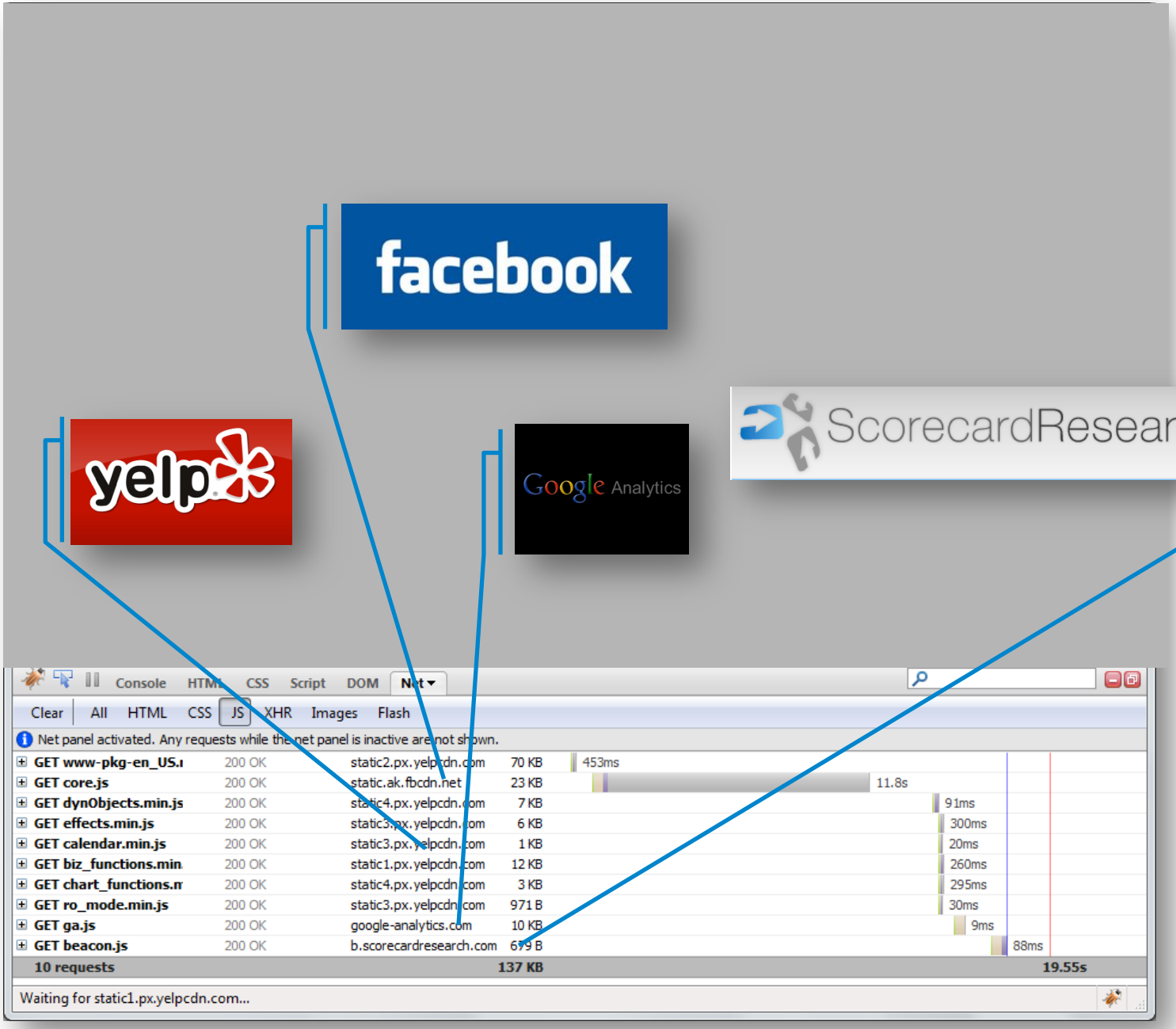
ConScript

Specifying and Enforcing Fine-Grained Security Policies for JavaScript in the Browser

