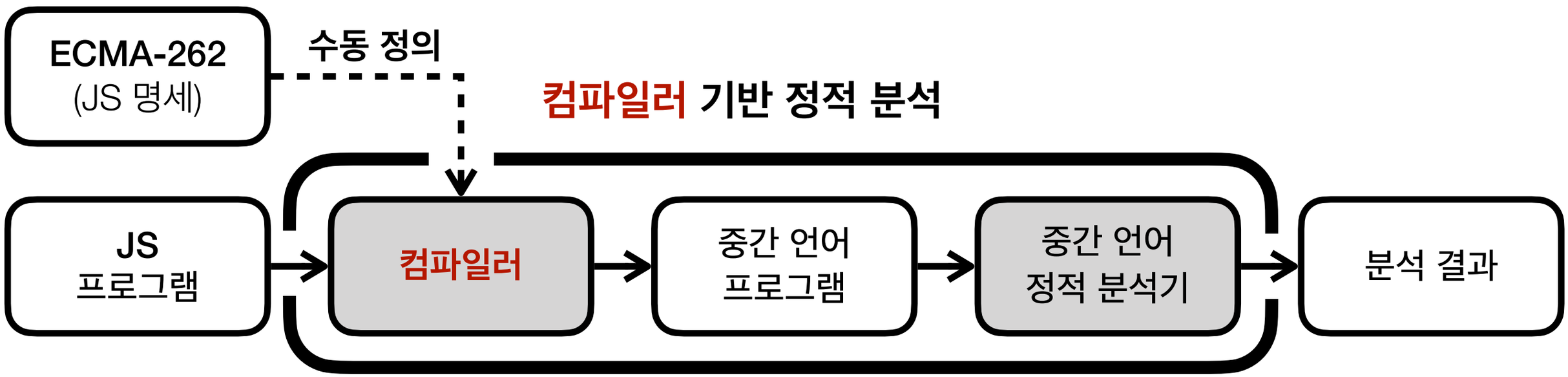
*“Right now, we are running Test262 and the V8 and Nashorn unit test suites in our CI for every change, it might make sense to **add your suite as well**.”*

