

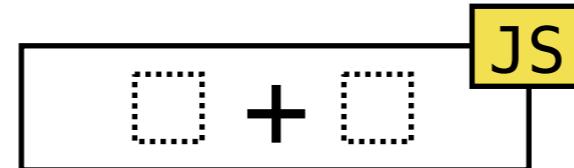
Trusted JavaScript Language Environments with ESMeta

Jihyeok Park

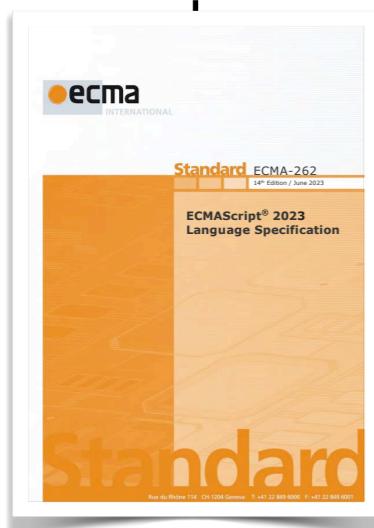


2025.06.17 @ RPLS 2025

Language Specification (ECMA-262) of JavaScript



TC
39



ECMA-262
(JavaScript Spec.)

Syntax

AdditiveExpression [?Yield, ?Await] :

MultiplicativeExpression [?Yield, ?Await]

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

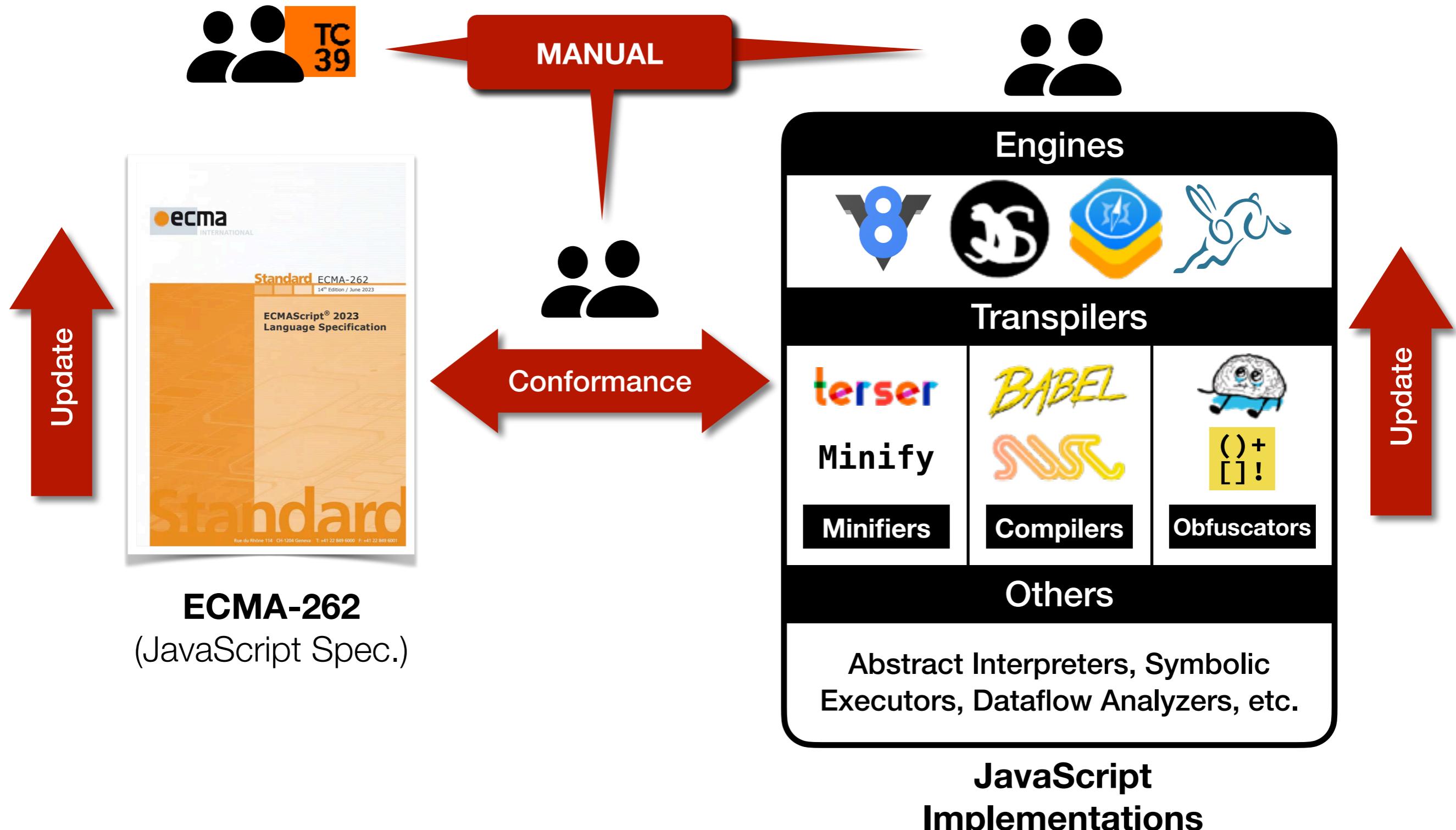
AdditiveExpression [?Yield, ?Await] - *MultiplicativeExpression* [?Yield, ?Await]

AdditiveExpression : *AdditiveExpression* + *MultiplicativeExpression*

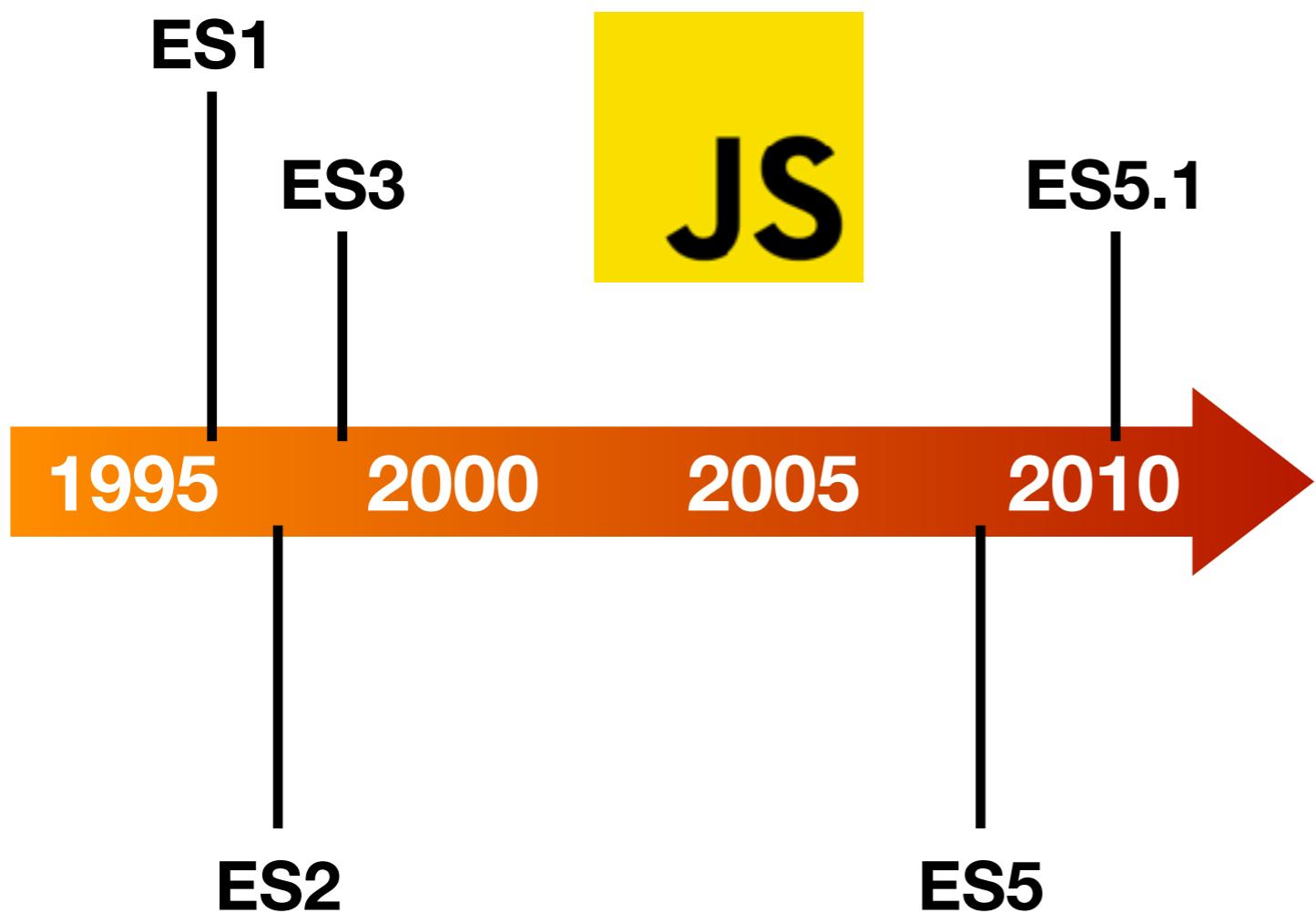
1. Return ? *EvaluateStringOrNumericBinaryExpression*(*AdditiveExpression*, +, *MultiplicativeExpression*).

