KOREA UNIVERSITY

PLRG

JavaScript 언어 생태계 자동화 연구를 하기까지의 여정

박지혁 교수

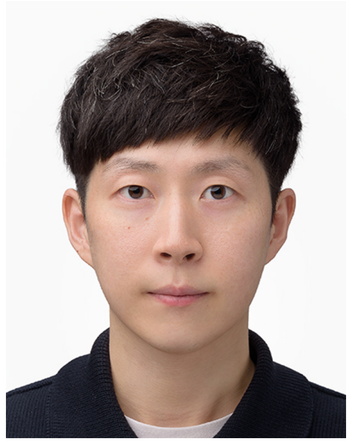
고려대학교 정보대학 컴퓨터학과

KSA 수리정보과학부 콜로퀴엄

2024.05.03

고려대학교 박지혁 교수

2011 NUS High School 교환학생



PLRG

(프로그래밍 언어 연구실)

컴퓨터학과
조교수

2009

2012

전산학과 & 수학과
복수전공

2016

프로그래밍 언어
연구실

2022

2023

PLRG

프로그래밍 언어 연구란?

1. **프로그래밍 언어를 설계** - 기존 언어의 확장 / 새로운 언어의 설계
2. **프로그램을 입력**으로 받거나 **결과**로 내뱉는 **소프트웨어를 개발**

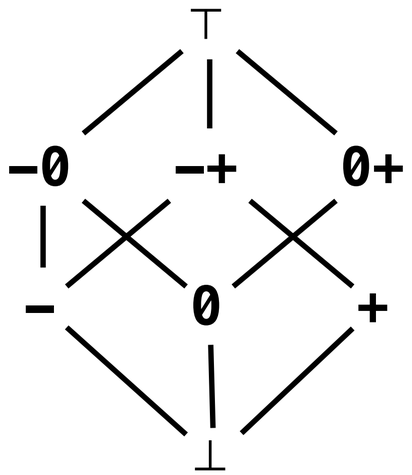
PLRG

예시 - 프로그램 분석

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

항상 음이 아닌 정수 반환

요약 도메인
(Abstract Domain)



⊤ : 모든 정수　　**⊥ : 공집합**
- : 음수　　**0 : 0**
+ : 양수

PLRG

어쩌다가 프로그래밍 언어 연구를?
그 중, 왜 **JavaScript** 연구를?

PLRG

안동시

KAIST

초등학생 / 중학생	고등학생	학부생
절차지향 언어	객체지향 언어	동적 타입 언어
알고리즘 대회	전산학 수업	개발 동아리

PLRG

CS320 프로그래밍 언어

↓

개별 연구

KAIST

류석영 교수님

> **JavaScript**로 개발한 소프트웨어를 **제대로 이해**하고 있나요?

PLRG

안동시

KSA

KAIST

KAIST

ORACLE

KOREA UNIVERSITY 1905

초등학생 / 중학생	고등학생	학부생	대학원생	회사 연구원	교수
					
절차지향 언어	객체지향 언어	동적 타입 언어	객체지향 + 함수형 언어		
					
알고리즘 대회	전산학 수업	개발 동아리	연구 및 개발		

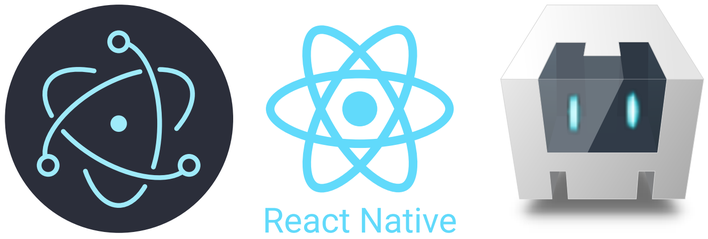
PLRG

JavaScript란 어떤 언어일까?