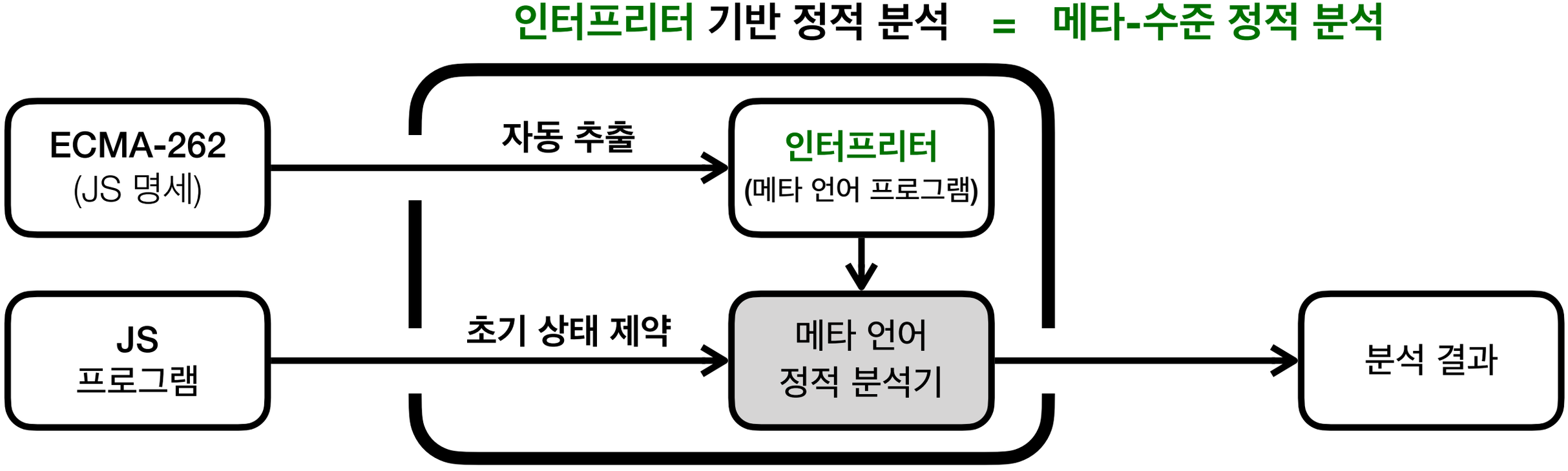
- A Developer of GraalVM



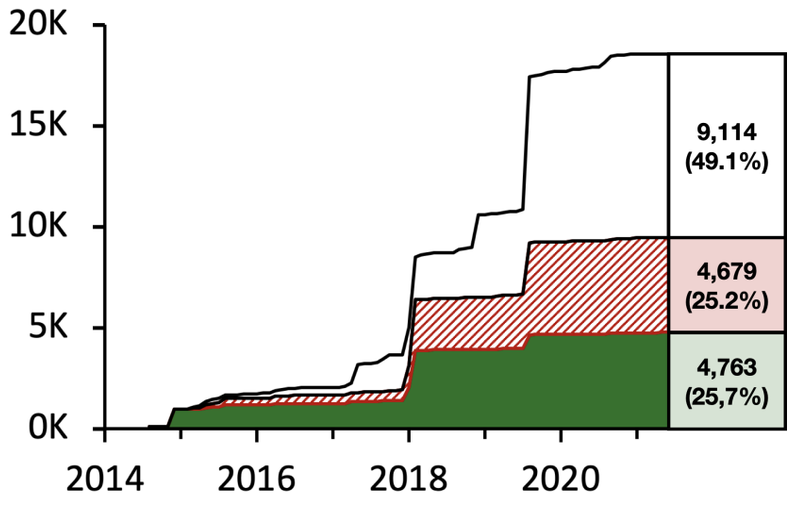
[FSE'22] J. Park, et al. "Automatically Deriving JavaScript Static