Semantics

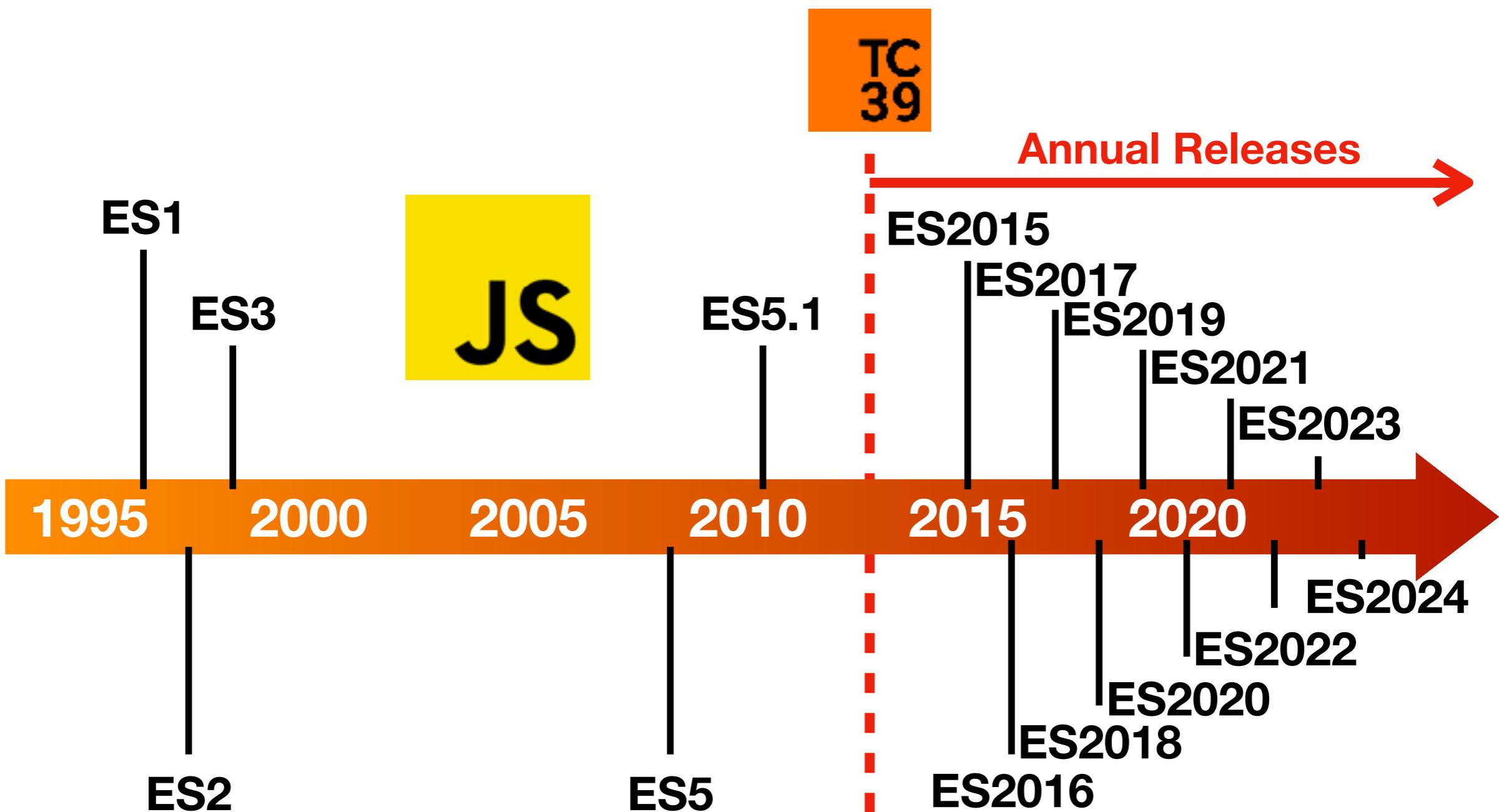
Design and Implementation of JavaScript



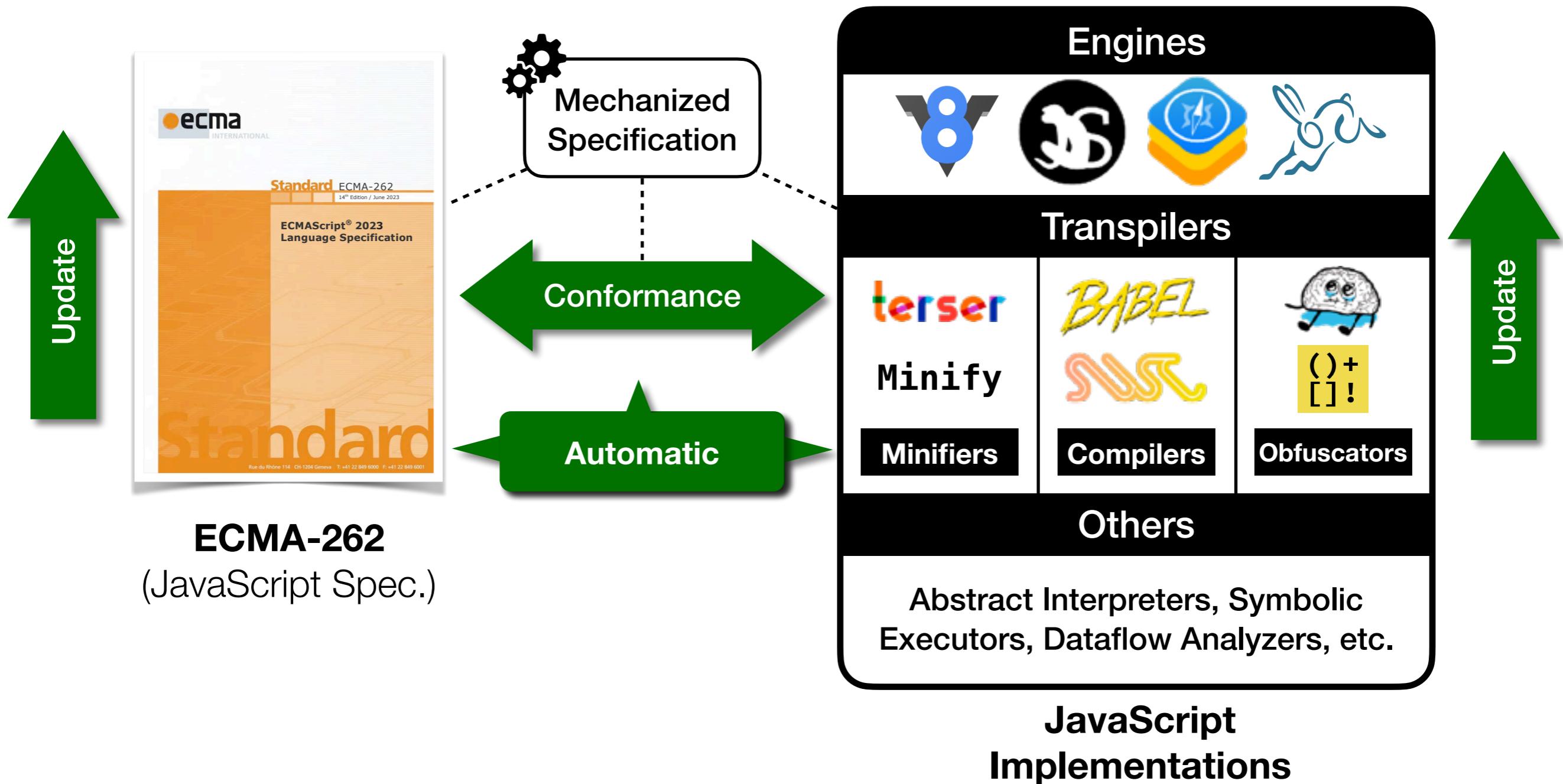
Problem - Fast Evolving JavaScript

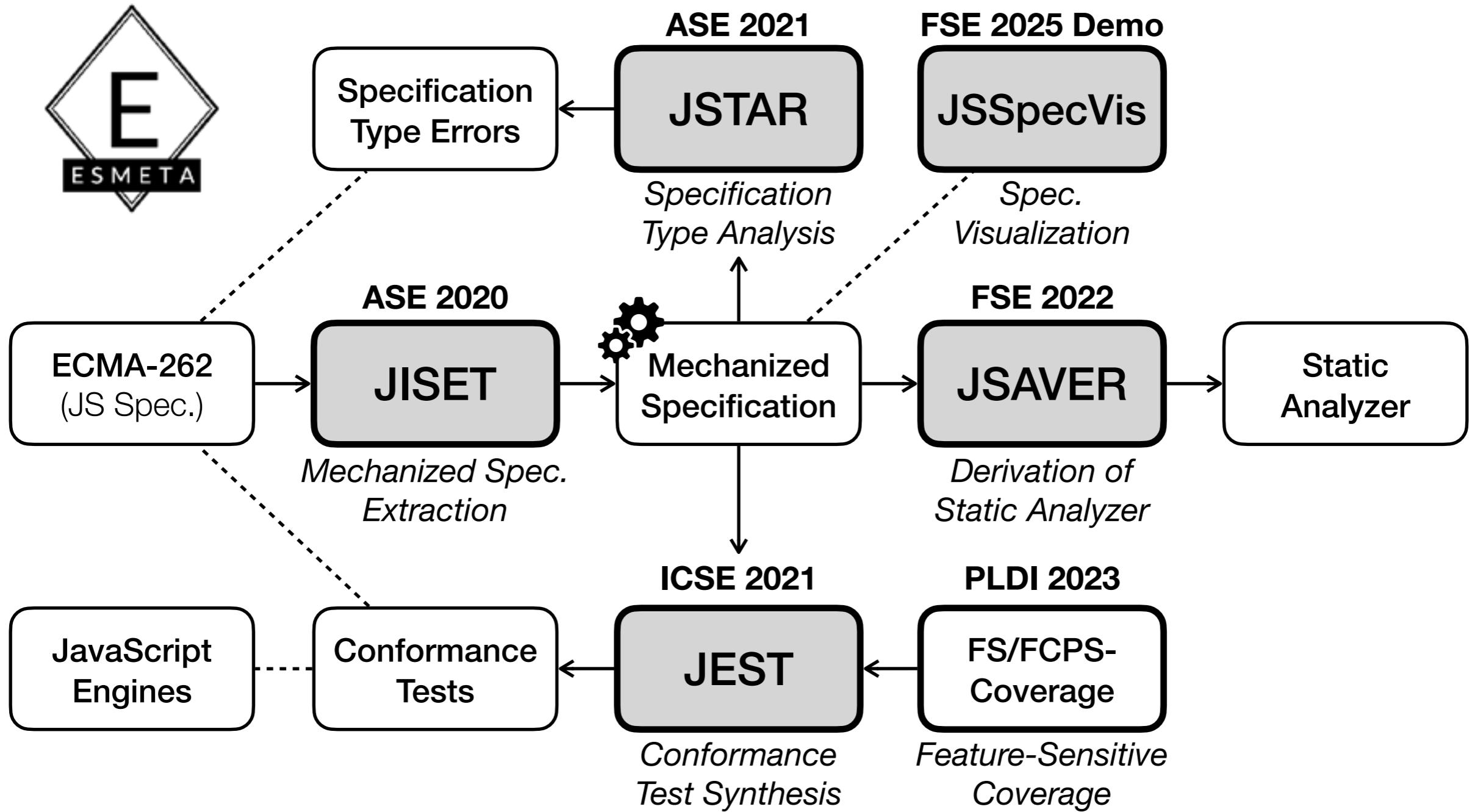


Problem - Fast Evolving JavaScript

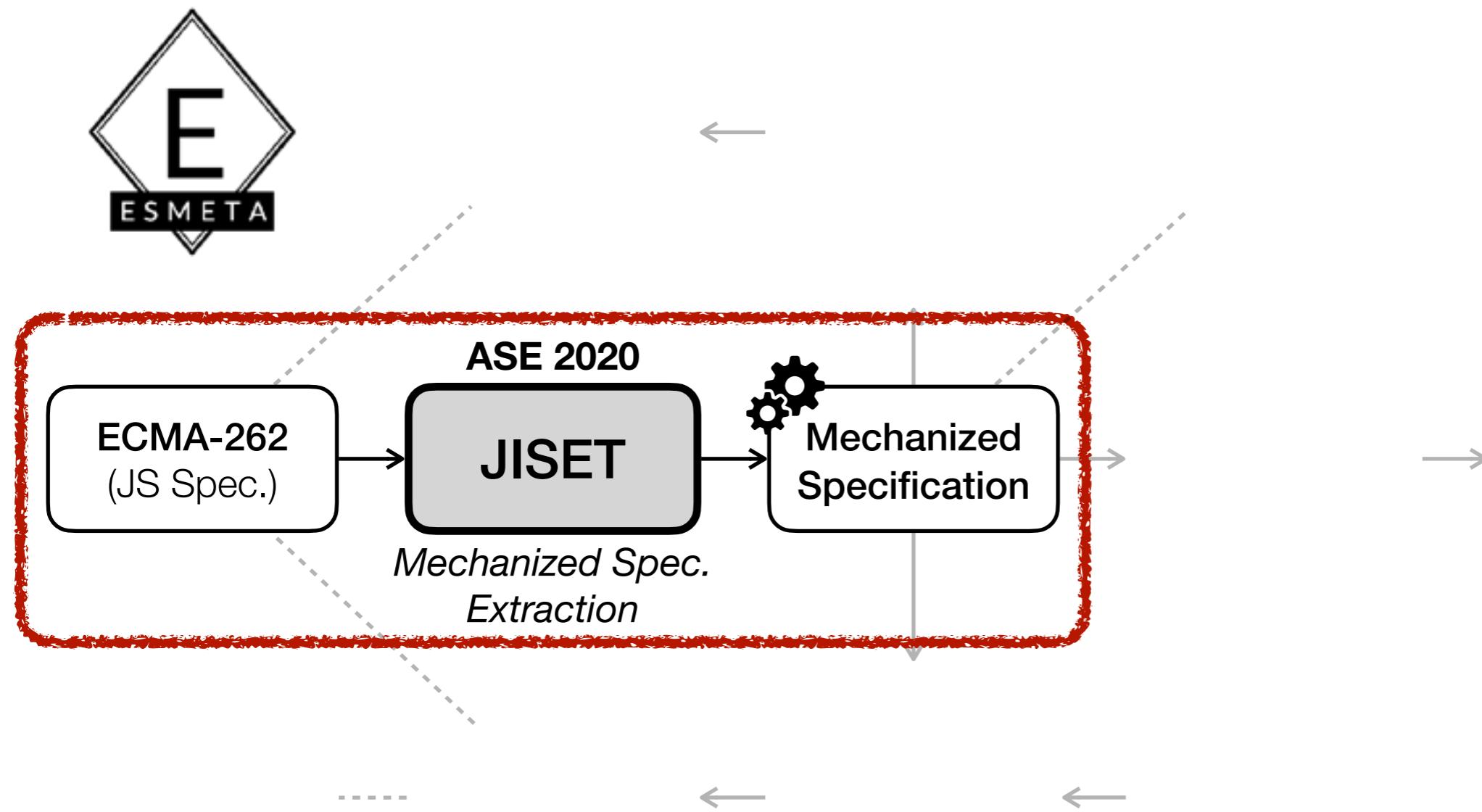


Solution - Mechanized Language Specification

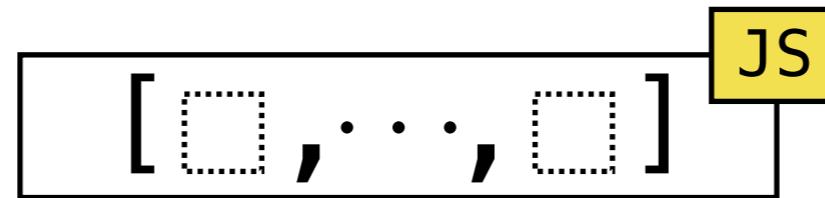




[ASE 2020] J. Park et al., “JISET: JavaScript IR-based Semantics Extraction Toolchain”



JISET - Patterns in Abstract Algorithms



Semantics

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*.