PLRG

JavaScript는 어디에나 있다

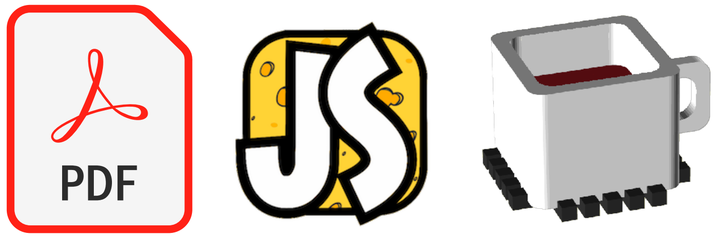
클라이언트-사이드 프로그래밍



모바일/데스크탑 어플리케이션

JS

서버-사이드 프로그래밍

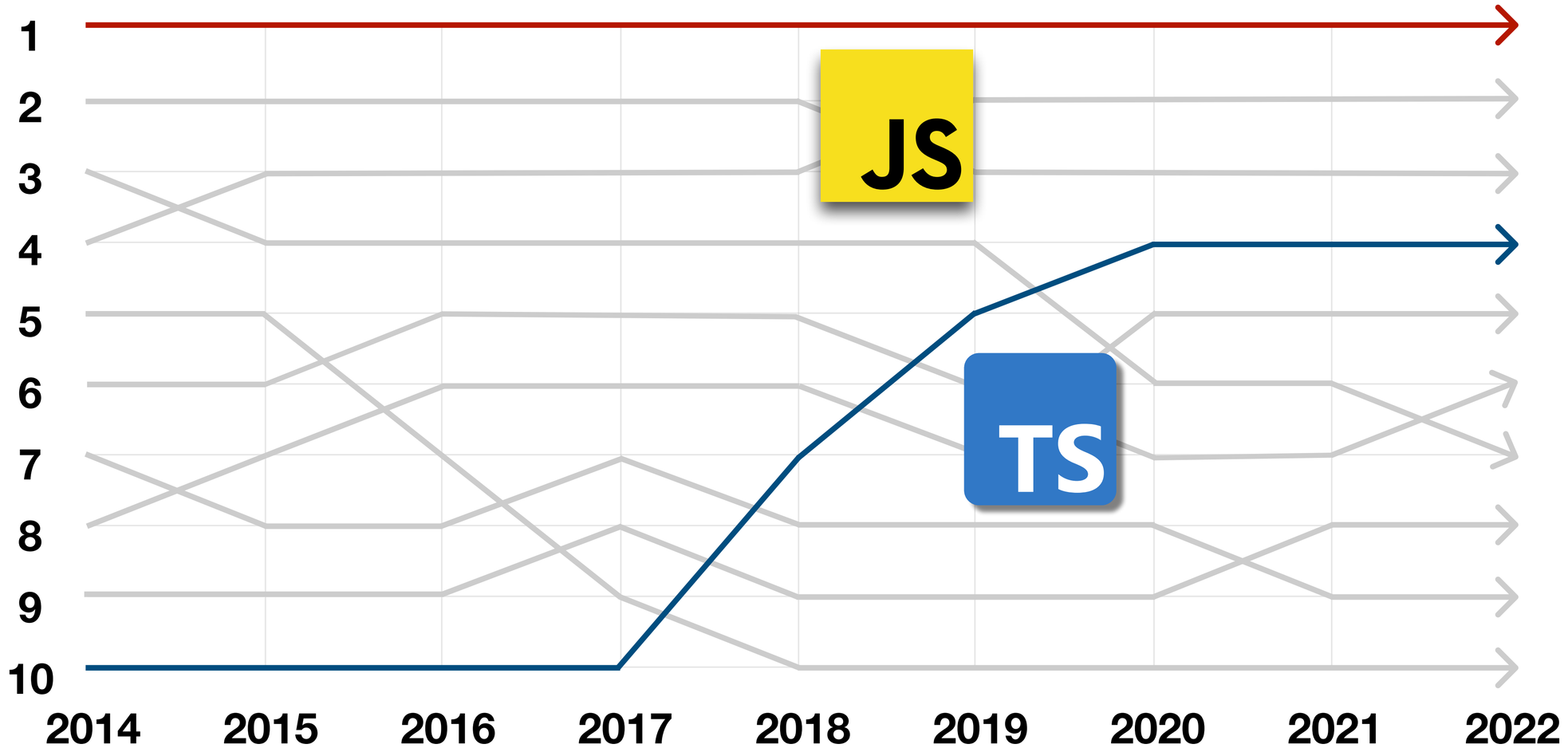


그 외 (PDF, IoT, Microcontrollers, etc.)

PLRG

JavaScript는 어디에나 있다

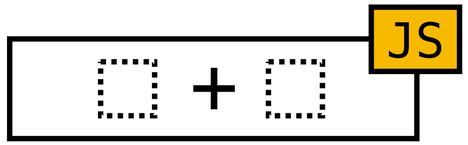
GitHub

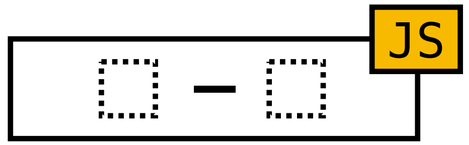


https://octoverse.github.com/

PLRG

하지만, JavaScript는 복잡하다





JavaScript를 어떻게 이해할까?

```
JS
4 + 2
```
6

```
JS
4 + "2"
```
"42"

```
JS
4 + 2n
```
TypeError

```
JS
"ab" + "cd"
```
"abcd"

```
JS
4 - "2"
```
2

```
JS
4 + [2n]
```
"42"

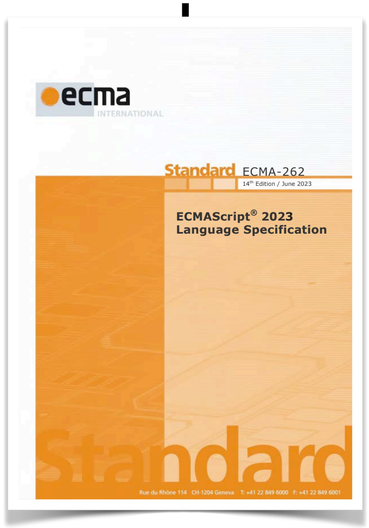
```
JS
(![]+[])[+[]]                    +   // "f"
(![]+[])[+!+[]]                  +   // "a"
([![]]+[][[]])[+!+[]+[+[]]] +   // "i"
(![]+[])[!+[]+!+[]]                  // "l"
```
"fail"

PLRG

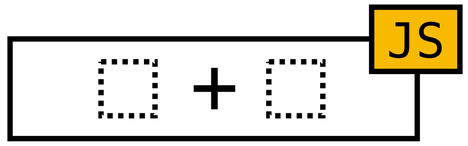
JavaScript의 공식 언어 명세 = ECMA-262

언어 공식 관리 위원회

TC 39



ECMA-262
(JavaScript 언어 명세)



구문(Syntax)

AdditiveExpression`[Yield, Await]` :

MultiplicativeExpression`[?Yield, ?Await]`

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

AdditiveExpression`[?Yield, ?Await]` - *MultiplicativeExpression*`[?Yield, ?Await]`

의미(Semantics)

AdditiveExpression : *AdditiveExpression* + *MultiplicativeExpression*

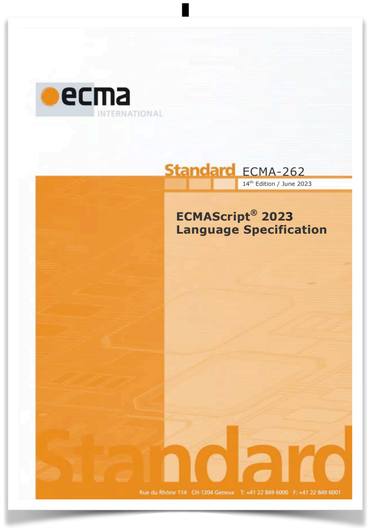
1. Return ? EvaluateStringOrNumericBinaryExpression(*AdditiveExpression*, +, *MultiplicativeExpression*).

