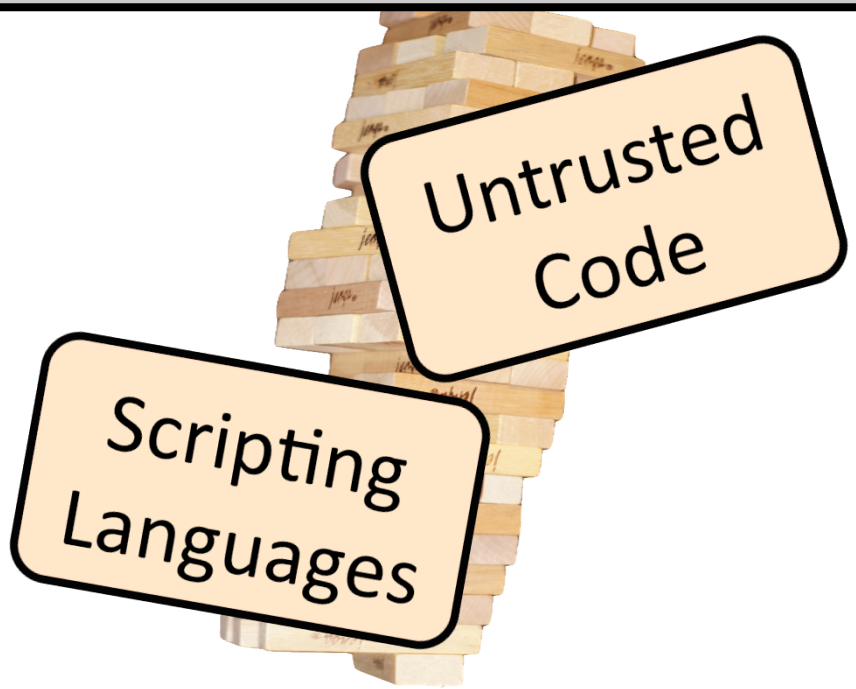
Modern Type Systems for Dynamic Languages

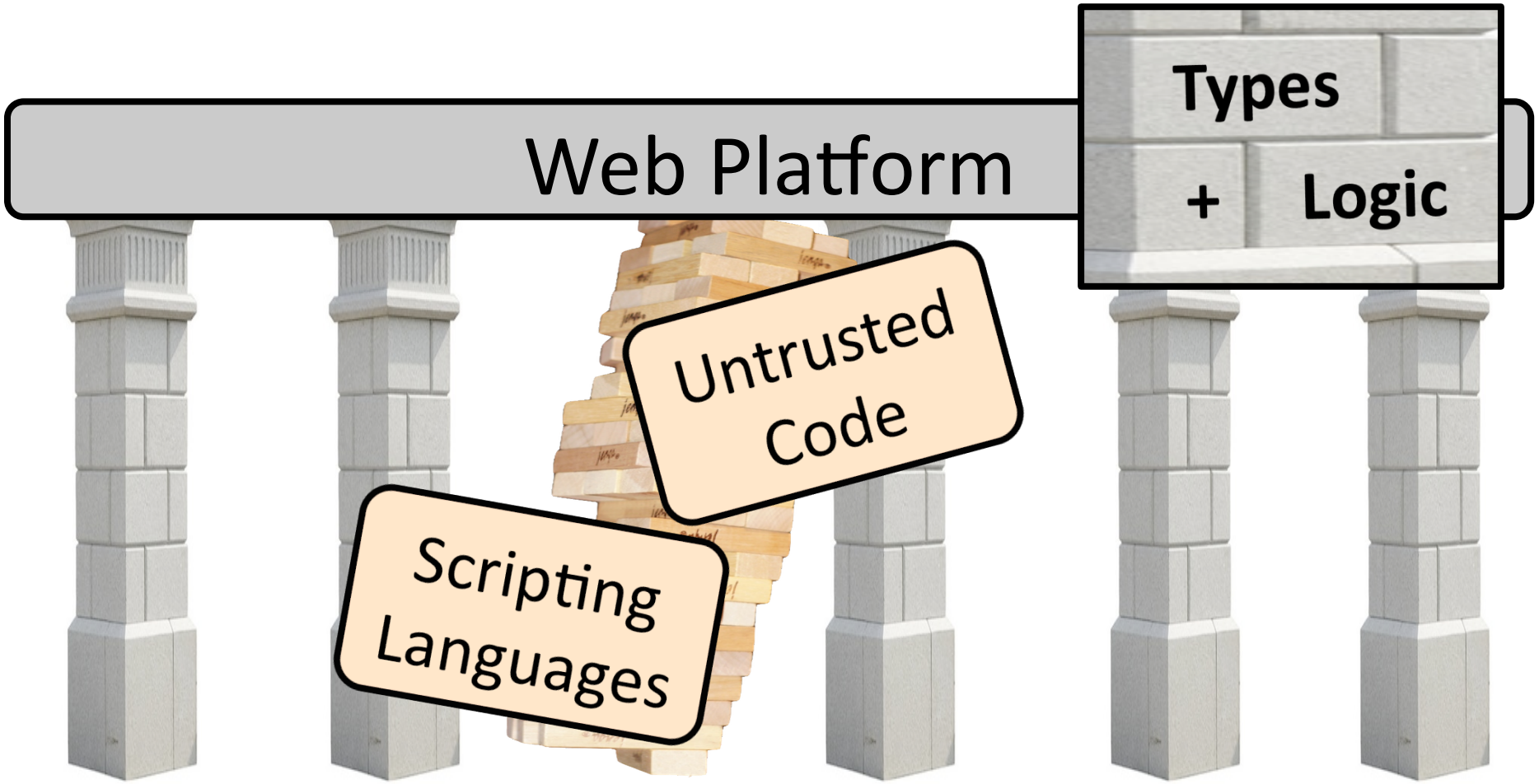
Ravi Chugh

UCSDCSE
Computer Science and Engineering

Web Platform



Goal: Rigorous Foundations



1995 Web Page = Static HTML + Links + Client-Side Code

```
<html>
```

Username

Password

Login

```
</html>
```

```
<script type="javascript">

loginButton.onclick = function () {
  if (passwordBox.value == "") {
    alert("Enter password!");
  } else {
    ...
} }

</script>
```

1995

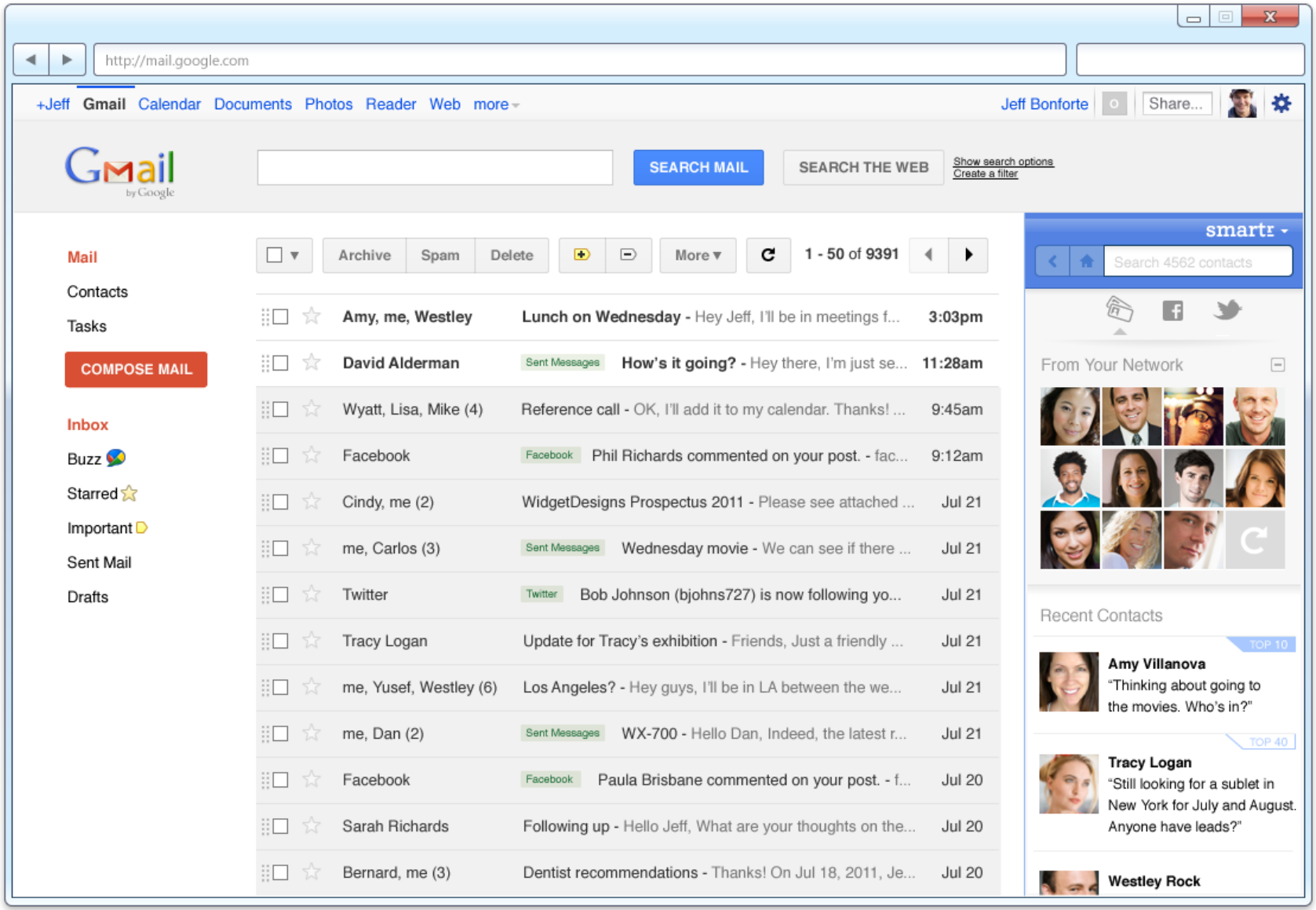
No Static Types

"**Dynamic**" Features Make it Easy to Experiment…

… an easy-to-use "scripting language" as a companion to Java

Brendan Eich,
JavaScript creator

2014


http://mail.google.com
+Jeff Gmail Calendar Documents Photos Reader Web more
Jeff Bonforte
Share...
Gmail by Google
SEARCH MAIL
SEARCH THE WEB
Show search options
Create a filter
Mail
Contacts
Tasks
COMPOSE MAIL
Inbox
Buzz
Starred
Important
Sent Mail
Drafts
Archive
Spam
Delete
More
1 - 50 of 9391
Amy, me, Westley Lunch on Wednesday - Hey Jeff, I'll be in meetings f... 3:03pm
David Alderman Sent Messages How's it going? - Hey there, I'm just se... 11:28am
Wyatt, Lisa, Mike (4) Reference call - OK, I'll add it to my calendar. Thanks! ... 9:45am
Facebook Facebook Phil Richards commented on your post. - fac... 9:12am
Cindy, me (2) WidgetDesigns Prospectus 2011 - Please see attached ... Jul 21
me, Carlos (3) Sent Messages Wednesday movie - We can see if there ... Jul 21
Twitter Twitter Bob Johnson (bjohns727) is now following yo... Jul 21
Tracy Logan Update for Tracy's exhibition - Friends, Just a friendly ... Jul 21
me, Yusef, Westley (6) Los Angeles? - Hey guys, I'll be in LA between the we... Jul 21
me, Dan (2) Sent Messages WX-700 - Hello Dan, Indeed, the latest r... Jul 21
Facebook Facebook Paula Brisbane commented on your post. - f... Jul 20
Sarah Richards Following up - Hello Jeff, What are your thoughts on the... Jul 20
Bernard, me (3) Dentist recommendations - Thanks! On Jul 18, 2011, Je... Jul 20
smartr
Search 4562 contacts
From Your Network
Recent Contacts
TOP 10
Amy Villanova
"Thinking about going to the movies. Who's in?"
TOP 40
Tracy Logan
"Still looking for a sublet in New York for July and August. Anyone have leads?"
Westley Rock

2014

Asked whether Gmail would ever open up an API for developers to code add-ons and plug-ins with more official support and stability, Kowitz suggested it was unlikely. **Gmail is based on "hundreds of thousands" of lines of JavaScript**, Kowitz said, and **it's a code base that "changes so quickly."** That said, Jackson said the Gmail team "loves" to see third-party functionality being developed, and the team tries to make developers aware of changes they know will affect those products.

Article on LifeHacker

2014

Asked whether Gmail would ever open up an API for developers to code add-ons and plug-ins with more official support and stability, Kowitz suggested it was unlikely. **Gmail is based on "hundreds of thousands" of lines of JavaScript**, Kowitz said, and **it's a code base that "changes so quickly."** That said, Jackson said the Gmail team "loves" to

"Scripting"

?!?

2014

JavaScript is Pervasive

Count	Tag
599,605	C#
583,322	Java
555,164	JavaScript
532,851	PHP
273,086	Python
268,725	C++
129,483	C
97,154	Ruby
15,044	Haskell
2,175	OCaml

http://stackoverflow.com/tags

02-26-14

2014

JavaScript is Pervasive

Other Dynamic Languages Are, Too

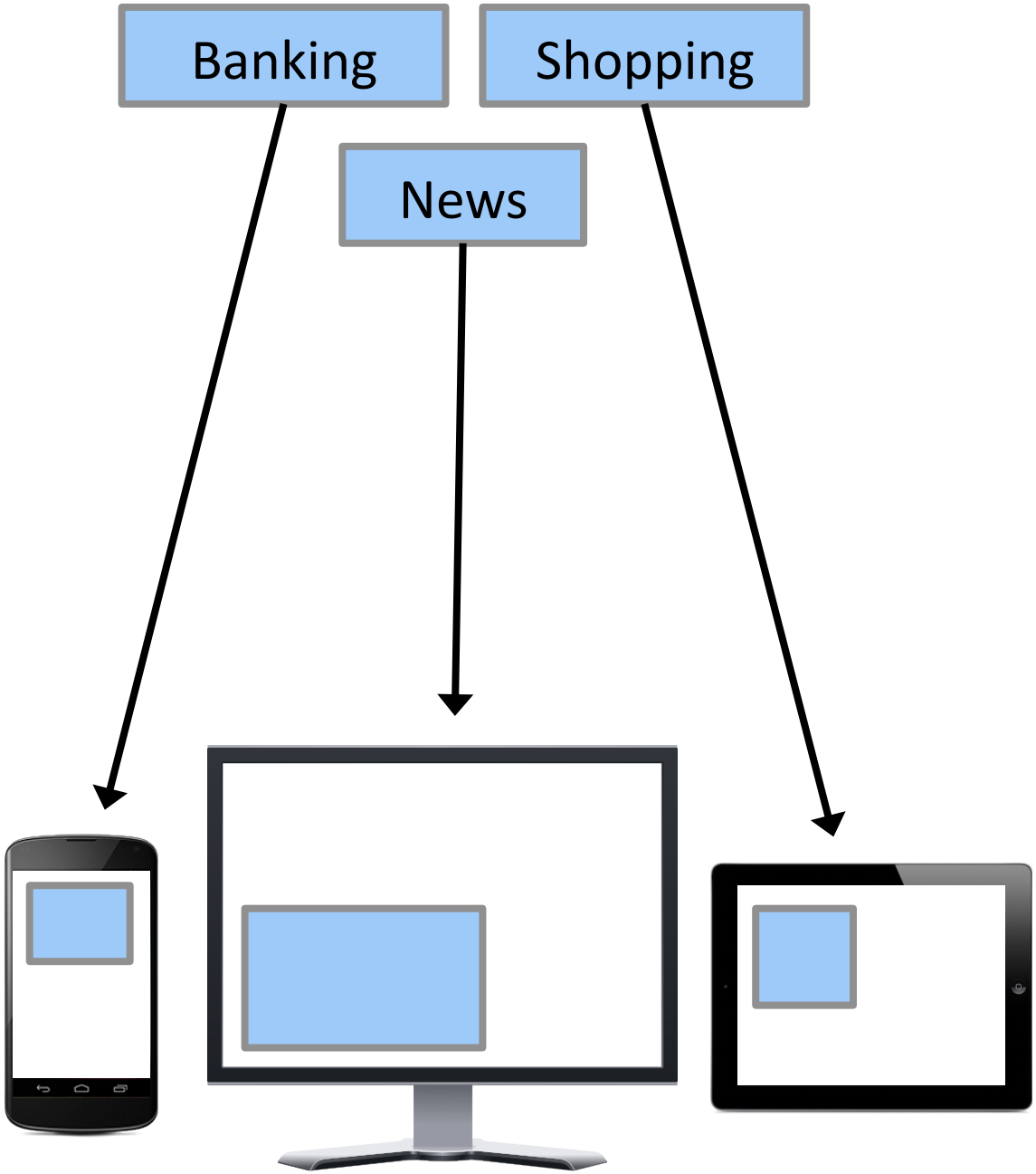
Count	Tag
599,605	C#
583,322	Java
555,164	JavaScript
532,851	PHP
273,086	Python
268,725	C++
129,483	C
97,154	Ruby
15,044	Haskell
2,175	OCaml

