International Workshop on Smalltalk Technologies Gdansk, Poland; July 1st to 4th, 2025

# Challenges of Transpiling Smalltalk to JavaScript

Noury Bouraqadi & Dave Mason

IMT Nord Europe
École Mines-Télécom
IMT-Université de Lille

learn, code, make
nootrix

Toronto Metropolitan University

# Why Transpile Smalltalk to JavaScript?

# Why Transpile Smalltalk to JavaScript?

**ST**

Simple & clean semantics

Rich IDE & core libraries

**JS**

Ubiquitous deployment

Performance

Large ecosystem

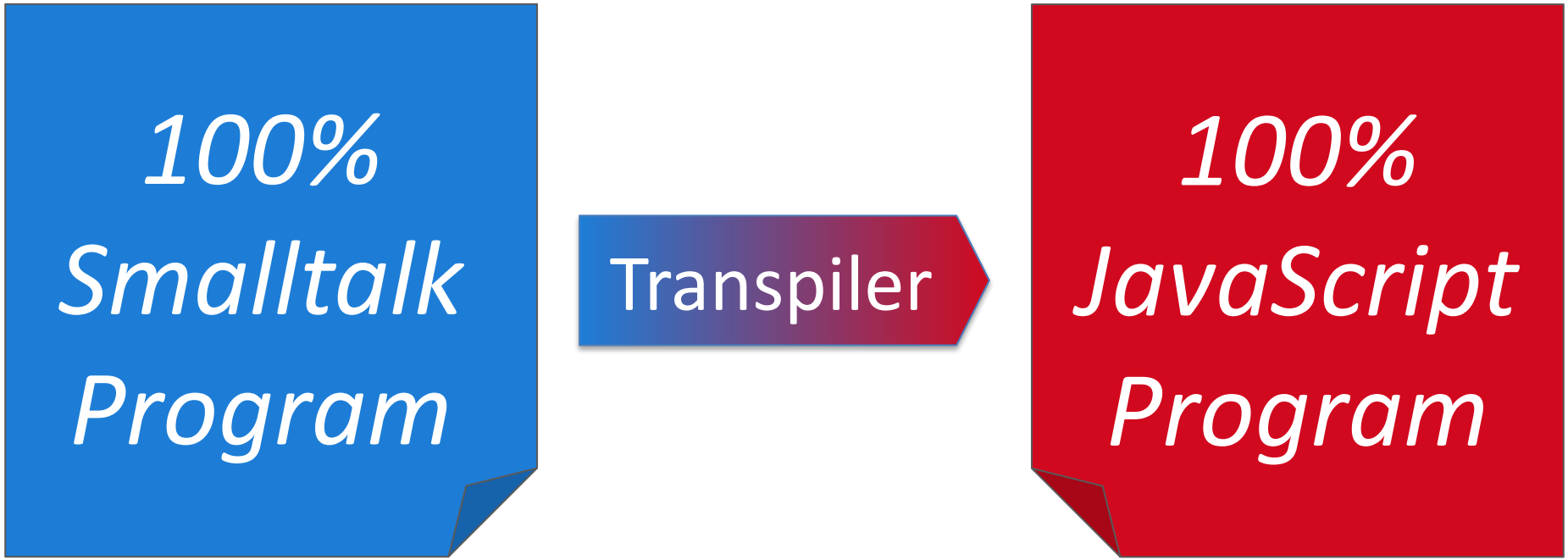
# Why Transpile Smalltalk to JavaScript?

| ST | JS |
| --- | --- |
| Simple & clean semantics | Ubiquitous deployment |
| Rich IDE & core libraries | Performance |
| | Large ecosystem |