PLRG

JavaScript를 위한 프로그래밍 언어 도구

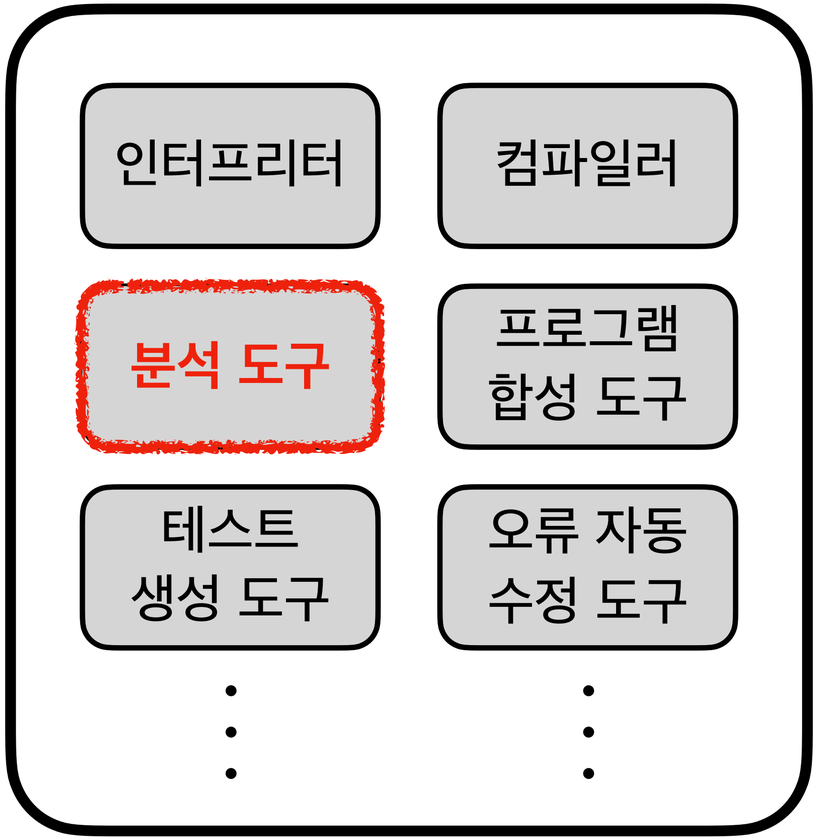
언어 공식 관리 위원회

TC 39



ECMA-262
(JavaScript 언어 명세)

