The Good, the Bad, and the Ugly: An Empirical Study of Implicit Type Conversions in JavaScript

Michael Pradel[1], Koushik Sen[2]

[1] TU Darmstadt, [2] UC Berkeley

JavaScript: An Unusual Language

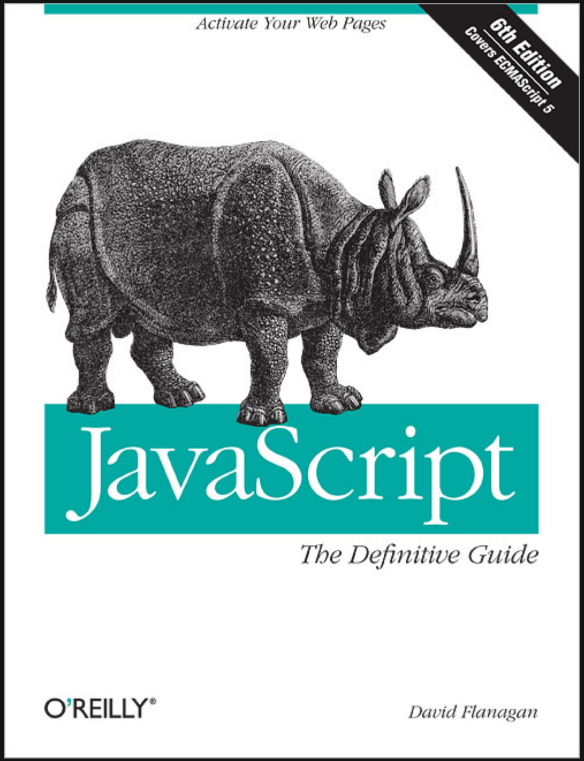
”We need a language for the web.”

JavaScript: An Unusual Language

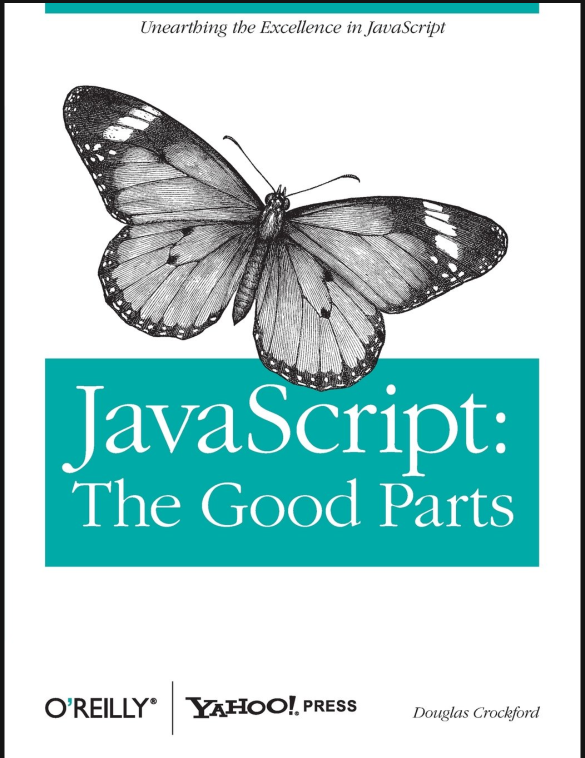
”We need a language for the web.”

”You have 10 days.”

JavaScript: An Unusual Language

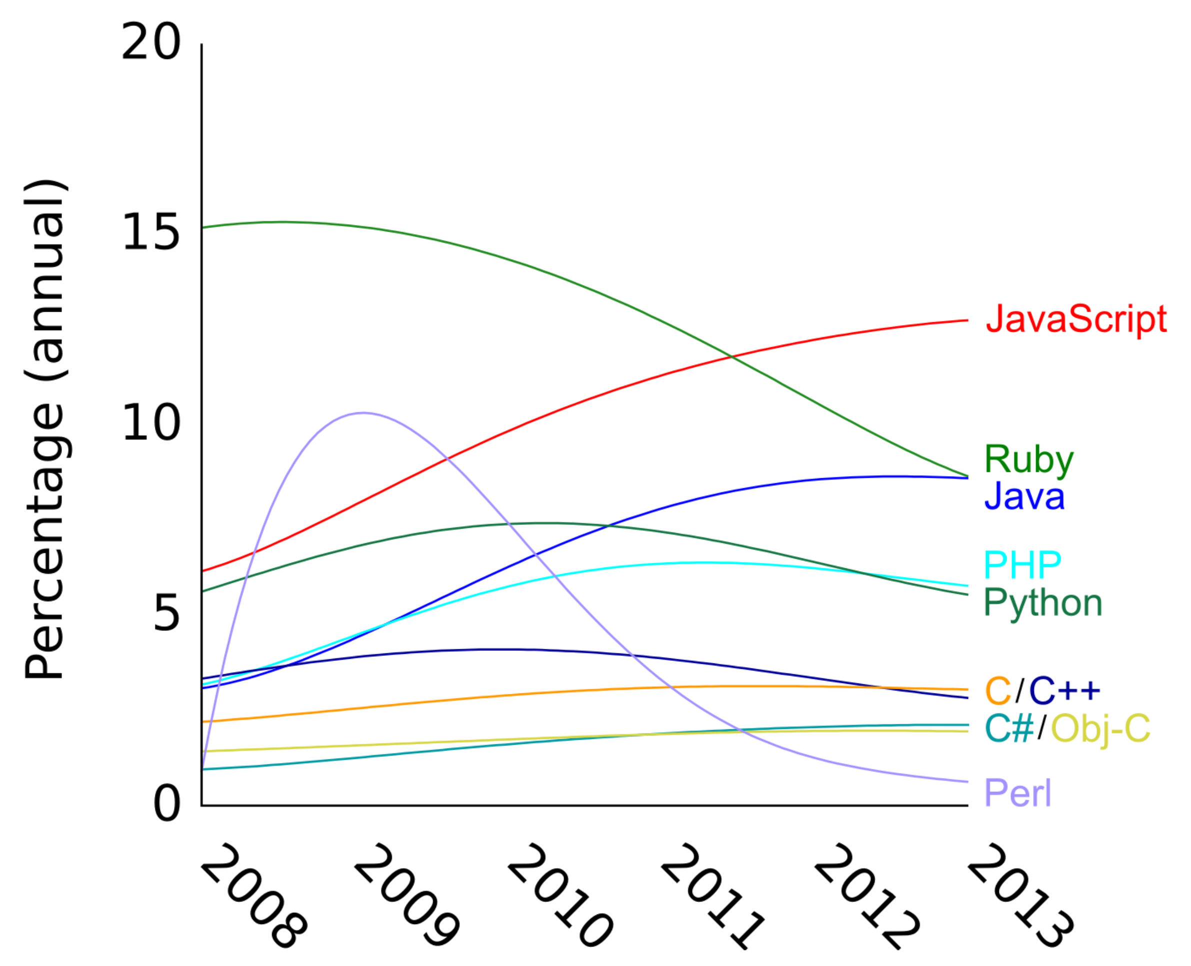


1096 pages



153 pages

JavaScript: An Unusual Language



New projects at Github

(Source: redmonk.com)

Type Coercions

Implicit conversion of a value of one type into a value of another type

Exist in many languages, e.g.

- Java, etc.: Upcasts to supertype
- C, Python, etc.: Integer vs. float

Heavily used in JavaScript

JavaScript Type Coercions

```
"false" == false

"0" == false
```

JavaScript Type Coercions

```
"false" == false   // false

"0" == false       // true
```

When compared to a boolean, strings coerce to numbers

JavaScript Type Coercions

```
new String("a") == "a"


"a" == new String("a")


new String("a") == new String("a")
```

JavaScript Type Coercions

```
new String("a") == "a"               // true

"a" == new String("a")               // true

new String("a") == new String("a")   // false
```

Equality is not transitive

JavaScript Type Coercions

```
[] << "2"

[1] << "2"

[1,2] << "2"
```

JavaScript Type Coercions

```
[] << "2"       // 0

[1] << "2"      // 4

[1,2] << "2"    // 0
```

Should these be valid at all?

Coercions are rarely used

Coercions are error-prone

Coercions make code hard to read

Coercions are rarely used

Really?

Coercions are error-prone

Coercions make code hard to read

This talk:

Empirical Study of JavaScript’s Type Coercions in Practice

Who Needs This Study?

Enables informed decisions

- Program analyses
- Language subsets
- Future languages

Methodology

Subject programs

- Top 100 web sites
- Octane and SunSpider benchmarks

Dynamic analysis

- All operations where coercions may occur
- Based on Jalangi [Sen et al., 2013]

132 programs, 139 million runtime events from 320.000 code locations

How prevalent are coercions?