Analyzers from Language Specifications"

아이디어: 인터프리터 기반 정적 분석

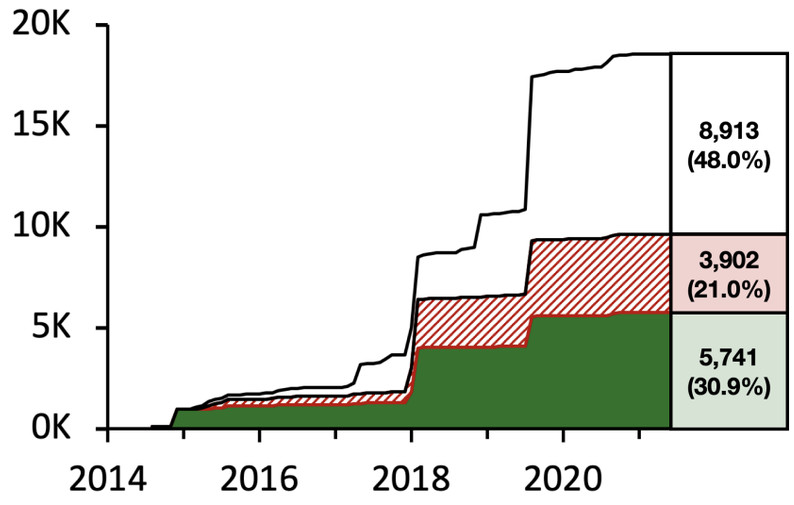




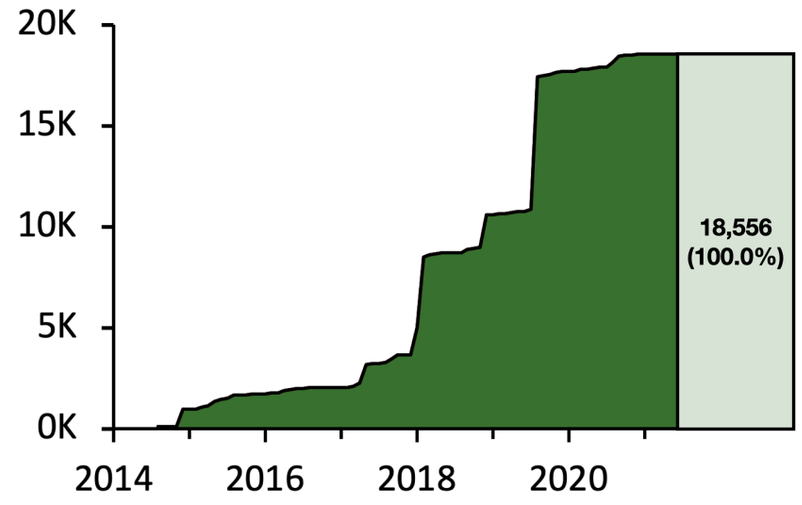
JSAVER - 평가 (RQ1: 올바른 분석)