http://stackoverflow.com/tags

02-26-14

2014

JavaScript is Pervasive



2014

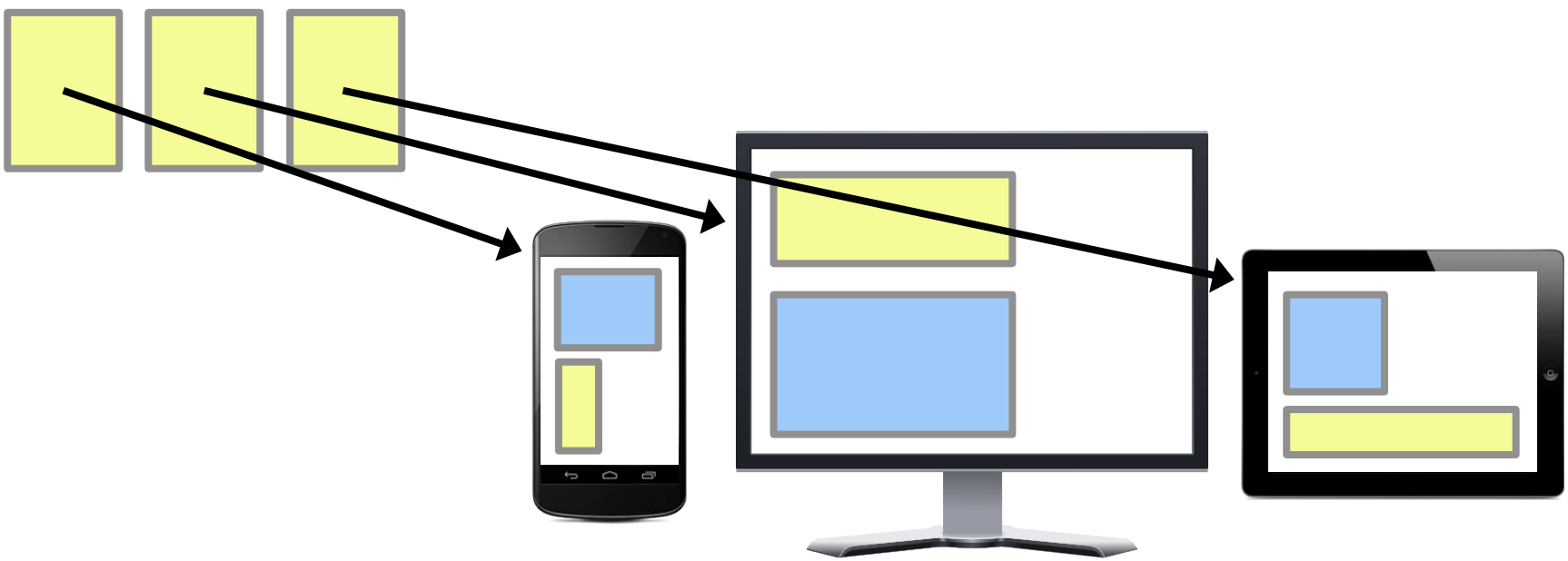
Banking

Shopping

News

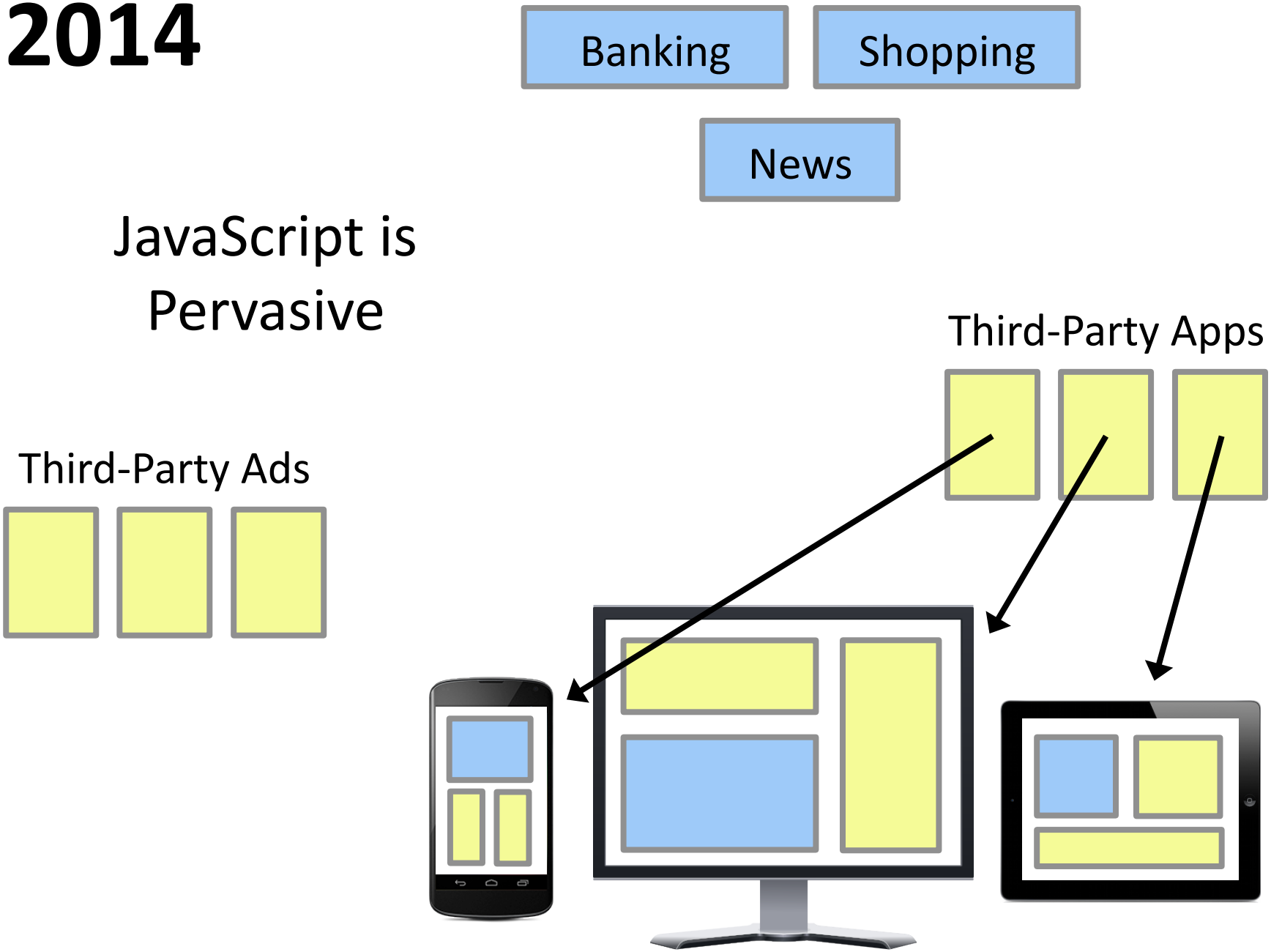
JavaScript is Pervasive

Third-Party Ads



2014

JavaScript is Pervasive



2014

Third-Party
JavaScript is
Pervasive

2014

No longer
just “scripting”

… an easy-to-use “scripting language”
as a companion to Java

Significant Interest For:

Closure IDE
IDE for Google Closure Tools

facebook developers

TypeScript PREVIEW

1. Development Tools
2. Reliability / Security
3. Performance

AD safe

asm.js

google-caja
Compiler for making third-party HTML, CSS and JavaScript safe for embedding

But JS is hard to reason about!

My Ph.D.

Static Types for Dynamic Languages

JavaScript, Python, Ruby, PHP
Share **Common Challenges**

New Techniques **Apply Broadly**,
Even to Statically Typed Languages

Static Types

Describe sets of run-time values

"booleans"

"integers"

"objects with `foo` field"

"non-`null` pointers"

"urls that are safe to visit"

Static Types

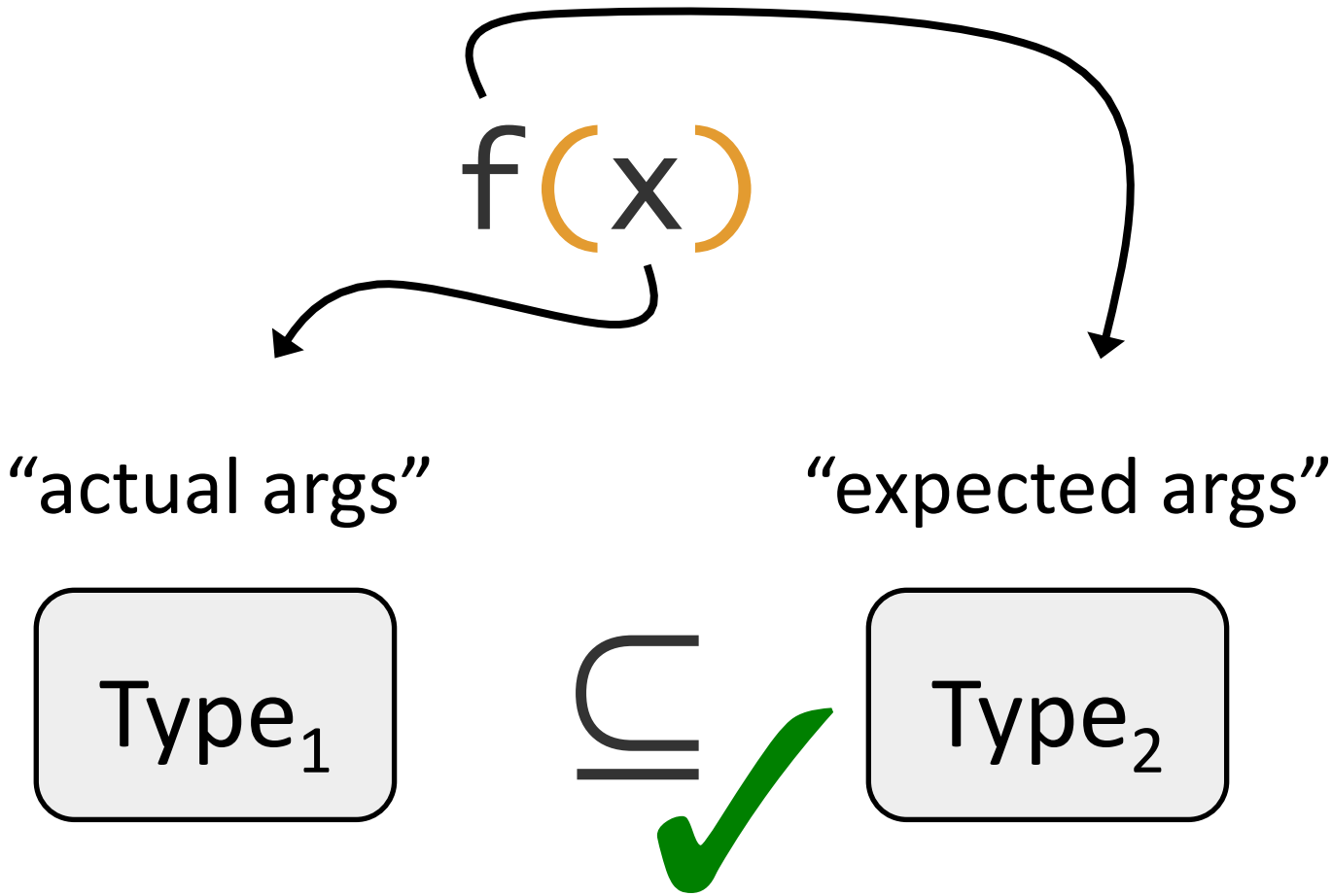
Prevent classes of run-time errors

"multiply int and bool"

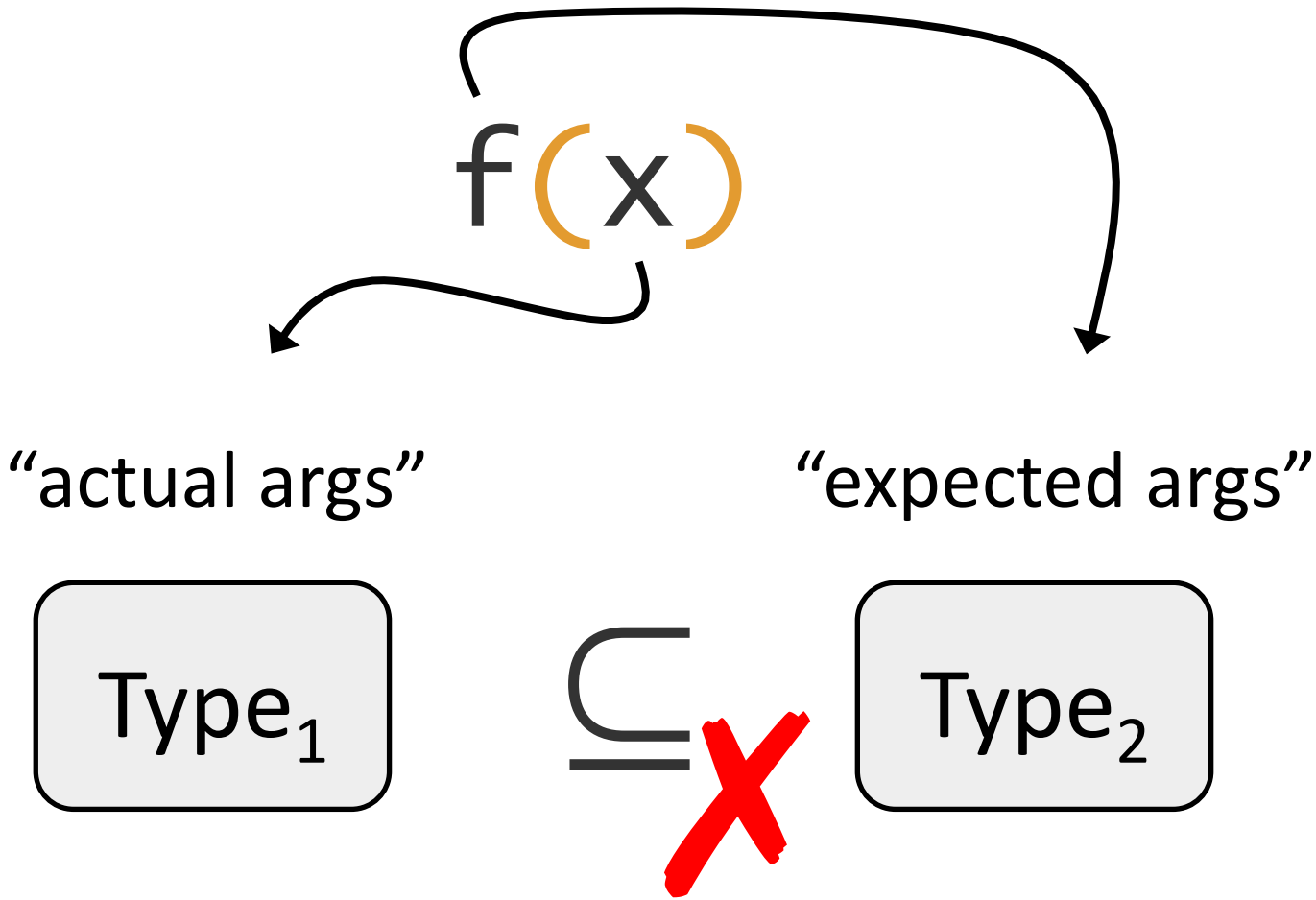
"`foo` field not found"

"`null` pointer exception"

"redirect to `evil.com`"



"function call produces **no** error"



"function call **may** produce error"

For all programs p in PROGRAMS

If p has a type T in TYPES

Then p does not fail
with any error e in ERRORS

GOAL: Reason About More Programs

For all programs p in PROGRAMS

If p has a type T in TYPES

Then p does not fail with any error e in ERRORS

GOAL: Reason About More Programs

For all programs p in PROGRAMS

If p has a type T in TYPES

Then p does not fail with any error e in ERRORS

GOAL: Eliminate More Errors

Types

Refinement Types

Types + Logic

TYPES

Refinement Types

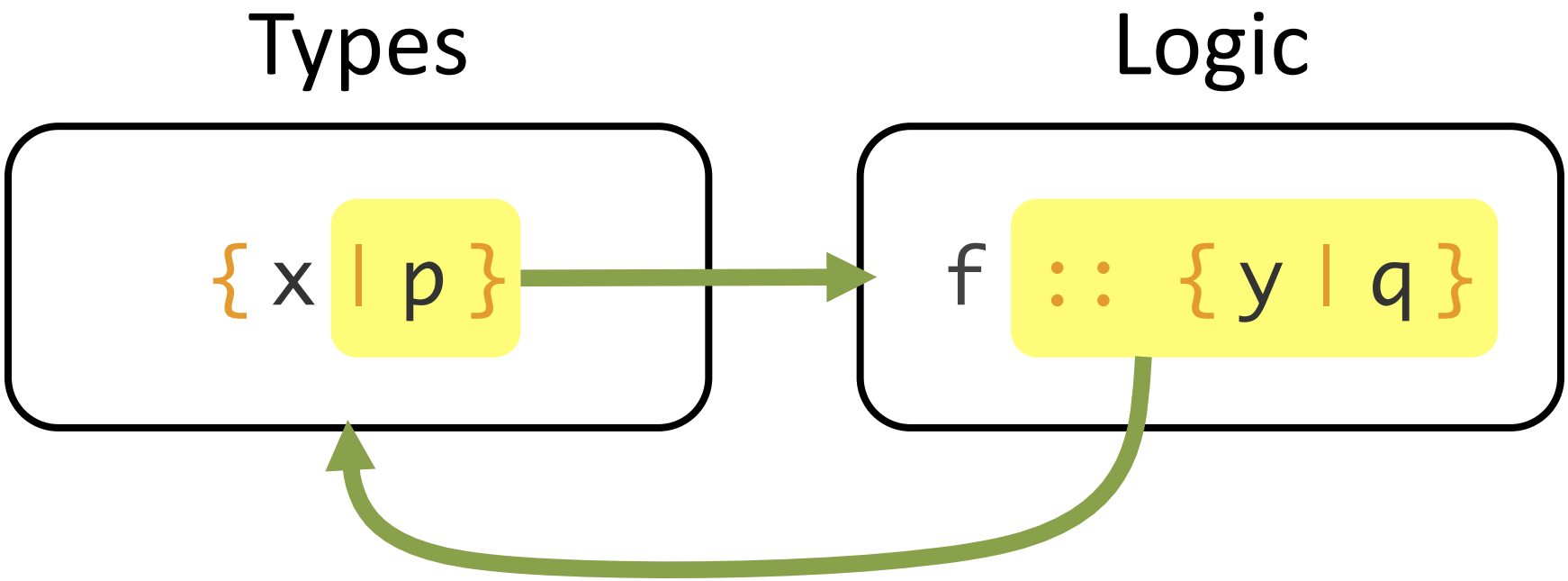
Types + Logic

"actual args" "expected args"

$$\{ x \mid p \} \subseteq \{ x \mid q \}$$

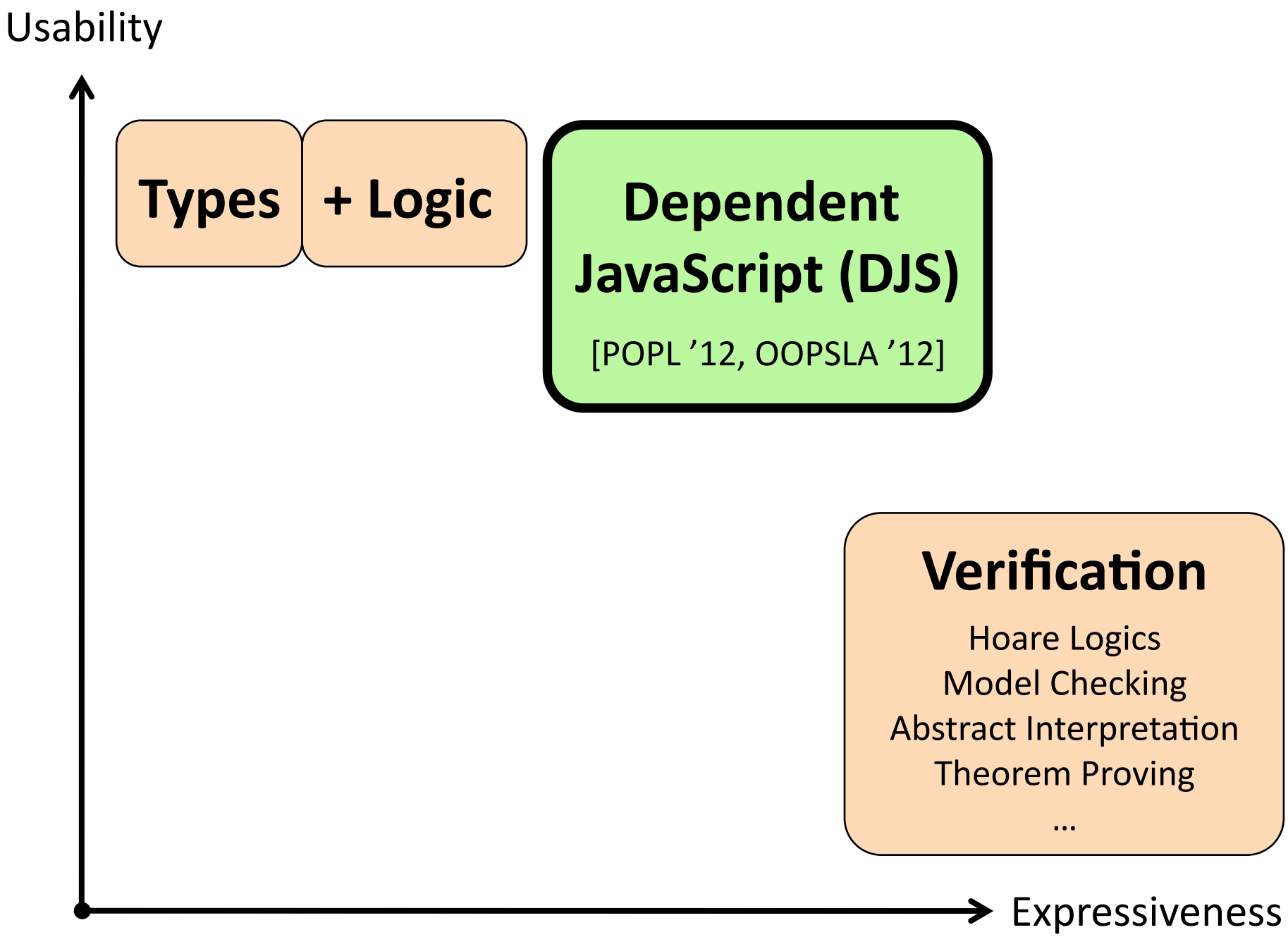
$$p \Rightarrow q$$

Refinement Types



Nested Refinements [POPL '12]

Key to Encode Dynamic Features


Usability
Types
+ Logic
Dependent JavaScript (DJS)
[POPL '12, OOPSLA '12]
Verification
Hoare Logics
Model Checking
Abstract Interpretation
Theorem Proving
...
Expressiveness

Outline

Why is JavaScript Important?

Why is JavaScript Hard? — PROGRAMS

Static Types via Logic — TYPES

Security via Logic — ERRORS

Looking Ahead

Outline

Why is JavaScript Important?

Why is JavaScript Hard? PROGRAMS

Static Types via Logic TYPES

Security via Logic ERRORS

Looking Ahead

Outline

Why is JavaScript Hard? PROGRAMS

- Lightweight Reflection
- Dictionary Objects
- Additional Challenges

```
function negate(x) {
  if (typeof x == "number") {

}
```

Q: What is my "type"?

All JavaScript Values

"number"
-1 17 42 3.14 ...

Q: What is my "type"?

X

"boolean"
true false

"object"
{ } { "f":17 } ... { "f":42, "g":true }

```
function negate(x) {
  if (typeof x == "number") {
```

Q: What is my "type"?