Photo: A.T.Bueta

Prevalence of Coercions

Function executions with at least one coercion:

- Average over all programs: 80.42%
- Range: 19.95% – 100%

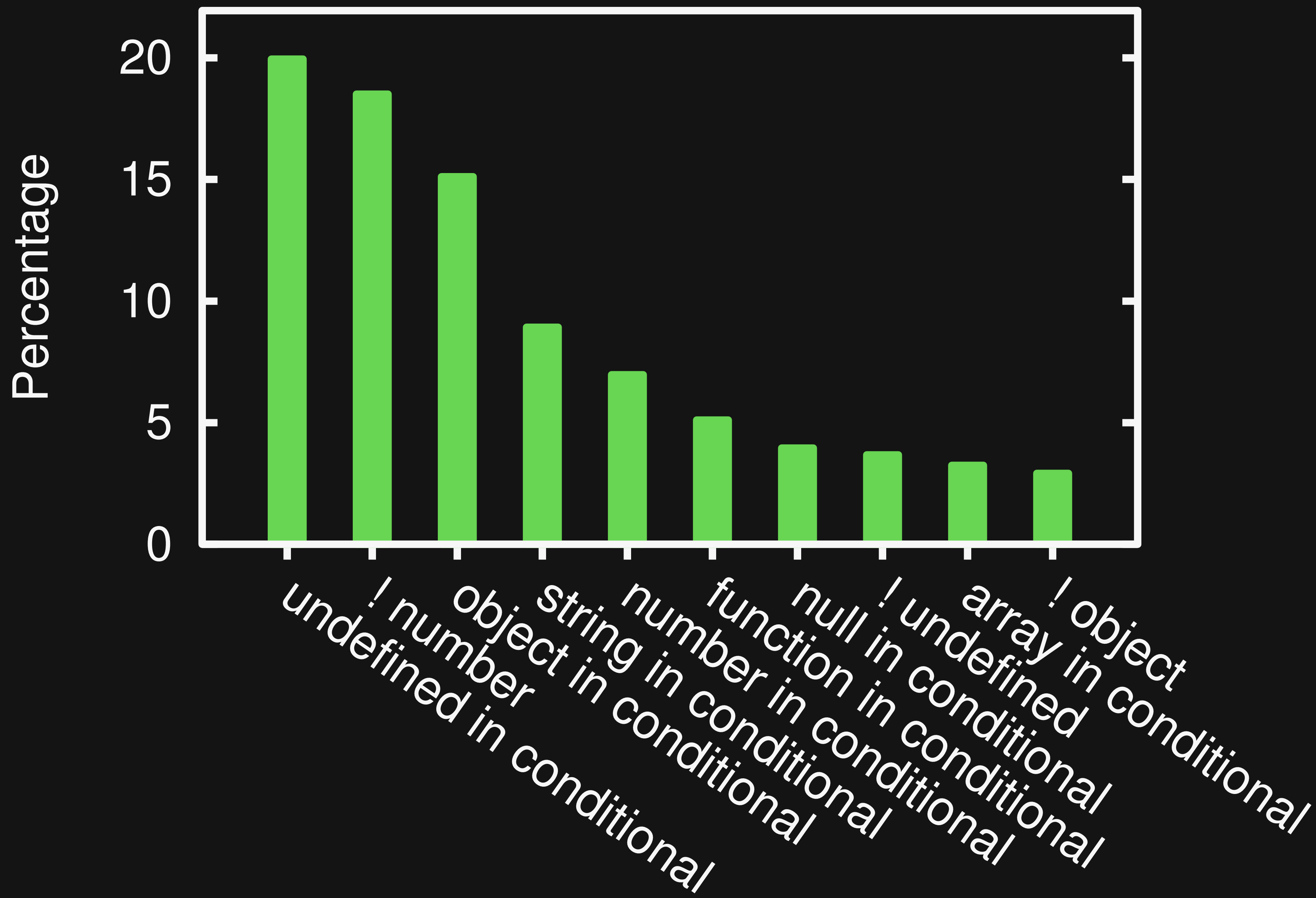
Prevalence of Coercions

Function executions with at least one coercion:

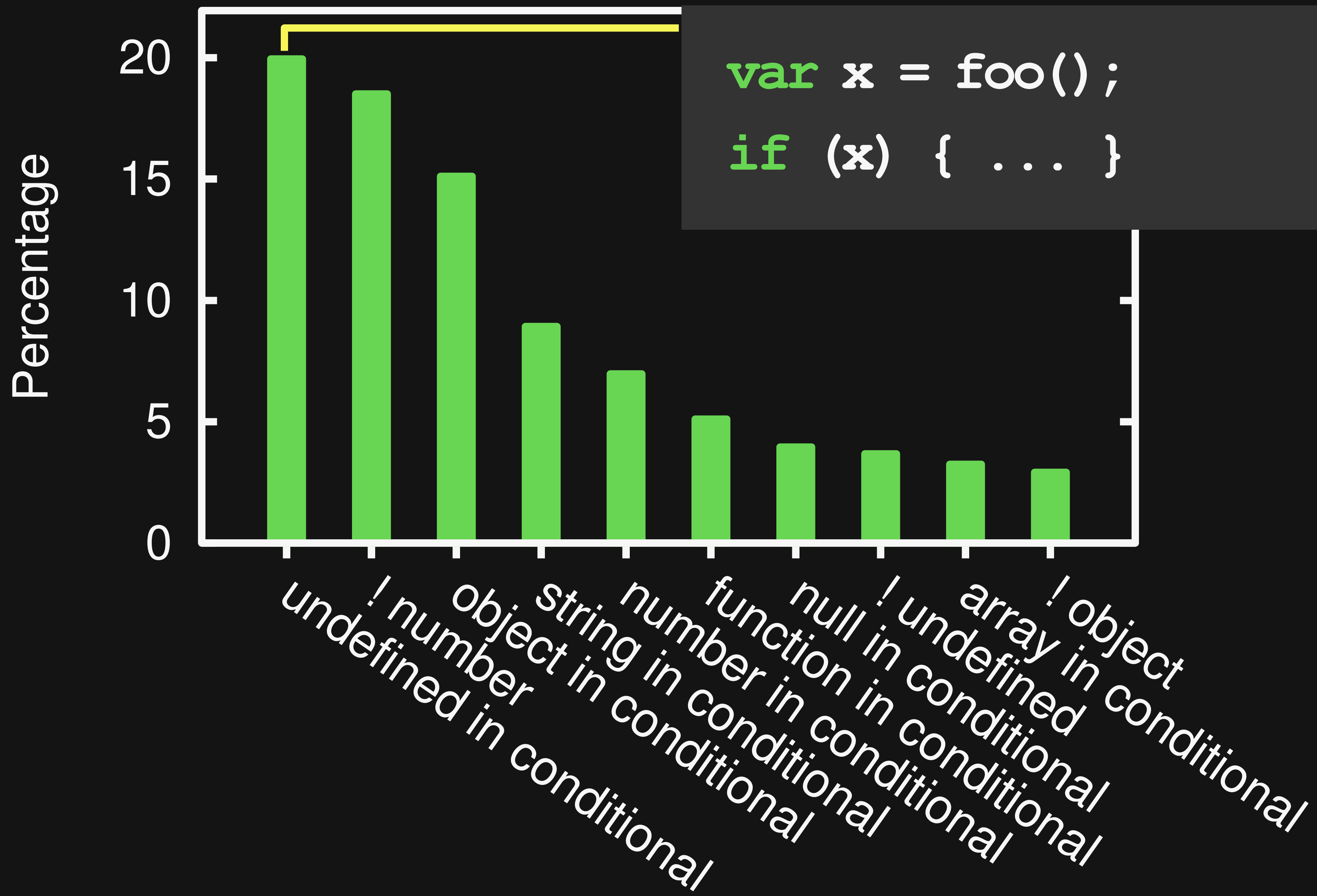
- Average over all programs: 80.42%
- Range: 19.95% – 100%

- Very prevalent
- Certainly non-negligible

What Are Coercions Used For?

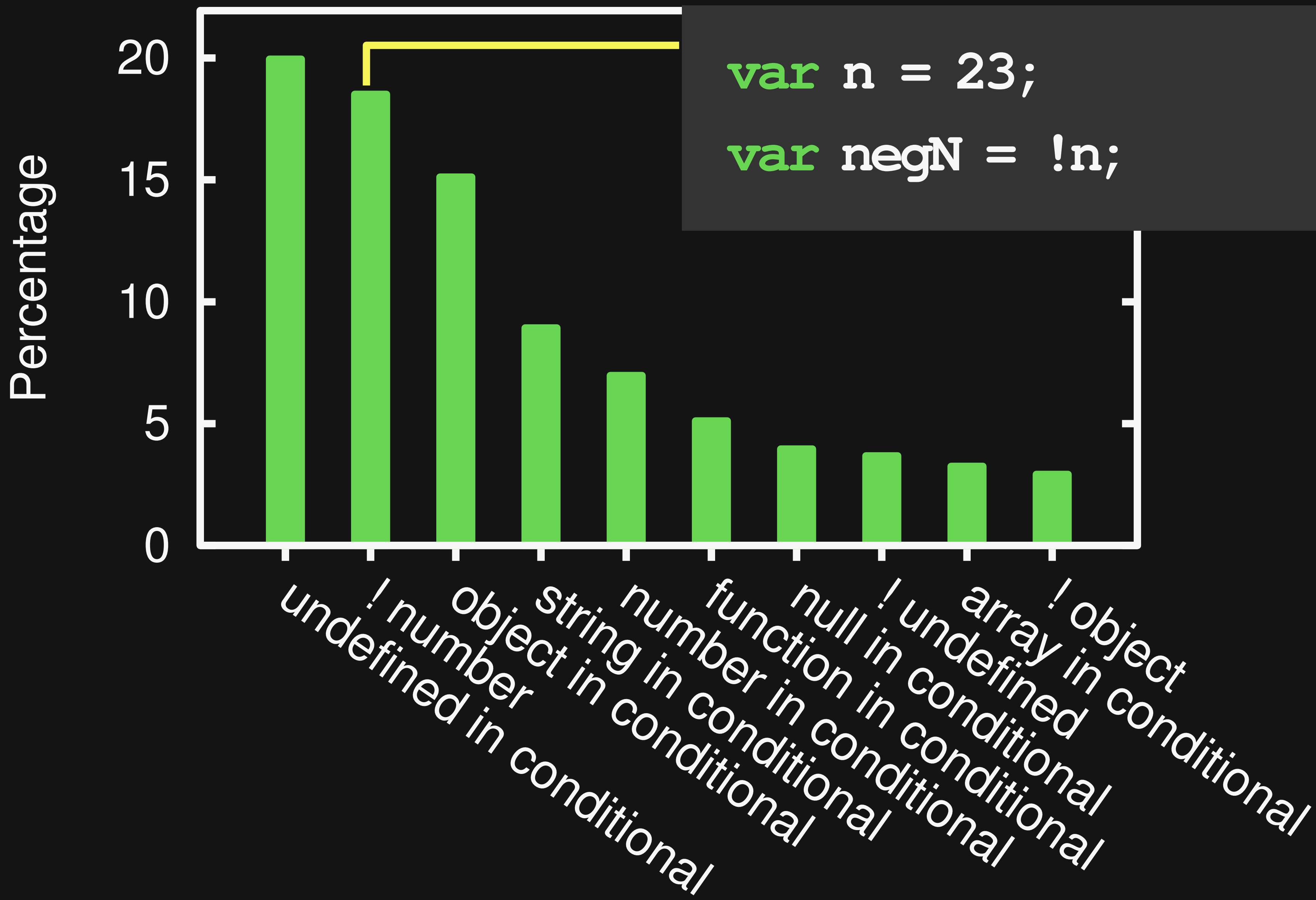


What Are Coercions Used For?