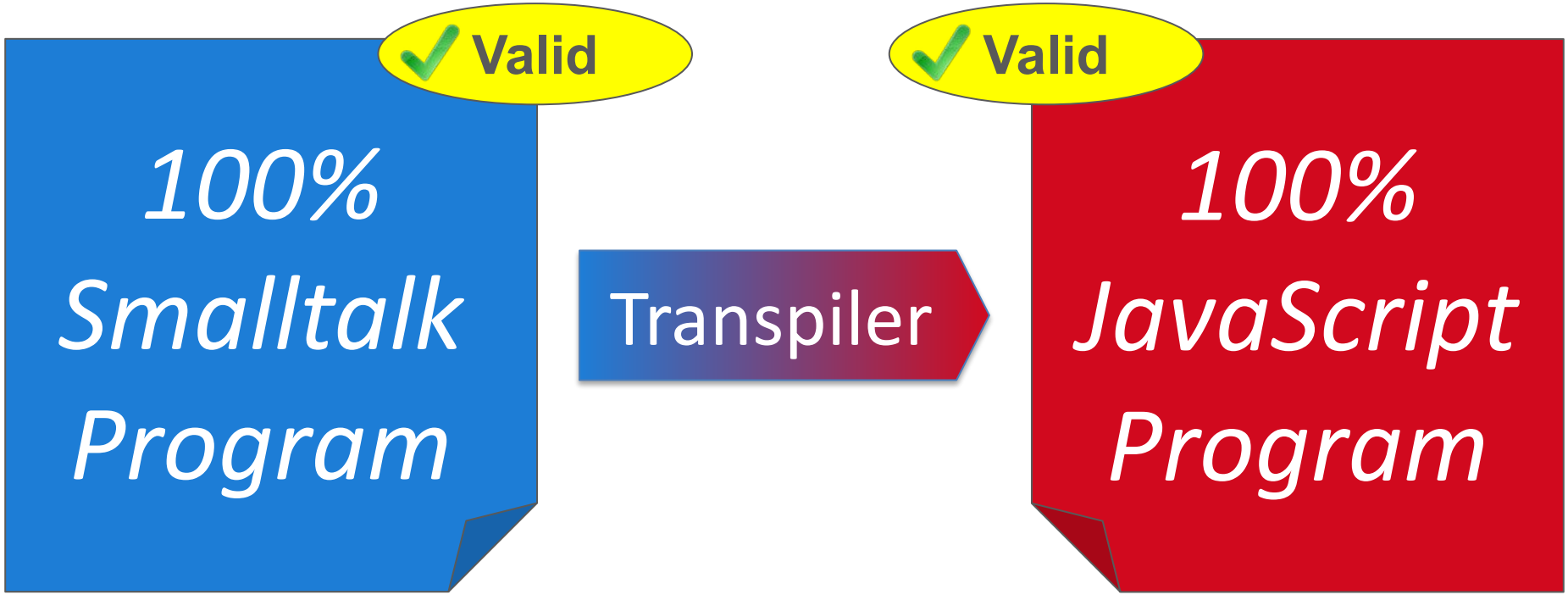
- ST + JS = Best of both worlds
- Smalltalk community interest in the Web/JavaScript
  - ESUG Main Track Talk **Smalltalk for the Web**