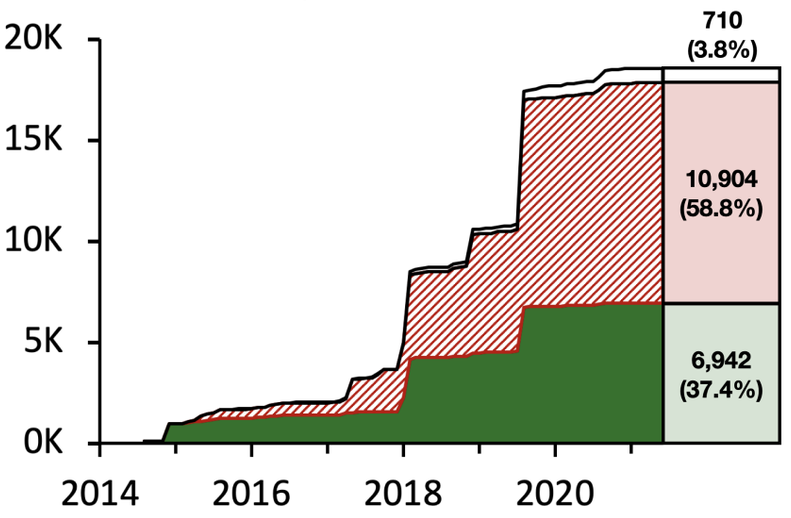
(a) Analysis results of TAJS



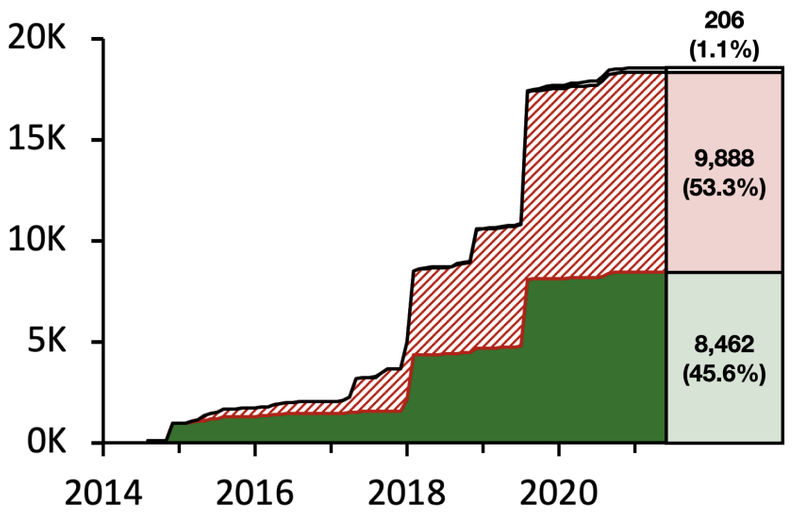
(b) Analysis results of SAFE