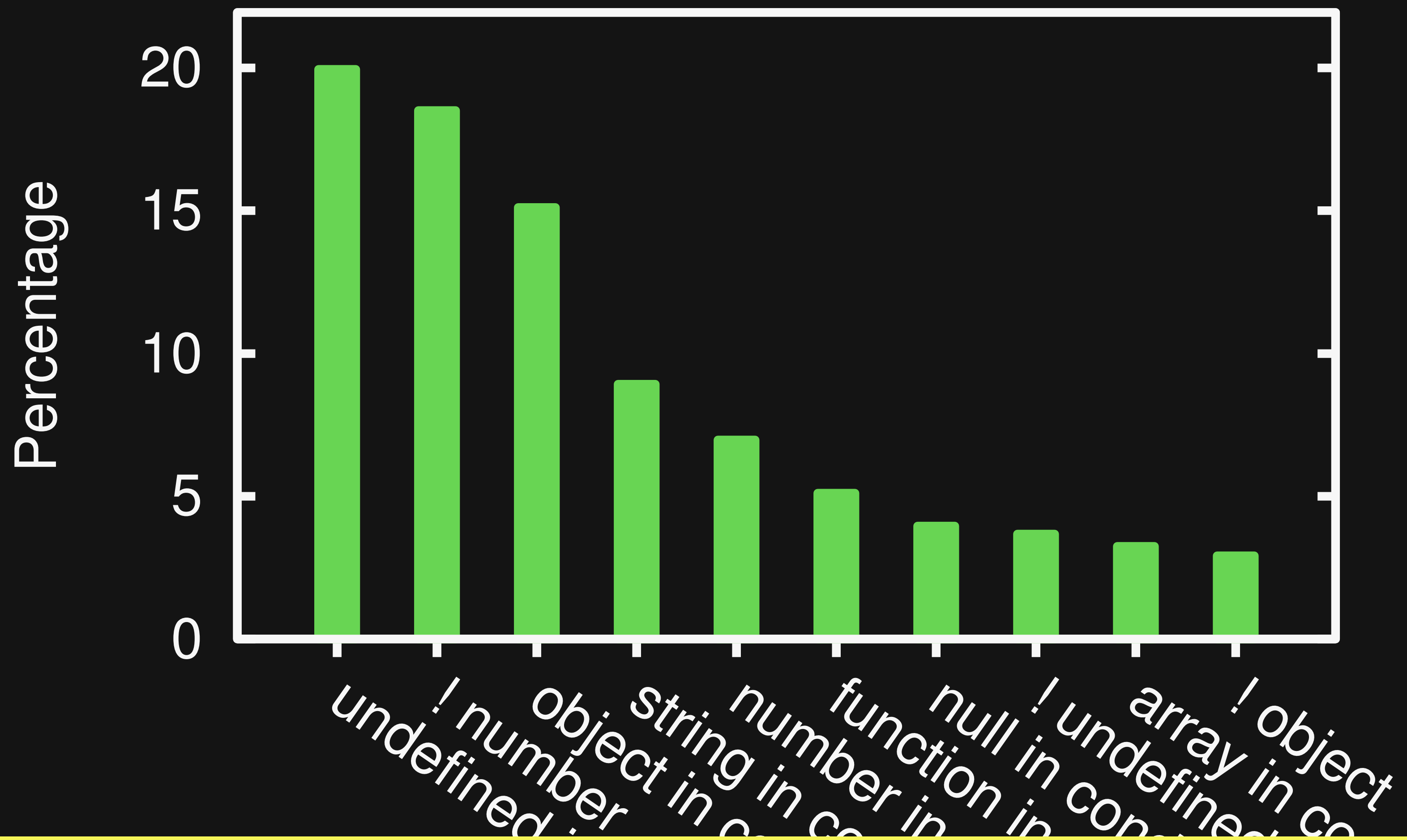
```
var x = foo();
if (x) { ... }
```

What Are Coercions Used For?



```
var n = 23;
var negN = !n;
```

What Are Coercions Used For?



- **Various kinds of coercions**
- **Most prevalent: Conditional-related**

What Are Coercions Used For?

Manual inspection of 30 code locations

- **10 checks if value defined before using it**
- **4 minified booleans: `!0` and `!1`**
- **3 checks if optional argument defined**
- **3 initialization patterns: `x=(x|0)+1`**

Implicit vs. Explicit Conversions

Do developers use explicit conversions?

- E.g., `Boolean(23)`
- 5,497,545 implicit vs. 20,407 explicit

Implicit vs. Explicit Conversions

Do developers use explicit conversions?

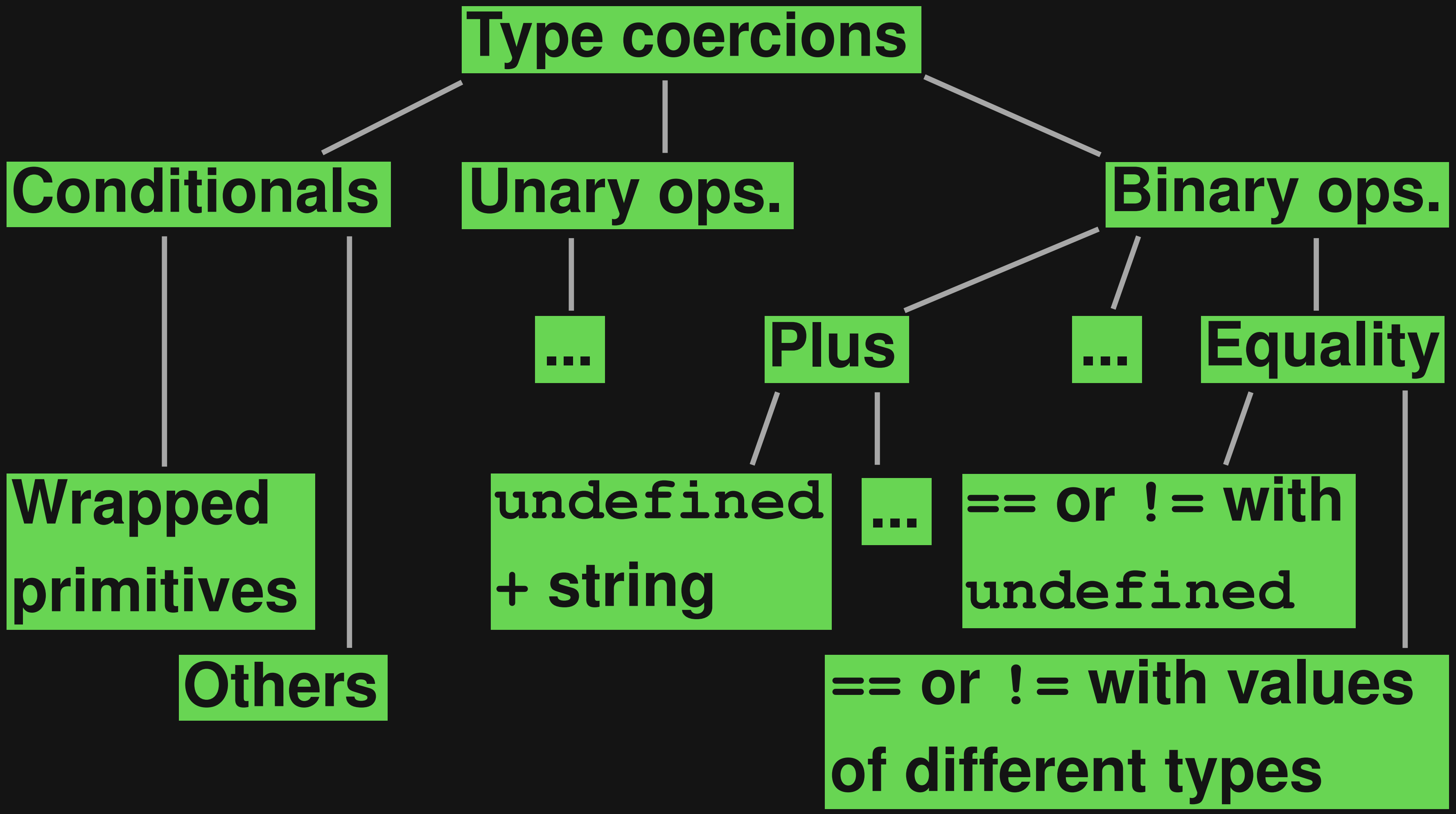
- E.g., `Boolean(23)`
- 5,497,545 implicit vs. 20,407 explicit

Developers prefer implicit conversions

Are coercions error-prone?