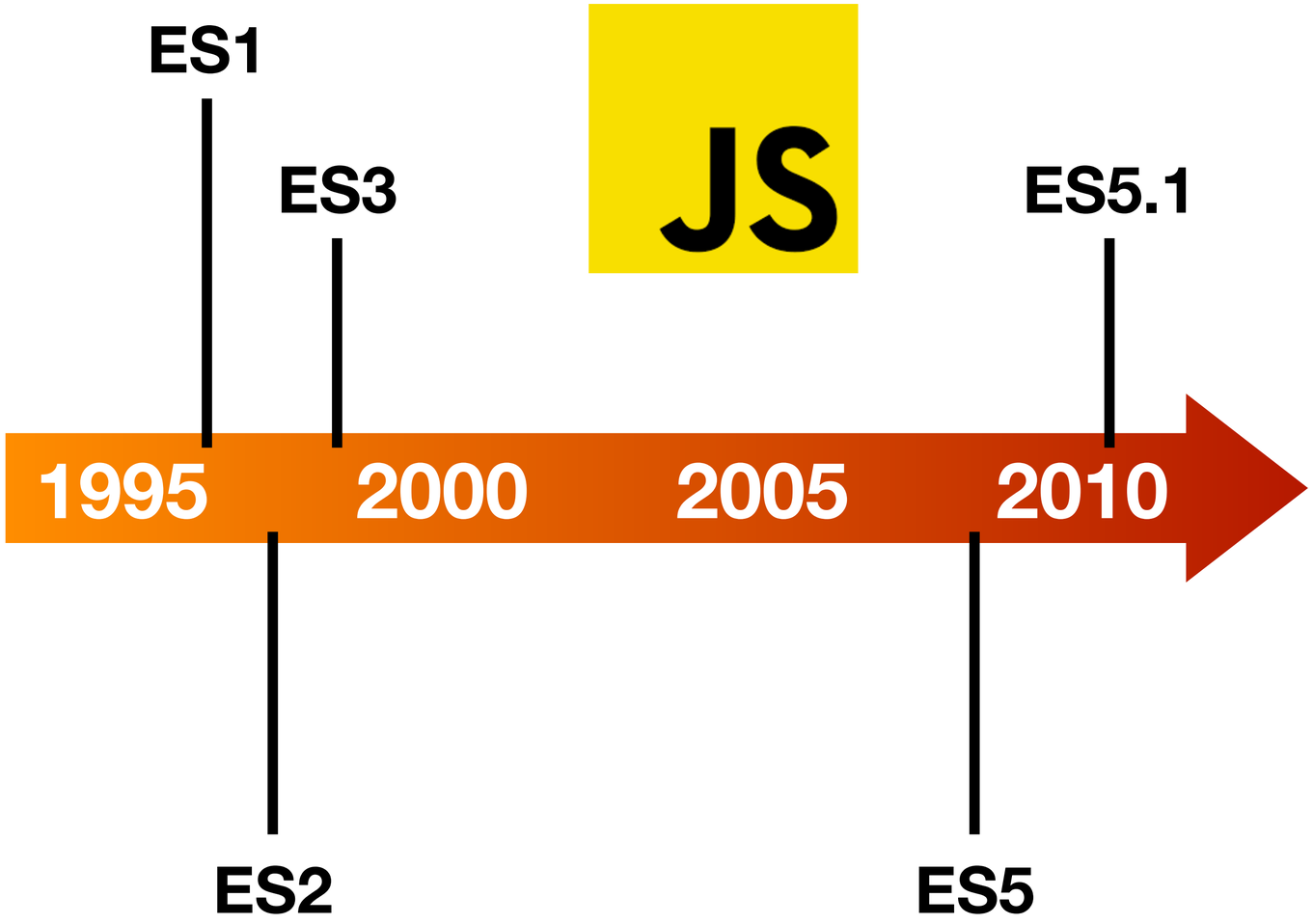
수동 개발



JavaScript용
프로그래밍 언어 도구들

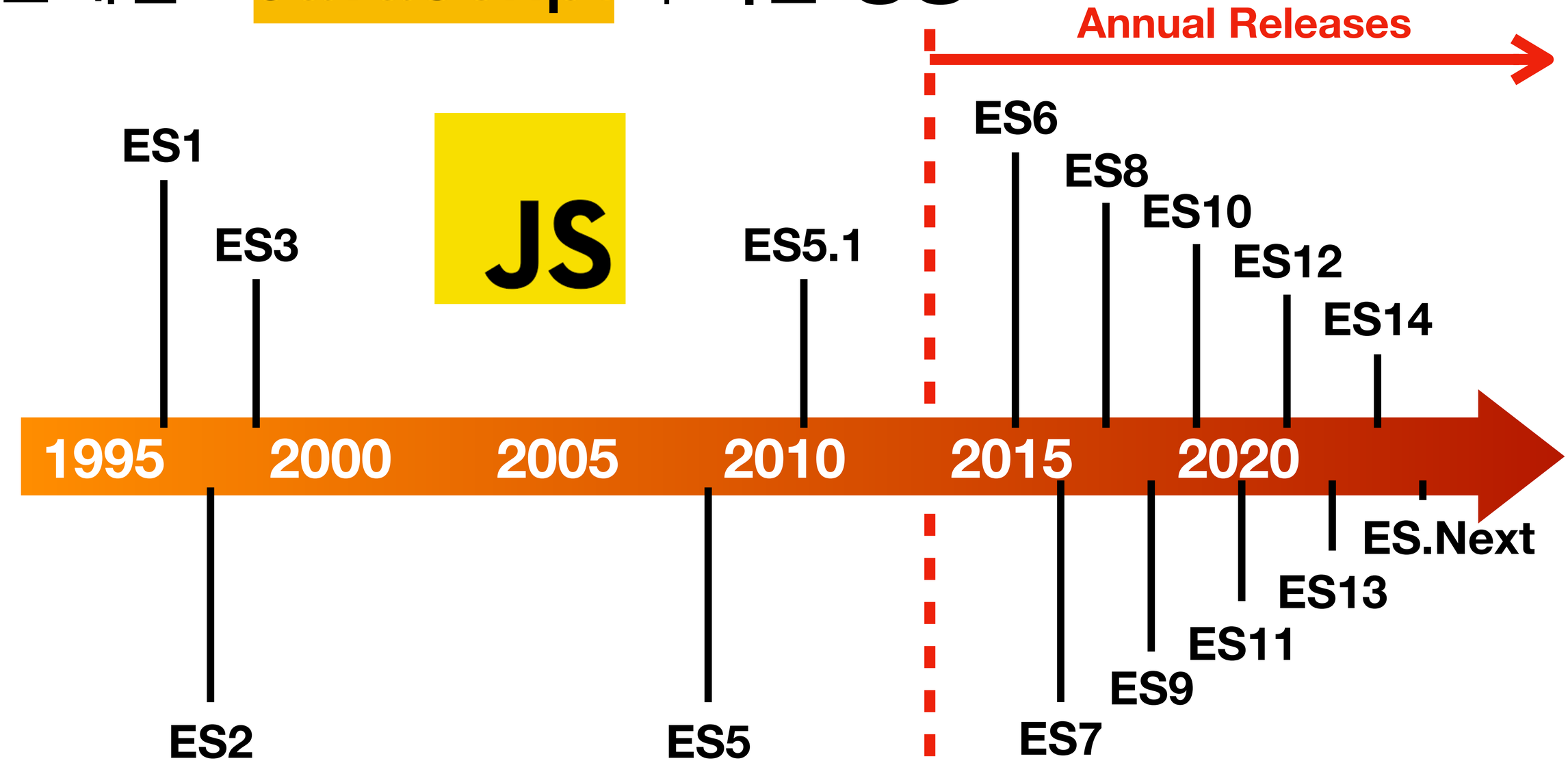
PLRG

문제점 - JavaScript의 빠른 성장



PLRG

문제점 - JavaScript의 빠른 성장



PLRG

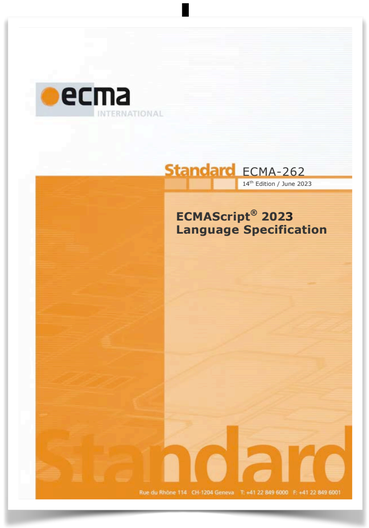
JavaScript 생태계 자동화 연구

PLRG

JavaScript를 위한 프로그래밍 언어 도구

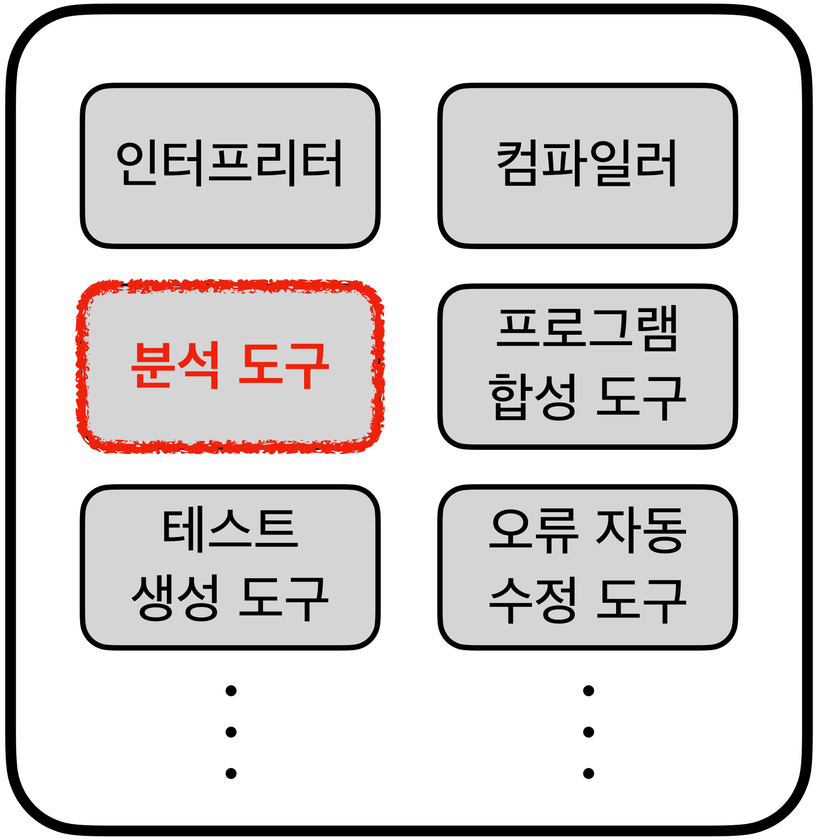
언어 공식 관리 위원회

TC 39



ECMA-262
(JavaScript 언어 명세)