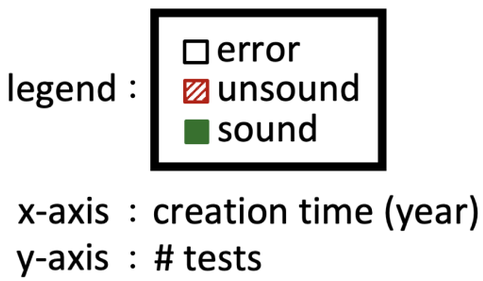
(c) Analysis results of JSA_{ES12}



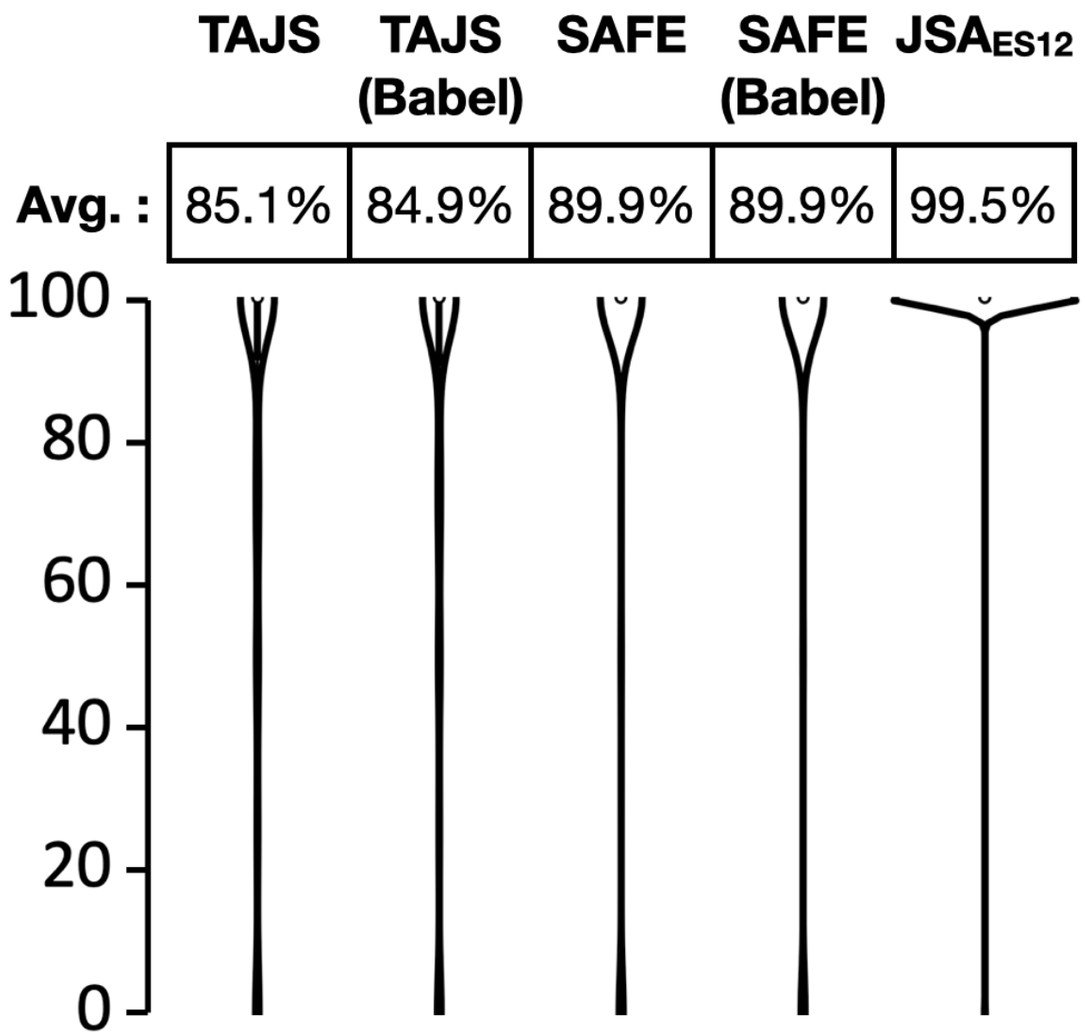
(d) Analysis results of TAJS with Babel