A: Described by **"tag"**

```
}
```

```
function negate(x) {
  if (typeof x == "number") {
    return 0 - x;
  } else {
    return !x;
  }
}
```

"expected args"

values with tag
"number" **OR** "boolean"

Outline

Why is JavaScript Hard? PROGRAMS

Lightweight Reflection

Dictionary Objects

Additional Challenges

`obj.f` means `obj["f"]`

Objects are that map string keys to values

"Dictionaries"

a.k.a.
"maps"
"hash tables"
"associative arrays"

```
function dispatch(table, op, i, j) {

}
```

method table

method to invoke

```
function dispatch(table, op, i, j) {
  var fn = table[op];
  return fn(i, j);
}
var integerOps =
  { "+" : function (n, m) { … } ,
    "-" : function (n, m) { … } };

dispatch (integerOps, "*", 17, 42);
```

"*" key not found

```
function dispatch(table, op, i, j) {
  var fn = table[op];
  return fn(i, j);
}
```

"expected args"

Dependency between types of arguments

`table` object with an `op` key that stores a function on numbers

Outline

Why is JavaScript Hard? PROGRAMS

Lightweight Reflection

Dictionary Objects

Additional Challenges

Outline

Why is JavaScript Hard? PROGRAMS

Lightweight Reflection

Dictionary Objects

Extensible Objects

Prototype Inheritance

JavaScript Peculiarities

Outline

Why is JavaScript Hard? PROGRAMS

Lightweight Reflection

Unsupported in Prior Work

- Dictionary Objects
- Extensible Objects
- Prototype Inheritance
- JavaScript Peculiarities

Outline

Why is JavaScript Important?

Why is JavaScript Hard? — PROGRAMS

Static Types via Logic — TYPES

Security via Logic — ERRORS

Looking Ahead

All JavaScript Values

“number”
-1 17 42 3.14 ...

{ } { “f”:17 }

... { “f”:42, “g”:true }

true false

All JavaScript Values

{ x | tag(x) = "number" }

-1 17 42 3.14 ...

{ x | tag(x) = "boolean" }

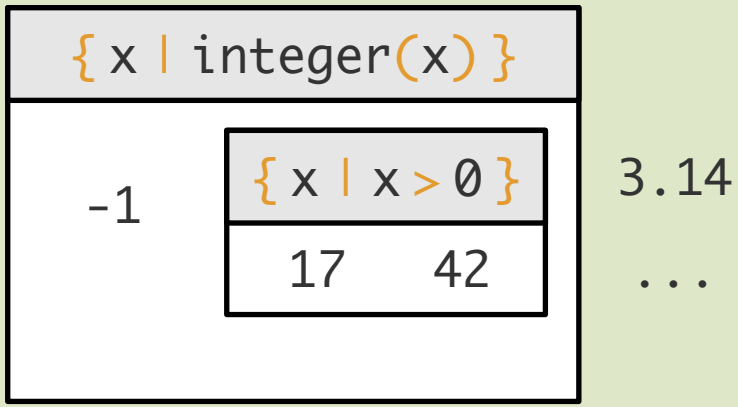
true false

{ x | tag(x) = "object" }

{ } { "f":17 }

... { "f":42, "g":true }

All JavaScript Values



{ } { "f" : 17 }

... { "f" : 42 , "g" : true }

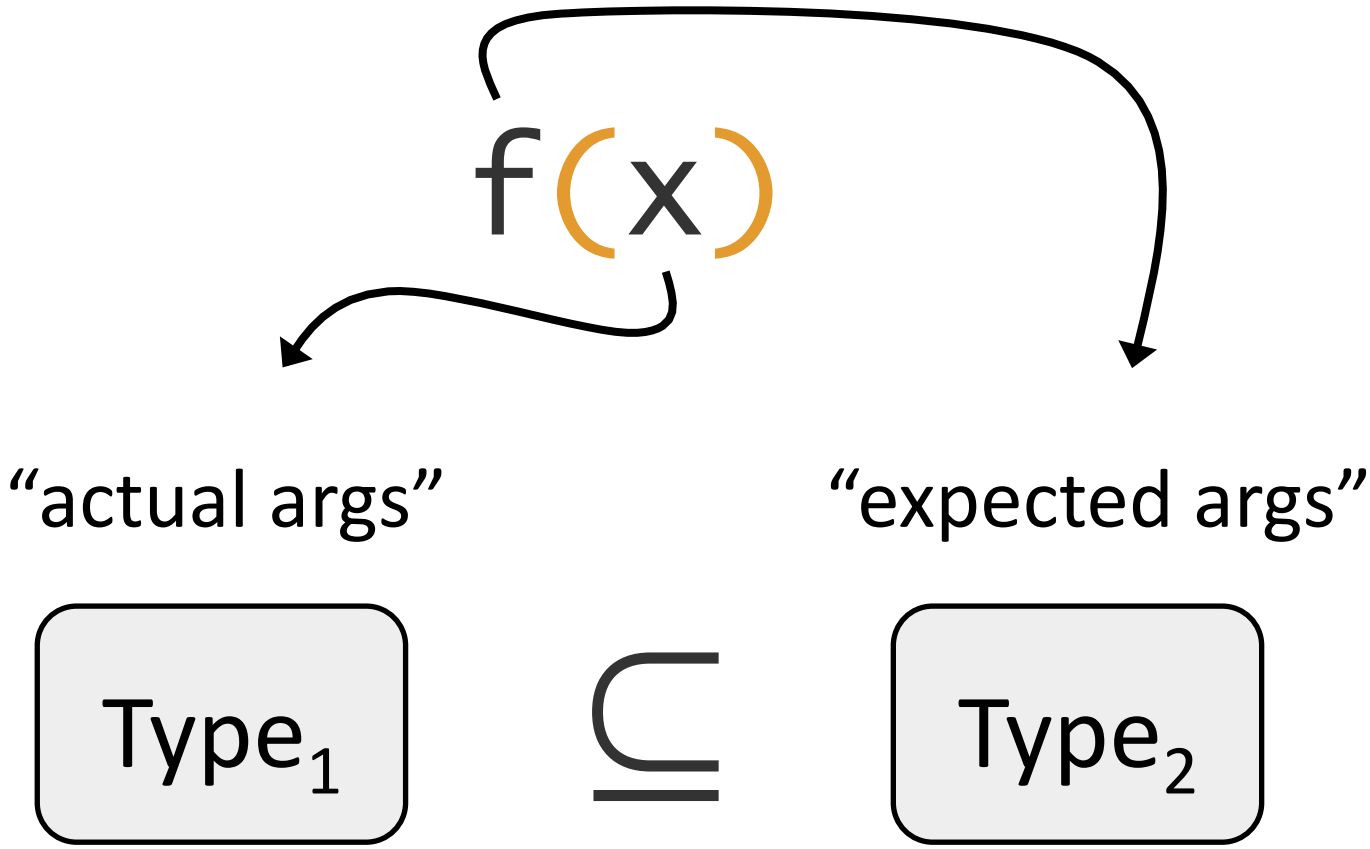
true false

Refinement Types

All JavaScript Values

$$\{ x \mid p \}$$

"set of values x s.t. formula p is true"


f(x)
"actual args"
"expected args"
Type1
⊆
Type2

$$f(x)$$

"actual args" "expected args"

$$\{x \mid p\} \subseteq \{x \mid q\}$$

$$p \Rightarrow q$$

Subtyping is **Implication**

Source of Expressive Power

$$\{ x \mid p \} \subseteq \{ x \mid q \}$$

$$p \Rightarrow q$$

Outline

Static Types via Logic

TYPES

- Lightweight Reflection
- Dictionary Objects
- Key Technical Innovations
- Evaluation

```
//: negate :: (x:NumOrBool) → NumOrBool
```

```
function negate(x) {
  if (typeof x == "number") {
    return 0 - x;
  } else {
    return !x;
} }
```

Type **Annotation** in Comments

Syntactic Sugar for Common Types

NumOrBool ≡

$$\{ v \mid tag(v) = \text{“number”} \vee tag(v) = \text{“boolean”} \}$$

```
//: negate :: (x:NumOrBool) → NumOrBool

function negate(x) {
  if (typeof x == "number") {
    return 0 - x;
  } else {
    return !x;
} }
```

"expected args"

$\{ x \mid tag(x) = \text{"number"} \} \subseteq \{ x \mid tag(x) = \text{"number"} \}$

$tag(x) = \text{"number"} \Longrightarrow tag(x) = \text{"number"}$ ✓

```
//: negate :: (x:NumOrBool) → NumOrBool

function negate(x) {
  if (typeof x == "number") {
    return 0 - x;
  } else {
    return !x;
} }
```

$\{ x \mid not(tag(x) = \text{“number”}) \wedge (tag(x) = \text{“number”} \vee tag(x) = \text{“boolean”}) \}$

$\subseteq$

“expected args”

$\{ x \mid tag(x) = \text{“boolean”} \}$

```
//: negate :: (x:NumOrBool) → NumOrBool

function negate(x) {
  if (typeof x == "number") {
    return 0 - x;
  } else {
    return !x;
} }
```

Logic Enables

Path-Sensitive

Reasoning on Branches

```
//: negate :: (x:NumOrBool) → { y | tag(y) = tag(x) }
function negate(x) {
  if (typeof x == "number") {
    return 0 - x;
  } else {
    return !x;
} }
```

Logic Enables Types to **Depend** on Values

Outline

Static Types via Logic

TYPES

- Lightweight Reflection
- Dictionary Objects
- Key Technical Innovations
- Evaluation

All JavaScript Values

$\{ x \mid tag(x) = \text{“object”} \}$

{ }

{ “f”:17 }

{ “f”:true }

{ “f”:42, “g”:null }

. . .

All JavaScript Values

{ }

{ x | tag(x) = "object" ∧ has(x, "f") }

{ "f": 17 }

{ "f": true }

{ "f": 42, "g": null }

. . .

All JavaScript Values

{ }

{ x | tag(x) = "object" ∧
has(x, "f") ∧
tag(sel(x, "f")) = "number" }

{ "f": 17 }

{ "f": true }

{ "f": 42, "g": null }

. . .

```
//: dispatch ::
//:   ({ table | tag(table) = "object" },
//:    { op    | tag(op) = "string"
//:            ∧ has(table,op)
//:            ∧ sel(table,op) :: (Num,Num) → Num }
//:    { i     | tag(i) = "number" },
//:    { j     | tag(j) = "number" })
//: → Num
function dispatch(table, op, i, j) {
  var fn = table[op];
  return fn(i, j);
}
```

```
//: dispatch ::
//:   (table : { sel(table,op) :: (Num,Num) → Num },
//:    op    : Str,
//:    i     : Num,
//:    j     : Num)
//:
//:
//: → Num
function dispatch(table, op, i, j) {
  var fn = table[op];
  return fn(i, j);
}
```

```
var integerOps = { "+" : …, "-" : … };
dispatch (integerOps, "*", 17, 42);
```

"actual args"

$\{ op \mid op = \text{"*"} \} \subseteq \{ op \mid has(integerOps, op) \}$

$op = \text{"*"} \Rightarrow$ ✗ $has(integerOps, op)$

Type Checking Prevents Run-Time Error

Outline

Static Types via Logic

- Lightweight Reflection
- Dictionary Objects
- Key Technical Innovations
- Evaluation

	Prior Refinement Types	**Dependent JavaScript (DJS)** [POPL '12, OOPSLA '12]
lightweight reflection	✓	✓
dictionary objects	✗	✓
extensible objects	✗	✓
prototype inheritance	✗	✓
JavaScript peculiarities	✗	✓

Prior Refinement Types

Dependent JavaScript (DJS)

[POPL '12, OOPSLA '12]

dictionary objects ✗ ✓

Key Challenge:

Function Subtyping as Logical Implication

```
42 + negate(17)
```

```
{ f | f :: (x:NumOrBool) → { y | tag(y) = tag(x) } }
```

$\subseteq$

```
{ f | f :: (x:Num) → Num }
```

"expected function"

How to Prove

```
f :: (x:NumOrBool) → { y | tag(y) = tag(x) }
```

⇒

```
f :: (x:Num) → Num
```

?

How to Encode Type System **Inside** Logic?

Prior Refinement Systems
Disallow Function Types Inside Logic!

Types

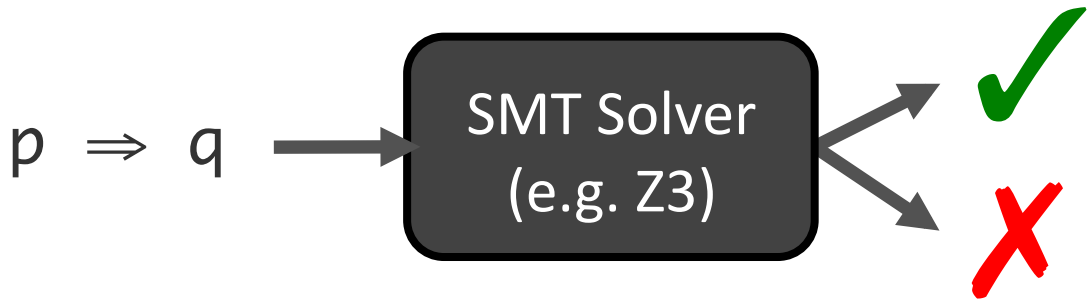
$$T ::= \{ x \mid p \}$$
$$\mid \; T_1 \rightarrow T_2$$

Refinement Logic

[Prior State-of-the-Art]

<u>Satisfiability Modulo Theories (SMT)</u>

SAT + Decidable Theories



Types

$$T ::= \{ x \mid p \} \mid T_1 \rightarrow T_2$$

Refinement Logic

$$p ::= p \wedge q \mid p \vee q \mid \neg p \mid x = y \mid x > y \mid \ldots \mid \texttt{tag}(x) = \text{“number”} \mid \ldots \mid \texttt{sel}(x,k) = 17 \mid \ldots$$

- Boolean Connectives
- Equality
- Linear Arithmetic
- Uninterpreted Functions
- McCarthy Arrays

Types

$$T ::= \{ x \mid p \} \mid T_1 \rightarrow T_2$$

Refinement Logic

$$p ::= p \wedge q \mid p \vee q \mid \neg p \mid x = y \mid x > y \mid \ldots \mid \mathtt{tag}(x) = \text{“number”} \mid \ldots \mid \mathtt{sel}(x, k) = 17 \mid \ldots$$

Enables Union Types and **Lightweight Reflection**

$$\mathtt{NumOrBool} \equiv \{ v \mid \mathtt{tag}(v) = \text{“number”} \vee \mathtt{tag}(v) = \text{“boolean”} \}$$

Types

$$T ::= \{ x \mid p \} \mid T_1 \rightarrow T_2$$

Refinement Logic

$$p ::= p \wedge q \mid p \vee q \mid \neg p \mid x = y \mid x > y \mid \ldots \mid \mathtt{tag}(x) = \text{“number”} \mid \ldots \mid \mathtt{sel}(x,k) = 17 \mid \ldots$$

Enables **Dictionary Types** with Dynamic Keys …

But **Disallows** Functions in Dictionaries!

$$\{ d \mid \mathtt{tag}(\boxed{\mathtt{sel}(d,k)}) = \text{“boolean”} \wedge \mathtt{tag}(\mathtt{sel}(d,\text{“f”})) = \text{“function”} \wedge \boxed{\mathtt{sel}(d,\text{“f”})\ ???} \}$$

Types

$$T ::= \{ x \mid p \} \quad \cancel{\mid\ T_1 \to T_2}$$

Refinement Logic

$$\begin{aligned} p ::= \ & p \wedge q \mid p \vee q \mid \neg p \\ \mid\ & x = y \mid x > y \mid \ldots \\ \mid\ & \mathtt{tag}(x) = \text{“number”} \mid \ldots \\ \mid\ & \mathtt{sel}(x,k) = 17 \mid \ldots \\ \mid\ & \mathtt{sel}(x,k) :: T_1 \to T_2 \mid \ldots \end{aligned}$$

<u>Goal</u>: Refer to Type System Inside Logic Without Giving up Decidability

<u>Solution</u>: **Nested Refinements!**

Types

```
T ::= { x | p }
```

Refinement Logic

```
p ::= p ∧ q | p ∨ q | ¬p
    | x = y | x > y | …
    | tag(x) = "number" | …
    | sel(x,k) = 17 | …
    | sel(x,k) :: T1 → T2 | …
```

Nested Refinements [POPL '12]

1. Decidable Encoding
2. Precise Subtyping
3. Type Soundness Proof

Decidable Encoding

```
f :: (x:NumOrBool) → { y | tag(y) = tag(x) }
⇒ f :: (x:Num) → Num
```

Uninterpreted Predicate in Logic

Distinct **Uninterpreted** Constants in Logic…

Decidable Encoding

f :: (x:NumOrBool) → { y | tag(y) = tag(x) }

⟹ f :: (x:Num) → Num

Uninterpreted
Predicate in Logic

Distinct **Uninterpreted**
Constants in Logic…
w/ Types **Nested** Inside

Decidable Encoding

```
f :: (x:NumOrBool) → { y | tag(y) = tag(x) }
```

⟹

```
f :: (x:Num) → Num
```

SMT Solver ✗

Trivially Decidable, but **Imprecise**

Precise Subtyping

$$\{ f \mid p \} \subseteq \{ f \mid f :: (x{:}T_1) \rightarrow T_2 \}$$

Step 1: Find $(x{:}S_1) \rightarrow S_2$ such that

$$p \Rightarrow f :: (x{:}S_1) \rightarrow S_2$$

SMT Solver

Step 2: Compare $(x{:}S_1) \rightarrow S_2$ and $(x{:}T_1) \rightarrow T_2$

"contra-variance" $T_1 \subseteq S_1$

"co-variance" $x{:}T_1 \wedge S_2 \subseteq T_2$

Precise Subtyping

$\{ f \mid f = \texttt{negate} \} \subseteq \{ f \mid f :: (x{:}\texttt{Num}) \rightarrow \texttt{Num} \}$

Step 1:

$f = \texttt{negate}$

SMT Solver ✓ $\Rightarrow$ $f :: (x{:}\texttt{NumOrBool}) \rightarrow \{ y \mid \texttt{tag}(y) = \texttt{tag}(x) \}$

Step 2:

Precise Subtyping

$\{ f \mid f = \texttt{negate} \} \subseteq \{ f \mid f :: (x{:}\texttt{Num}) \to \texttt{Num} \}$ ✓

Step 1:

$f = \texttt{negate}$

SMT Solver ✓ $\Rightarrow$ $f :: (x{:}\texttt{NumOrBool}) \to \{ y \mid \texttt{tag}(y) = \texttt{tag}(x) \}$

Step 2:

$\texttt{Num} \subseteq \texttt{NumOrBool}$ ✓

$x{:}\texttt{Num} \wedge \{ y \mid \texttt{tag}(y) = \texttt{tag}(x) \} \subseteq \texttt{Num}$ ✓

Precise Subtyping

{ f | f = negate } ⊆ { f | f :: (x:Num) → Num }

Step 1: **Decidable** Reasoning via SMT

+

Step 2: **Precise** Function Subtyping via Syntactic Techniques

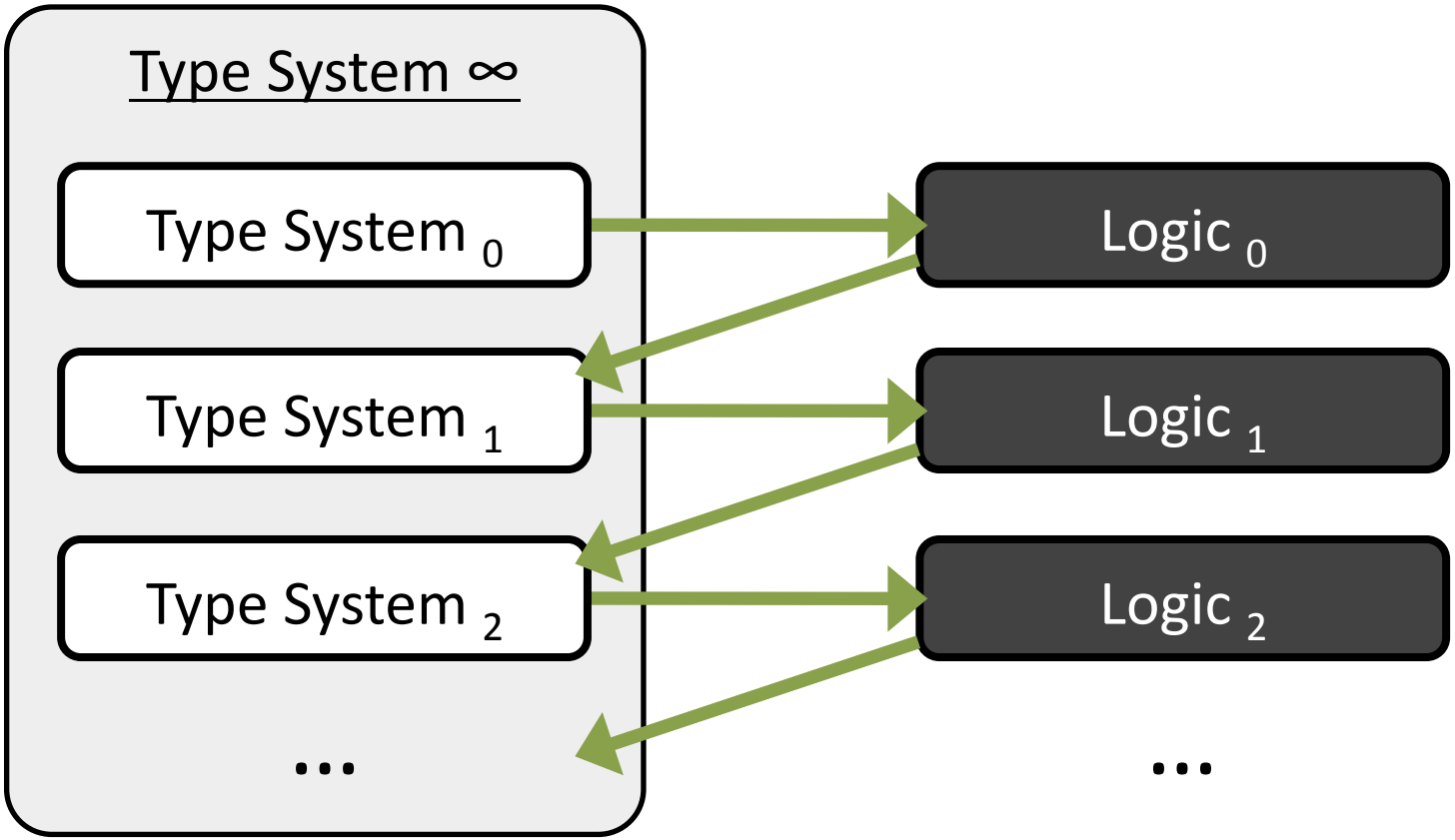
Type Soundness Proof

Conventional Proofs
Require Logic to be an **Oracle** …

… but Nesting Introduces **Circularity**
That Prevents Typical Inductive Proofs

Type Soundness Proof

Proof Technique:
Stratify into Infinite Hierarchy



Types

$$T ::= \{ x \mid p \}$$

Refinement Logic

$$\begin{aligned} p ::= \; & p \wedge q \mid p \vee q \mid \neg p \\ \mid \; & x = y \mid x > y \mid \ldots \\ \mid \; & \mathtt{tag}(x) = \text{“number”} \mid \ldots \\ \mid \; & \mathtt{sel}(x,k) = 17 \mid \ldots \\ \mid \; & \mathtt{sel}(x,k) :: T_1 \rightarrow T_2 \mid \ldots \end{aligned}$$

Nested Refinements [POPL ’12]

Key to Encode Dictionary Objects

(Even in Statically Typed Languages!)