Wednesday
July 2nd 2025

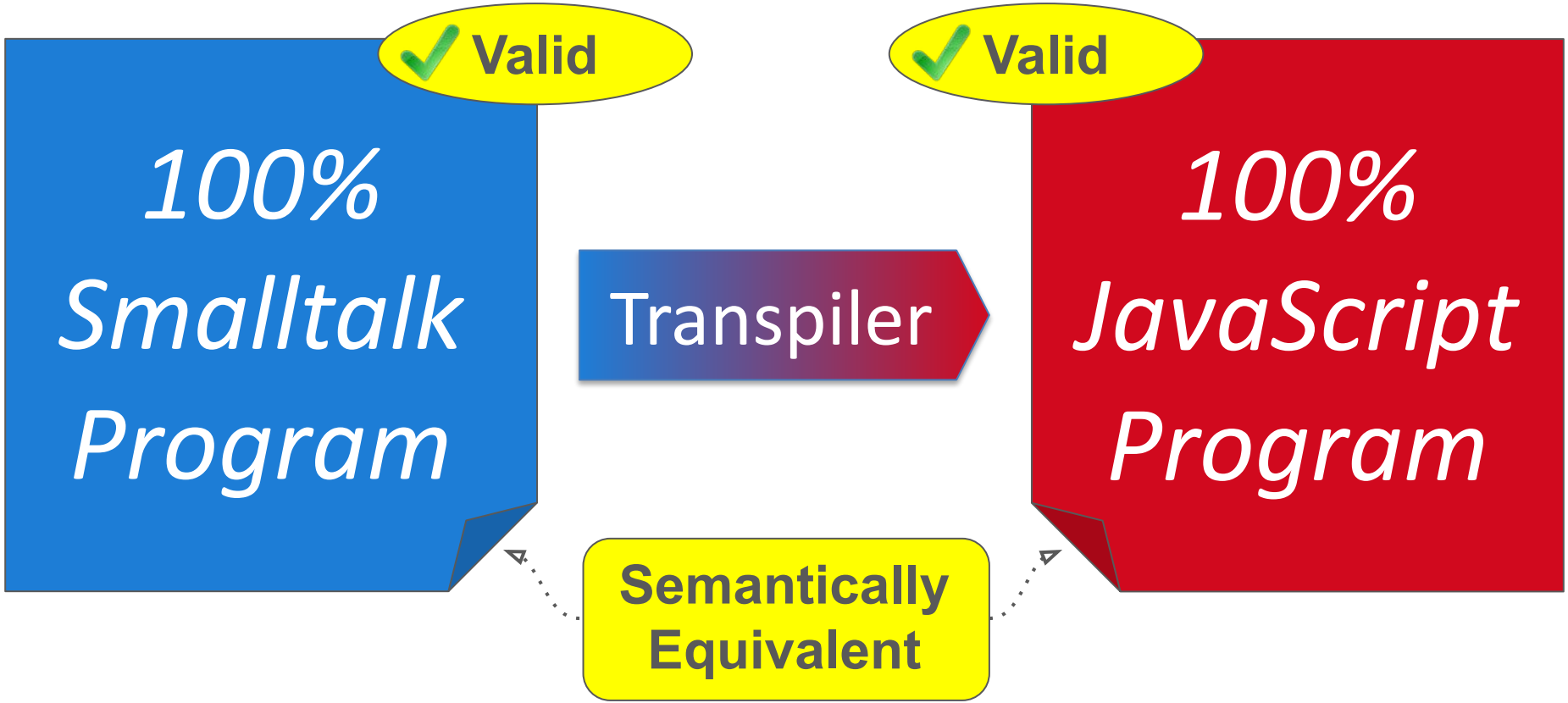
# Transpilation = TRANSlation & comPILATION