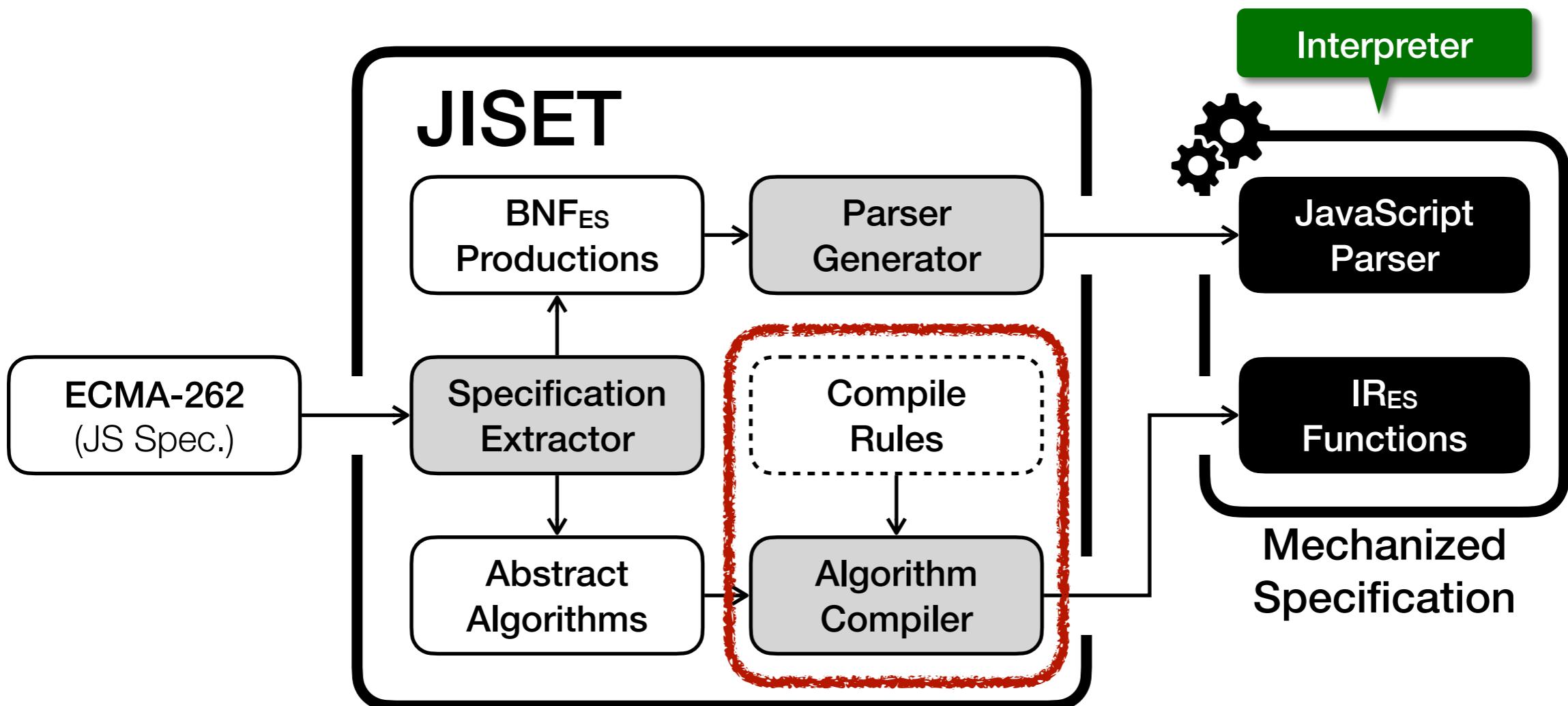
JISET - Metalanguage for ECMA-262

IR_{ES} - Intermediate Representation for ECMA-262

Programs	$\mathfrak{P} \ni P ::= f^*$
Functions	$\mathcal{F} \ni f ::= \text{syntax? def } x(x^*) \{ [\ell : i]^* \}$
Variables	$\mathcal{X} \ni x$
Labels	$\mathcal{L} \ni \ell$
Instructions	$\mathcal{I} \ni i ::= r := e \mid x := \{\} \mid x := e(e^*)$ $\quad \mid \text{if } e \ell \ell \mid \text{return } e$
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	$v \in \mathbb{V} = \mathbb{A} \cup \mathbb{V}^p \cup \mathbb{T} \cup \mathcal{F}$
Primitive Values	$v^p \in \mathbb{V}^p = \mathbb{V}_{\text{bool}} \cup \mathbb{V}_{\text{int}} \cup \mathbb{V}_{\text{str}} \cup \dots$
JS ASTs	$t \in \mathbb{T}$

JISET

(JavaScript IR-based Semantics Extraction Toolchain)



JISET - Algorithm Compilers

Abstract algorithm for *ArrayLiteral* in ES13

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*.

118 compile rules for steps in abstract algorithms

Semantics

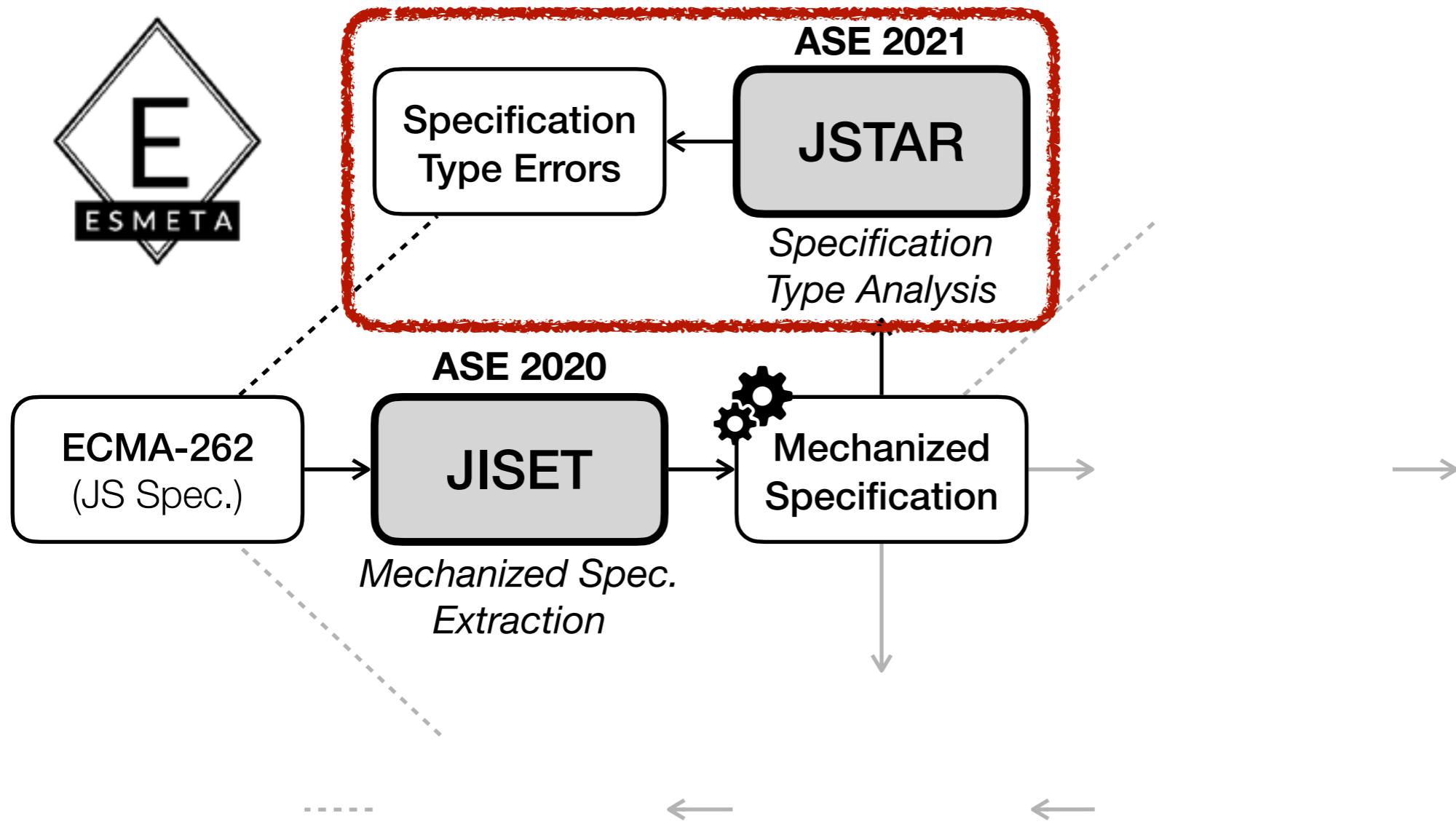
[\square , ..., \square]

JS

```
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
}
```

IR_{ES} function for *ArrayLiteral* in ES13

[ASE 2021] J. Park et al., “JSTAR: JavaScript Specification Type Analyzer using Refinement”



JSTAR - Specification Type Analysis

20.3.2.28 Math.round (x)

$x : \text{String} \mid \text{Boolean} \mid \text{Number} \mid \text{Object} \mid \dots$

$n : \text{Number}$

Filter Exception

Number | Exception

1. Let n be ?`ToNumber(x)`.

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

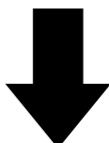
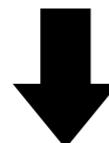
3. If $x < 0.5$ and $x > 0$, return +0.

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

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

...

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



3. If $n < 0.5$ and $n > 0$, return +0.

Fixed

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/575149cf77aebcf3a129e165bd89e14caafc31c>

JSTAR - Evaluation

- Type analysis on **864 versions** of ECMA-262 in 3 years

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
	DuplicatedVar		45 / 46		46 / 47	
Arity	MissingParam	4 / 4	4 / 4	4 / 4	4 / 4	/
Assertion	Assertion	4 / 56	4 / 56	4 / 31	4 / 31	/ -25
Operand	NoNumber	22 / 113	2 / 65	22 / 44	2 / 6	/ -69
	Abrupt		20 / 48		20 / 38	
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

JSTAR - Occurrence Typing (On Going)

22.2.4.1 RegExp (*pattern*, *flags*)

• *pattern*: ESValue

1. Let *patternIsRegExp* be $\text{?IsRegExp}(\text{pattern})$.

Normal[Boolean]
{ Normal[True] => *pattern*: Object }

RETURN

• *patternIsRegExp*: Boolean { True => *pattern*: Object }
pattern: ESValue

2. ...

b. If *patternIsRegExp* is true and *flags* is undefined, then

• *patternIsRegExp*: True { True => *pattern*: Object }
pattern: ESValue → *pattern*: Object

i. Let *patternConstructor* be $\text{?Get}(\text{pattern}, \text{"constructor"})$.

...