	Prior Refinement Types	Dependent JavaScript (DJS) [POPL '12, OOPSLA '12]	
lightweight reflection	✓	✓	**path-sensitivity**
dictionary objects	✗	✓	**nested refinements**
extensible objects	✗	✓	**flow-sensitivity**
prototype inheritance	✗	✓	**logical encoding**
JavaScript peculiarities	✗	✓	**logical encoding**

Dependent JavaScript (DJS)

[POPL '12, OOPSLA '12]

JavaScript, Python, Ruby

lightweight reflection

dictionary objects

extensible objects

JavaScript

prototype inheritance

JavaScript peculiarities

- ✓ **path-sensitivity**
- ✓ **nested refinements**
- ✓ **flow-sensitivity**
- ✓ **logical encoding**
- ✓ **logical encoding**

Statically Typed Languages
(Java, C++, OCaml, Haskell, …)

- lightweight reflection
- dictionary objects
- extensible objects
- prototype inheritance
- JavaScript peculiarities

Dependent JavaScript (DJS)
[POPL '12, OOPSLA '12]

- ✓ **path-sensitivity**
- ✓ **nested refinements**
- ✓ **flow-sensitivity**
- ✓ **logical encoding**
- ✓ **logical encoding**

Statically Typed Languages
(Java, C++, OCaml, Haskell, …)

Dependent JavaScript (DJS)

[POPL '12, OOPSLA '12]

General Techniques Can Be Used to Eliminate More

ERRORS

✓ **path-sensitivity**

✓ **nested refinements**

✓ **flow-sensitivity**

Outline

Static Types via Logic

TYPES

Lightweight Reflection

Dictionary Objects

Key Technical Innovations

Evaluation

Benchmarks	LOC before	LOC after
13 Excerpts from: *JavaScript, Good Parts* SunSpider Benchmark Suite Google Closure Library	306	408 (+33%)

Chosen to Stretch Expressiveness Limits of DJS

Benchmarks	LOC before	LOC after
13 Excerpts from: *JavaScript, Good Parts* SunSpider Benchmark Suite Google Closure Library	306	408 (+33%)
9 Browser Extensions from: [Guha et al. Oakland '11]	321	383 (+19%)
2 Examples from: Google Gadgets	1,003	1,027 (+2%)
TOTALS	**1,630**	**1,818** **(+12%)**

Type Annotations

Relatively Simple **Syntactic Sugar** and **Type Inference** Have Helped Significantly

Opportunities for Improved Type Inference:

- Iterative Predicate Abstraction
- Bootstrap by **Running Test Cases** [STOP '11]
- Translating Programs from TypeScript

TOTALS	1,630	1,818 (+12%)

Benchmarks	LOC before/after		Running Time
13 Excerpts from: *JavaScript, Good Parts* SunSpider Benchmark Suite Google Closure Library	306	408 (+33%)	10 sec
9 Browser Extensions from: [Guha et al. Oakland '11]	321	383 (+19%)	3 sec
2 Examples from: Google Gadgets	1,003	1,027 (+2%)	19 sec
TOTALS	**1,630**	**1,818** **(+12%)**	**32 sec**

Performance

Relatively Simple Optimizations So Far:

- Avoid SMT Solver When Possible
- Reduce Precision for Common Patterns
- Rely on Nested Refinements When Needed

TOTALS	**1,630**	**1,818 (+12%)**	**32 sec**

Outline

Static Types via Logic

TYPES

Lightweight Reflection

Dictionary Objects

Key Technical Innovations

Evaluation

Outline

Why is JavaScript Important?

Why is JavaScript Hard? PROGRAMS

Static Types via Logic TYPES

Security via Logic ERRORS

Looking Ahead

Outline

Security via Logic

ERRORS

Browser Extensions

Types for Fine-Grained Policies

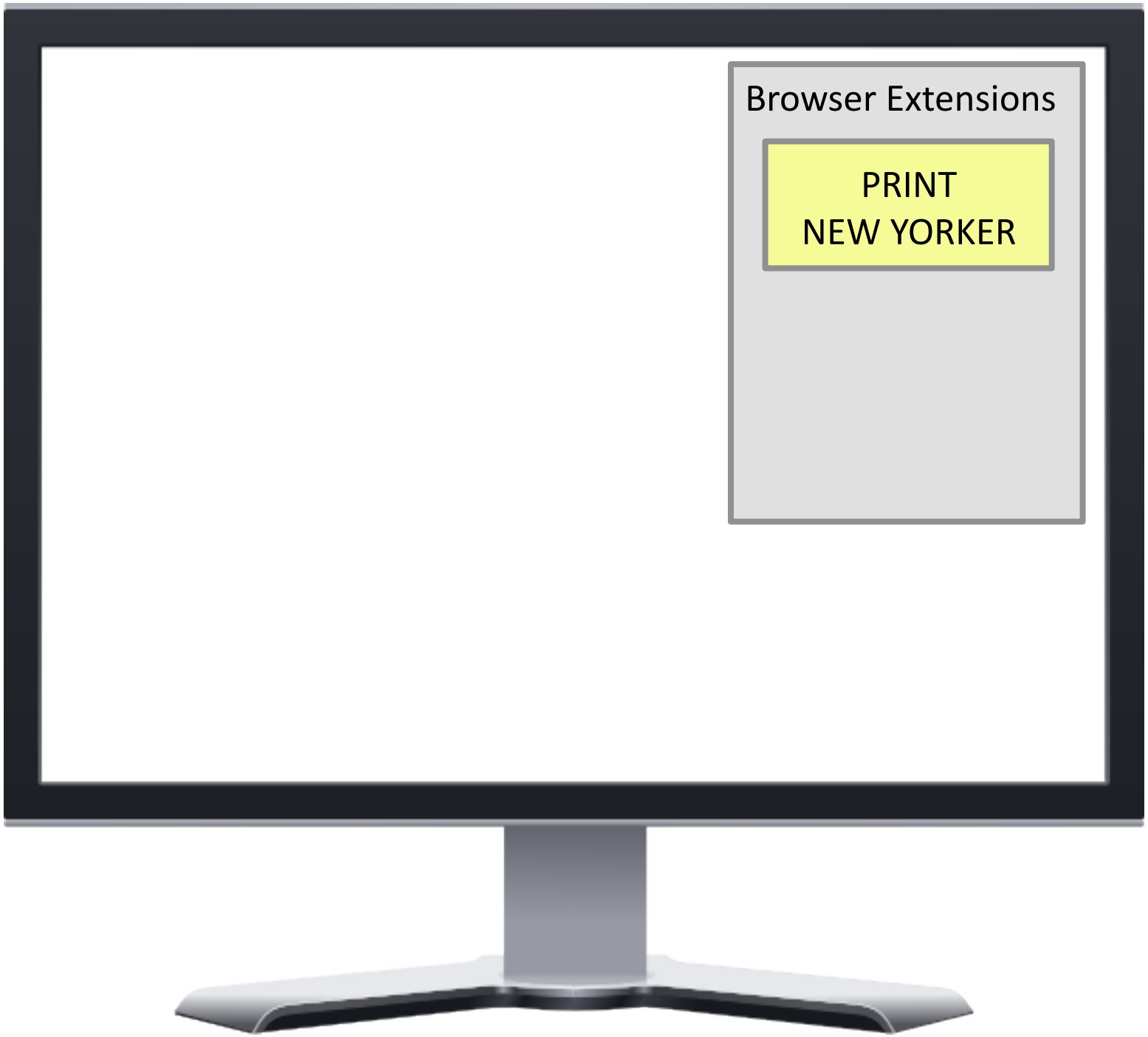

Dr. Gawande Answers Quest
well.blogs.nytimes.com/2008/07/04/dr-gawande-answers-questions-about-the-itch/
The New York Times Health | Science
Subscribe: Digital / Home Delivery | Log In | Register Now
Well
RECIPES FOR HEALTH
A Great Grain Adventure
FOOD
Your Twitter Tips for Going Vegan
Search Well
Go
FREE SHIPPING
ORDER NOW!
FAIRWAYGOLFUSA.COM
Advertise on NYTimes.com
JULY 4, 2008, 10:21 AM
Dr. Gawande Answers Questions About 'The Itch'
By TARA PARKER-POPE
WELL HOME PAGE »
PREVIOUS POST
For Beets, a Little More Respect, Please
NEXT POST
Cholesterol Drugs for Kids
ASKWell
Your health questions answered by Times journalists
All open questions »
What are good exercises for folks who spend much of their day sitting at a computer?
Answered by • 2 days ago
New treatments for motion sickness?
174
... New Yorker article, "tch," ...


Dr. Gawande Answers Quest
Scratching an itch through th
www.newyorker.com/reporting/2008/06/30/080630fa_fact_gawande
THE NEW YORKER
SUBSCRIBE and get a FREE 2013 CALENDAR!
SUBSCRIBE
RENEW
GIVE A GIFT
INTERNATIONAL ORDERS
ONLINE ARCHIVE
SUBSCRIBE
THIS WEEK'S ISSUE
NEWS
CULTURE
POLITICS
BOOKS & FICTION
CARTOONS
HUMOR
ARCHIVE
DOUBLE TAKE | PHOTO BOOTH | DAILY SHOUTS | PAGE-TURNER | DAILY COMMENT | AMY DAVIDSON | JOHN CASSIDY | BOROWITZ | RICHARD BRODY
THE NEW YORKER | REPORTING & ESSAYS
ANNALS OF MEDICINE
THE ITCH
Its mysterious power may be a clue to a new theory about brains and bodies.
BY ATUL GAWANDE
JUNE 30, 2008
Recommend 4.1k
Tweet 199
+1 67
It was still shocking to M. how much a few wrong turns could change your life. She had graduated from Boston College with a degree in psychology, married at twenty-five, and had two children, a son and a daughter. She and her family settled in a town on Massachusetts' southern shore. She worked for thirteen years in health care, becoming the director
WELCOME
SIGN IN | HELP | REGISTER
Search Web site
Find
MOST POPULAR
MOST E-MAILED
THIS ISSUE
1. Amy Davidson: Lance Armstrong's Flawed Confession
2. Andy Borowitz: Fox News to Shut Down for Routine Maintenance Monday Morning at 11:30
3. Maria Lokke: A Secret History of Women and Tattoo
4. Andy Borowitz: Republicans Accuse Obama of Using Position as President to Lead Country
5. Nathan Heller: Hello Laptop, My Old Friend
Advertisement
THE EVERYDAY COLLECTION


Dr. Gawande Answers Quest
Scratching an itch through th
www.newyorker.com/reporting/2008/06/30/080630fa_fact_gawande
Recommend 4.1k
Tweet 199
+1 67
It was still shocking to M. how much a few wrong turns could change your life. She had graduated from Boston College with a degree in psychology, married at twenty-five, and had two children, a son and a daughter. She and her family settled in a town on Massachusetts' southern shore. She worked for thirteen years in health care, becoming the director of a residence program for men who'd suffered severe head injuries. But she and her husband began fighting. There were betrayals. By the time she was thirty-two, her marriage had disintegrated. In the divorce, she lost possession of their home, and, amid her financial and psychological struggles, she saw that she was losing her children, too. Within a few years, she was drinking. She began dating someone, and they drank together. After a while, he brought some drugs home, and she tr
them. The drugs got harder. Eventual
they were doing heroin, which turne
to be readily available from a street
a block away from her apartment.
Scientists once saw itching as a form o
pain. They now believe it to be a different order of sensation. Photograph by Gerald Slota.
EMAIL SINGLE PAGE
AKLANDER,
LOGY;
LSIVE
PTION
3. Maria Lokke: A Secret History of Women and Tattoo
4. Andy Borowitz: Republicans Accuse Obama of Using Position as President to Lead Country
5. Nathan Heller: Hello Laptop, My Old Friend
Advertisement
THE EVERYDAY COLLECTION
EXPECT MORE. PAY LESS.®
discover coupons
THE NEW YORKER
NEW YEAR'S SPECIAL
BUY ONE GIFT SUBSCRIPTION - GET ONE FREE!
Advertisement
THE NEW YORKER
DIGITAL
TABLET, MOBILE, AND MORE

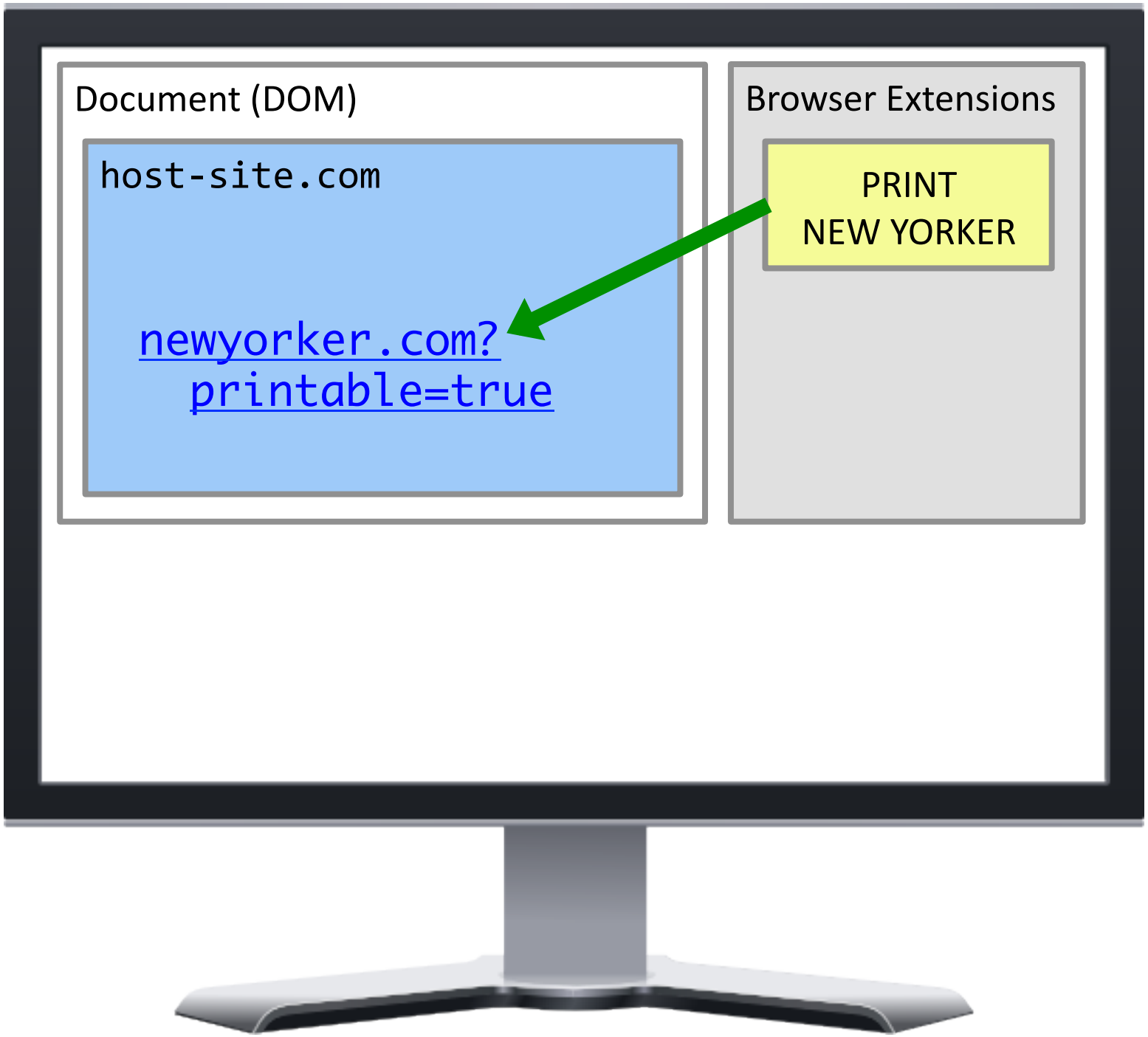

Dr. Gawande Answers Quest
Scratching an itch through t
www.newyorker.com/reporting/2008/06/30/080630fa_fact_gawa de?printable=true
"?printable=true"
THE NEW YORKER
ANNALS OF MEDICINE
THE ITCH
Its mysterious power may be a clue to a new theory about brains and
BY ATUL GAWANDE
JUNE 30, 2008
It was still shocking to M. how much a few wrong turns could change your life. She had graduated from Boston College with a degree in psychology, married at twenty-five, and had two children, a son and a daughter. She and her family settled in a town on Massachusetts' southern shore. She worked for thirteen years in health care, becoming the director of a residence program for men who'd suffered severe head injuries. But she and her husband began fighting. There were betrayals. By the time she was thirty-two, her marriage had disintegrated. In the divorce, she lost possession of their home, and, amid her financial and psychological struggles, she saw that she was losing her children, too. Within a few
"I want **every** article in print mode"