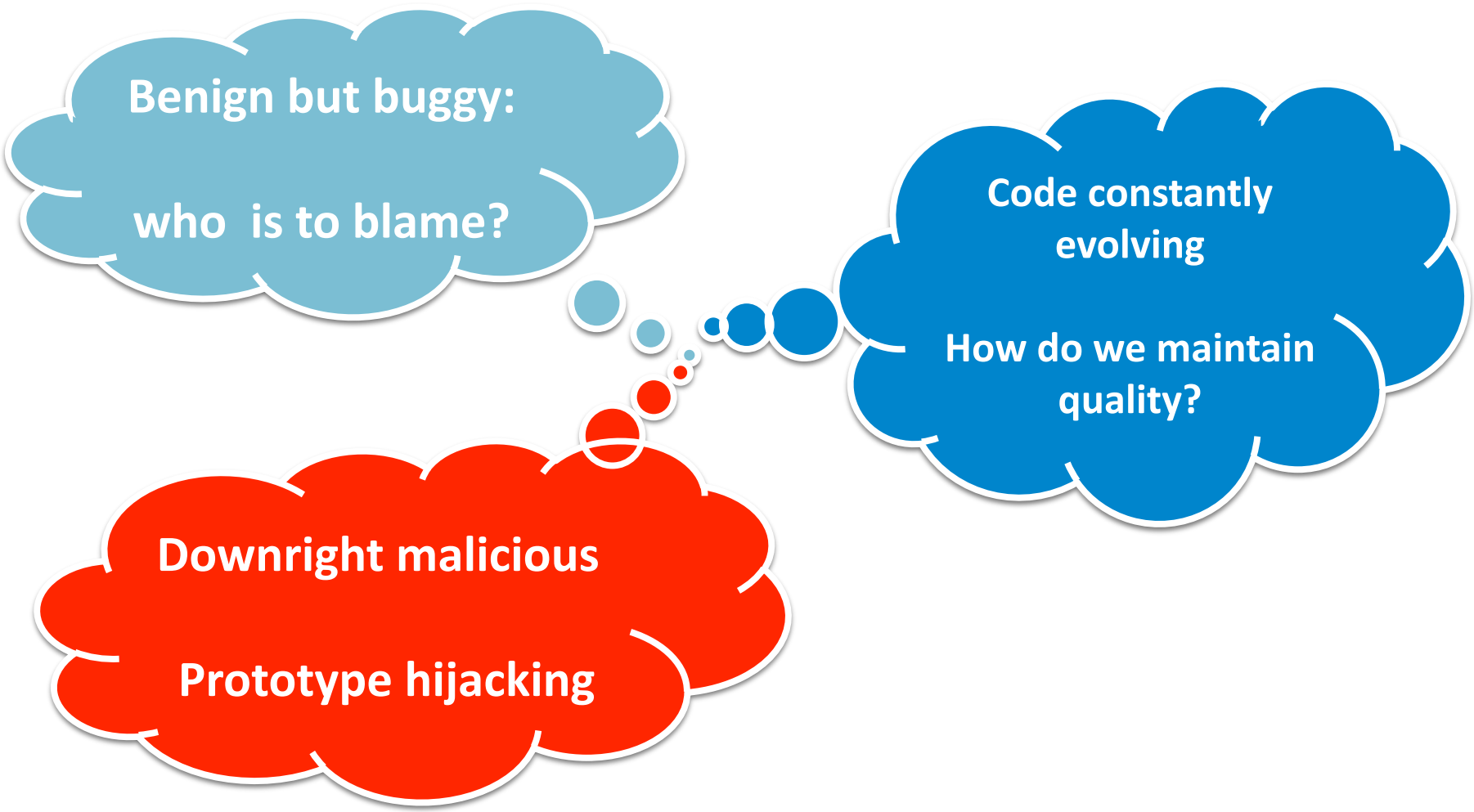
Leo Meyerovich
UC Berkeley

Benjamin Livshits
Microsoft Research

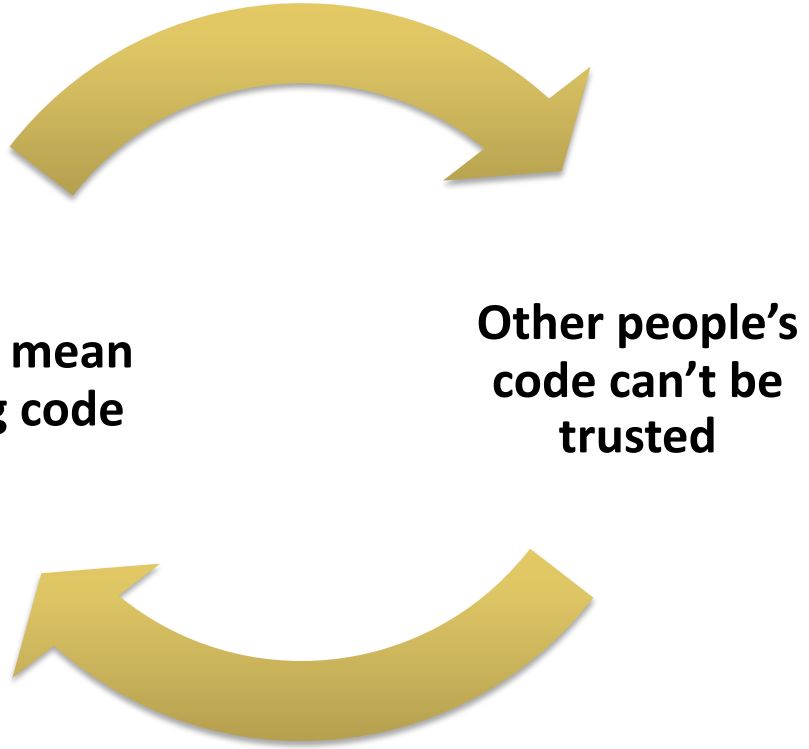
Microsoft®
Research


facebook
yelp
Google Analytics
ScorecardResearch
Console HTML CSS Script DOM Net
Clear All HTML CSS JS XHR Images Flash
Net panel activated. Any requests while the net panel is inactive are not shown.
GET www-pkg-en_US.i 200 OK static2.px.yelpcdn.com 70 KB 453ms
GET core.js 200 OK static.ak.fbcdn.net 23 KB 11.8s
GET dynObjects.min.js 200 OK static4.px.yelpcdn.com 7 KB 91ms
GET effects.min.js 200 OK static3.px.yelpcdn.com 6 KB 300ms
GET calendar.min.js 200 OK static3.px.yelpcdn.com 1 KB 20ms
GET biz_functions.min 200 OK static1.px.yelpcdn.com 12 KB 260ms