5. Else if *patternIsRegExp* is true, then

requires Object

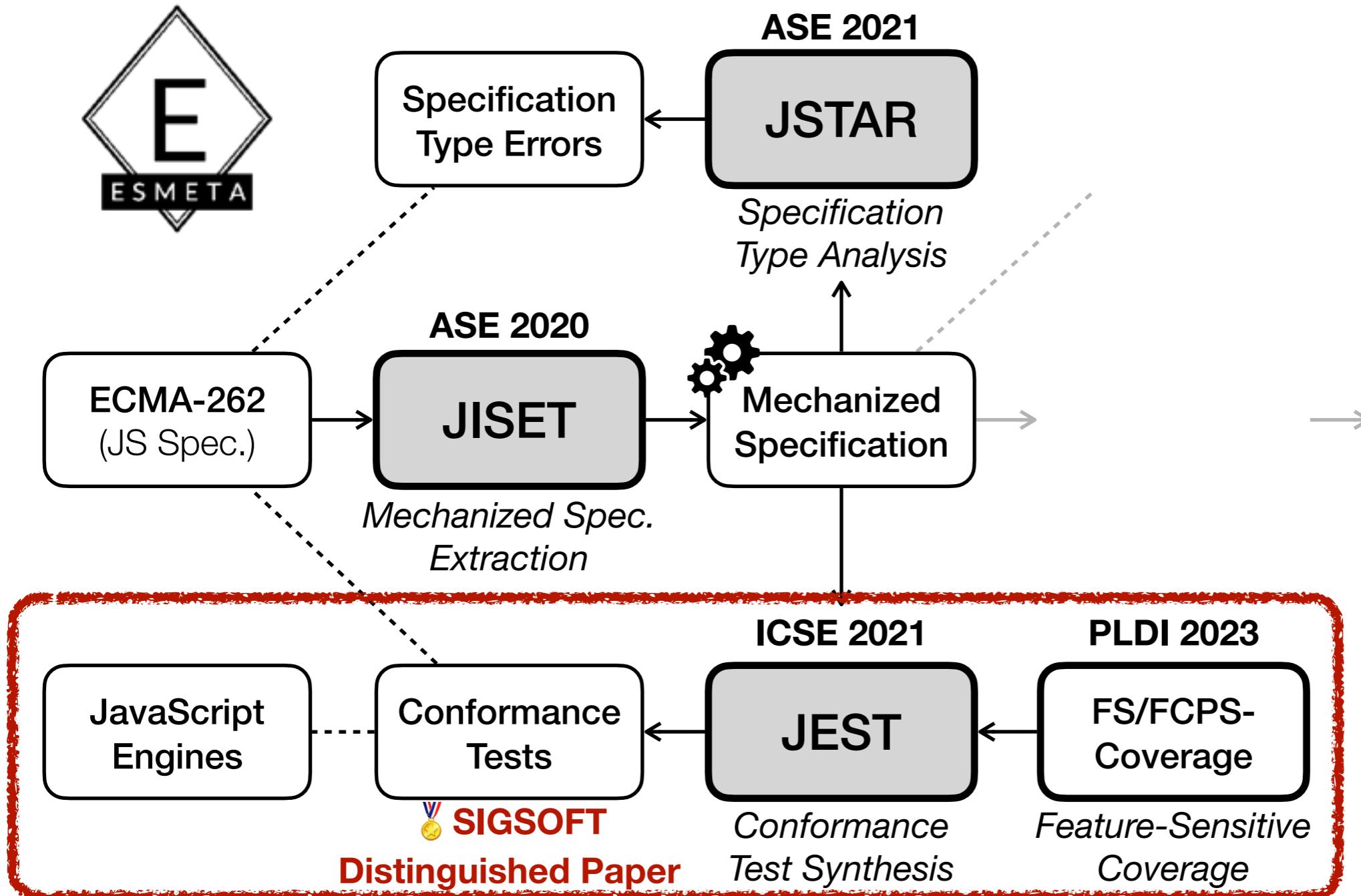
• *patternIsRegExp*: True { True => *pattern*: Object }
pattern: ESValue → *pattern*: Object

a. Let *P* be $\text{?Get}(\text{pattern}, \text{"source"})$.

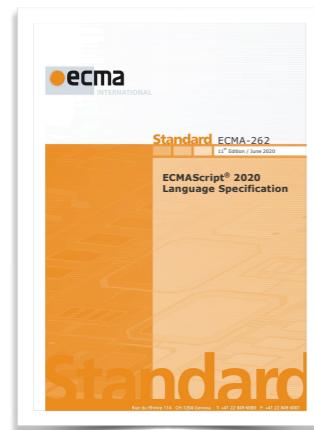
requires Object

[ICSE 2021] J. Park et al., “JEST: N +1-version Differential Testing of Both JavaScript Engines and Specification”

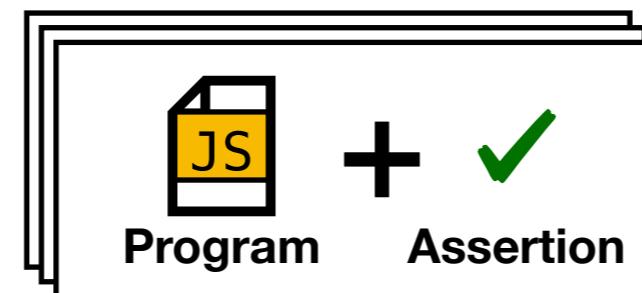
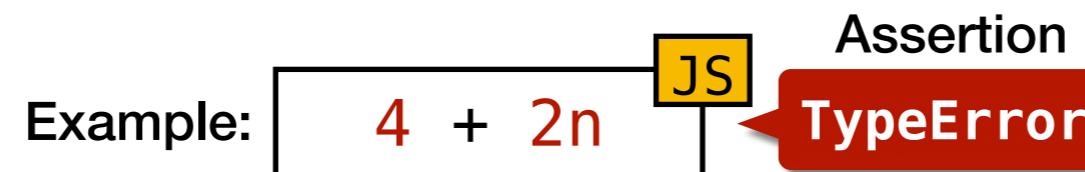
[PLDI 2023] J. Park et al., “Feature-Sensitive Coverage for Conformance Testing of Programming Language Implementations”



Conformance of JavaScript Engines



ECMA-262
(JavaScript Spec.)



Conformance Tests



Test262
(Official Test Suite)