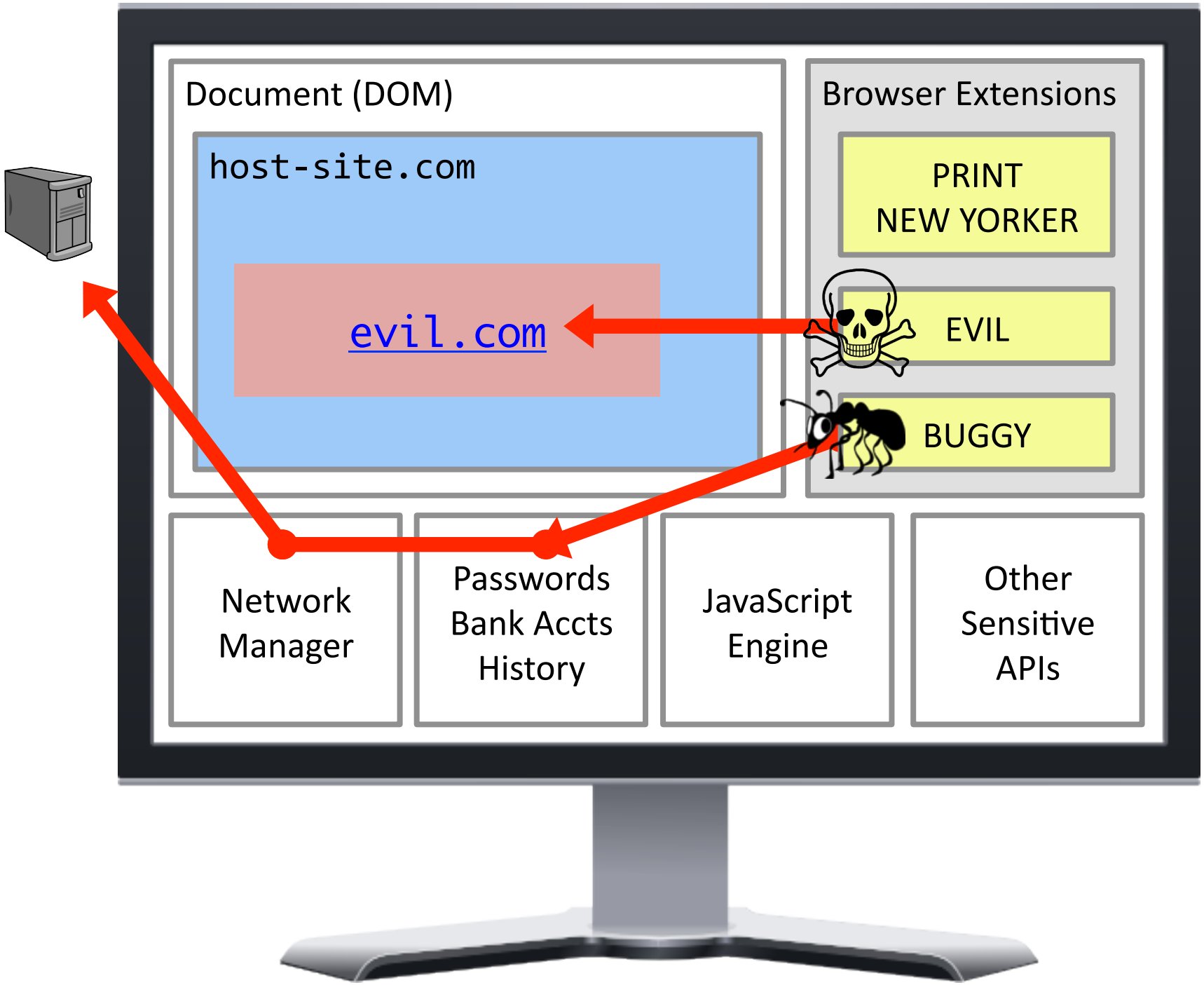

Browser Extensions
Third-Party
Developers
Customize
Browser

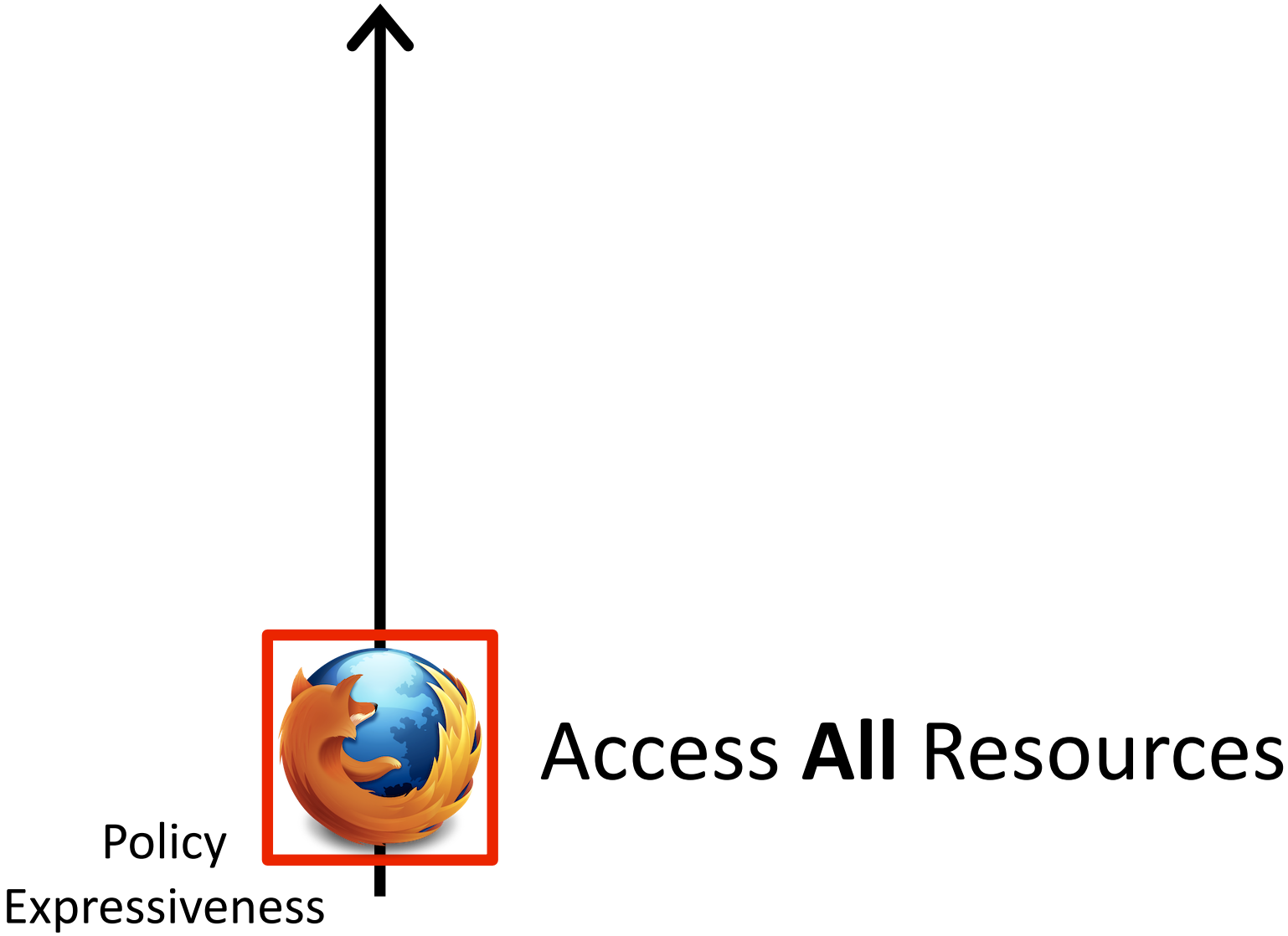

Browser Extensions
PRINT
NEW YORKER

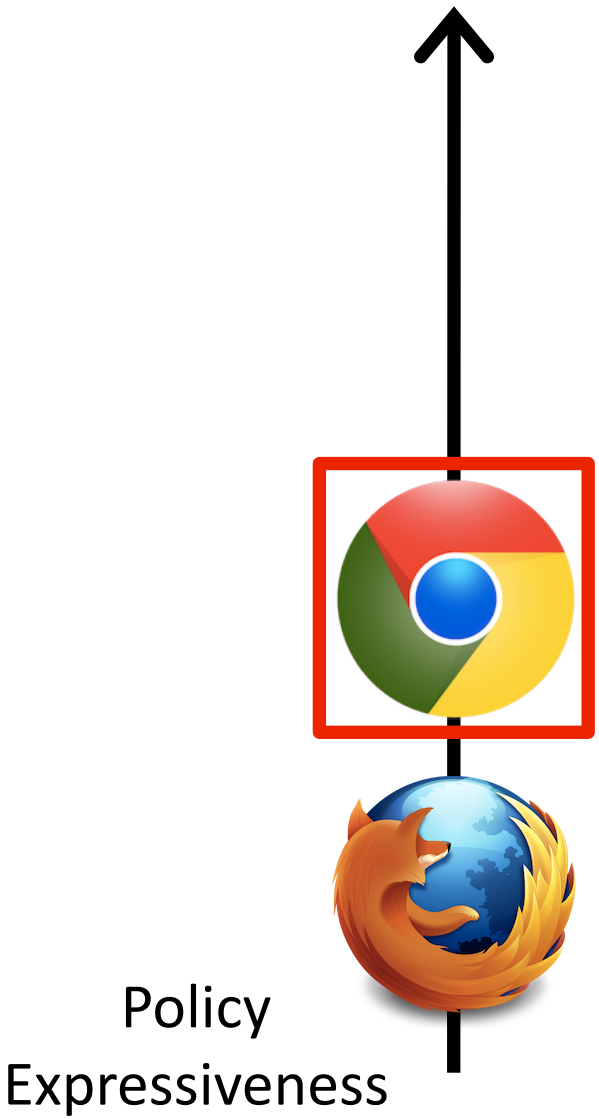
Browser Extensions

```
for (var elt in doc.getEltsByTagName("a")) {
  var url = elt.getAttr("href");
  if (url.match(/^newyorker.com/)) {
    elt.setAttr("href", url + "?printable=true")
} }
```


Document (DOM)
host-site.com
newyorker.com?
printable=true
Browser Extensions
PRINT
NEW YORKER

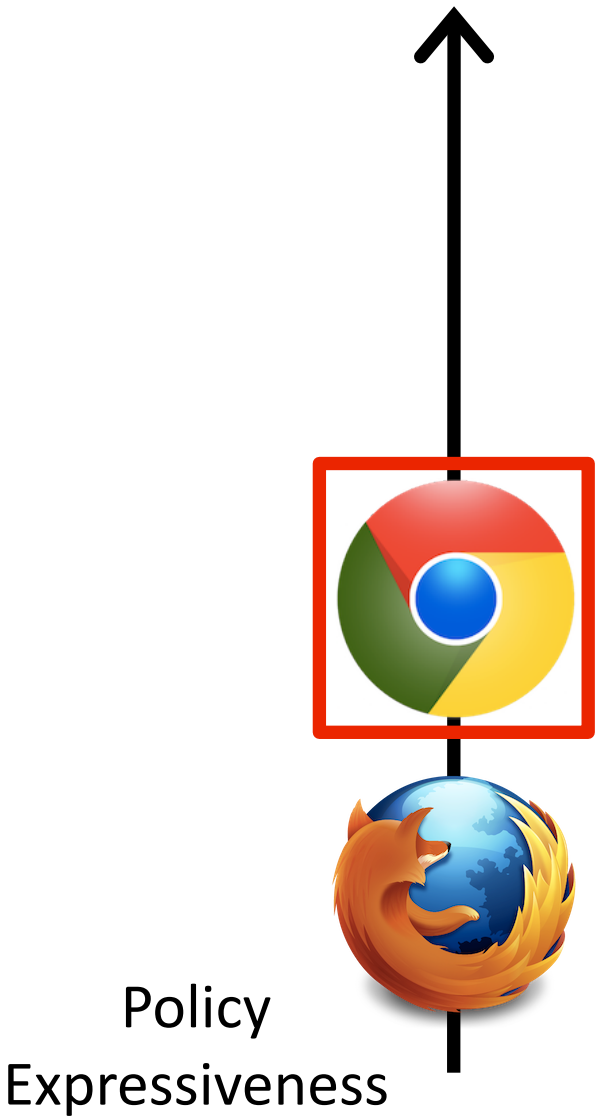

Document (DOM)
host-site.com
evil.com
Browser Extensions
PRINT
NEW YORKER
EVIL
BUGGY
Network
Manager
Passwords
Bank Accts
History
JavaScript
Engine
Other
Sensitive
APIs


Policy
Expressiveness
Access **All** Resources



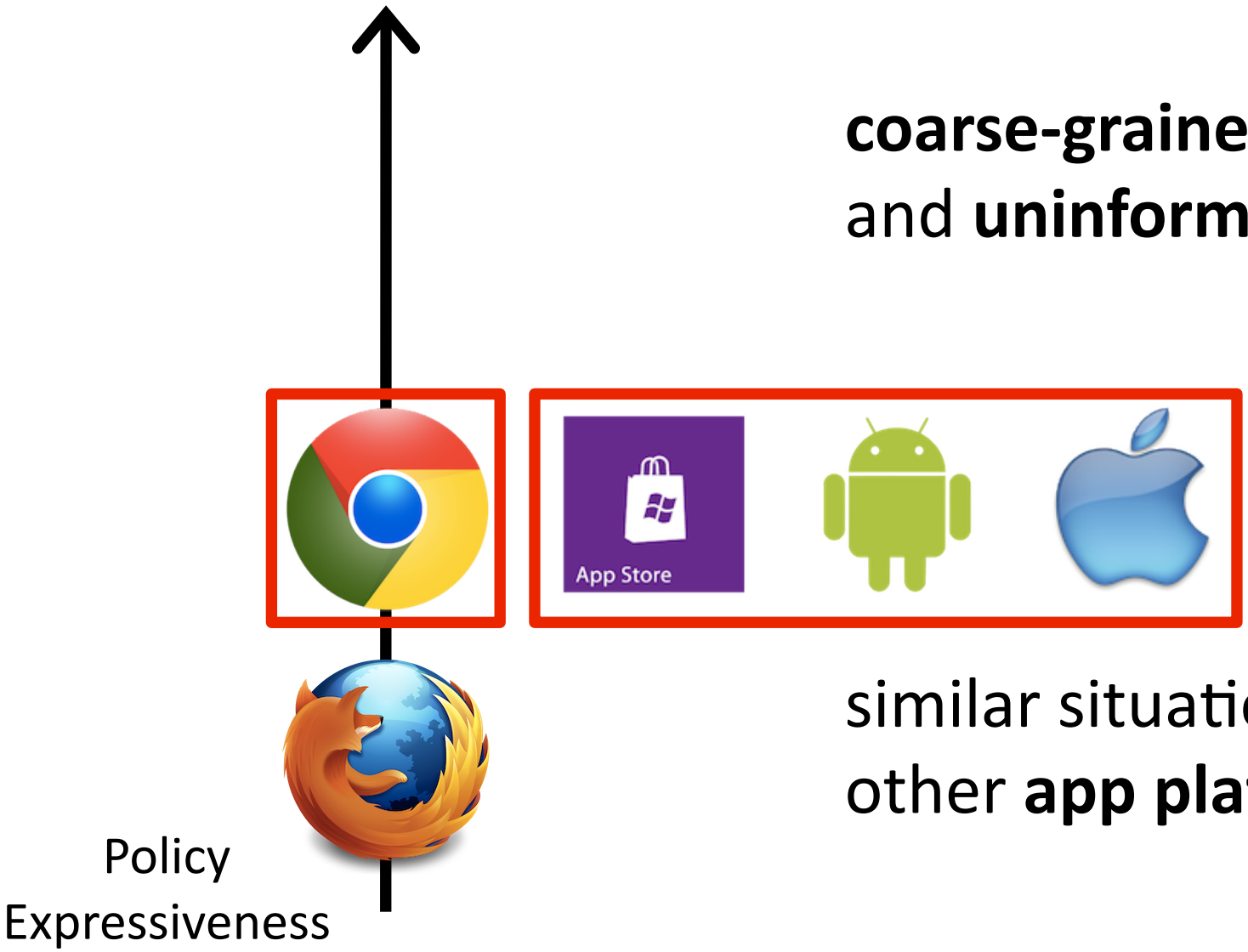
Access **Some** Resources

can access *DOM*
cannot access *History*



coarse-grained
and **uninformative**!

can access *DOM*
cannot access *History*



coarse-grained
and **uninformative**!

similar situation for
other **app platforms**

GOAL

Fine-Grained Policies

Policy Expressiveness

GOAL **Fine-Grained** Policies

can write "a" elements
but only "href" attribute

when original "href" value
starts with "newyorker.com"
and is **prefix of** new value

Language-Based Security

If Language Could Enforce Some Security-Related Policies …

Then **Reduce** Dependence On:

- Testing
- Code Reviews
- Dynamic Checking

For all programs p in PROGRAMS

If p has a type T in TYPES

Then p does not fail
with any error e in ERRORS

For all programs p in PROGRAMS

If p has a type T in TYPES

Then p does not fail
with any error e in ERRORS

GOAL: Eliminate Some Security-Related Errors

But How?

ERRORS

GOAL: Eliminate Some Security-Related Errors

Subtyping is **Implication**

Source of Expressive Power

$$\{ x \mid p \} \subseteq \{ x \mid q \}$$

$$p \Rightarrow q$$

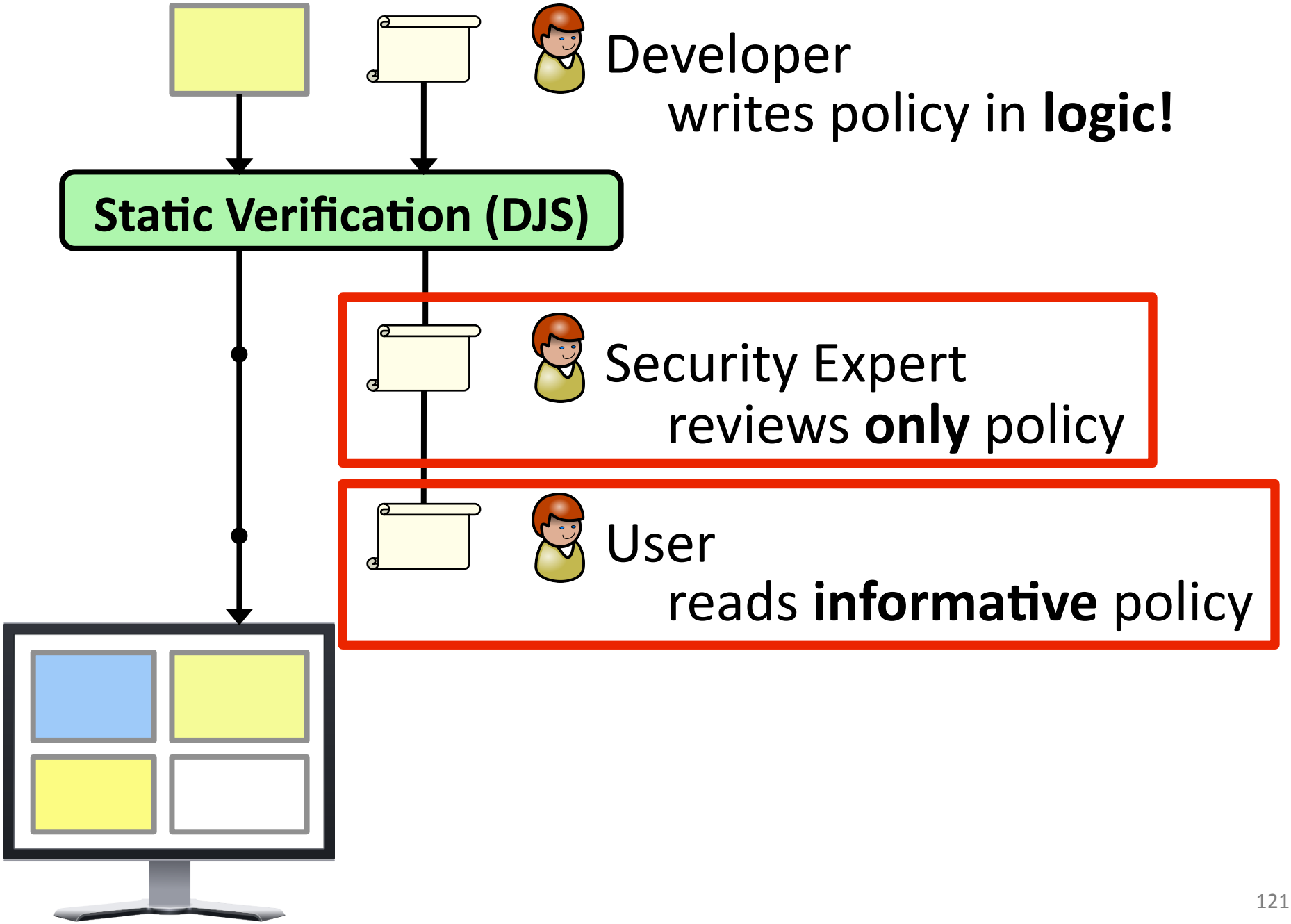
Outline

Security via Logic

ERRORS

Browser Extensions

Types for Fine-Grained Policies


Developer
writes policy in **logic!**
Static Verification (DJS)
Security Expert
reviews **only** policy
User
reads **informative** policy