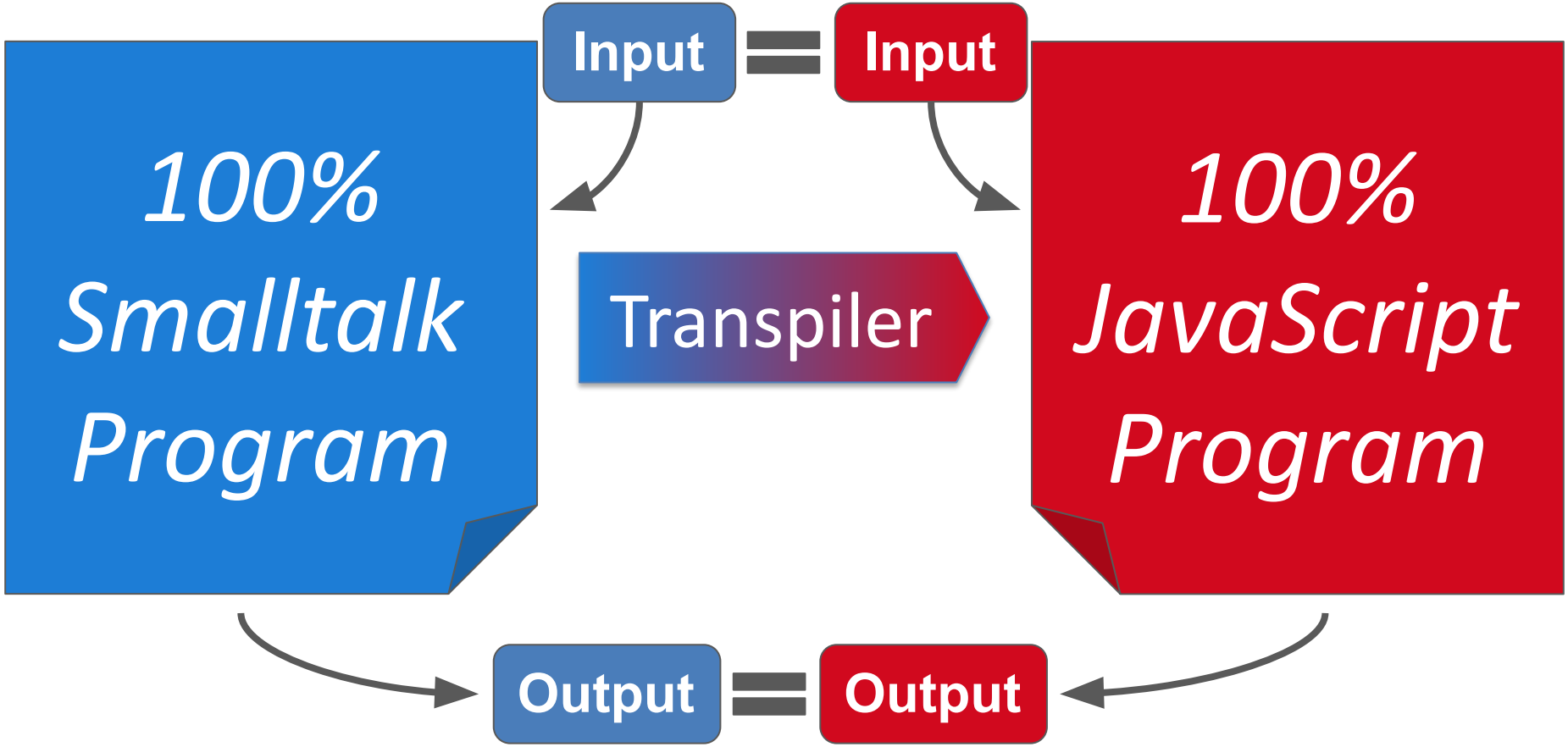
# Transpilation = TRANSlation & comPILATION



# Transpilation = TRANSlation & comPILATION



# Semantic Equivalence



# How to Transpile Smalltalk to JavaScript?

# How to Transpile Smalltalk to JavaScript?

- Map Smalltalk's **syntax** to JavaScript?
- Map Smalltalk's **reflective kernel** to JavaScript?
- Handle Smalltalk **dependencies with the runtime**?

# How to Transpile Smalltalk to JavaScript?

Empirical Study based on 10 Year experience with


**x** to JavaScript?

**tive kernel** to JavaScript?

**ndencies with the runtime**?

# How to Transpile Smalltalk to JavaScript?

Empirical Study based on 10 Year experience with


x to

tive

nd

## Challenges Catalog

1. Primitive Types & Literals
2. Messages
3. Block Closures
4. Classes
5. Reflection

# Challenges Catalog

# Challenges: Primitive Types and Literals

- JS *undefined* and *null* are **Not Objects**.
- JS has an **Impoverished Numeric Stack**.
- ST **Automatically Converts**
  - Between Small Integers & Large Ones.
- ST Supports **Fixed-Point Arithmetic**.
- ST has **Literal Symbols**.