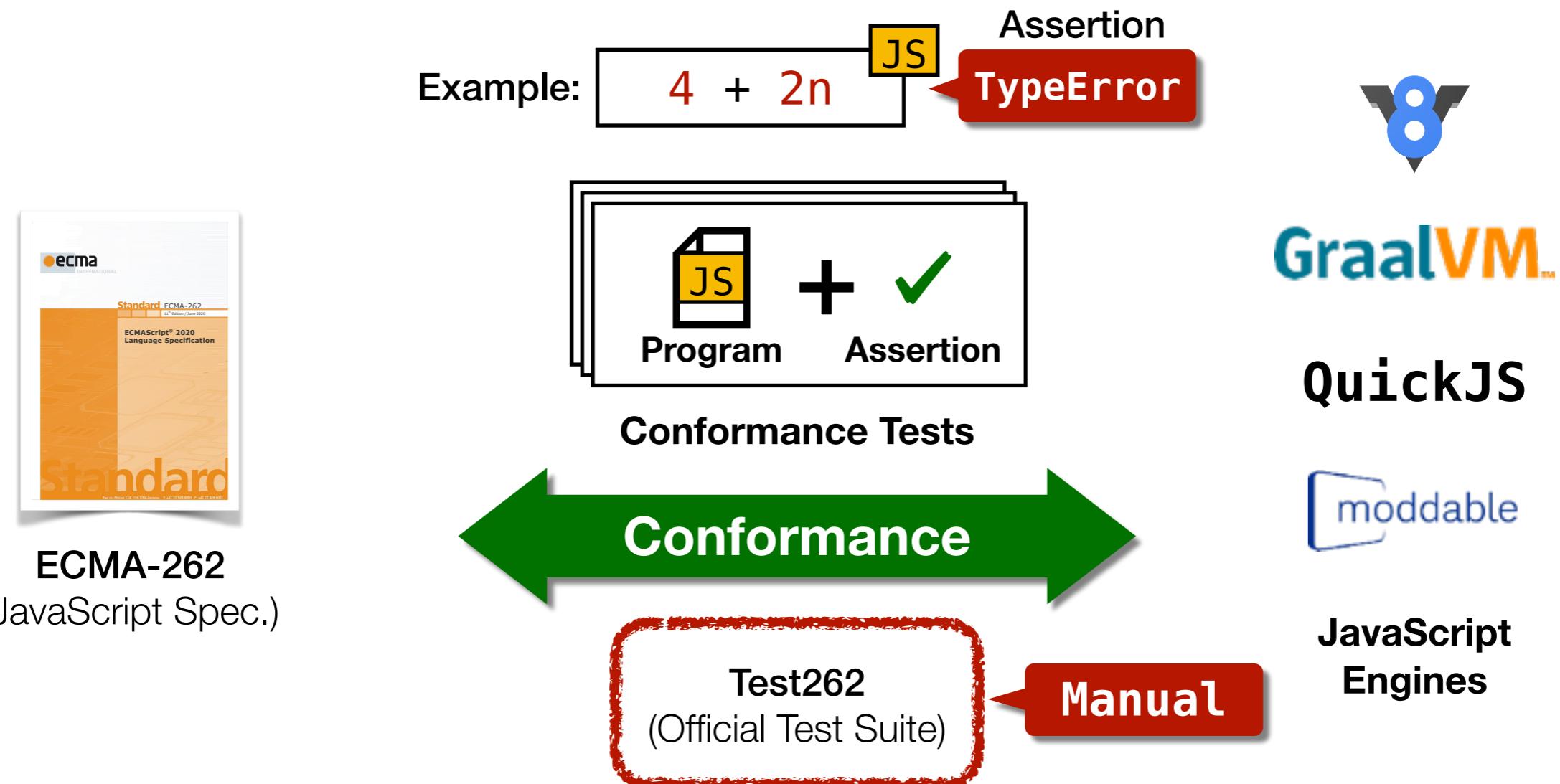
GraalVM

QuickJS



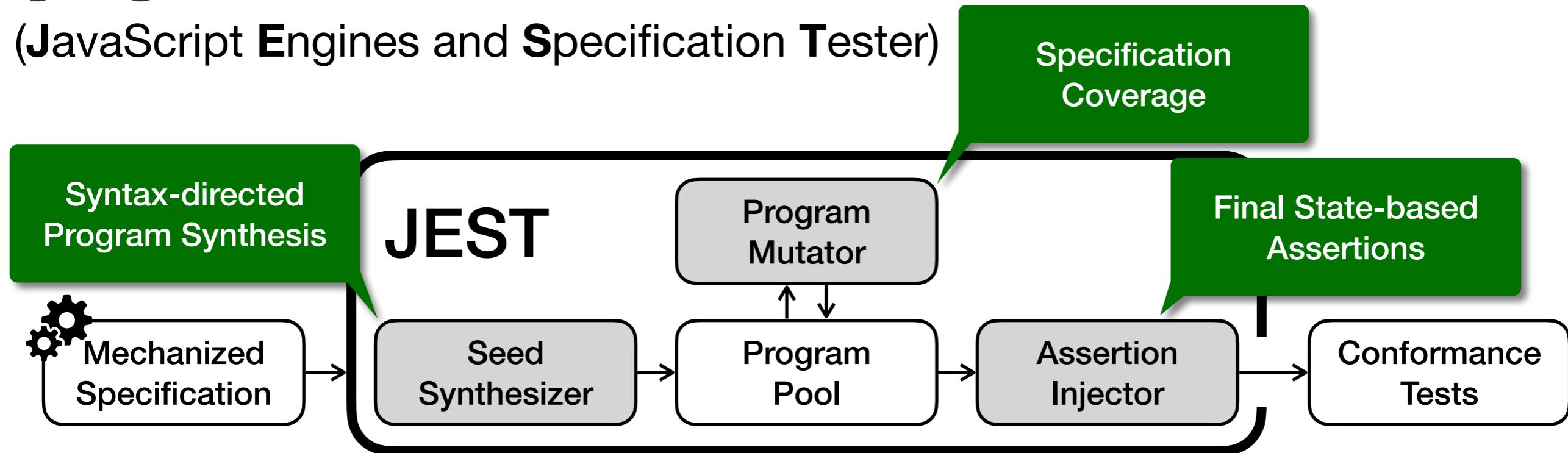
JavaScript
Engines

Problem - Manual Approach



JEST

(JavaScript Engines and Specification Tester)



Program Pool

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

• • •

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

• • •

• • •

• • •

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

• • •

JEST - Specification Coverage

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,

3 + 2

JS

1n - 2n

JS

4 * 2n

JS

JEST - Feature-Sensitive Coverage

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,

Language Features

Addition

3 + 2 JS

1n + 2n JS

17 + 2n JS

Subtraction

4 - 7 JS

7n - 2n JS

1n - 23 JS

Multiplication

2 * 9 JS

9n * 3n JS

3n * 45 JS

•

•

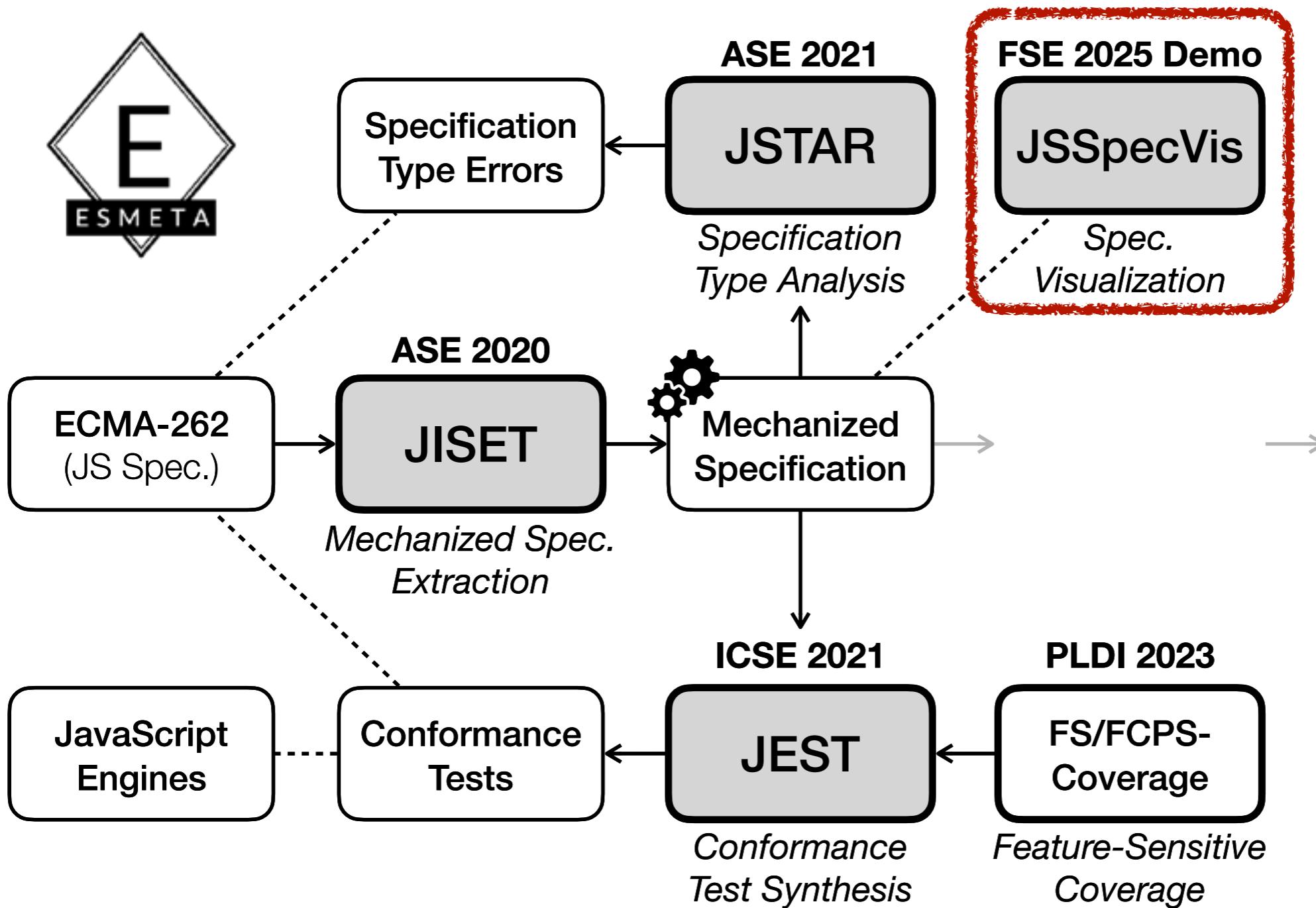
•

JEST - Evaluation

42 Bugs
In Engines

94 Bugs
In Transpilers

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	26	26
	GraalJS	v22.2.0	2022.07.26	11	11	11
	SpiderMonkey	v107.0b4	2022.10.24	2	4	4
	Total			42	44	45
Transpiler	Babel	v7.19.1	2022.09.15	37	37	39
	SWC	v1.3.10	2022.10.21	37	37	47
	Terser	v5.15.1	2022.10.05	19	19	19
	Obfuscator	v4.0.0	2022.02.15	1	2	7
	Total			94	95	112
Total				136	139	157



JSSpecVis - Specification Visualizer

The screenshot shows a browser-based interface for debugging ECMAScript code. On the left is a JavaScript code editor with the following code:

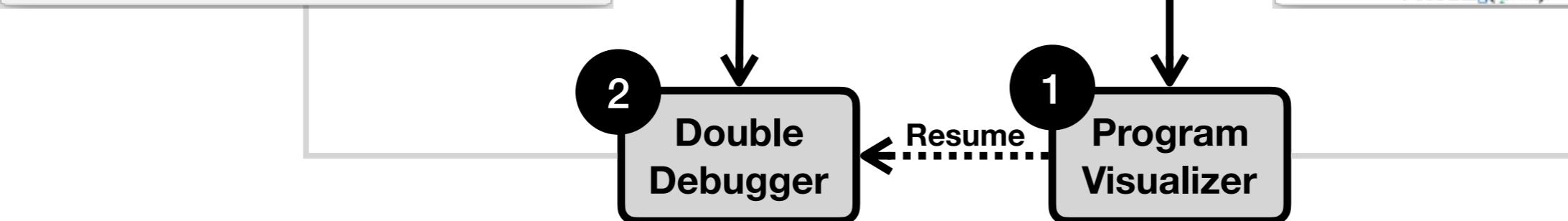
```
function add(left, right) {  
    left + right;  
}  
  
add(1, 1);
```

On the right is a sidebar titled "ECMA-262 Specification Environment" which lists the following properties:

- *lval*: true
- *opText*: "+"
- *rval*: 1n

Below this is the "JavaScript Environment" section with the following properties:

- *this*: undefined
- *left*: Record[MutableBinding]
 - *BoundValue*: true
 - *mutable*: true
 - *initialized*: true
 - *strict*: false
 - *right*: Record[MutableBinding]
 - *BoundValue*: 1n
 - *mutable*: true