수동 개발



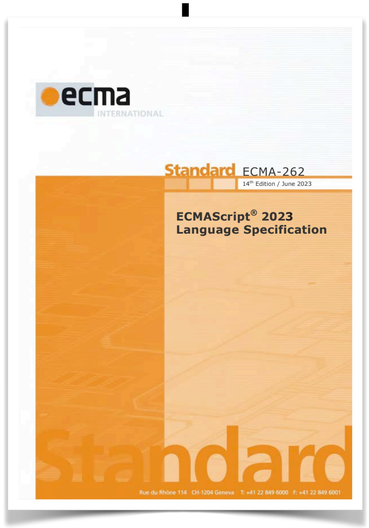
JavaScript용
프로그래밍 언어 도구들

PLRG

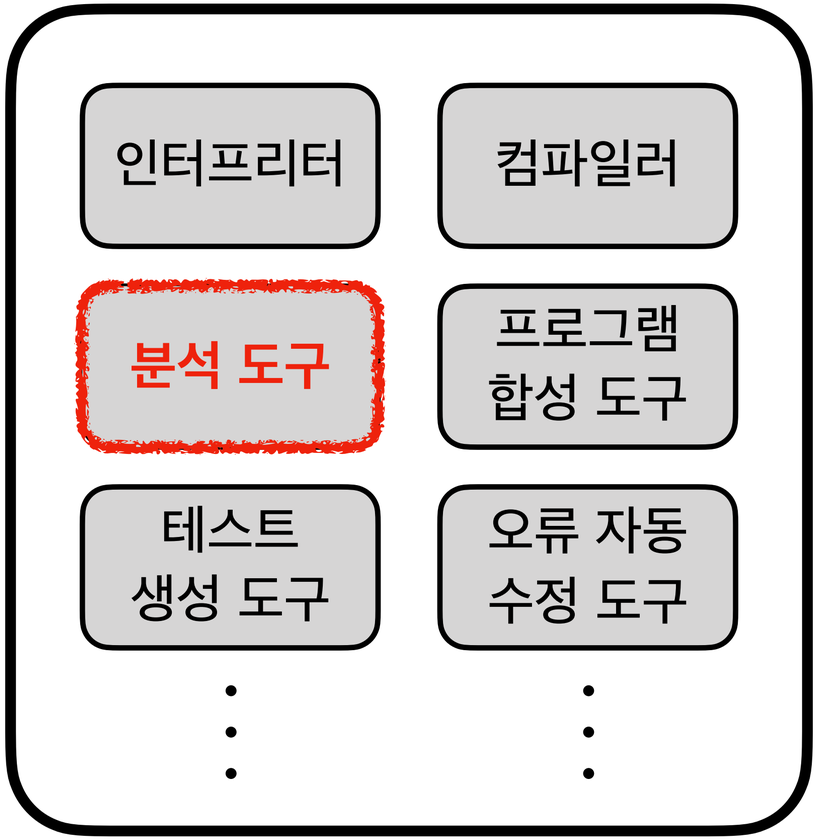
JavaScript를 위한 프로그래밍 언어 도구 자동 생성

언어 공식 관리 위원회

TC 39



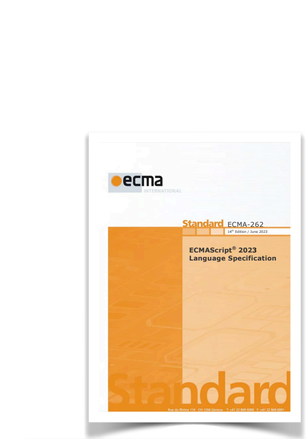
ECMA-262
(JavaScript 언어 명세)



JavaScript용
프로그래밍 언어 도구들

PLRG

JavaScript의 언어 명세에서 **기계화 명세**로 **추출**



ECMA-262

JavaScript
언어 공식 명세

JISET

JS

기계화 명세

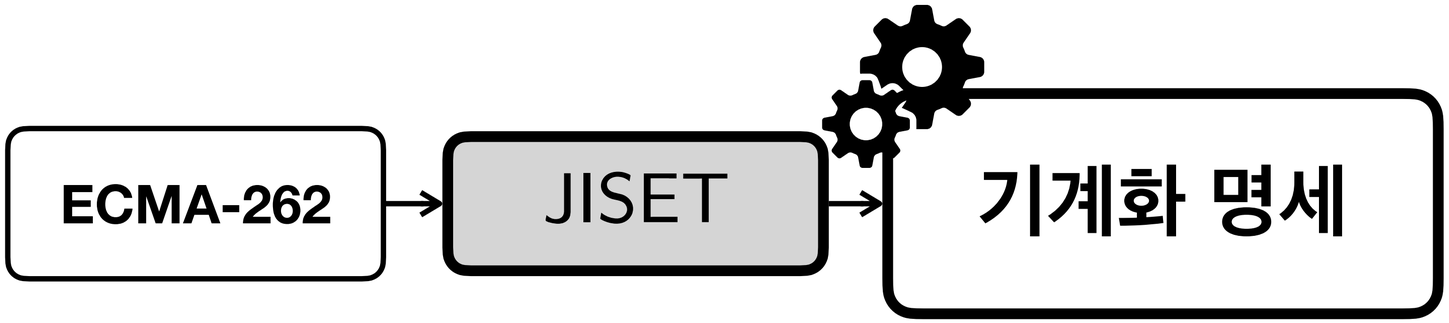
메타 언어(metalanguage)로
기술된 명세

프로그래밍 언어 도구
자동 생성

일치성 테스트
자동 합성

명세 타입
자동 분석

PLRG


ECMA-262
JISET
기계화 명세

ArrayLiteral[Yield, Await] :

[*Elision*opt]

[*ElementList*[?Yield, ?Await]]

[*ElementList*[?Yield, ?Await] , *Elision*opt]

구문(Syntax)

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*.