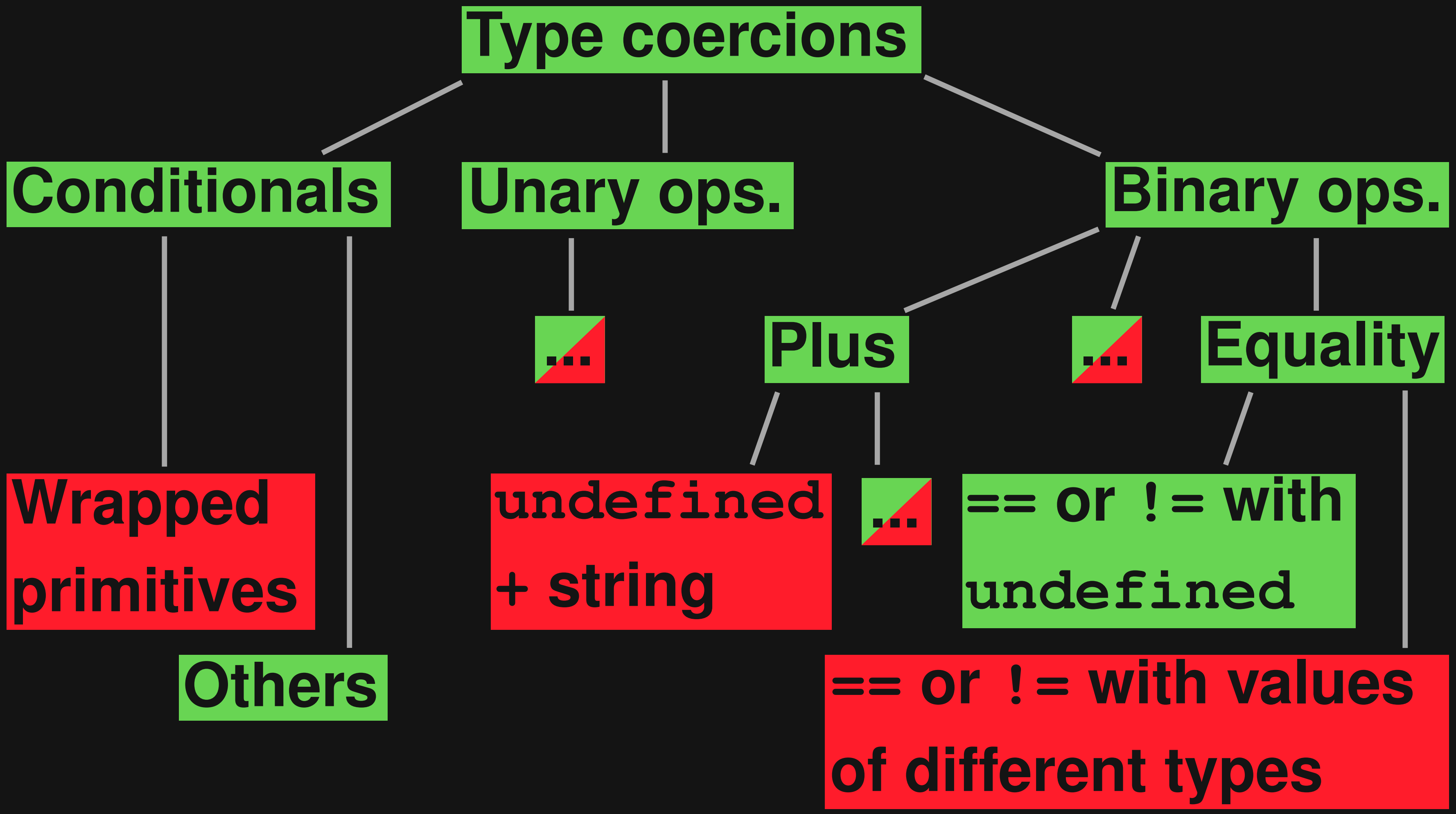
Photo: Raj Alive

Classification of Coercions



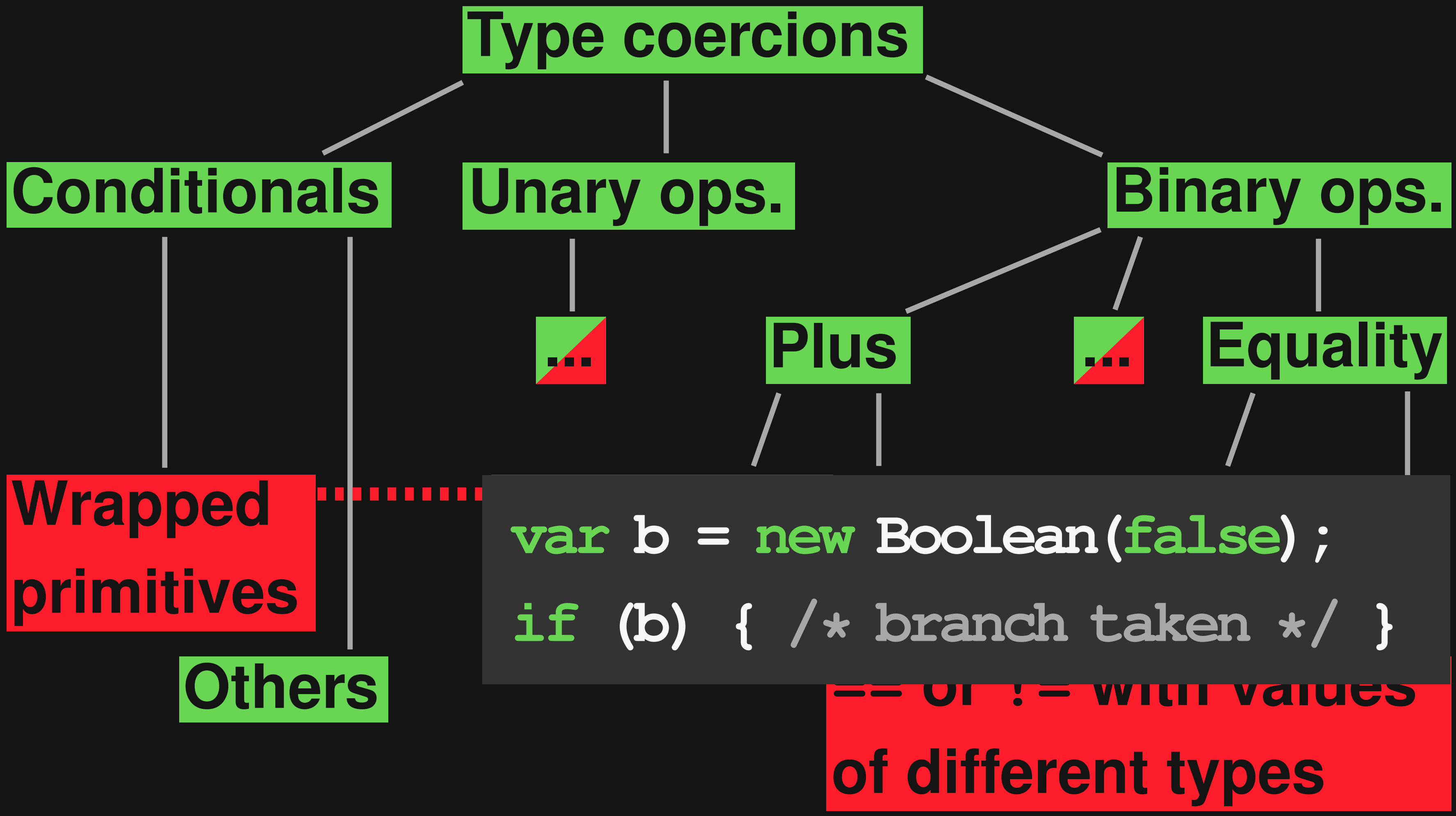
Total: 18 kinds of coercions

Classification of Coercions



Total: 7 harmless, 11 potentially harmful

Classification of Coercions



Type coercions

Conditionals

Unary ops.

Binary ops.

...

Plus

...

Equality

Wrapped primitives

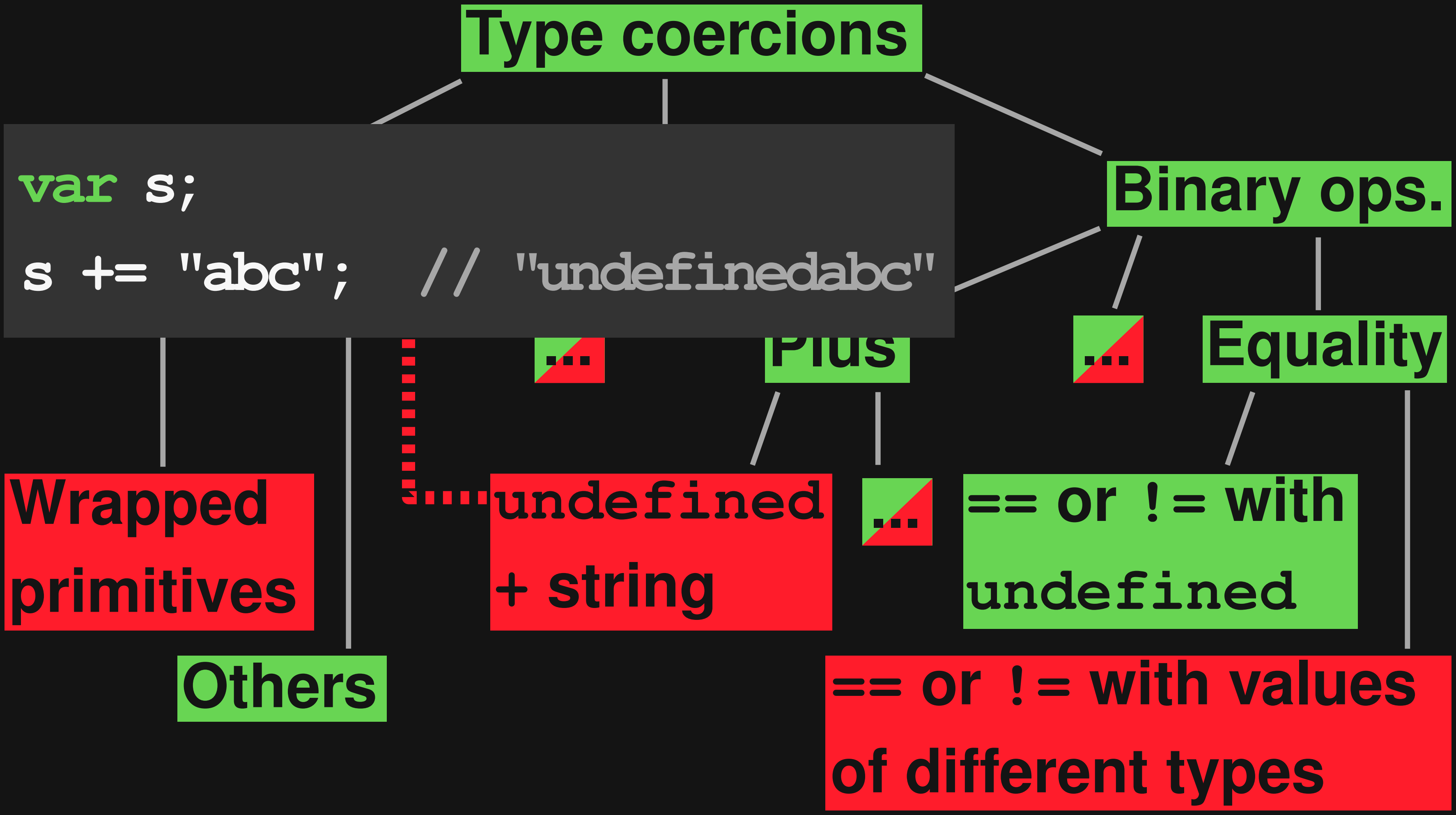
Others

== or != with values of different types

```
var b = new Boolean(false);
if (b) { /* branch taken */ }
```