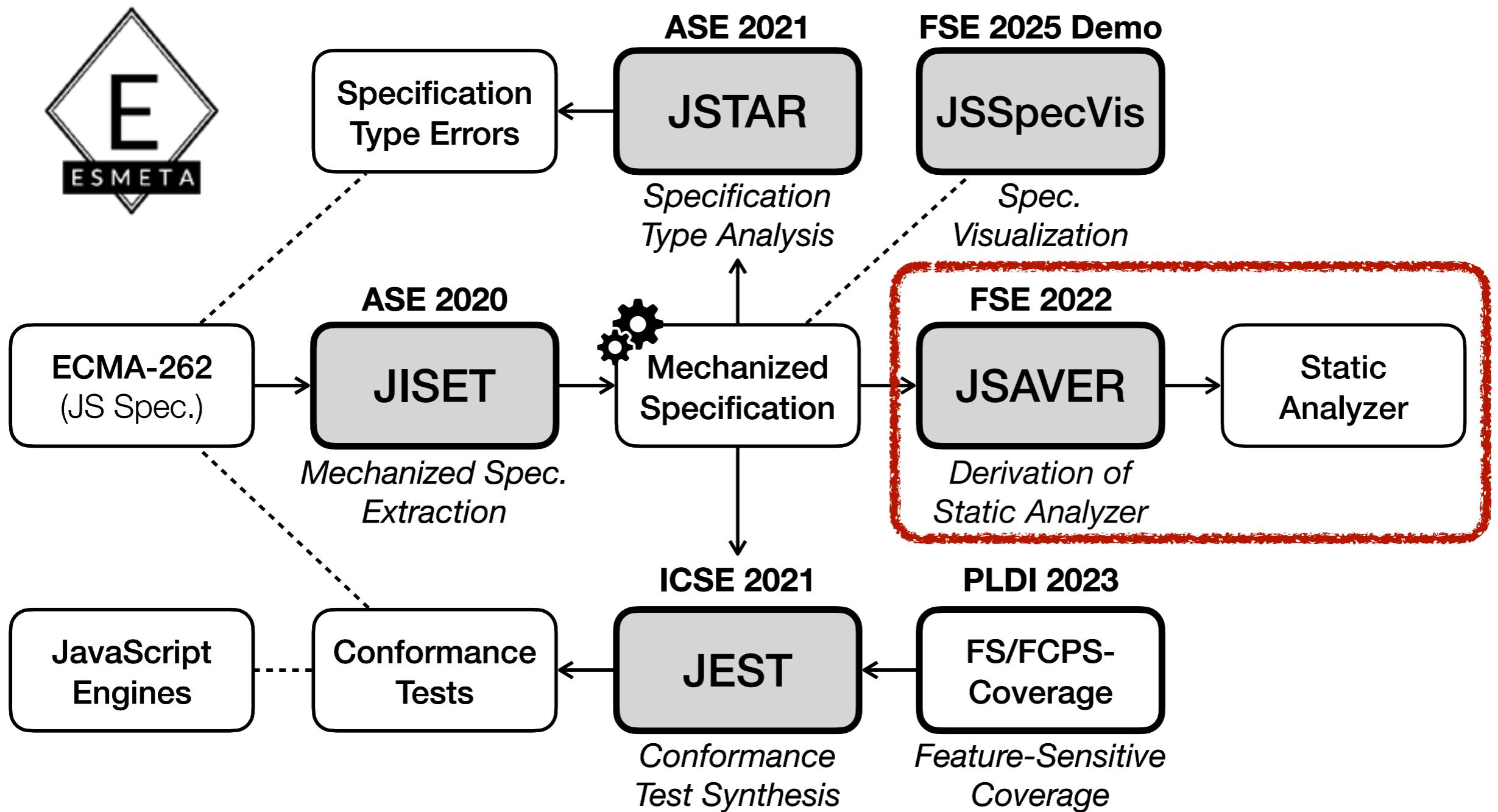
The screenshot shows a detailed view of the **ApplyStringOrNumericBinaryOperator** operation in the ECMA-262 specification.

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

The abstract operation **ApplyStringOrNumericBinaryOperator** takes arguments *lval* (an ECMAScript language value), *opText* (★★, ★, /, %, +, -, <<, >>, >>>, &, ^, or |), and *rval* (an ECMAScript language value) and returns either a **normal completion** containing either a String, a BigInt, or a Number, or a **throw completion**. It performs the following steps when called:

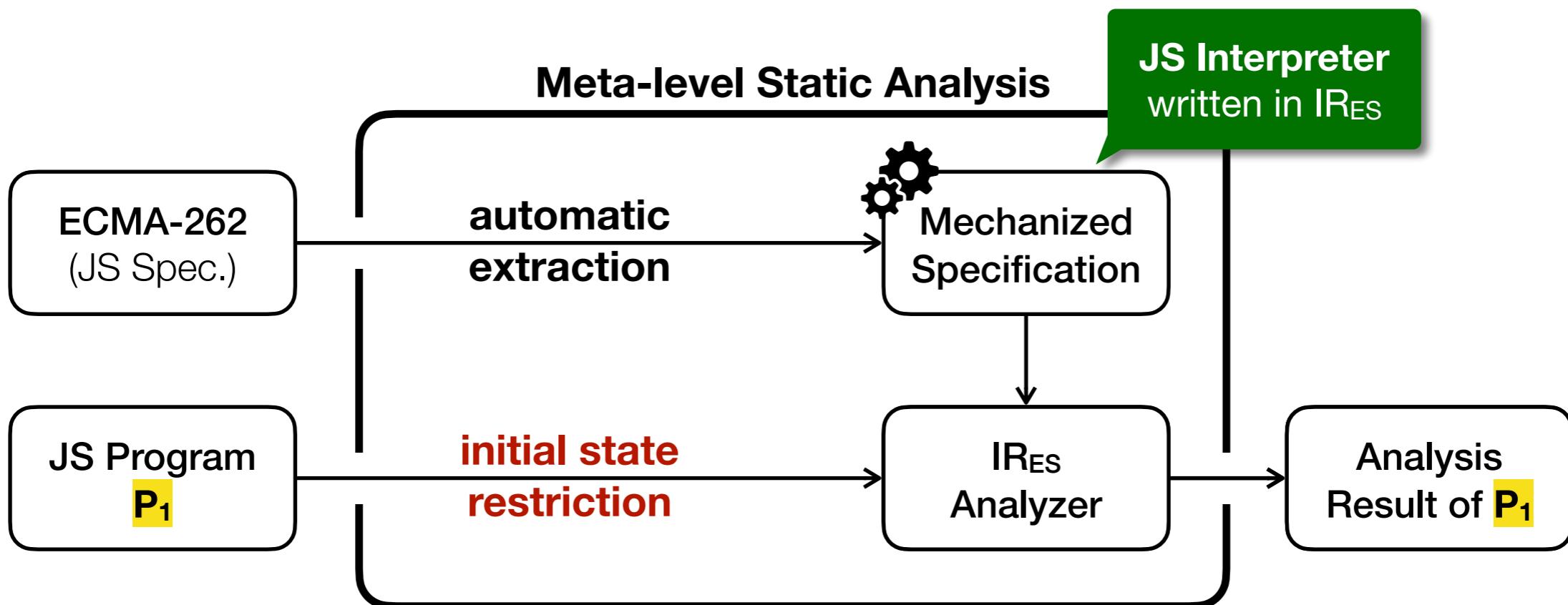
1. If *opText* is +, then
 - a. Let *lprim* be ? ToPrimitive(*lval*).
 - b. Let *rprim* be ? ToPrimitive(*rval*).
 - c. If *lprim* is a String or *rprim* is a String, then
 - i. Let *lstr* be ? ToString(*lprim*).
 - ii. Let *rstr* be ? ToString(*rprim*).
 - iii. Return the string-concatenation of *lstr*.

[FSE 2022] J. Park et al., “Automatically Deriving JavaScript Static Analyzers from Specifications using Meta-level Static Analysis”