Network Manager

`request`, …

JS Engine

`eval`, …

DOM

`getAttr`, `setAttr`, …

```
request ::
    url:Str
 →  Str
```

Network Manager

request, …

JS Engine

eval, …

DOM

getAttr, setAttr, …

```
request ::
    {url:Str|WhiteListed(url)}
 →  Str
```

Network Manager

`request`, …

JS Engine

`eval`, …

DOM

`getAttr`, `setAttr`, …

```
eval ::
        s:Str
  →     Void
```

Network Manager

`request, …`

JS Engine

`eval, …`

DOM

`getAttr, setAttr, …`

```
eval ::
    {s:Str|not(Tainted(s))}
  → Void
```

Network Manager
`request, …`

JS Engine
`eval, …`

DOM
`getAttr, setAttr, …`

```
getAttr ::
    elt:Element
 →  attr:Str
 →  Str
```

Network Manager
`request, …`

JS Engine
`eval, …`

DOM
`getAttr, setAttr, …`

```
getAttr ::
      elt:Element
  → {attr:Str|CanRead(elt,attr)}
  →   Str
```

Network Manager
`request, …`

JS Engine
`eval, …`

DOM
`getAttr, setAttr, …`

```
setAttr ::
      elt:Element
  →   attr:Str
  →   y:Str
  →   Void
```

Network Manager
`request, …`

JS Engine
`eval, …`

DOM
`getAttr, setAttr, …`

```
setAttr ::
      elt:Element
  →   attr:Str
  →   {y:Str|CanWrite(elt,attr,y)}
  →   Void
```

Network Manager
`request`, …

JS Engine
`eval`, …

DOM
`getAttr`, `setAttr`, …

Refined APIs Mediate Access

Typecheck Extension with **Policy**

```
forall url.
    not (url = "evil.com")
 ⇒ WhiteListed (url)
```

finite **blacklist**

```
var s = request("evil.com");
```

```
forall url.
    not (url = "evil.com")
 ⇒ WhiteListed (url)
```

```
//: url:Str
var s = request(url);
```

might be **blacklisted**!

```
forall url.
    not (url = "evil.com")
  ⇒ WhiteListed (url)
```

```
if (url != "evil.com") {
  var s = request(url);
}
```

```
forall url.
    not (url = "evil.com")
 ⇒ WhiteListed (url)
```

```
if (url != "evil.com") {
  var s = request(url);

  eval(s);
}
```

s:Str might be **tainted**!

```
forall url.
    not (url = "evil.com")
 ⇒ WhiteListed (url)
```

```
if (url != "evil.com") {
  var s = request(url);
  s = sanitize(s);
  eval(s);
}
```

s:Str

{s:Str|not(Tainted(s))}

STATICALLY VERIFIED!

```
WhiteListed ("ny.com")
WhiteListed ("golf.com")
forall url.
    WhiteListed (url)
 ⇒  not (Tainted (request (url)))
```

trust that
whitelisted servers
return **untainted** strings

```
WhiteListed ("ny.com")
WhiteListed ("golf.com")
forall url.
    WhiteListed (url)
 ⇒ not (Tainted (request (url)))
```

```
var s = request("ny.com");
eval(s);
```

STATICALLY VERIFIED!

no need to sanitize

```
forall e, oldUrl, newUrl.
    ValueOf (e, "href", oldUrl) ∧
    StrPrefix ("newyorker.com", oldUrl) ∧
    StrPrefix (oldUrl, newUrl)
      ⇒ CanWrite (e, "href", newUrl)
```

```
for (var elt in doc.getEltsByTagName("a")) {
  var url = elt.getAttr("href");
  if (url.match(/^newyorker.com/)) {
    elt.setAttr("href", url + "?printable=true");
} }
```

```
ValueOf (elt, "href", url) ∧
StrPrefix ("newyorker.com", url) ∧
StrPrefix (url, url + "?printable=true")
  ⇒ CanWrite (e, "href", url+"?printable=true")
```

```
for (var elt in doc.getEltsByTagName("a")) {
  var url = elt.getAttr("href")
  if (url.match(/^newyorker.com/)) {
    elt.setAttr("href", url + "?printable=true");
} }
```

```
ValueOf (elt, "href", url) ∧
StrPrefix ("newyorker.com", url) ∧
StrPrefix (url, url + "?printable=true")
  ⇒ CanWrite (e, "href", url+"?printable=true")
```

```
for (var elt in doc.getEltsByTagName("a")) {
  var url = elt.getAttr("href");
  if (url.match(/^newyorker.com/)) {
    elt.setAttr("href", url + "?printable=true");
} }
```

```
ValueOf (elt, "href", url) ∧
StrPrefix ("newyorker.com", url) ∧
StrPrefix (url, url + "?printable=true")
    ⇒ CanWrite (e, "href", url+"?printable=true")
```

```
for (var elt in doc.getEltsByTagName("a")) {
  var url = elt.getAttr("href");
  if (url.match(/^newyorker.com/)) {
    elt.setAttr("href", url + "?printable=true");
} }
```

```
ValueOf (elt, "href", url) ∧
StrPrefix ("newyorker.com", url) ∧
StrPrefix (url, url + "?printable=true")
  ⇒ CanWrite (e, "href", url+"?printable=true")
```

```
for (var elt in doc.getEltsByTagName("a")) {
  var url = elt.getAttr("href");
  if (url.match(/^newyorker.com/)) {
    elt.setAttr("href", url + "?printable=true")
} }
```

STATICALLY VERIFIED!

```
ValueOf (elt, "href", url) ∧
StrPrefix ("newyorker.com", url) ∧
StrPrefix (url, url + "?printable=true")
  ⇒ CanWrite (e, "href", url+"?printable=true")
```

```
for (var elt in doc.getEltsByTagName("a")) {
  var url = elt.getAttr("href");
  if (url.match(/^newyorker.com/)) {
    elt.setAttr("href", "evil.com");
} }
```

```
ValueOf (elt, "href", url) ∧
StrPrefix ("newyorker.com", url) ∧
StrPrefix (url, "evil.com")
  ⇒ CanWrite (e, "href", "evil.com")
```

```
for (var elt in doc.getEltsByTagName("a")) {
  var url = elt.getAttr("href");
  if (url.match(/^newyorker.com/)) {
    elt.setAttr("href", "evil.com");
} }
```

```
ValueOf (elt, "href", url) ∧
StrPrefix ("newyorker.com", url) ∧
StrPrefix (url, "evil.com")
  ⇒ CanWrite (e, "href", "evil.com")
```

```
for (var elt in doc.getEltsByTagName("a")) {
  var url = elt.getAttr("href");
  if (url.match(/^newyorker.com/)) {
    elt.setAttr("href", "evil.com")
} }
```

Preliminary Benchmarks

Secure Extensions Written in OCaml

Types for Security [ESOP '10, Guha et al. '11]

\+ JavaScript Verification [POPL '12, OOPSLA '12]

= **Fine-Grained Web Security** [current]

Preliminary Benchmarks

Secure Extensions Written in OCaml

Ported to DJS (~400 LOC so far)

Well-typed programs
don't have **run-time** errors
and **conform to security policies**

Guha et al. '11

Outline

Security via Logic

ERRORS

Browser Extensions

Types for Fine-Grained Policies

Outline

Why is JavaScript Important?

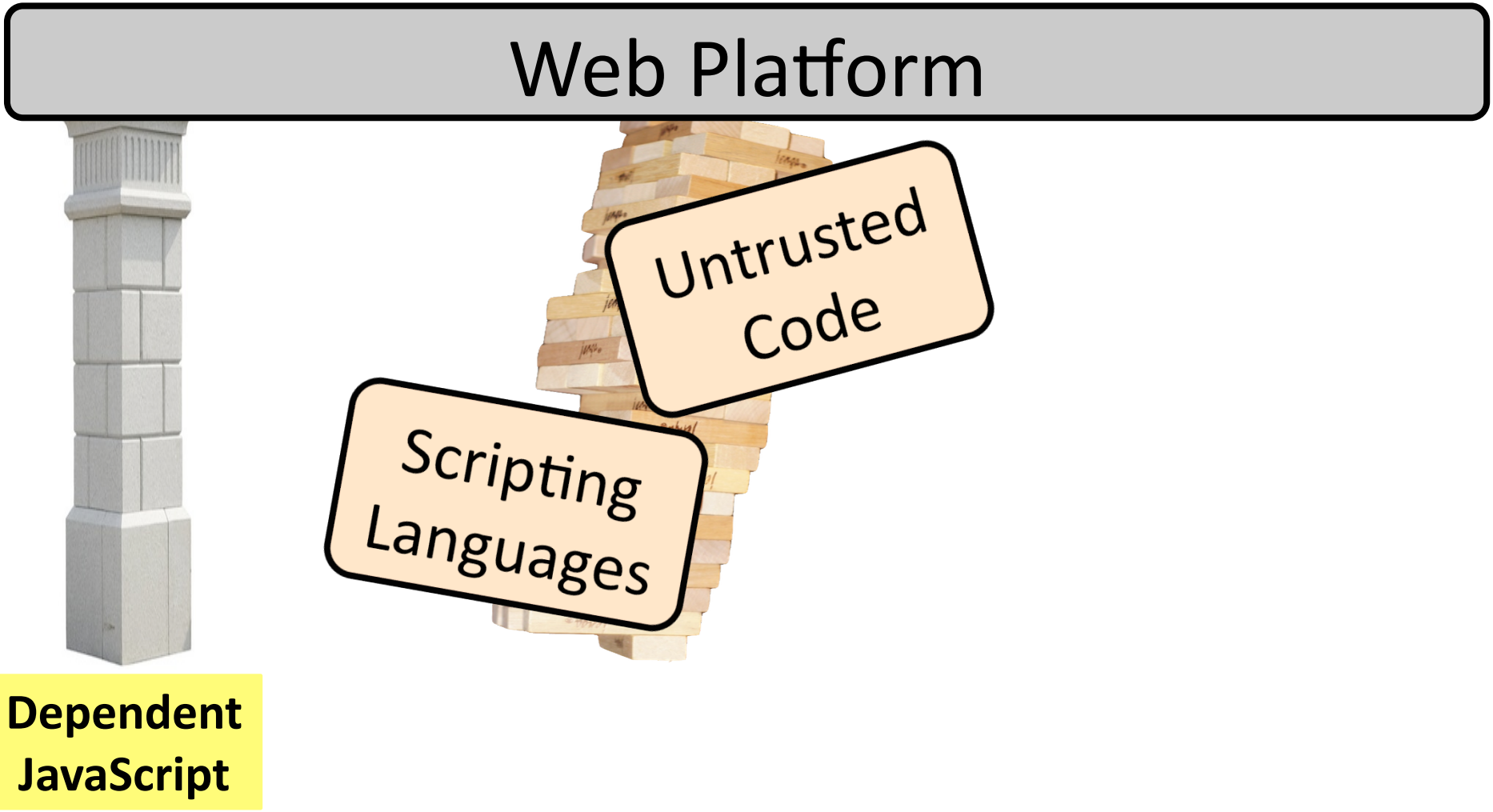
Why is JavaScript Hard? PROGRAMS

Static Types via Logic TYPES

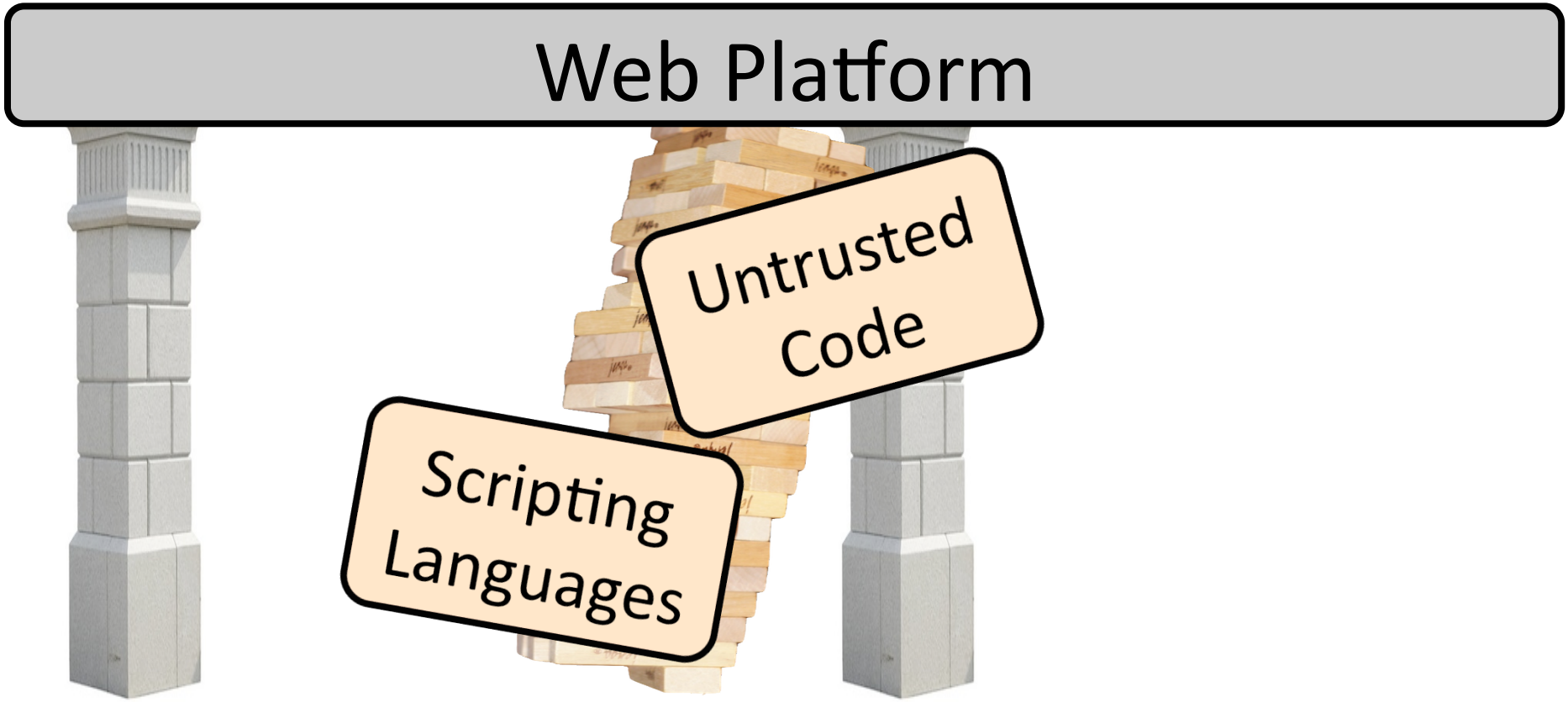
Security via Logic ERRORS

Looking Ahead

>>>

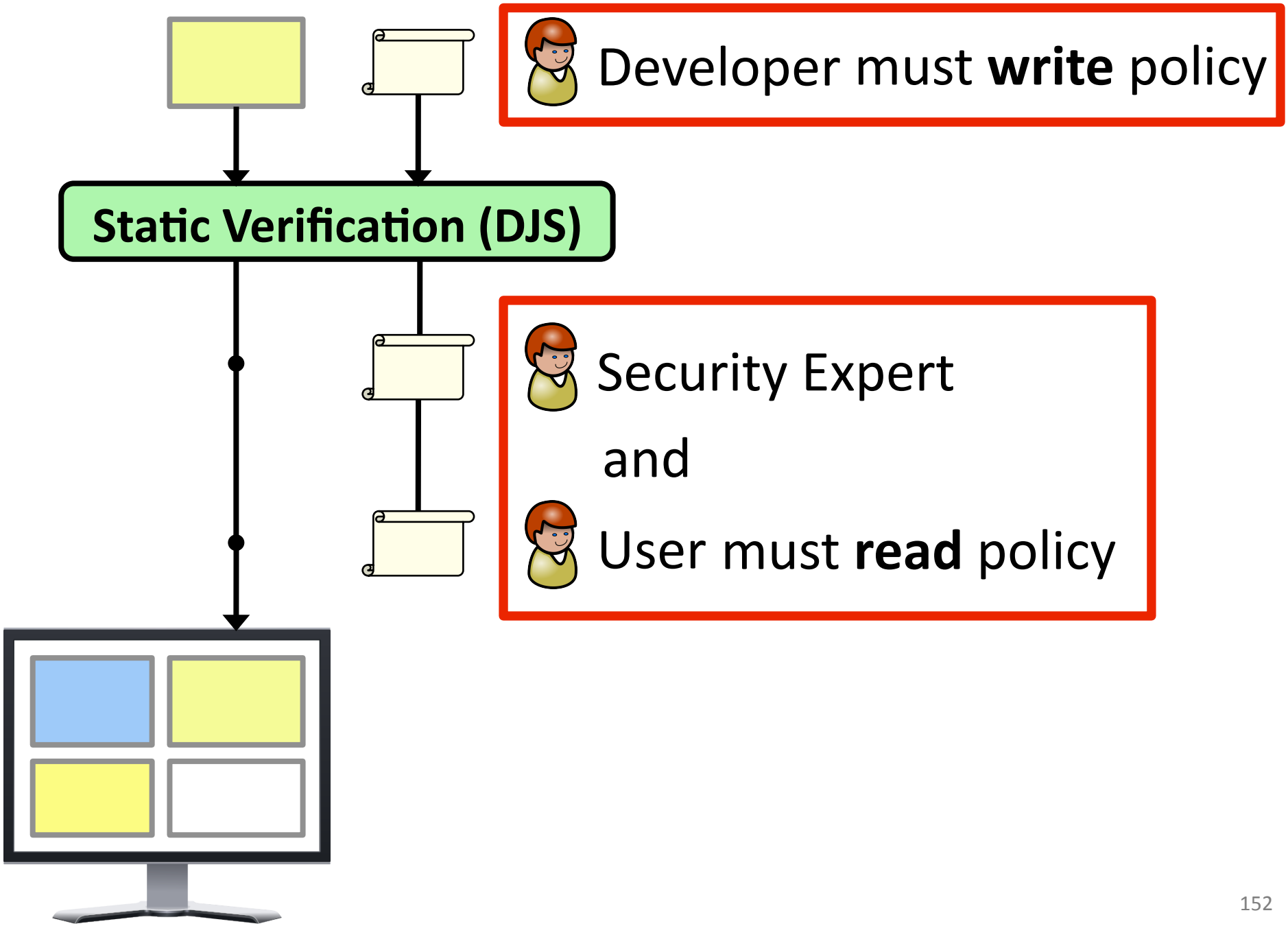

Web Platform
Untrusted Code
Scripting Languages
Dependent JavaScript

>>>



Dependent JavaScript

Policy Specification


Developer must **write** policy
Static Verification (DJS)
Security Expert
and
User must **read** policy