의미(Semantics)

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

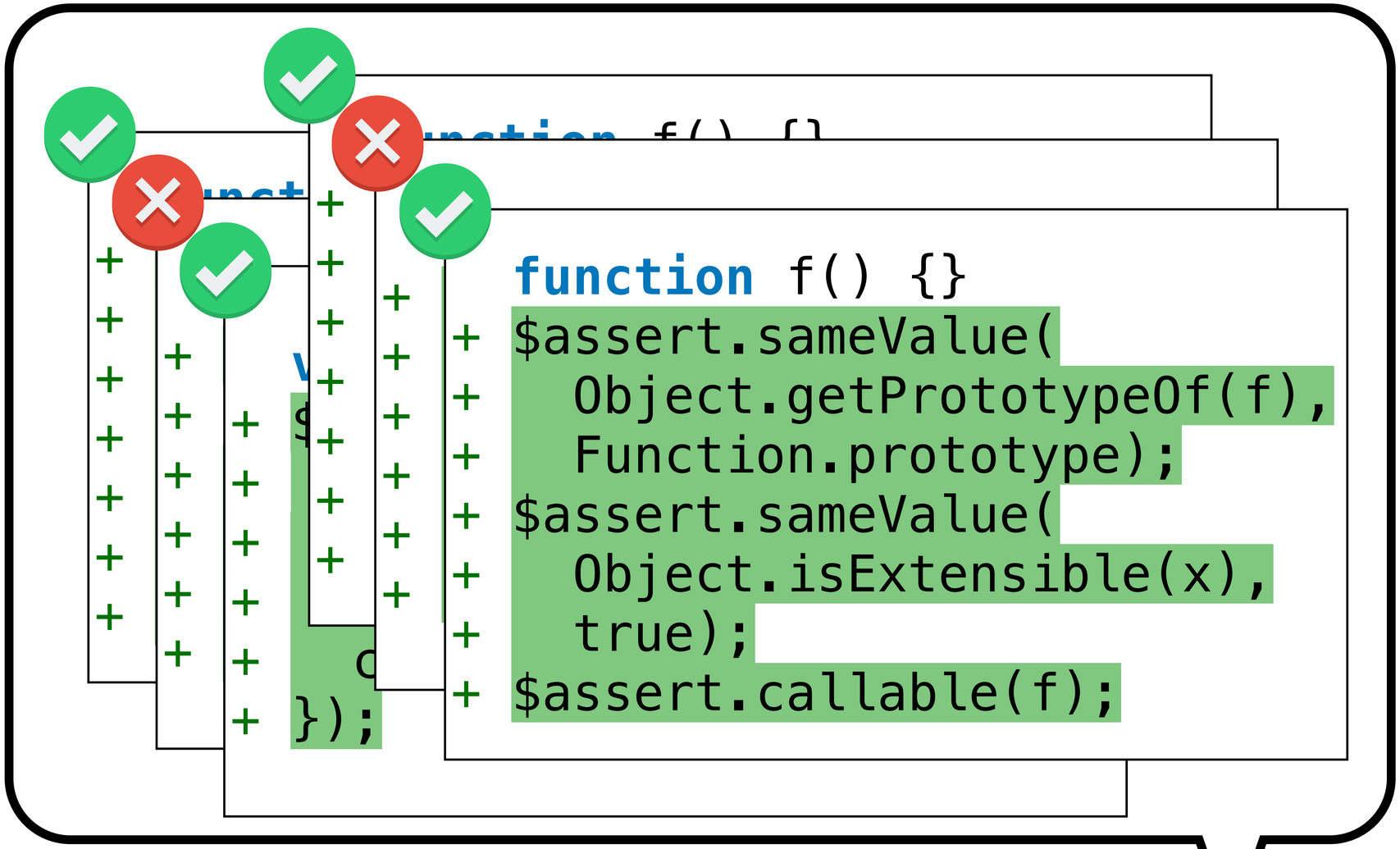
파서(Parser)

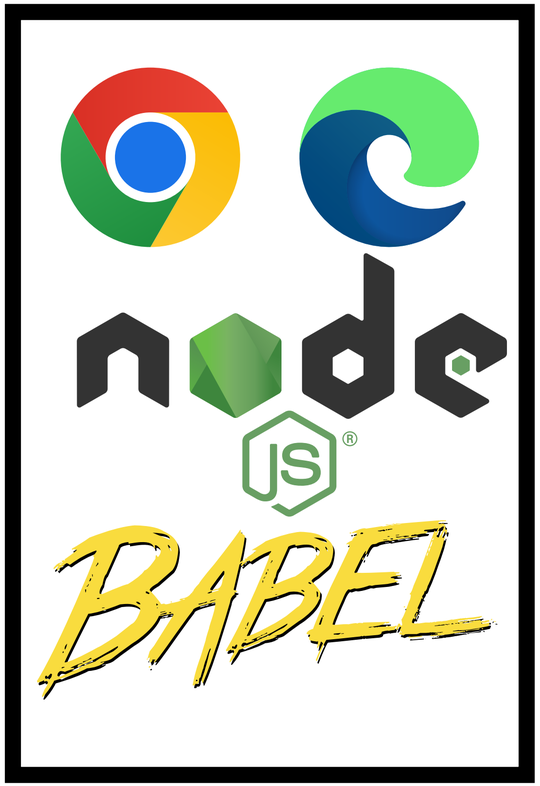
```
syntax def ArrayLiteral[2].Evaluation(
  this, ElementList, Elision
){
  let array = [! (ArrayCreate 0)]
  let nextIndex =
    [? (ElementList.ArrayAccumulation array 0)]
  if (! (= Elision absent))
    [? (Elision.ArrayAccumulation array nextIndex)]
  return array
}
```