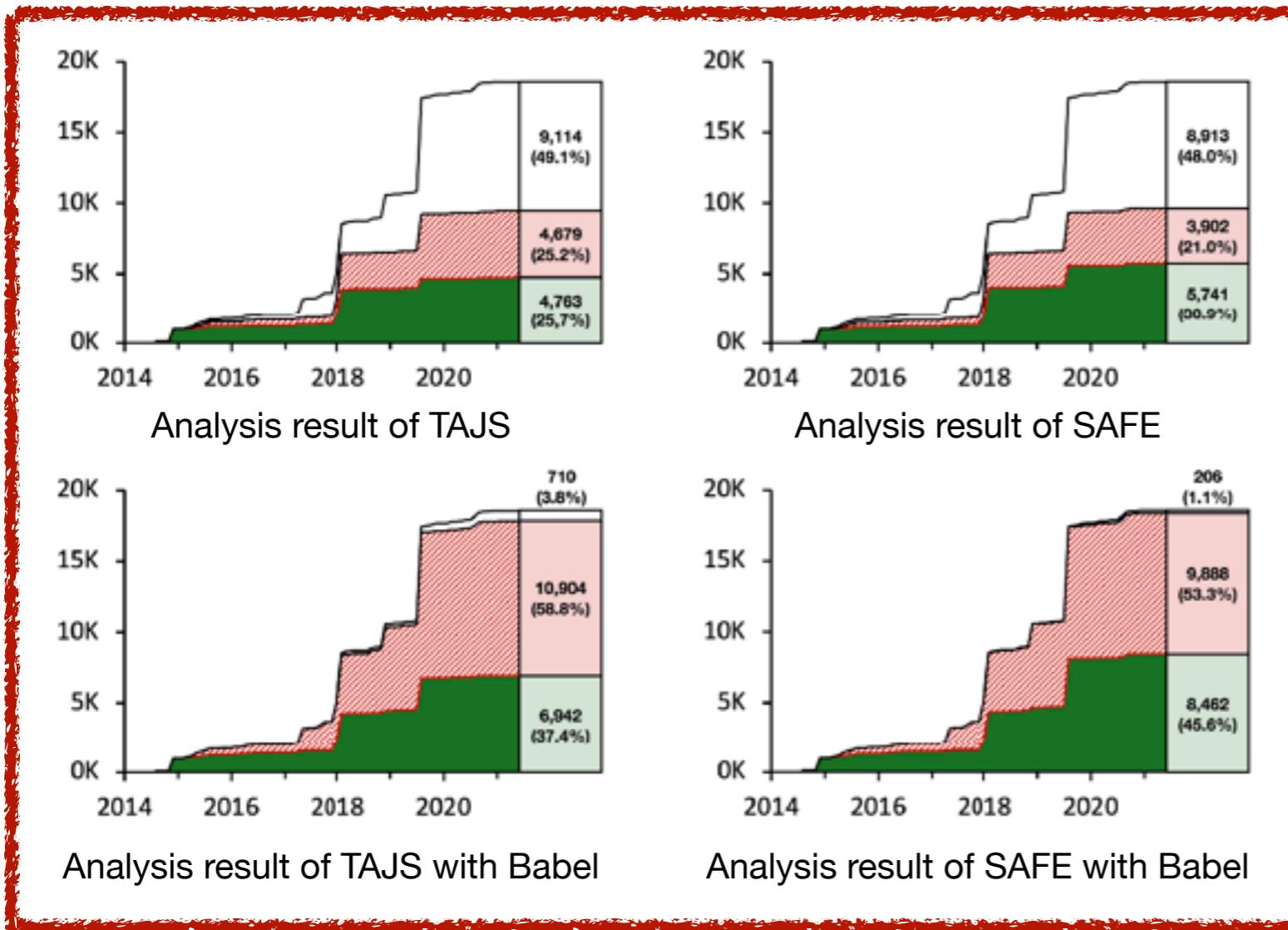
Meta-Level Static Analysis

How to perform **static analysis** on **JavaScript** programs using language specification?

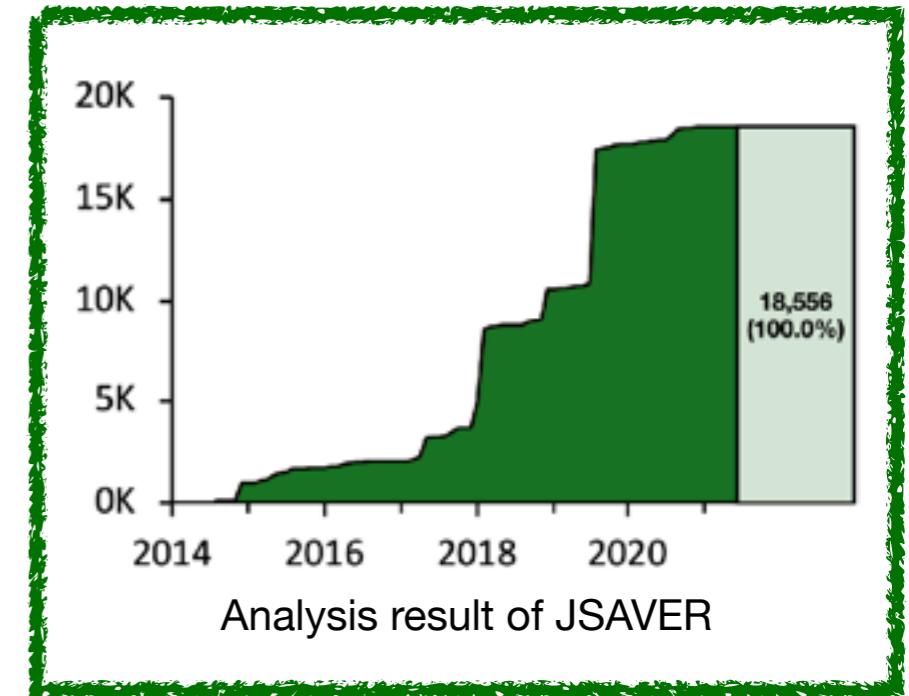


JSAVER - Evaluation

Manual



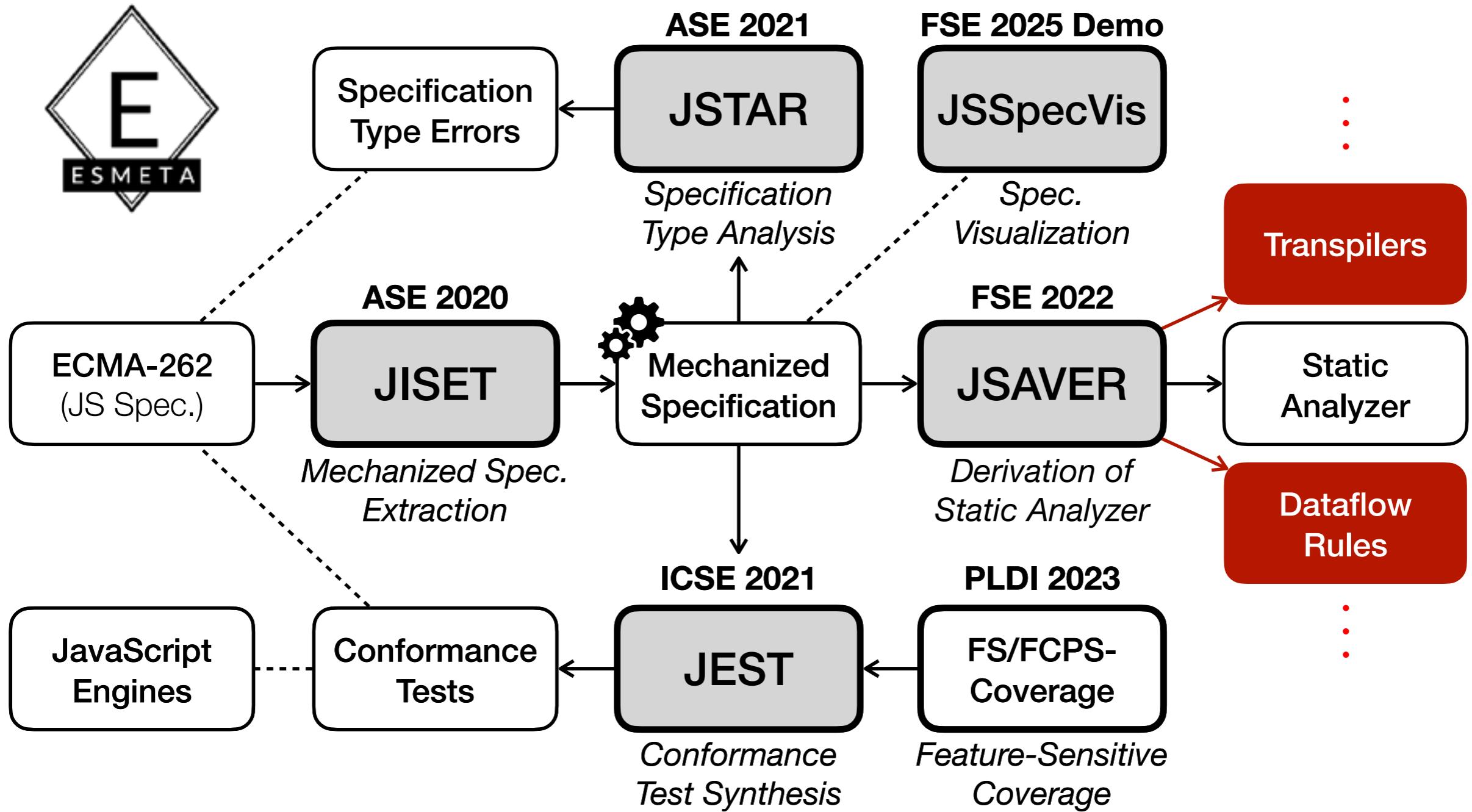
Derived



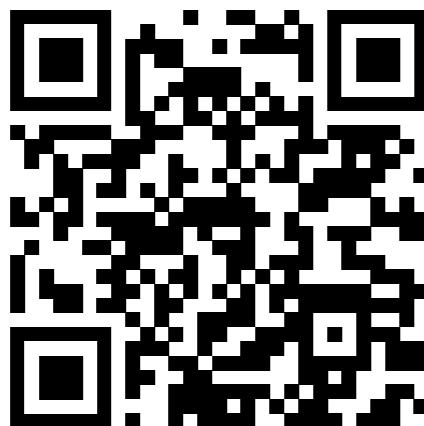
legend :

- error
- unsound
- sound

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



**Official tool used in CI system of
ECMA-262 and Test262**



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

The screenshot shows the GitHub repository page for 'esmeta' (es-meta/esmeta). The repository name is 'esmeta - es-meta/esmeta: ECMAScript Specifi...'. It has 156 stars, 12 forks, 8 watching, 12 branches, 15 tags, and an activity section. The main branch is 'main'. The repository is a public repository. The last commit was by 'jhnaldo' 6 months ago, updating the version. Other commits include adding post-submit test262 tests, updating the client, and removing implicit wrapping/unwrapping. The repository uses GitHub Actions for CI.