Total: 7 harmless, 11 potentially harmful

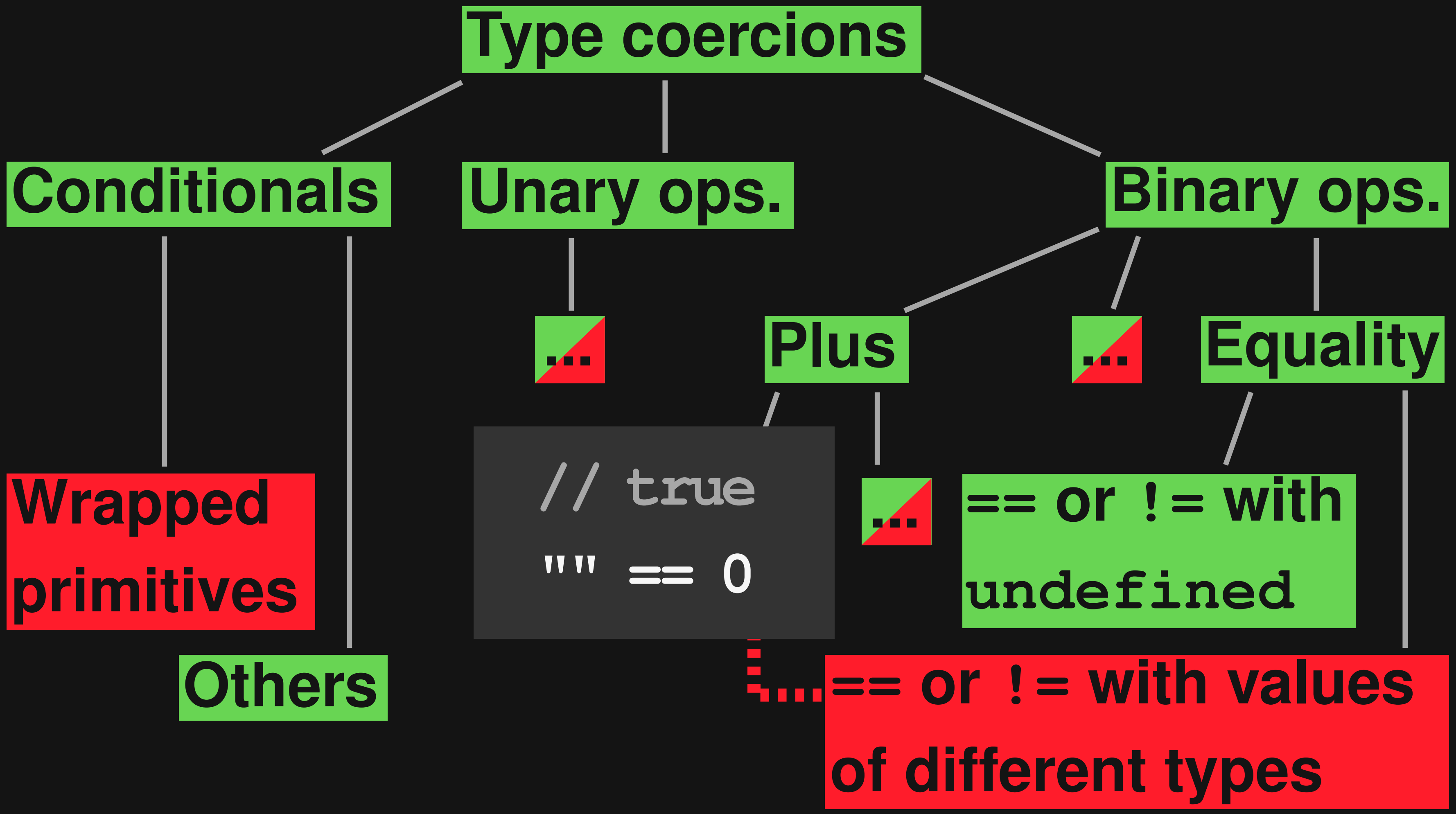
Classification of Coercions



```
var s;
s += "abc";   // "undefinedabc"
```

Total: 7 harmless, 11 potentially harmful

Classification of Coercions



Type coercions

Conditionals

Unary ops.

Binary ops.

...

Plus

...

Equality

Wrapped primitives

Others

```
// true
"" == 0
```

...

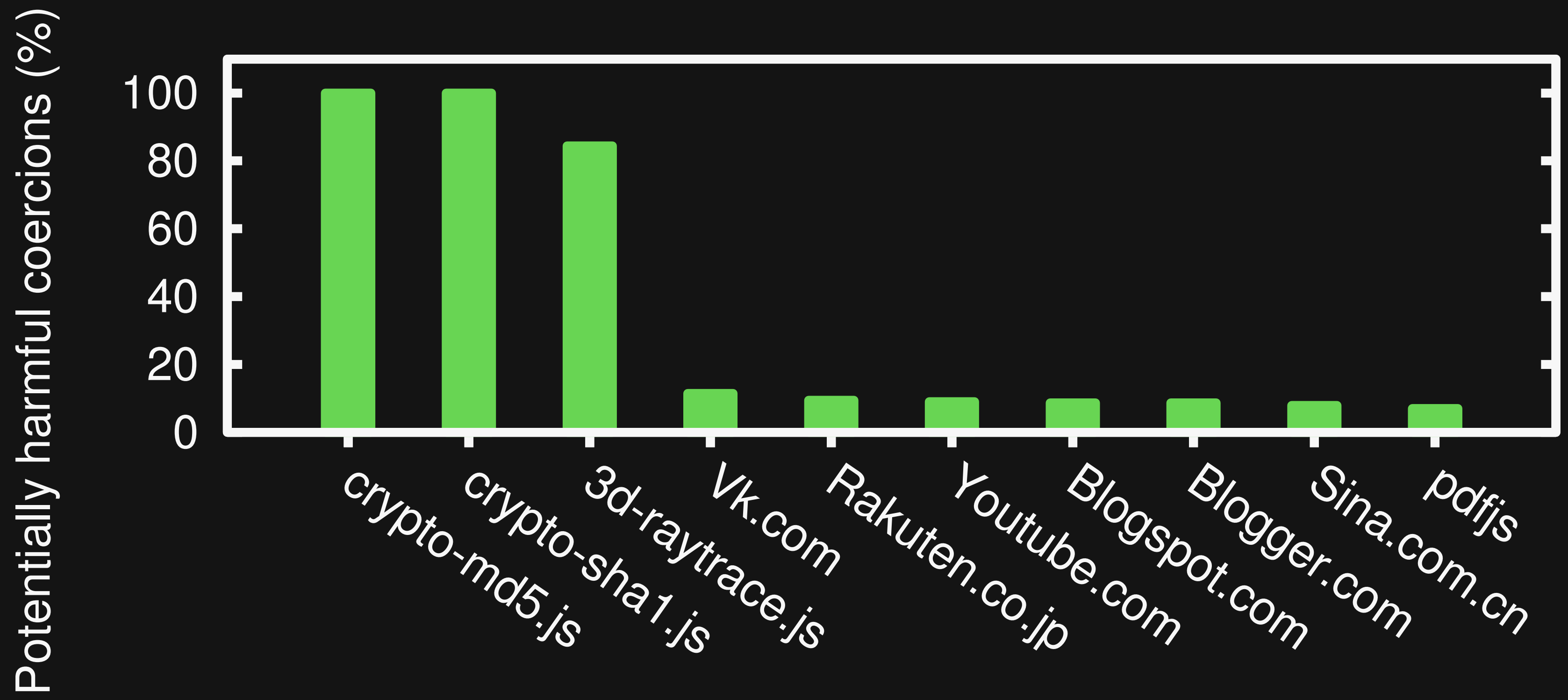
== or != with undefined

== or != with values of different types

Total: 7 harmless, 11 potentially harmful

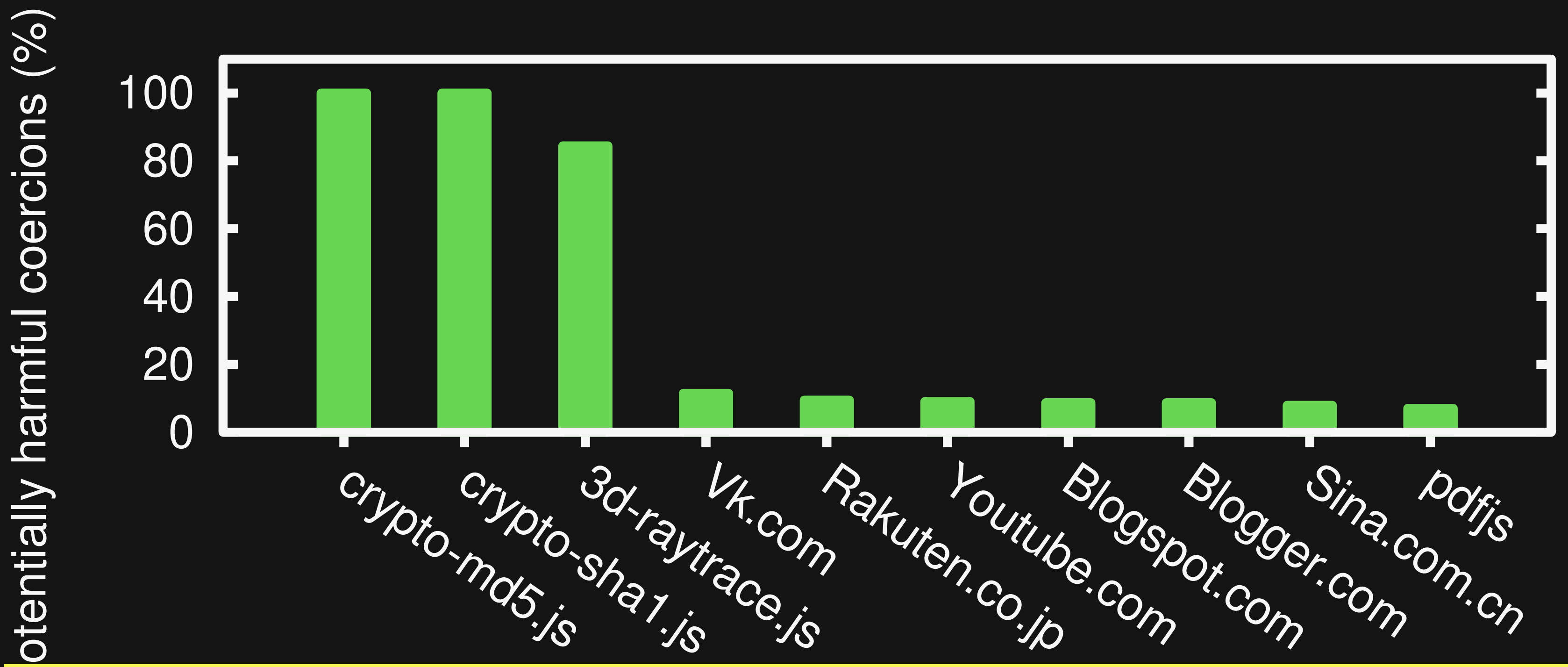
Are Coercions Harmful?