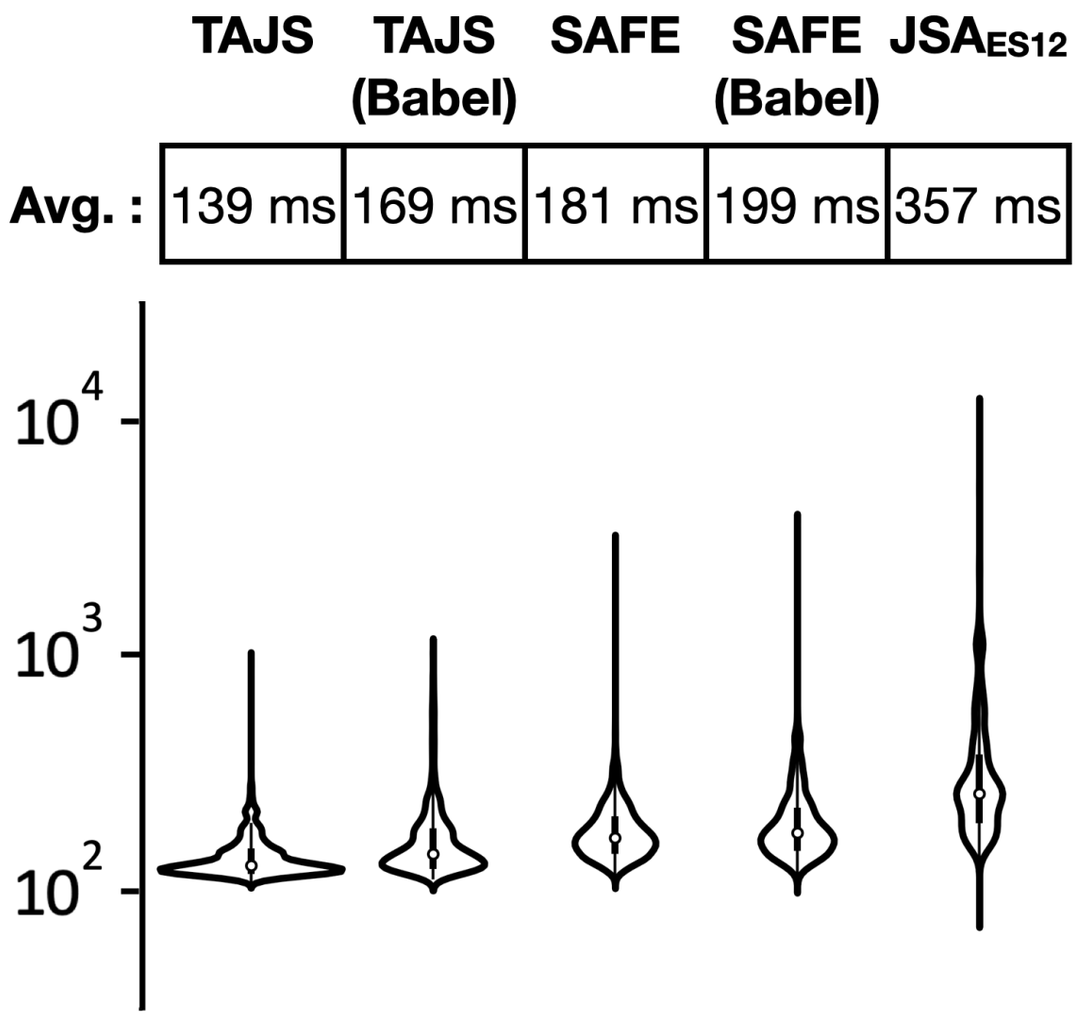
(e) Analysis results of SAFE with Babel



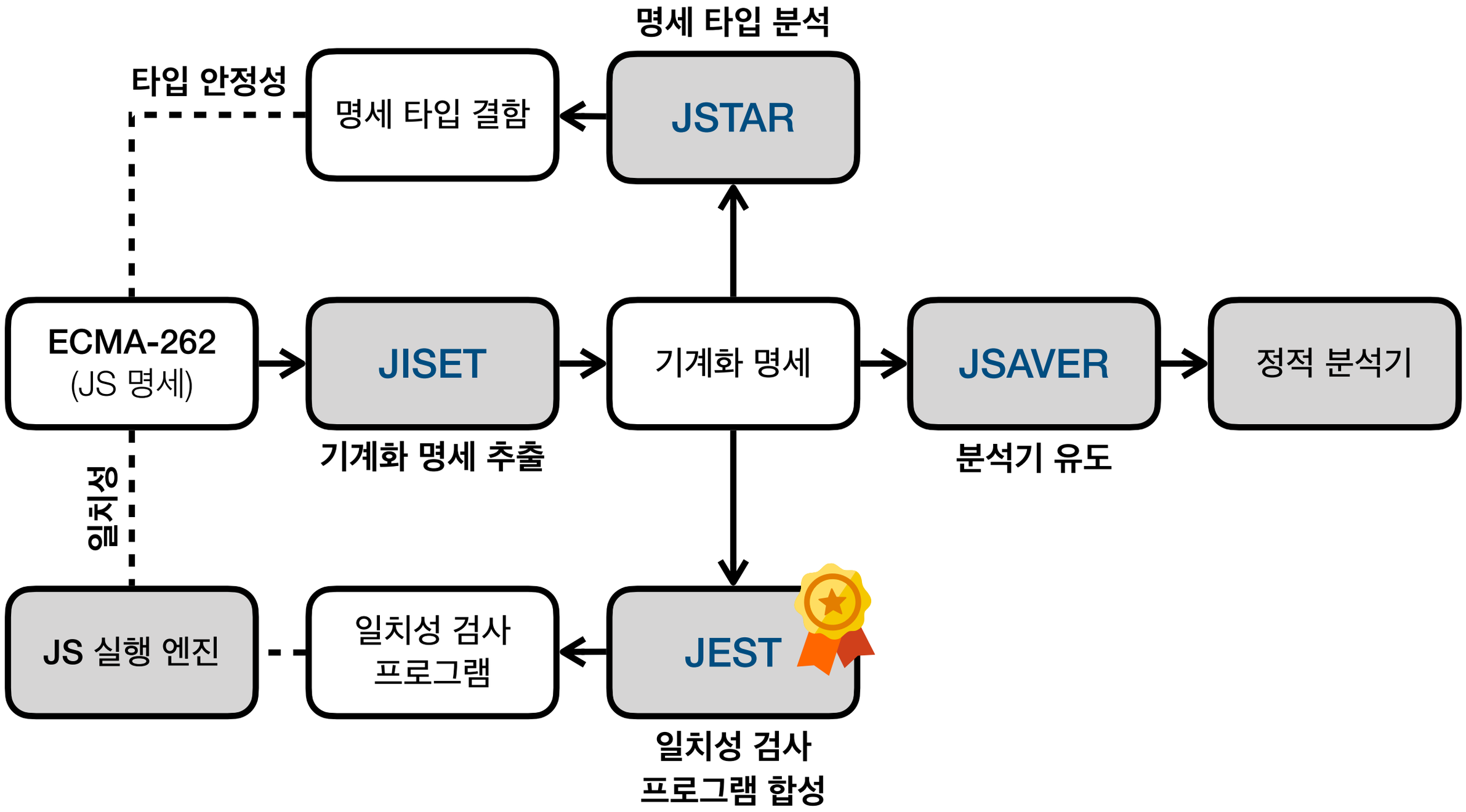
JSAVER - 평가 (RQ2: 정확도 / 속도)