Developer must **write** policy

- Infer from traces
- Incentives from platform

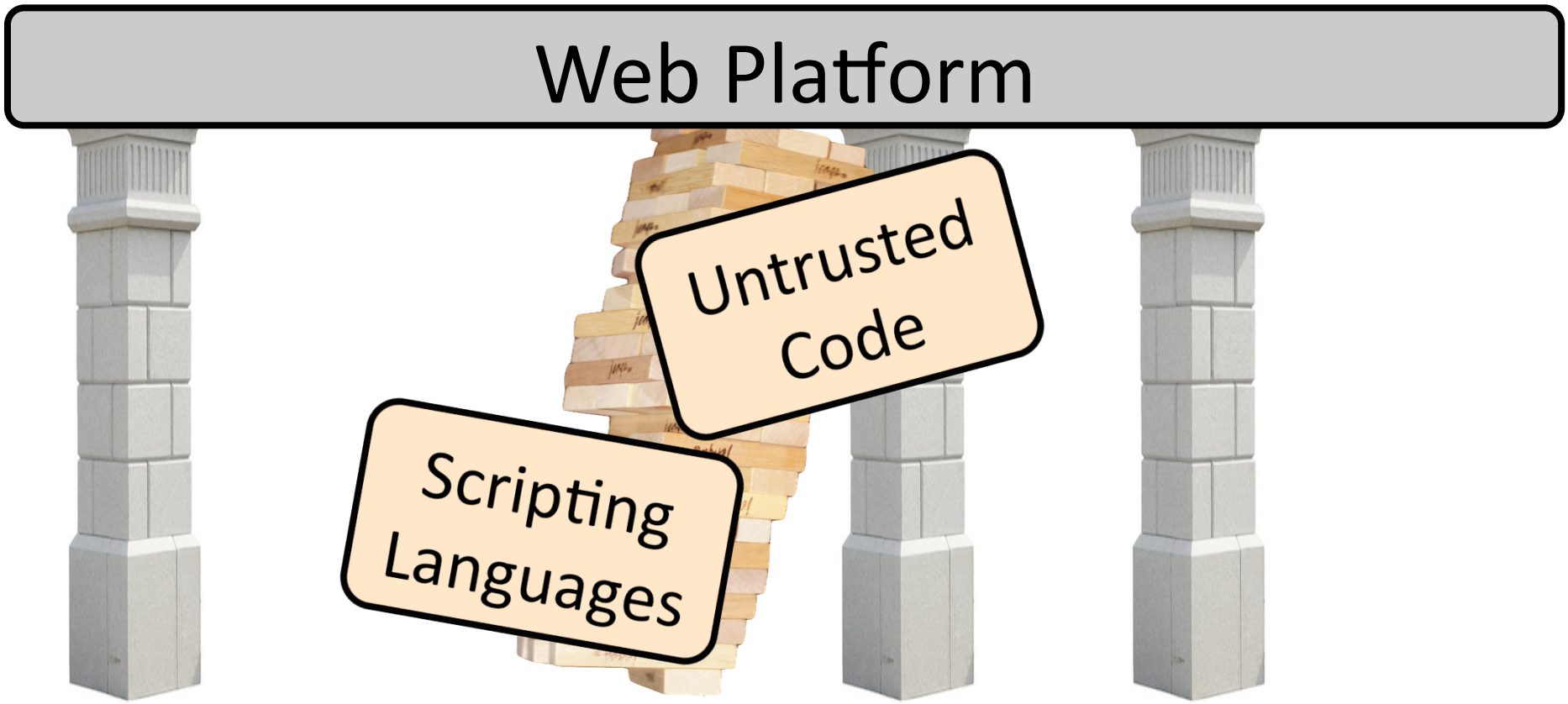
Security Expert must **read** policy

- Modeling tools to analyze policies

User must **read** policy

- Traces to demonstrate behavior
- Community review-and-recommend services

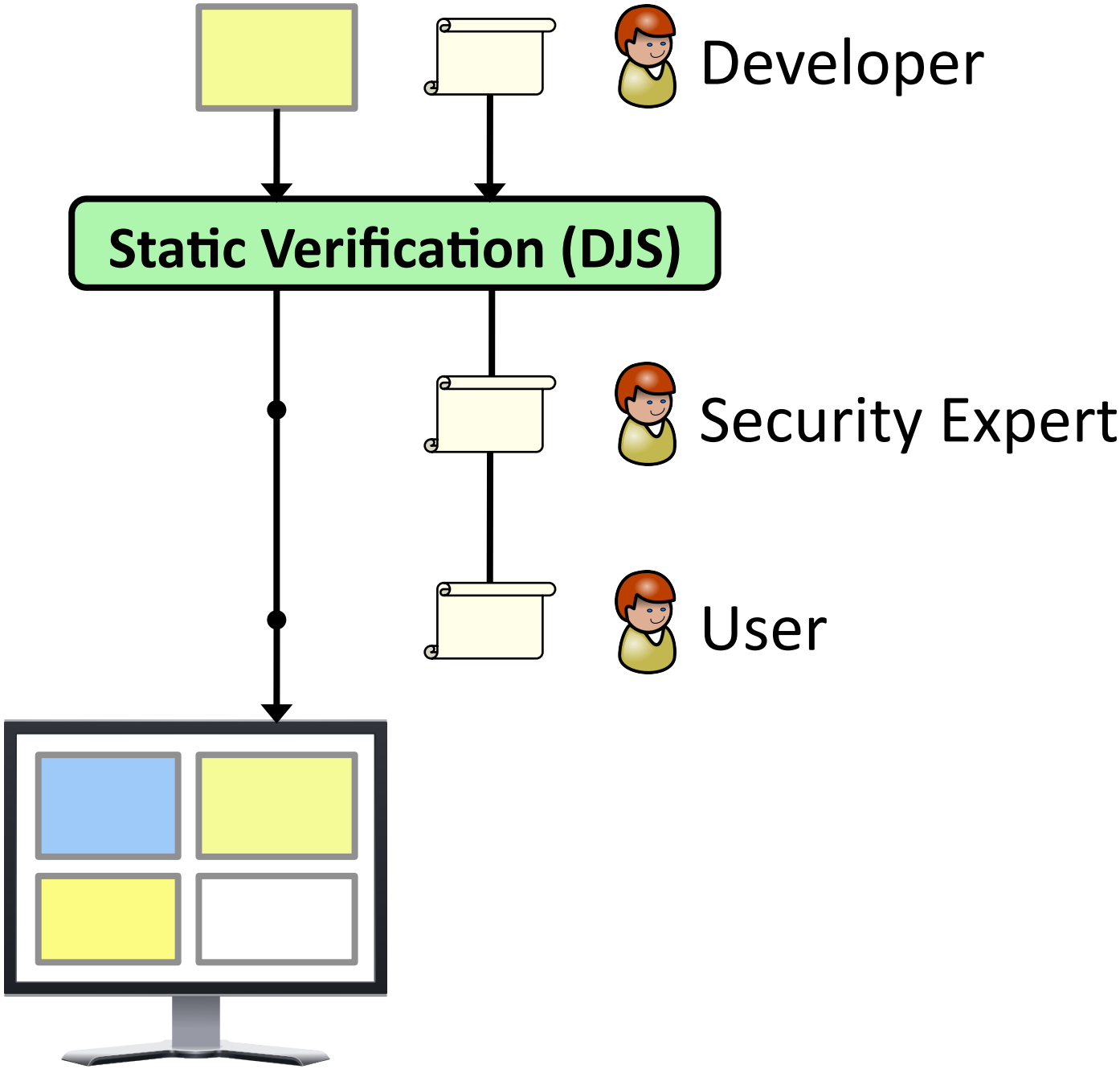
>>>



Dependent JavaScript

Policy Specification

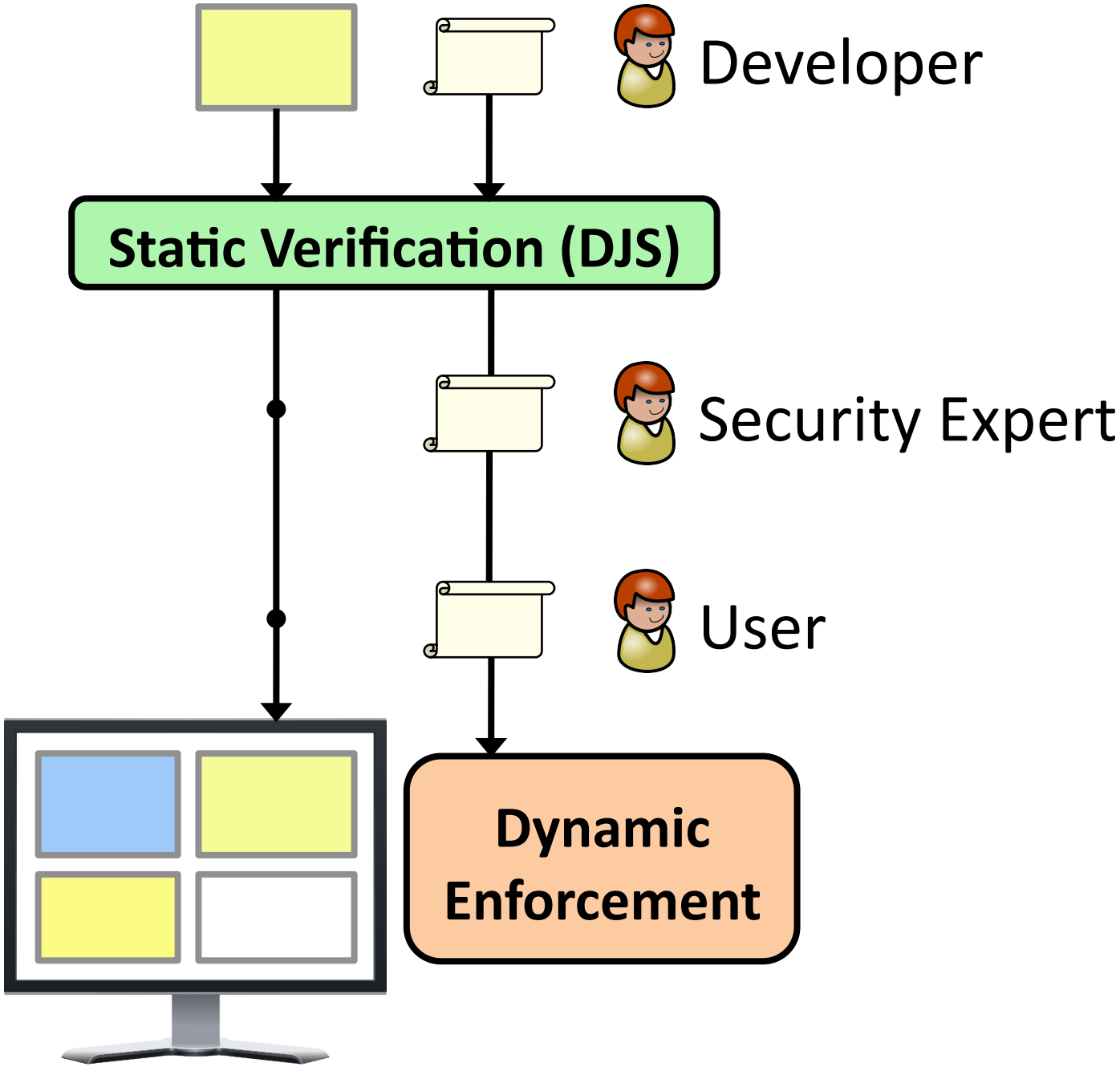
Dynamic Enforcement

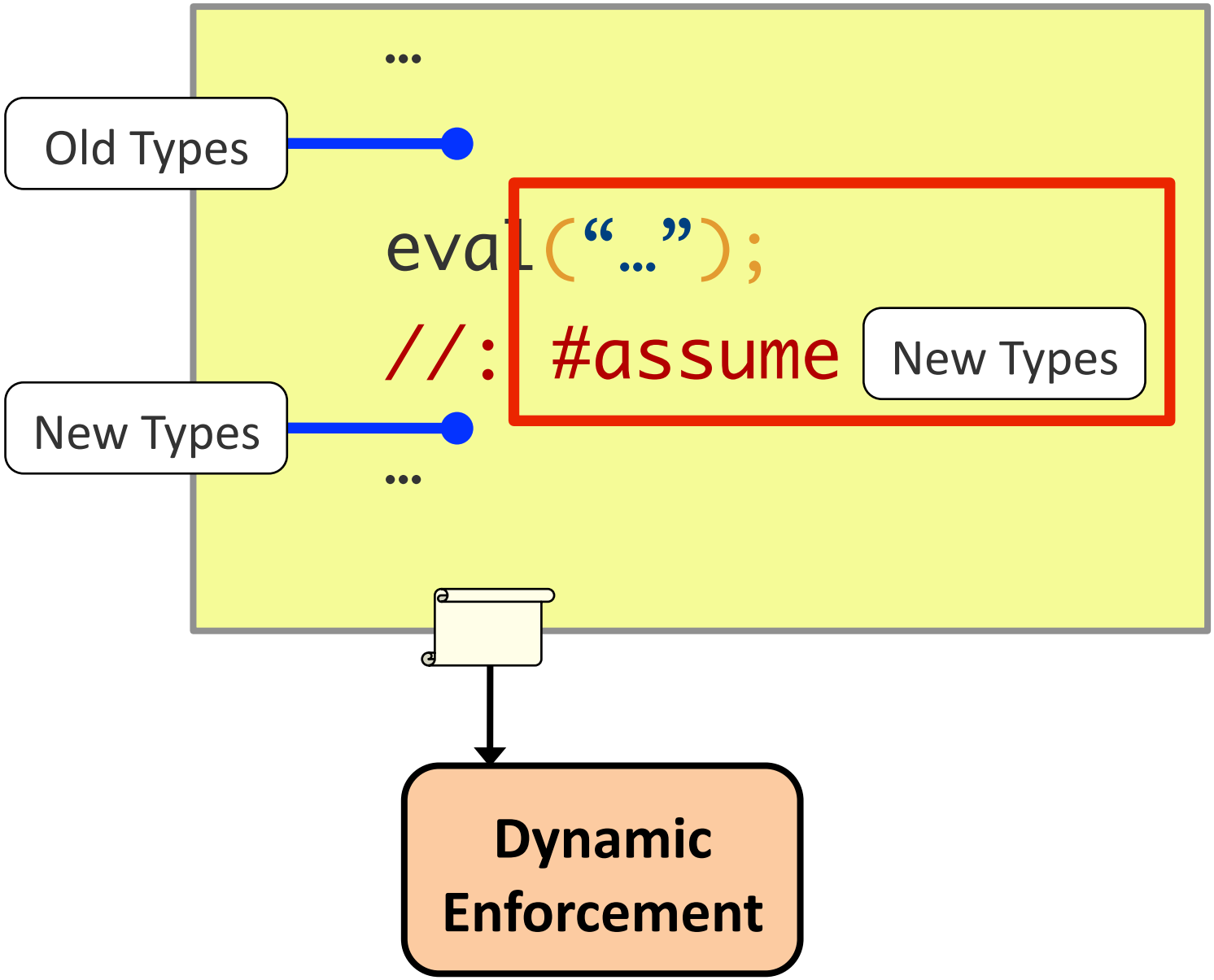

Developer
Static Verification (DJS)
Security Expert
User

```
...

eval("...");

...
```

Dynamic
Code Loading

Impossible to **statically** analyze code you don't have!


Developer
Static Verification (DJS)
Security Expert
User
Dynamic Enforcement


...
Old Types
eval("...");
//: #assume
New Types
New Types
...
Dynamic Enforcement

Limits of Static Reasoning

"actual" "expected"

$p \Rightarrow q$ ✓ **Definitely Safe**

$p \Rightarrow q$ ✗ **Maybe Unsafe**
(But Maybe Safe…)

Limits of Static Reasoning

"actual" "expected"

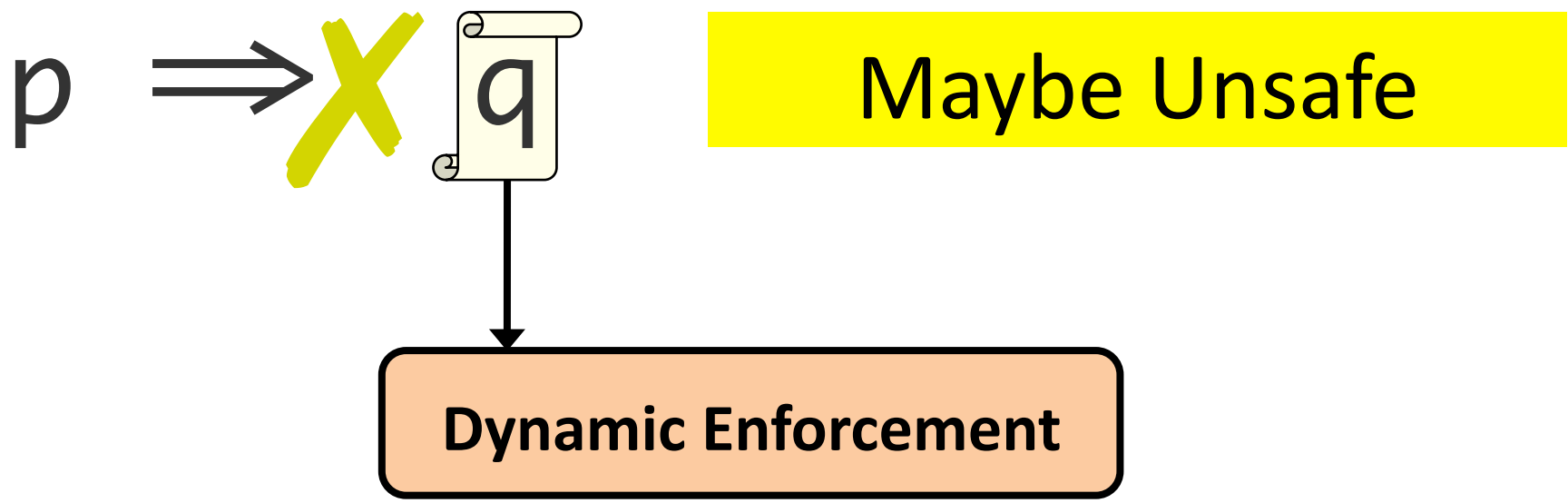
$p \Rightarrow q$ ✓ — Definitely Safe

$p \Rightarrow q$ ✗ — Maybe Unsafe

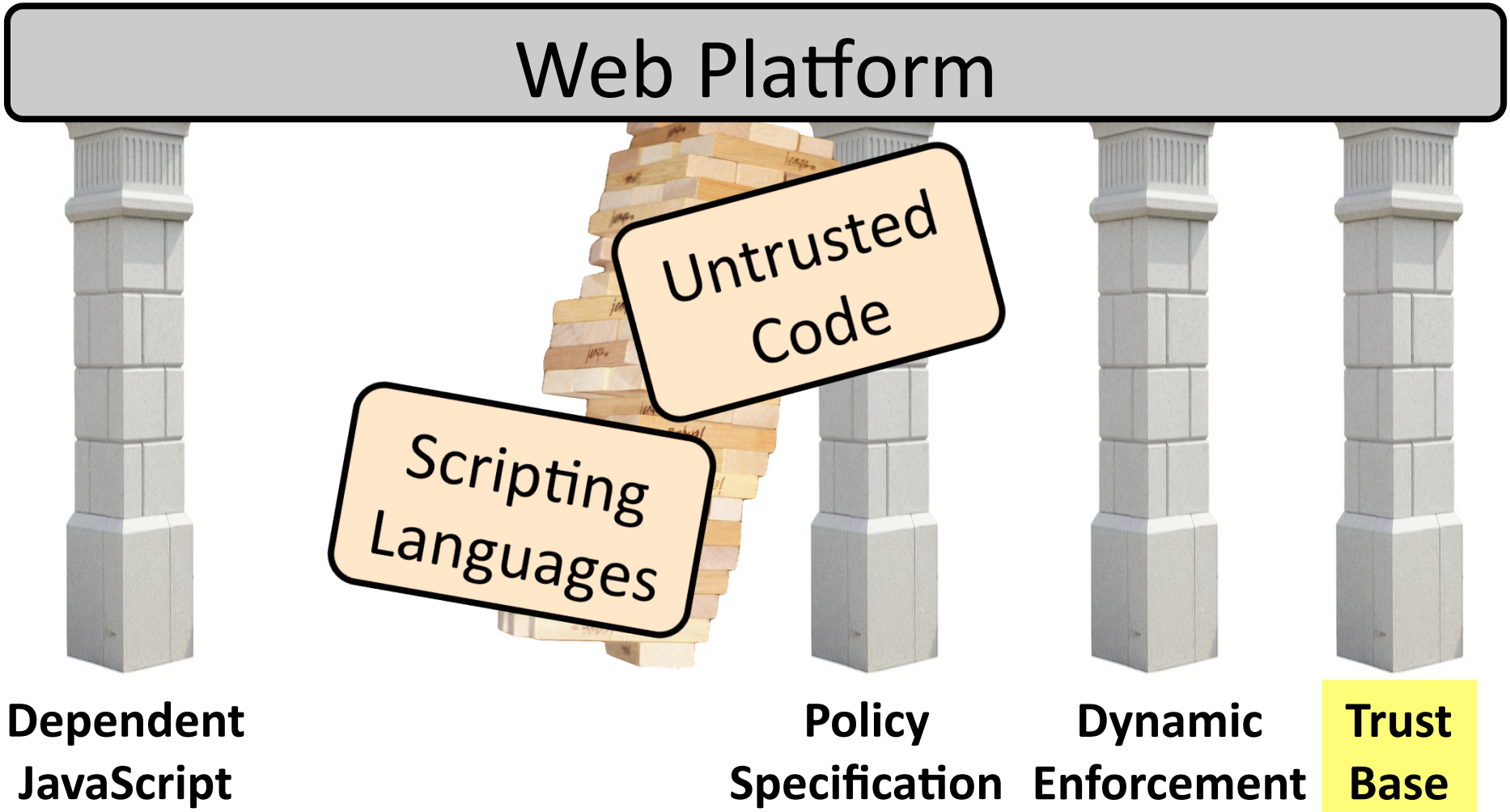
$p \Rightarrow \neg q$ ✓ — Definitely Unsafe