1.15% of all coercions are potentially harmful



Are Coercions Harmful?

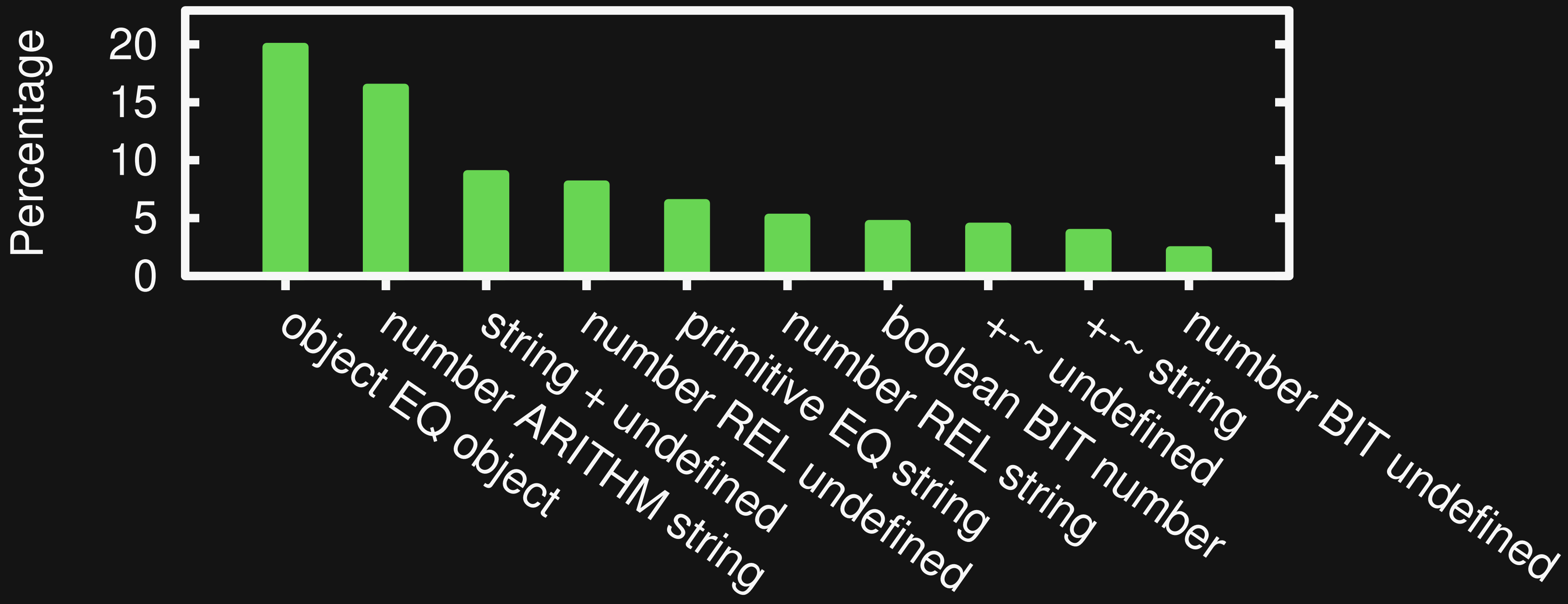
1.15% of all coercions are potentially harmful



Most coercions are harmless

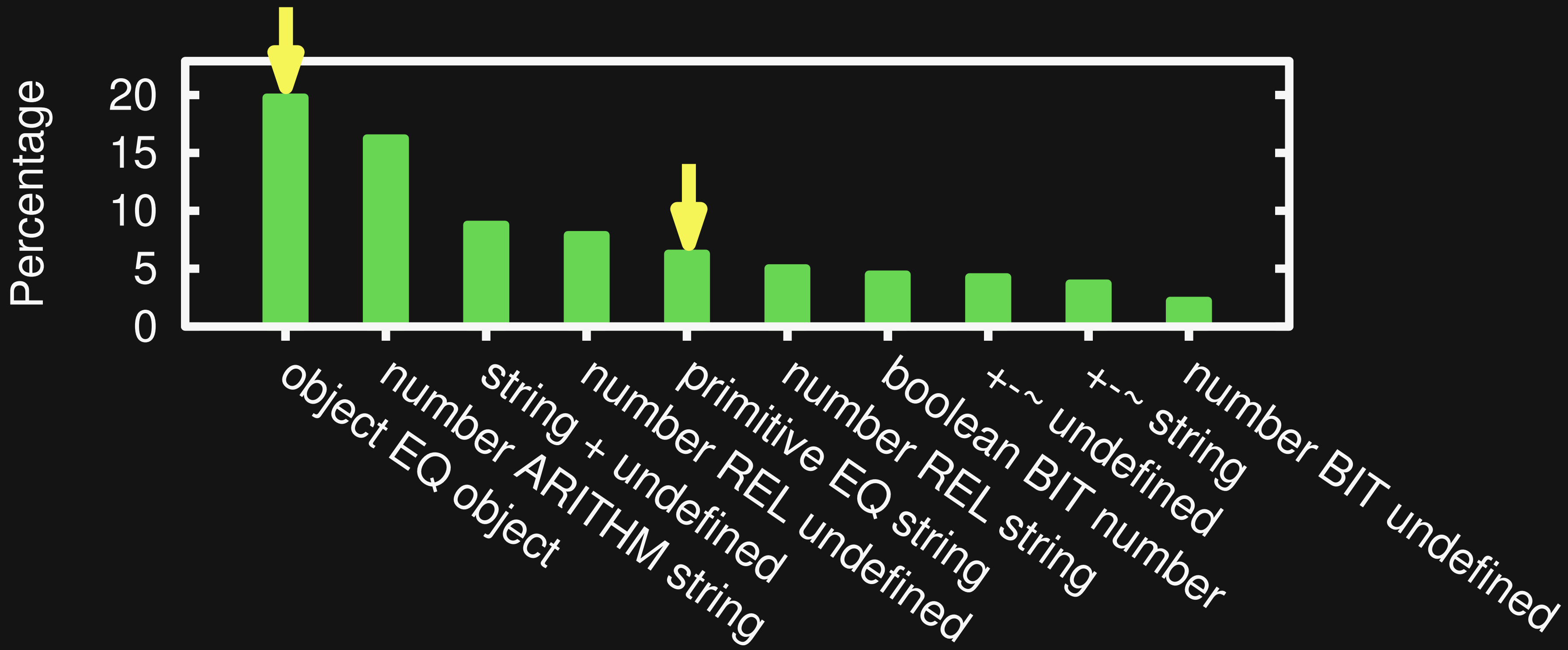
Potentially Harmful Coercions

Which harmful coercions are the most prevalent?



Potentially Harmful Coercions

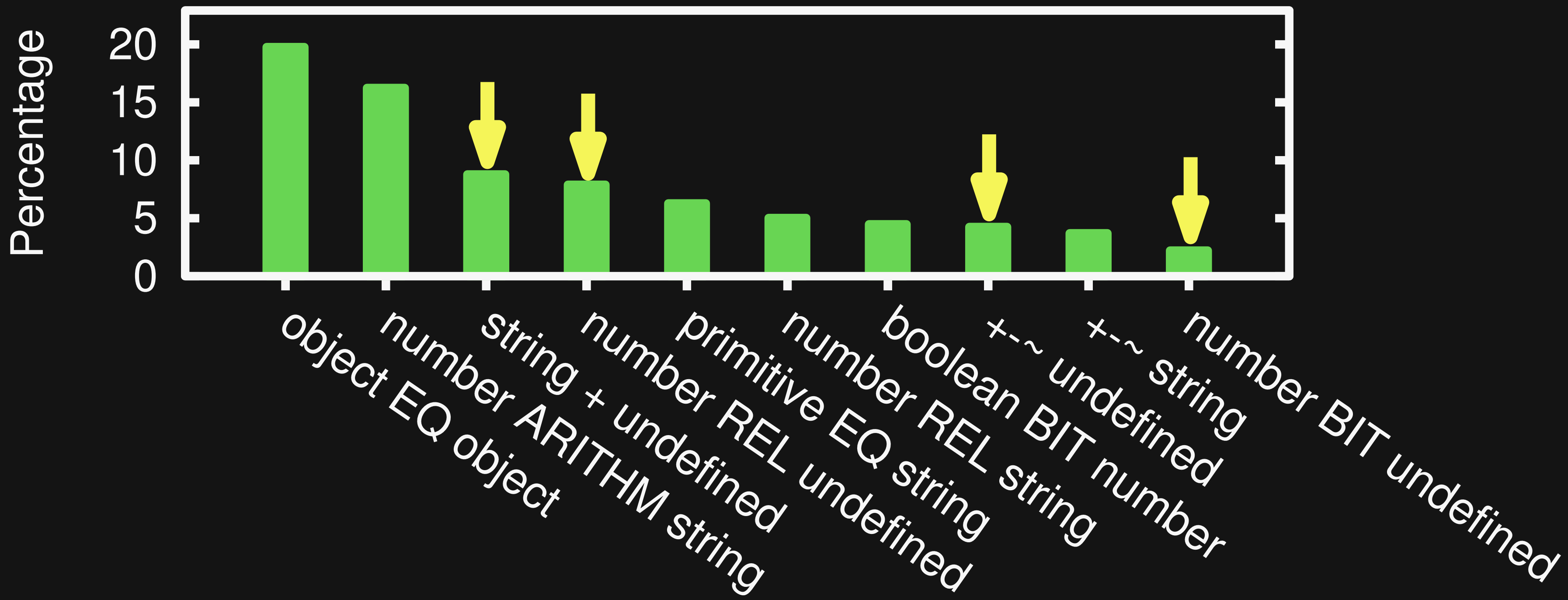
Which harmful coercions are the most prevalent?