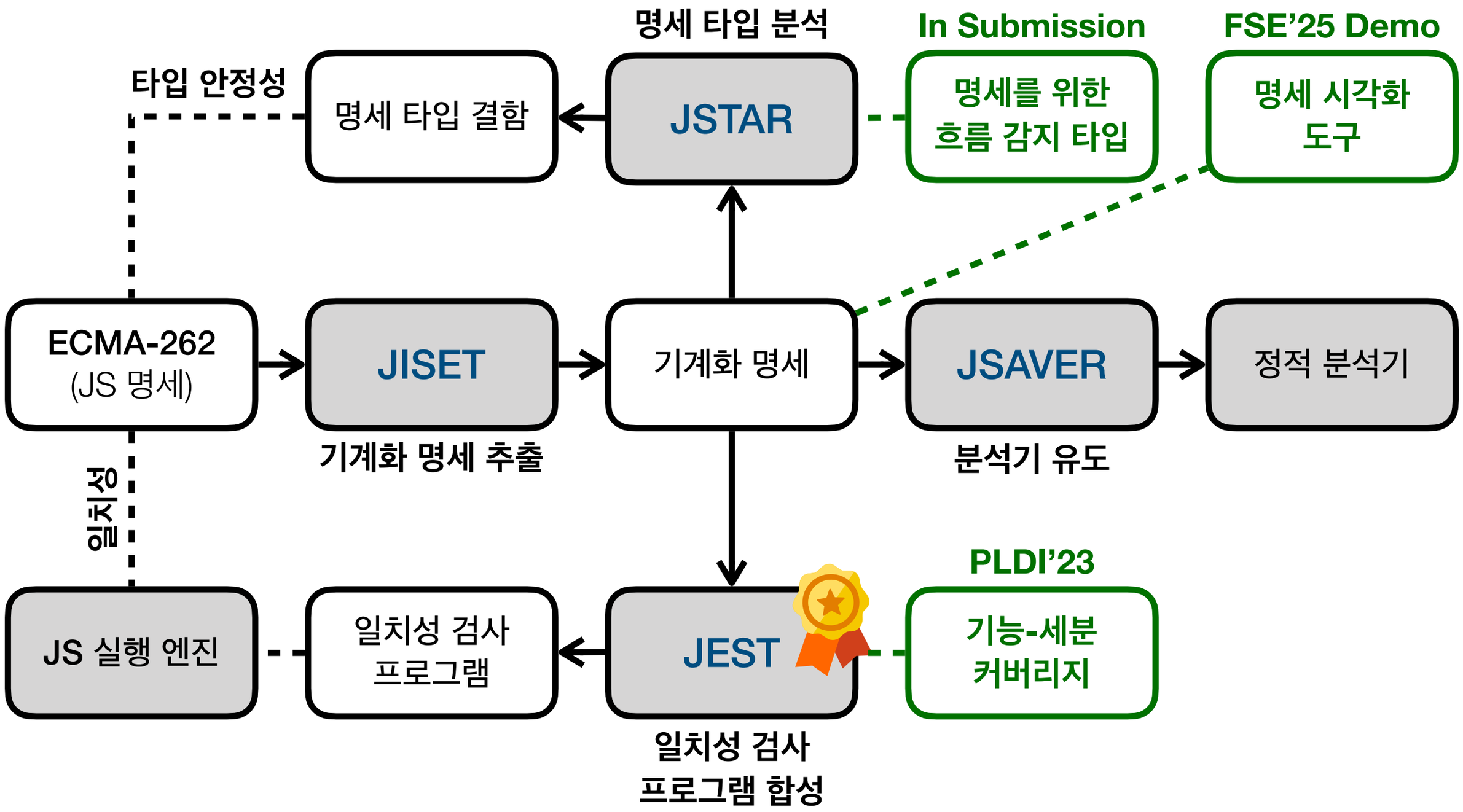


(a) The analysis precision



(b) The analysis performance


명세 타입 분석
타입 안정성
명세 타입 결함
JSTAR
ECMA-262
(JS 명세)
JISET
기계화 명세
JSAVER
정적 분석기
기계화 명세 추출
분석기 유도
일치성
JS 실행 엔진
일치성 검사
프로그램
JEST
일치성 검사
프로그램 합성


명세 타입 분석
In Submission
FSE'25 Demo
타입 안정성
명세 타입 결함
JSTAR
명세를 위한
흐름 감지 타입
명세 시각화
도구
ECMA-262
(JS 명세)
JISET
기계화 명세
JSAVER
정적 분석기
기계화 명세 추출
분석기 유도
일치성
PLDI'23
JS 실행 엔진
일치성 검사
프로그램
JEST
기능-세분
커버리지
일치성 검사
프로그램 합성

명세 타입 분석

In Submission

FSE'25 Demo

타입 안정성

명세 타입 결함

JSTAR

명세를 위한
흐름 감지 타입

명세 시각화
도구

명세 타입
자동 수정

ECMA-262
(JS 명세)

JISET

기계화 명세

JSAVER

정적 분석기

기계화 명세 추출

분석기 유도

일치성

명세 합성

PLDI'23

JS 실행 엔진

일치성 검사
프로그램

JEST

기능-세분
커버리지

트랜스파일러
자동 생성

일치성 검사
프로그램 합성

일치성 검증

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

https://plrg.korea.ac.kr/