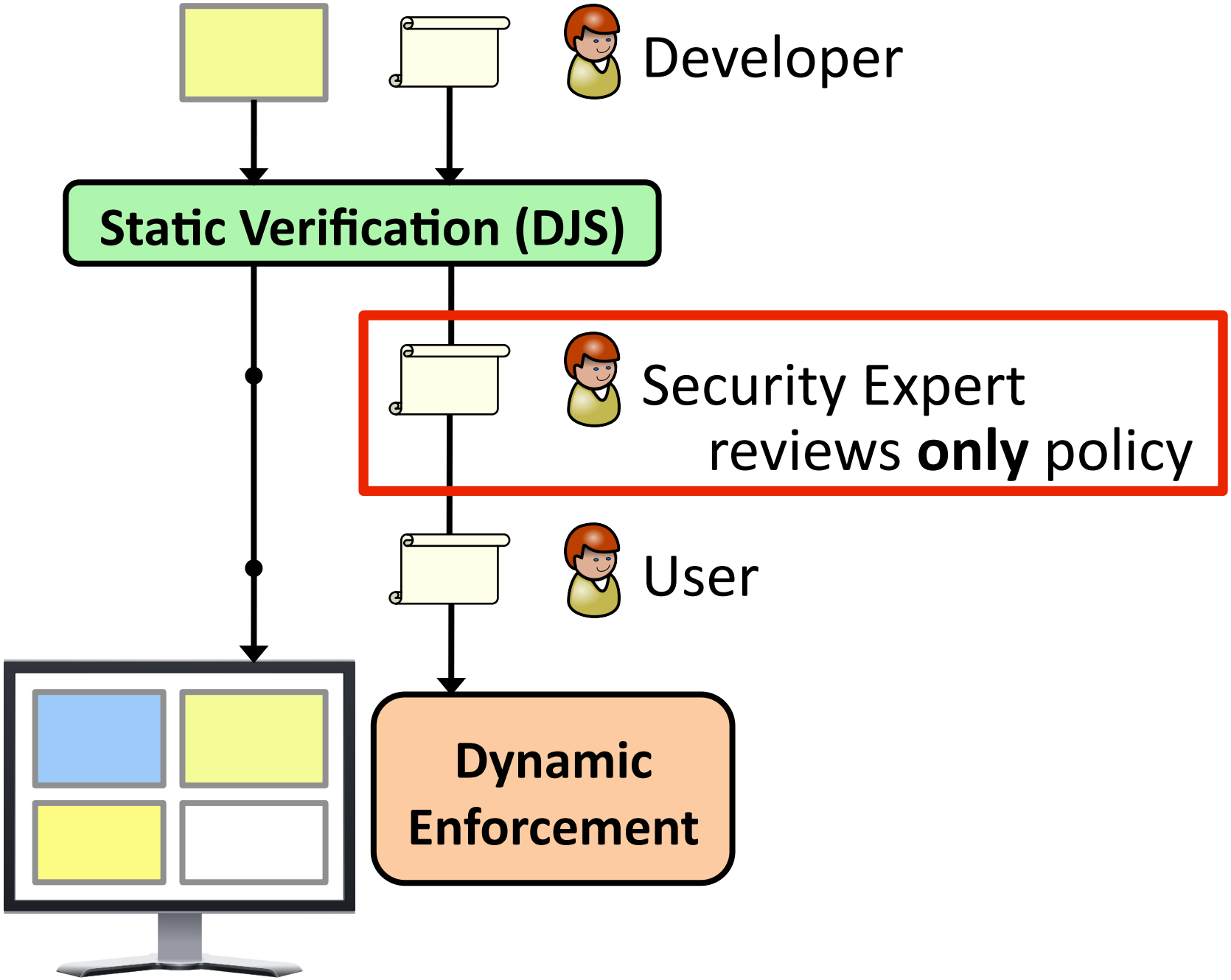
Limits of Static Reasoning

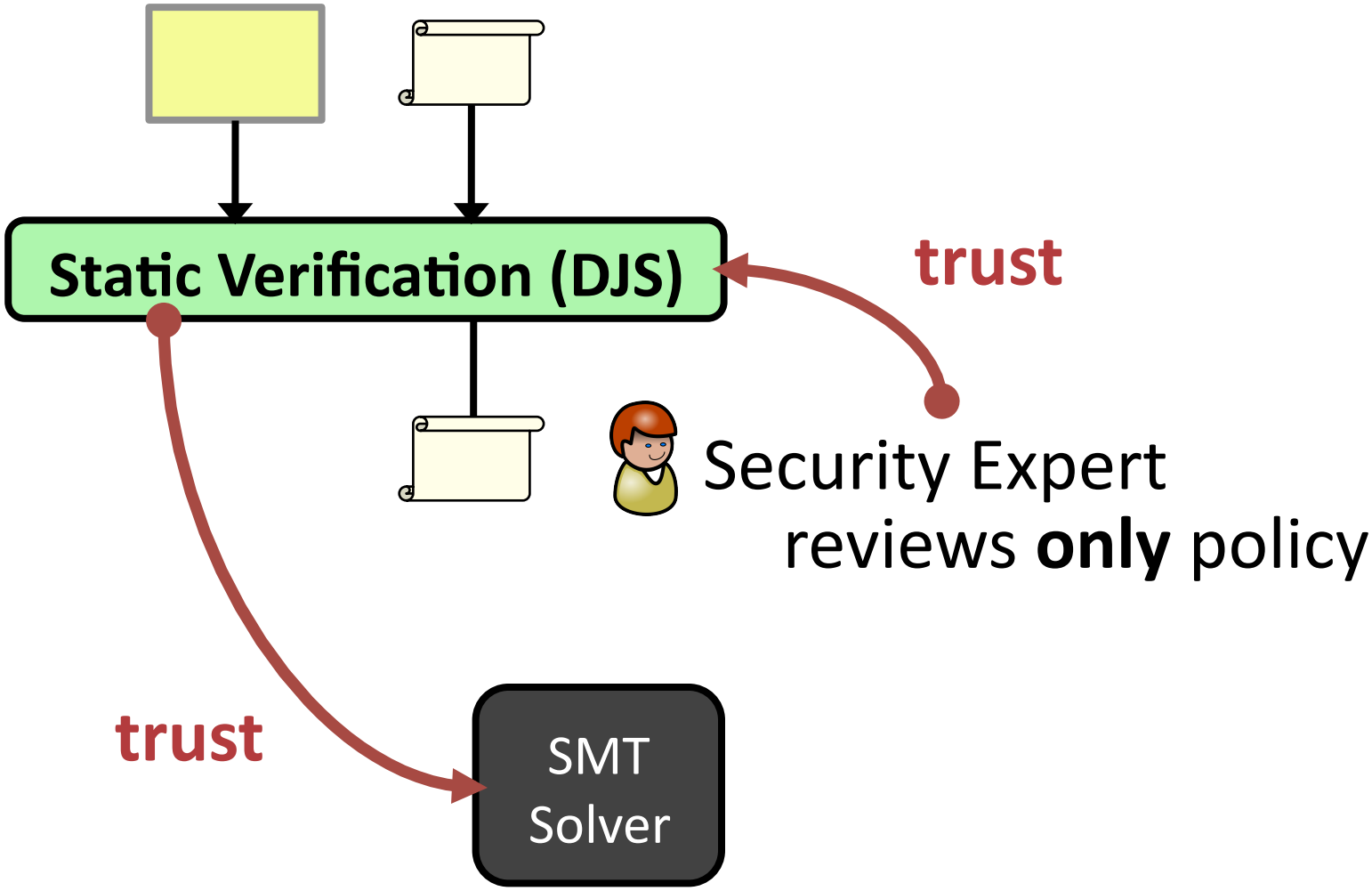
- Accept Program, Rather than Reject
- Rely on **Programmer-Trusted** or **Dynamically-Checked** Assumption



>>>


Web Platform
Untrusted Code
Scripting Languages
Dependent JavaScript
Policy Specification
Dynamic Enforcement
Trust Base


Developer
Static Verification (DJS)
Security Expert
reviews **only** policy
User
Dynamic
Enforcement


Static Verification (DJS)
trust
Security Expert
reviews **only** policy
trust
SMT
Solver

Eliminate from TCB via Formal Verification

Static Verification (DJS)

trust

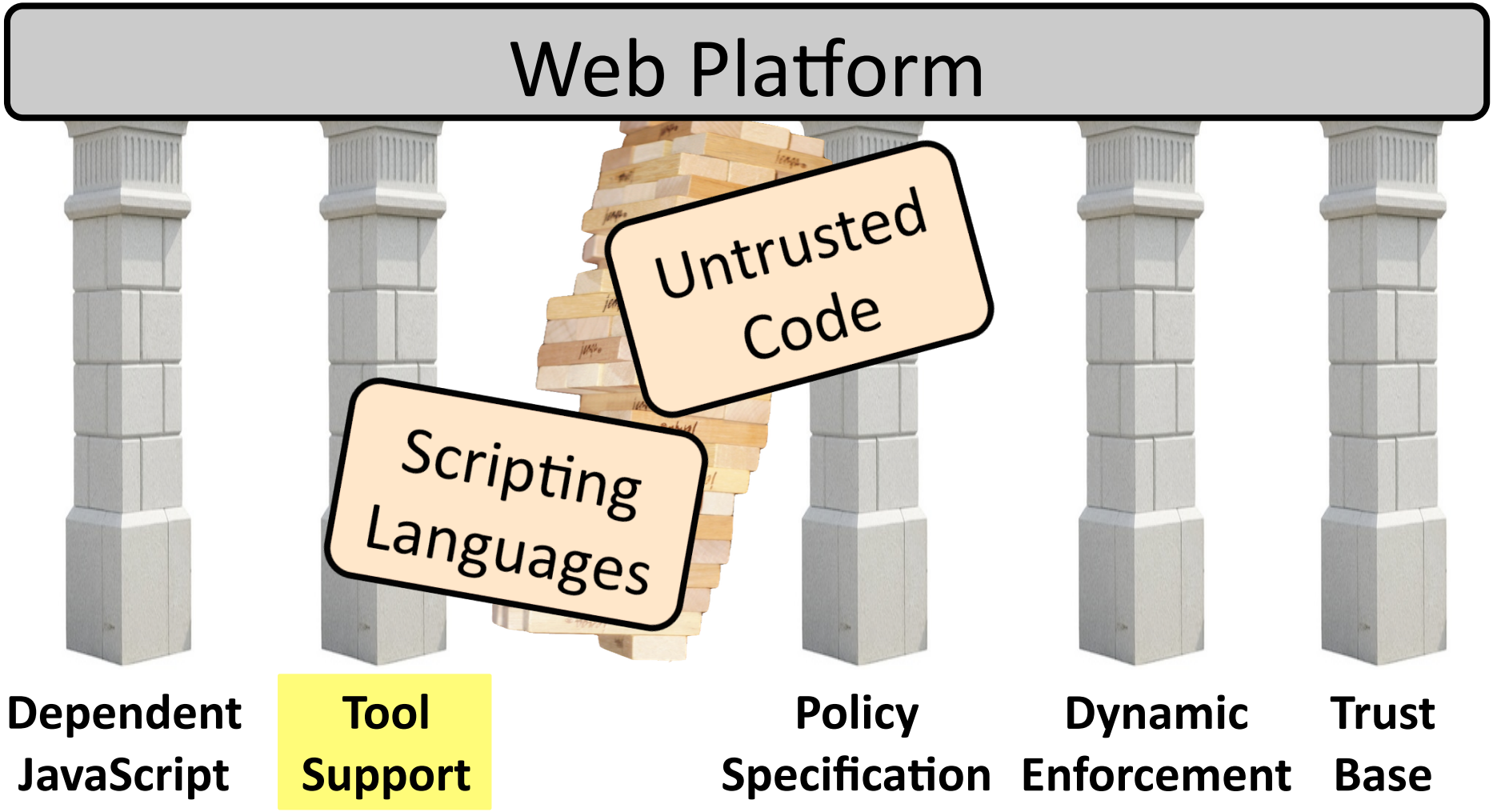
Security Expert

trust

SMT Solver

Eliminate from TCB à la Proof-Carrying Code [PLDI '10]

>>>


Web Platform
Untrusted Code
Scripting Languages
Dependent JavaScript
Tool Support
Policy Specification
Dynamic Enforcement
Trust Base

Tools for Web Development

- Generate Tests and Find Bugs for **Unannotated Programs**
- Support Refactoring and Code Completion for **Annotated Programs**
- Provide Analysis and Documentation Tools for Large, **Popular Libraries** like jQuery
- Web-Based Interfaces to **Collect Statistics** about Programming Patterns, Errors

Chrome File Edit View History Bookmarks Window Help

New Tab | Source Code Annotations:

file://localhost/Users/ravi/impscript/tests/objects-2/test-02.is.html

```
1  //: extern val _plus : (num, num) -> (num) ;
2
3  //:                                  ?(ref(`x))} ;
4
5  //:                                  t) -> (any /
6  var f = function (xs) {
7    //: xs = (all L_. ?(ref(L_)) => ref(L_)) (xs) ;
8    //: unfold (numlist, xs) ;
9    _plus (1 , xs["hd"]) ;
10   _plus (2 , xs["hd"]) ;
11
12   //: xs["tl"] = (all L_. ?(ref(L_)) => ref(L_)) (xs
13   //: unfold (numlist, xs["tl"]) ;
14   _plus (3 , xs["tl"]["hd"]) ;
15   _plus (4 , xs["tl"]["hd"]) ;
16
17   //: xs["tl"]["tl"] = (all L_.                    tl"]) ;
18   //: unfold (numlist, xs["tl"]["
19   _plus (5 , xs["tl"]["tl"]["hd"
20
21   //: fold (numlist, xs["tl"]["tl"]) ;
22   //: fold (numlist, xs["tl"]) ;
23   //: fold (numlist, xs) ;
24   return xs ;
25 } ;
```

Inferred Annotations are in Comments

Flow-Sensitive Type Information by Hovering over Program Point

```
undefined

*xs        : ref(L)
*L         : {hd: num, tl: ref(L1_unpack)}
*L1_unpack : {hd: num, tl: ?(ref(L2_unpack))}
*L2_unpack : numlist
```

Chrome File Edit View History Bookmarks Window Help

New Tab | Source Code Annotations:

file://localhost/Users/ravi/impscript/tests/objects-2/test-02.is.html

```
//: extern val _plus : (num, num) -> (num) ;

//:                                  ?(ref(`x))} ;

//:                                  t) -> (any /

var f = function (xs) {
  //: xs = (all L_. ?(ref(L_)) => ref(L_)) (xs) ;
  //: unfold (numlist, xs) ;
  _plus (1 , xs["hd"]) ;
  _plus (2 , xs["hd"]) ;

                                     (xs

                                                  l"]) ;
                                     m, tl: ref(L1_unpack)}
                                     m, tl: ?(ref(L2_unpack))}

  //: fold (numlist, xs["tl"]) ;
  //: fold (numlist, xs) ;
  return xs ;
} ;
```

Inferred Annotations are in Comments

Flow-Sensitive Type Information by Hovering over Program Point

Many Software Engineering Challenges!

Tools for Education

Learning **Complex Languages** is Difficult

Static Error Messages can be Mysterious

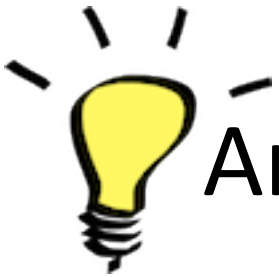
Generate Concrete Run-Time Traces?

- Execution may be easier to understand
- Help motivate compile-time guarantees

Tools for Education

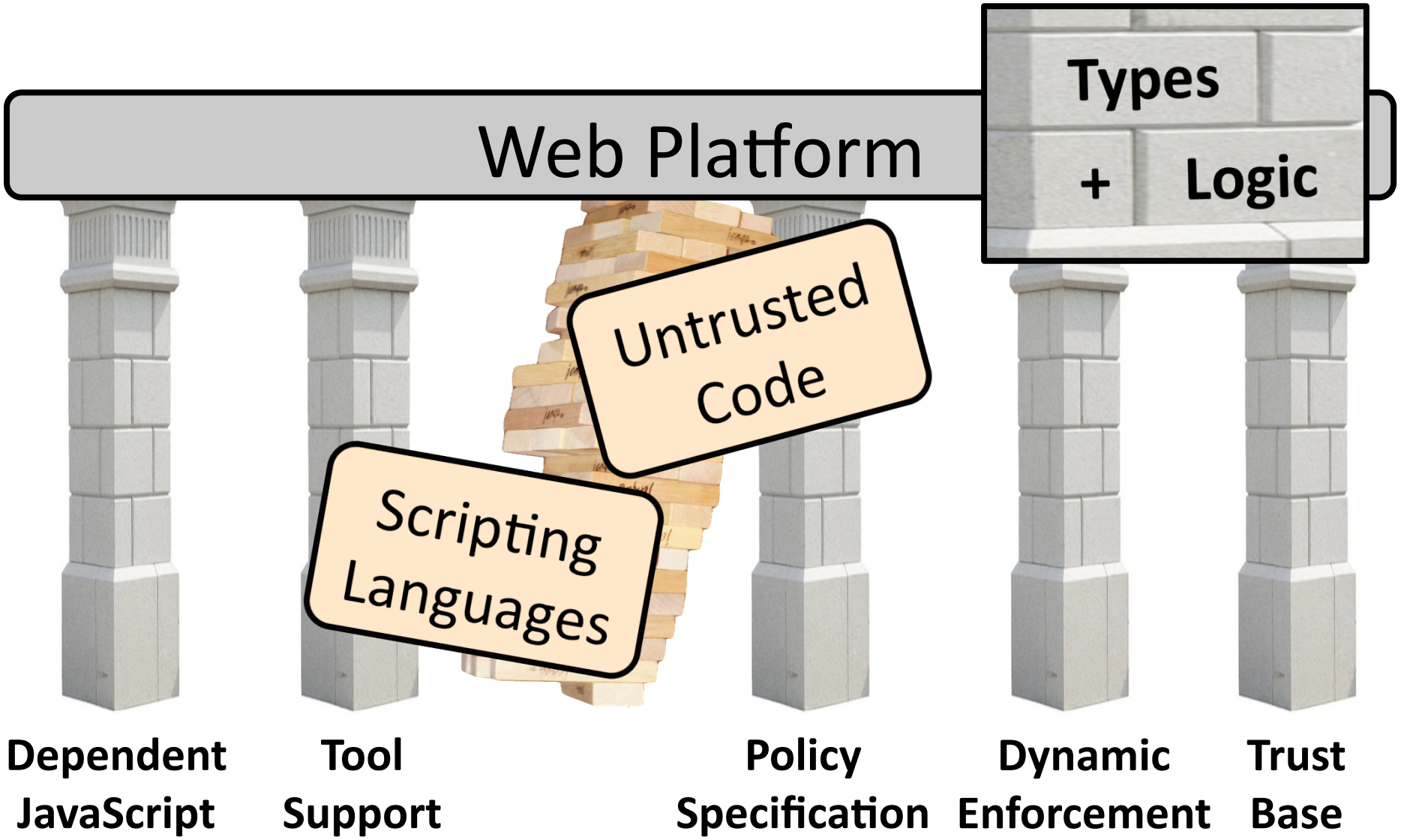
Learning **in Large Classes** is Difficult

Clickers and Peer Instruction Can Help

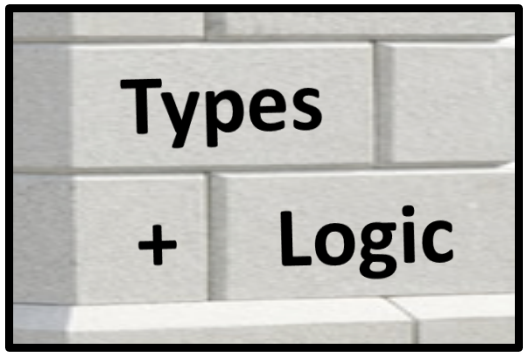


Analyze Live In-Class Performance?

- To guide subsequent topics during class
- To provide personalized feedback after class
- To suggest well-balanced study groups


Web Platform
Types
+ Logic
Untrusted Code
Scripting Languages
Dependent JavaScript
Tool Support
Policy Specification
Dynamic Enforcement
Trust Base

PL Techniques



Program Analysis for Security

[PLDI '09]

[ESOP '10]
[PLDI '10]

[STOP '11]
[POPL '12]
[OOPSLA '12]
[ML '13]

Dataflow Analysis for Race Detection

[PLDI '08]

Thanks!

Dynamic Languages are Pervasive

DJS: Precise Static Types via Logic

Towards Fine-Grained Web Security

UCSDCSE
Computer Science and Engineering

ravichugh.com