Confusing equality semantics

Potentially Harmful Coercions

Which harmful coercions are the most prevalent?



Propagated `undefined` values

Potentially Harmful Coercions (2)

Manual inspection of 30 potentially harmful coercions

- 22 probably correct
- 1 clear bug
- 3 maybe buggy
- 4 unclear

Potentially Harmful Coercions (2)

Manual inspection of 30 potentially harmful coercions

- 22 probably correct
- 1 clear bug
- 3 maybe bu
- 4 unclear

- Represent number as string (10x)
- string + `undefined` (3x)
- typeA == typeB (2x)

Potentially Harmful Coercions (2)

Buggy coercion on *www.sina.com.cn*

```
flashVer: function() {
  if (m & 8192 != 8192) {
    return ""
  }
  ..
}
```

Potentially Harmful Coercions (2)

Buggy coercion on *www.sina.com.cn*

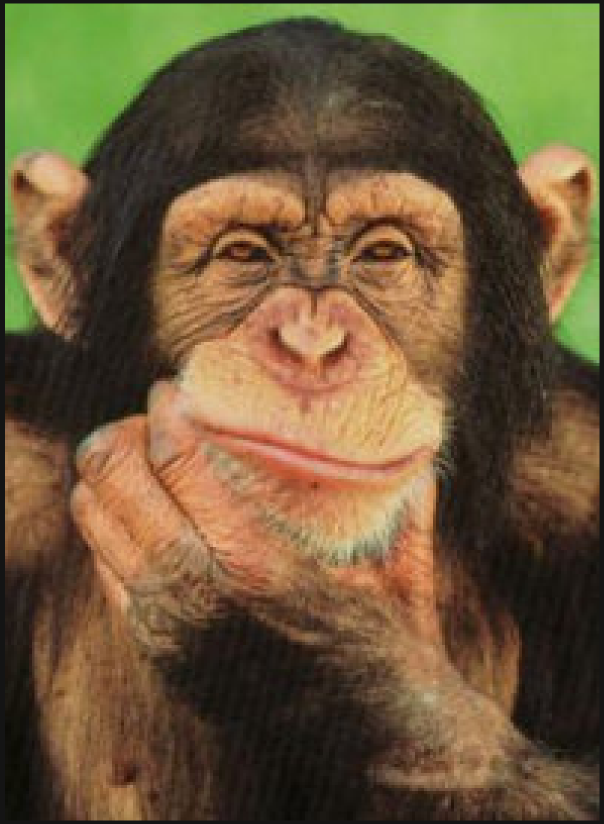
```
flashVer: function() {
  if (m & 8192 != 8192) {
    return ""
  }
  ..
}
```

$\leadsto$ `m & false`

$\leadsto$ `0`

$\leadsto$ `false`

Do coercions harm code understandability?