메타 언어(metalanguage) 함수

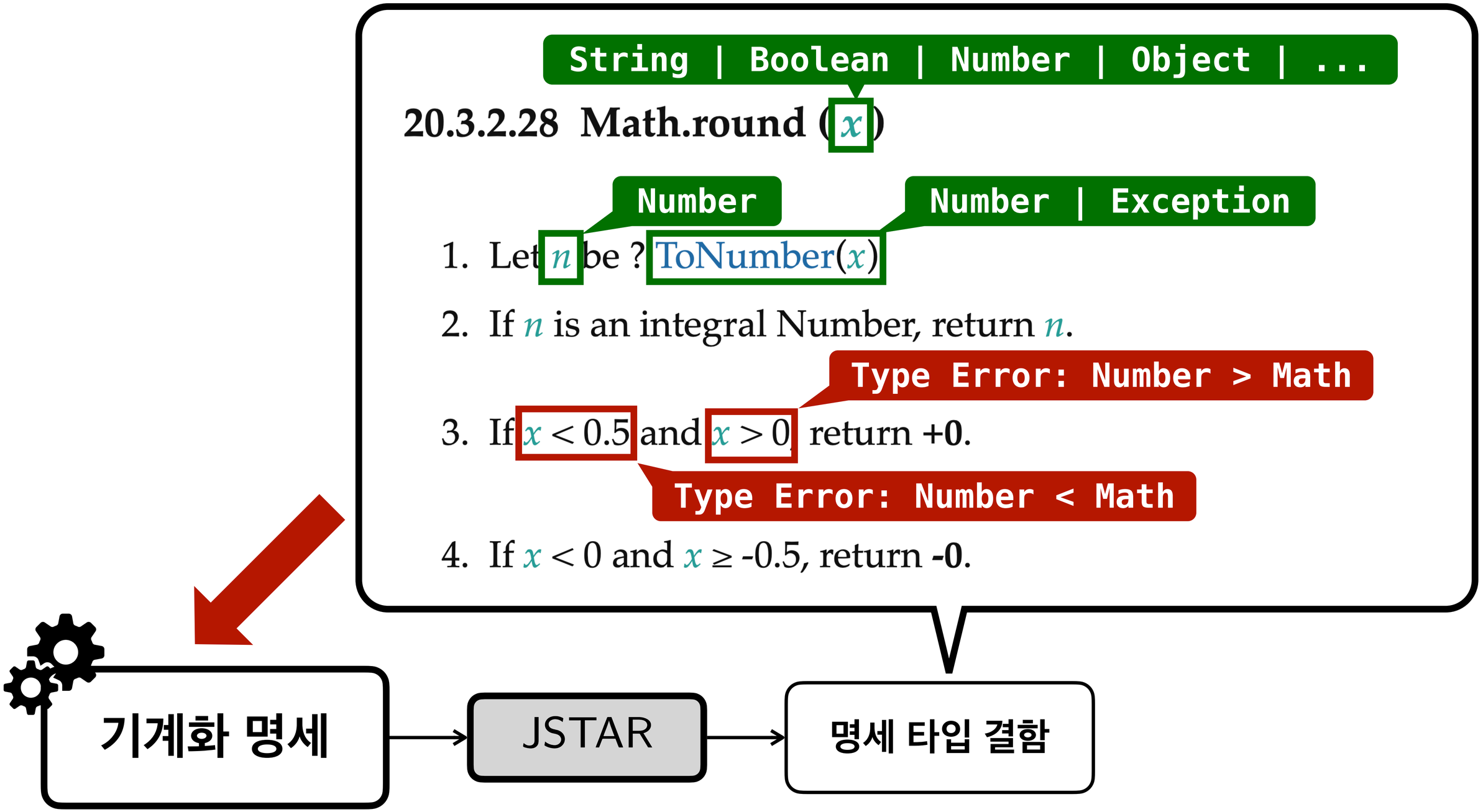
ECMA-262 → JISET → 기계화 명세

기계화 명세

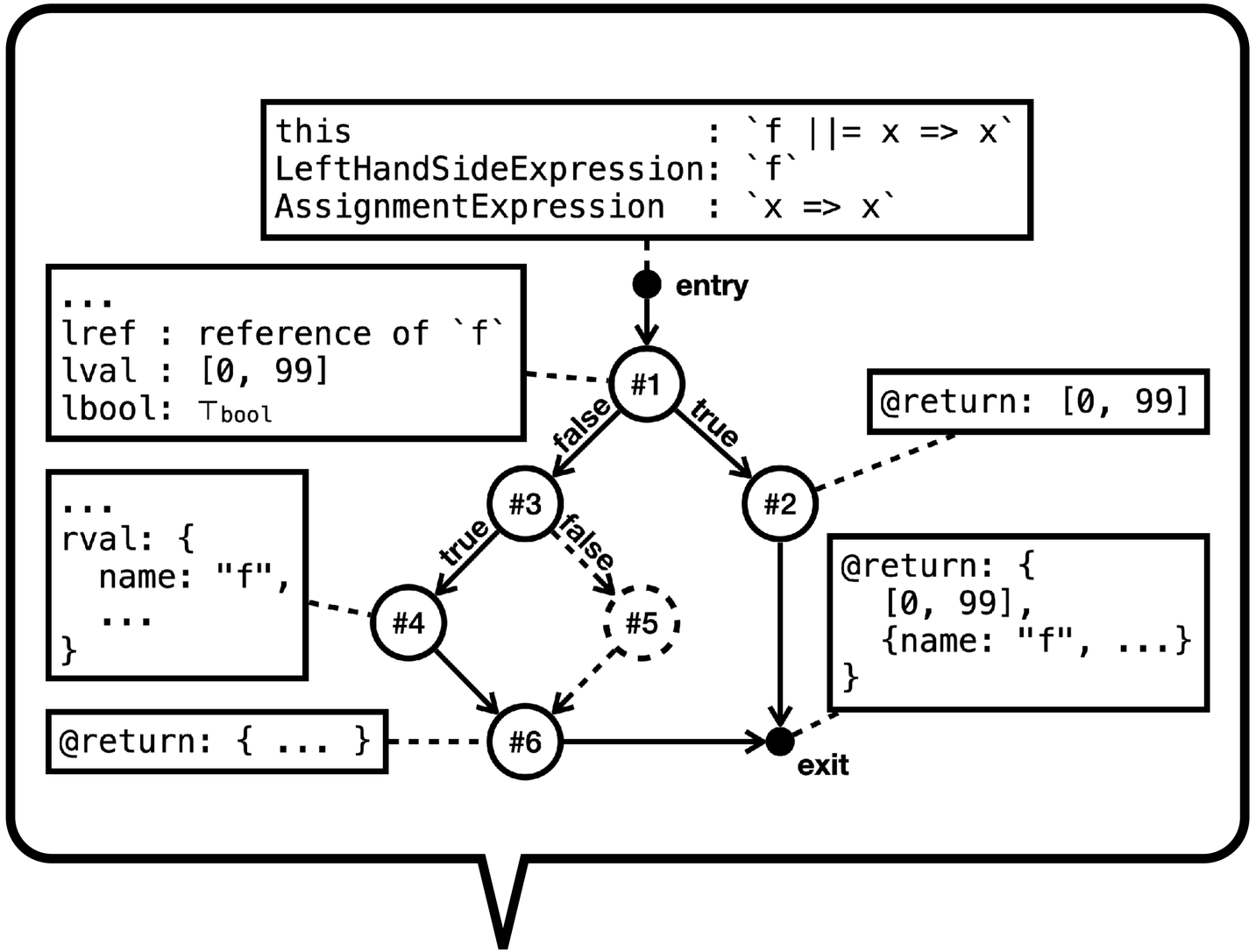




기계화 명세 → JEST → 일치성 테스트


String | Boolean | Number | Object | ...
20.3.2.28 Math.round (x)
Number
Number | Exception
1. Let n be ? ToNumber(x).
2. If n is an integral Number, return n.
Type Error: Number > Math
3. If x < 0.5 and x > 0, return +0.
Type Error: Number < Math
4. If x < 0 and x ≥ -0.5, return -0.