# Example: ST Literal Symbols

- JS has a *Symbol* class, but:
  - No literal symbols
    - Simple to fix
      - Map ST symbols to instances of JS Symbol
  - JS Symbol class is not related to String!
    - Complex to fix
      - Make Symbol subclass of String
        - Side-effects?
      - Override some String methods

# Challenges: Messages

- **Non-Alphanumeric** Characters in ST Message Selectors
- JS Math-Like Message **Priorities**
- ST Message **Cascading**

# Example: Non-Alphanumeric Characters in Selectors

- Simple to fix:
  - Replace non-alphanumeric characters with their Ascii code
  - ST keyword selectors: 1 string concatenating keywords
- Complex to fix: None

# Challenges: Block Closures

Smalltalk Blocks:

- Always **Bind** the Outer Context
- Always **Answer** Some Value
- Support **Non-Local Returns**

# Example: Block Returns

- Simple to fix: ST blocks always answer some value
  - *ST* result := [123]
  - *JS* result = () =>{**return** 123}
- Complex to fix: ST blocks support non-local returns
  - *ST* condition ifTrue: [^123]
  - *JS* rely on exceptions

# Challenges: ST Classes - 1

- Class Variables
- Pool Variables
- Class Extensions
- Stateful Traits

# Challenges: ST Classes - 2

- Class Initialization & Startup/Shutdown Lists
- Methods Always Have a Return Value
- Methods can Have Pragmas
    - Primitive Pragmas Refer to the Virtual Machine

# Examples - 1

- Simple to fix: Class Variables
  - JS Classes are objects with attributes
  - JS encapsulation is optional

```
class A{
    anInstanceMethod(){
        x = A.someClassVariable + 42; }
}
A.someClassVariable = 37;
```

# Examples - 2

- Complex to fix: Pragmas such as #primitive:
  - Implement the primitive behavior

# Challenges: Reflection

- ST **Reifies Messages** Upon Handling Type Errors
- ST Reifies **Execution Contexts**
- ST and JS have different solutions for **Intercepting Method Evaluation**
- Pharo ST **Reifies Slots**
- Pharo ST classes define **Object Format/Layout**

# Examples - 1

- Simple to fix: ST DNU Reifies Messages
  - Extend **JS Object** with
    - doesNotUnderstand() method
    - default methods for every sent message in ST Code

```
Object.prototype.zork = function(arg1, arg2){

        return this.doesNotUnderstand(“zork”, arguments)}
```

# Examples

- Complex to fix: ST Reifies Execution Contexts
  - *ST* thisContext

# Conclusion

Smalltalk & JavaScript

**100% semantic equivalence**

via transpilation is

**difficult if not impossible!**

# Conclusion: Not all challenges are equal

- Some challenges are “easy” to address
  - Primitive Types & Literals,
  - Messages,
  - Block Closures
- Several ST capabilities are complex to implement in JS
  - Primitives,
  - thisContext,
  - Reified Slots…

# Future Work

- Beyond Transpilation
  - ST and JS **Run-time interoperability** (Production)
  - **Live-coding** with JS objects from the ST (Development)
  - **Reuse JS libraries** (code + globals) in code transcribed from ST
  - **Transpile ST to produce JS libraries** for 3rd parties.

- Methodology?
  - Generalizable to other language pairs? Smalltalk & Python?

MIT License

Kindly Supported by

EUROPEAN SMALLTALK USER GROUP
esug
www.esug.org

# PharoJS.org

## Develop in Pharo, Run on JavaScript

GitHub

Thanks to all the contributors!








International Workshop on Smalltalk Technologies Gdansk, Poland; July 1st to 4th, 2025

# Challenges of Transpiling Smalltalk to JavaScript

Noury Bouraqadi & Dave Mason

IMT Nord Europe
École Mines-Télécom
IMT-Université de Lille

learn, code, make
nootrix

Toronto Metropolitan University