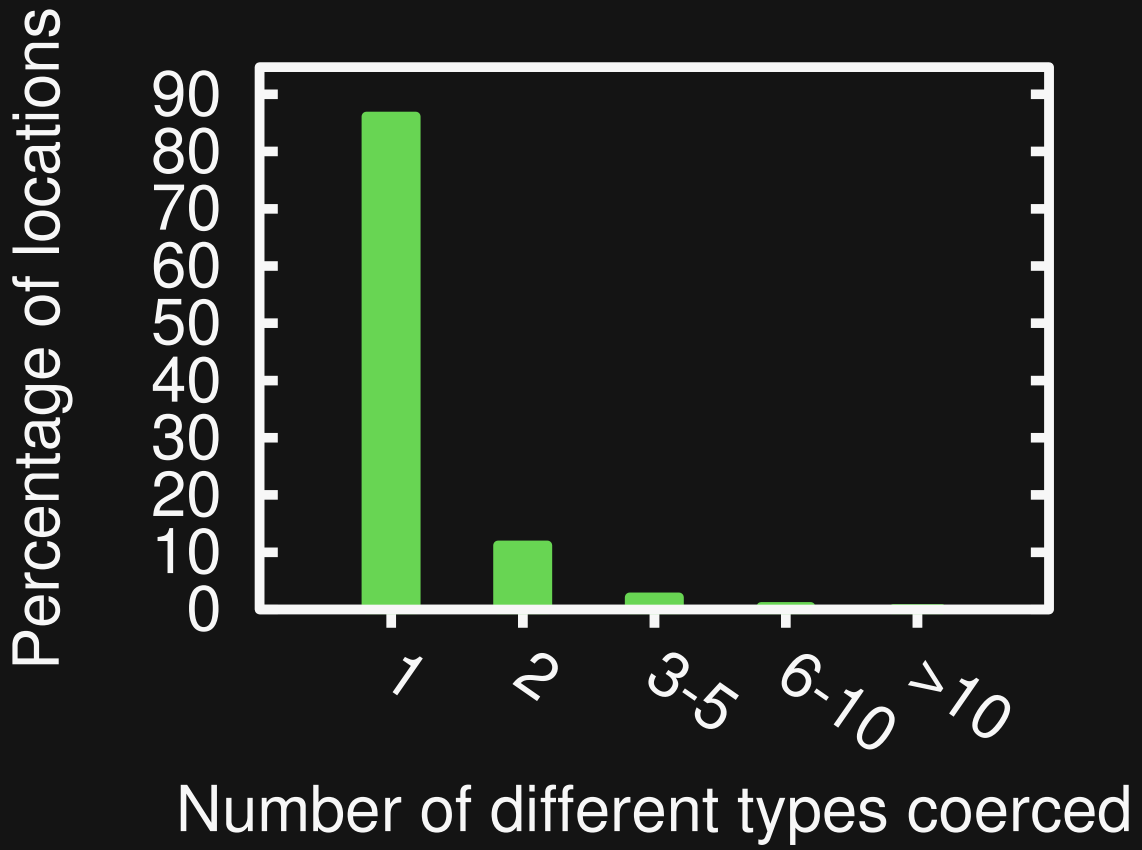


Coercions vs. Understandability

Polymorphic code locations

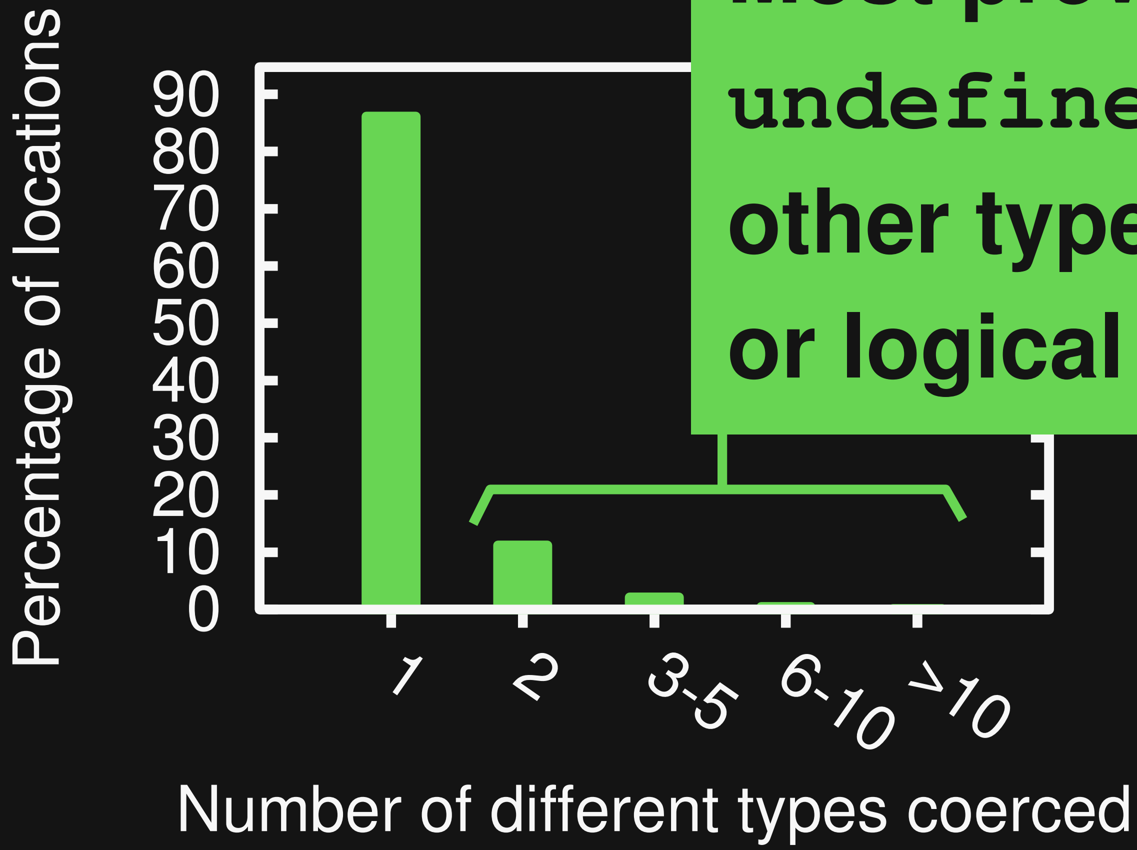
Coercions vs. Understandability

Polymorphic code locations



Coercions vs. Understandability

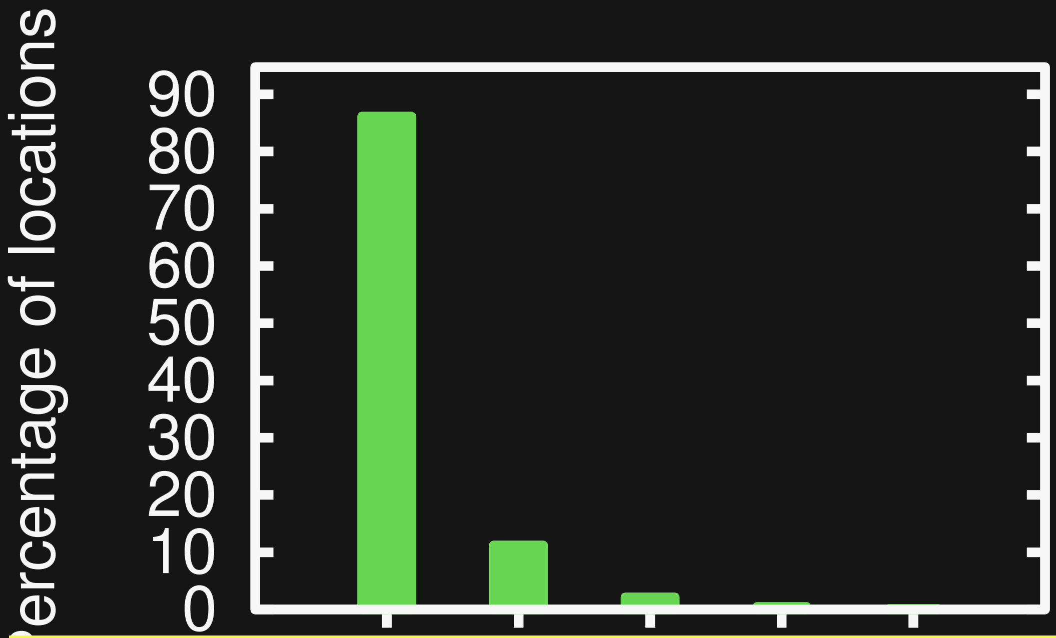
Polymorphic code locations



Most prevalent: `undefined` vs. some other type in conditional or logical negation

Coercions vs. Understandability

Polymorphic code locations



- Most locations are monomorphic
- Polymorphism: Mostly expected

Coercions vs. Understandability (2)

Strict vs. non-strict equality

- `===` and `!==`
- Equal only if same type

- `==` and `!=`
- Considers coercions

Common advice:
Avoid non-strict checks

Coercions vs. Understandability (2)

Strict vs. non-strict equality

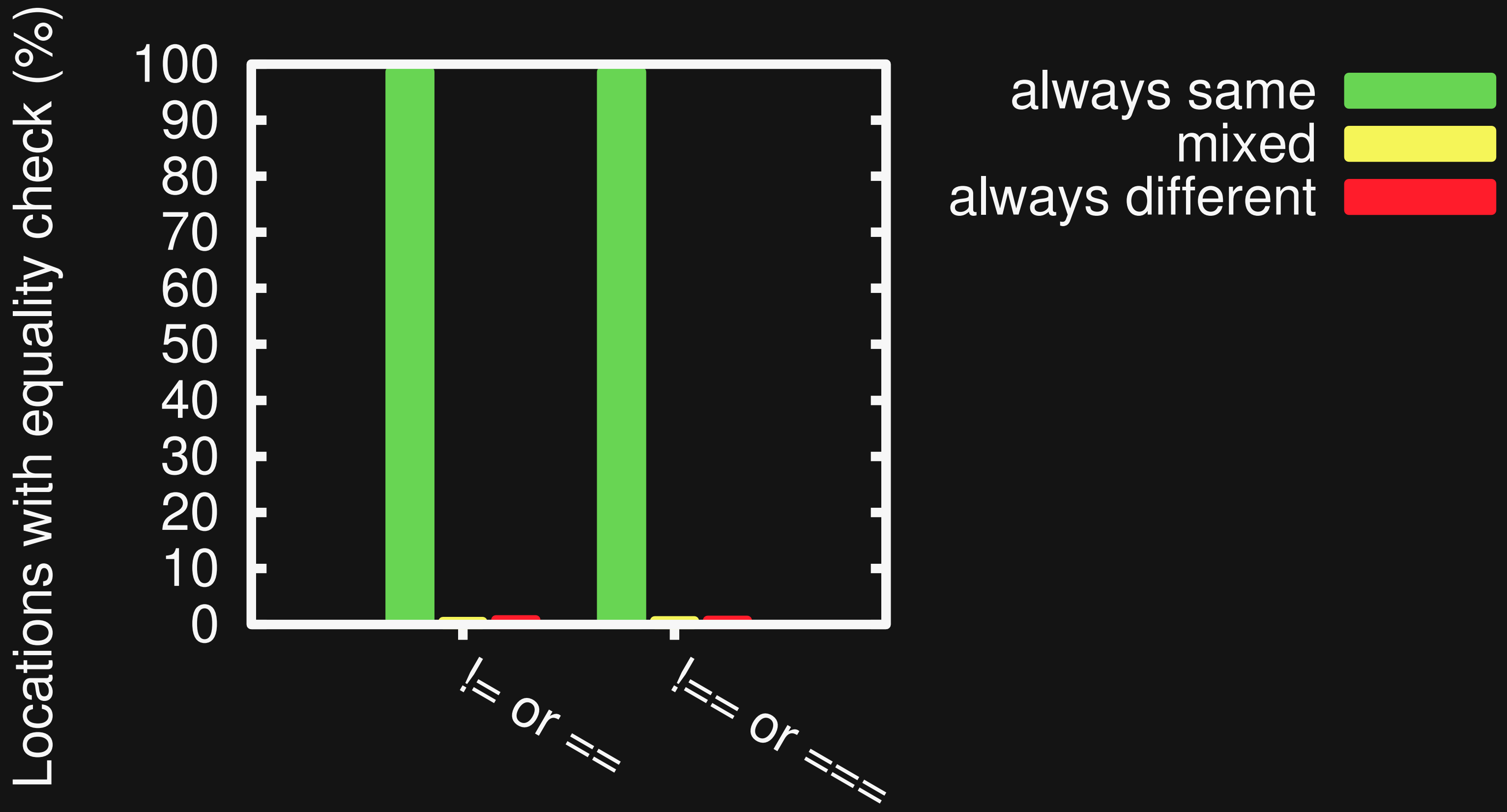
2,026,782 occurrences

3,143,592 occurrences

Coercions vs. Understandability (2)

Strict vs. non-strict equality

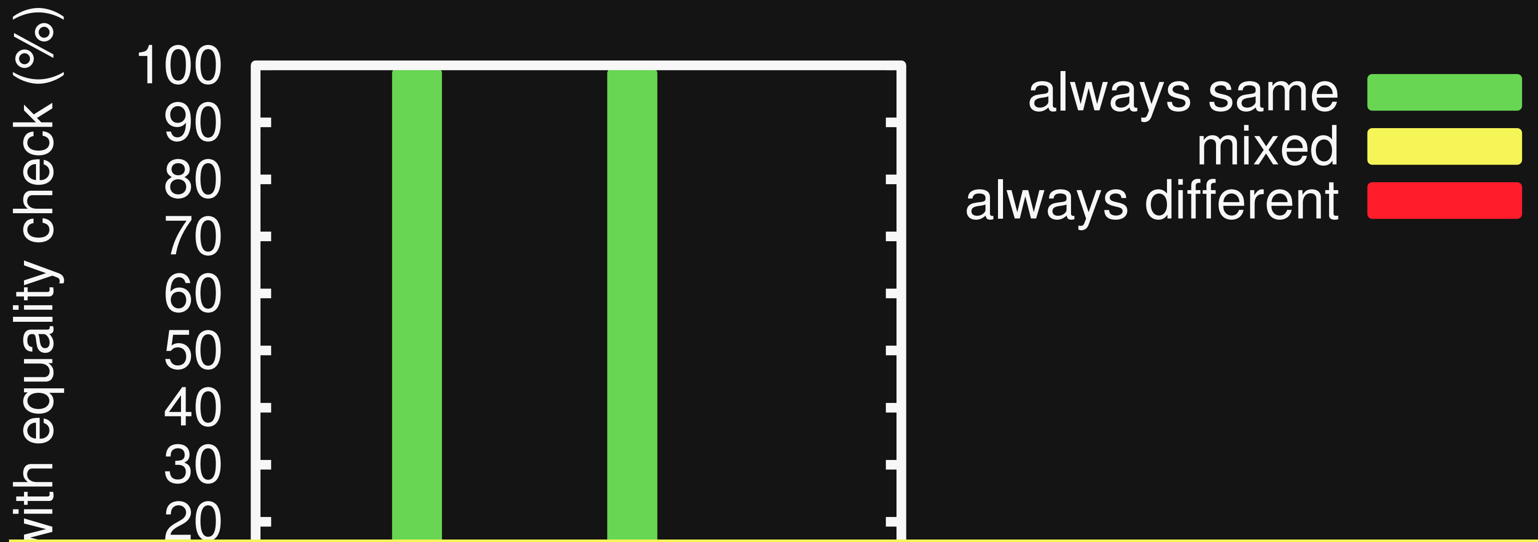
Do developers distinguish between them?



Coercions vs. Understandability (2)

Strict vs. non-strict equality

Do developers distinguish between them?



- **Confusing semantics**
- **May refactor into strict checks**

Threats to Validity

- Dynamic analysis: Underestimations
- Harmless vs. harmful: Subjective
- Representativeness of programs
- JavaScript only

Related Work

Studies on language usage

- Dynamic analysis
  Knuth1971, Richards2010/11
- Static analysis
  Tempero2008, Muschevici2008, Malayeri2009
- Humans
  Hanenberg2010

Analyze and restrict usage of types

- Type inference and checking
  Thiemann2005, Jensen2009, Chugh2012
- Language subsets
  strict mode, restrict mode

Conclusions

In-depth study of type coercions

- Coercions are widely used
- Most coercions are harmless
- Equality checks difficult to understand

Implications for future research

- Static analyses must consider coercions
- Languages: Disallow some coercions
- Refactoring of equality checks

http://mp.binaervarianz.de/ecoop2015

Conclusions

In-depth study of type coercions

- Coercions are widely used
- Most coercions are harmless
- Equality checks difficult to understand

Implications for future research

- Static analyses must consider coercions
- Languages: Disallow some coercions
- Refactoring of equality checks

Thanks!

http://mp.binaervarianz.de/ecoop2015