기계화 명세
JSTAR
명세 타입 결함

```
let f = input(0, 99);
f ||= x => x;
let y = f.name;
```



기계화 명세 → JSAVER → 정적 분석 도구

메타 언어(metalanguage)로
기술된 명세

ECMA-262

JavaScript
언어 공식 명세

JISET

기계화 명세 추출

기계화 명세

JSAVER

정적 분석 도구

정적 분석 도구
자동 생성

일치성 테스트

JEST

일치성 테스트 합성

JSTAR

명세 타입 결함

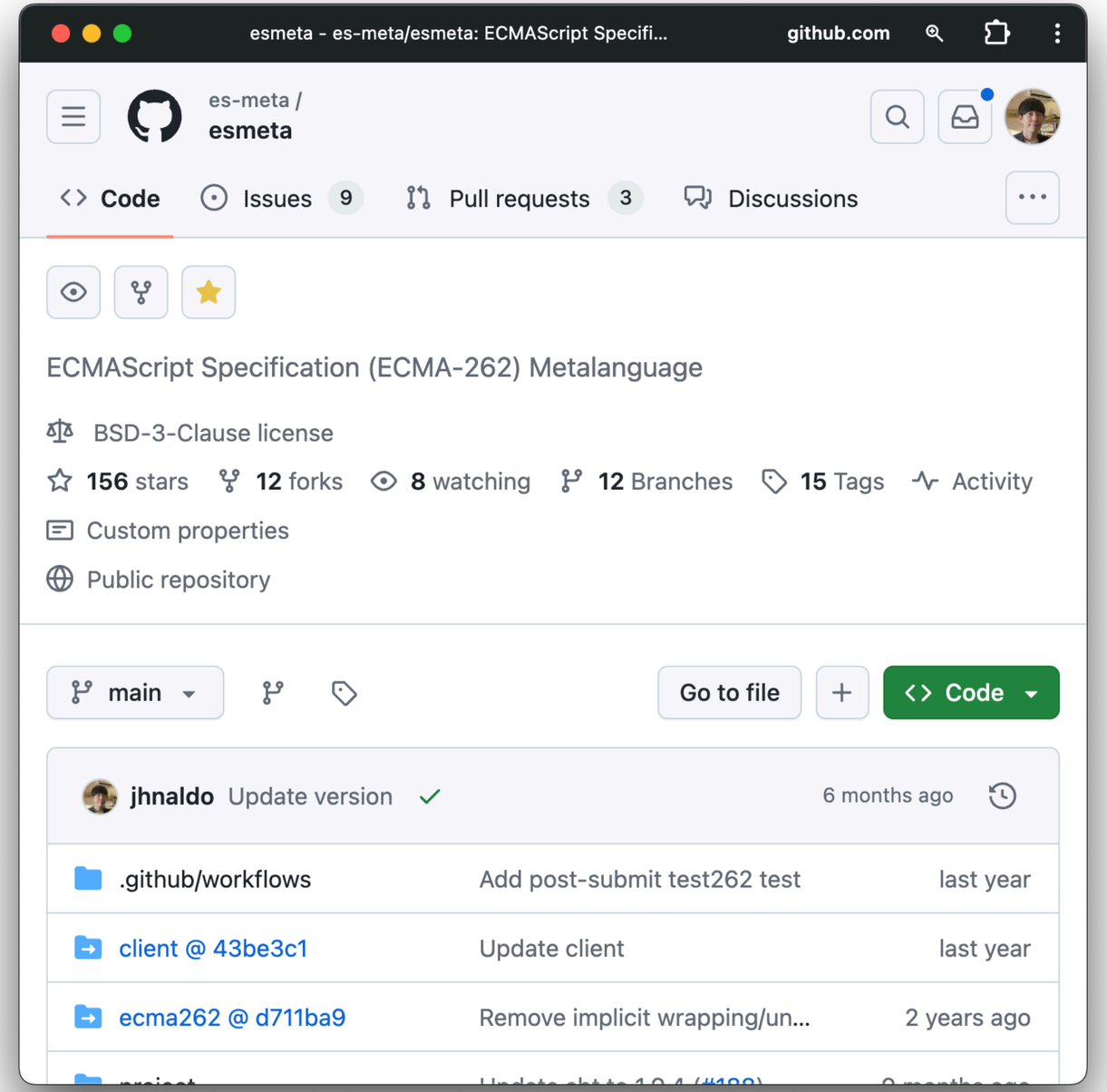
명세 타입 분석

PLRG

JavaScript 언어 명세
공식 검사 도구로 사용 중

ESMETA

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



PLRG