GET chart_functions.m 200 OK static4.px.yelpcdn.com 3 KB 295ms
GET ro_mode.min.js 200 OK static3.px.yelpcdn.com 971 B 30ms
GET ga.js 200 OK google-analytics.com 10 KB 9ms
GET beacon.js 200 OK b.scorecardresearch.com 679 B 88ms
10 requests 137 KB 19.55s
Waiting for static1.px.yelpcdn.com...

Complications



Developer's Dilemma



Only Allow `eval` of JSON

- Idea for a policy:
  - Parse input strings instead of running them
  - Use ConScript to *advise* **`eval`** calls

- AspectJ advice for Java

```
void around call Window::eval (String s) { … }
```

- How to do advice in JavaScript?
  - No classes to speak of

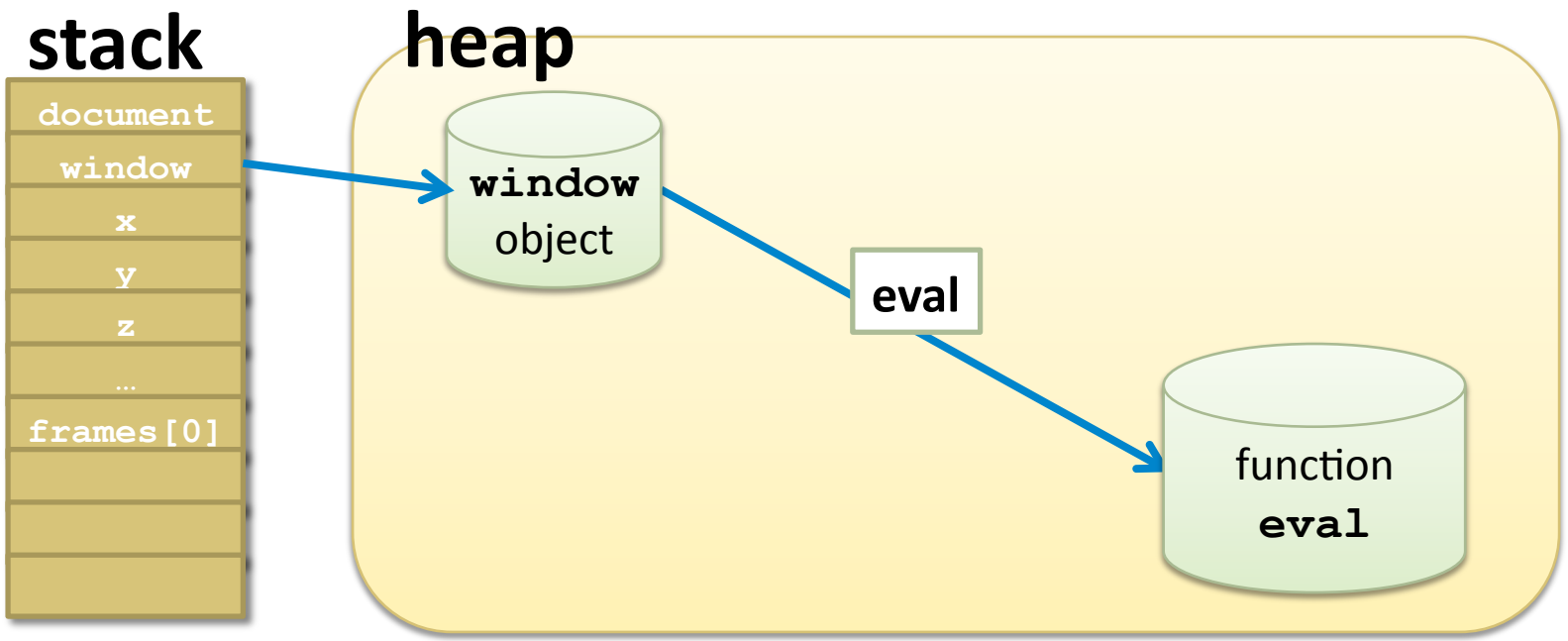
Only Allow `eval` of JSON

eval(“(xhr.open(‘evil.com’);)”)

eval(“([{‘hello’: ‘Oakland’}, 2010])”)

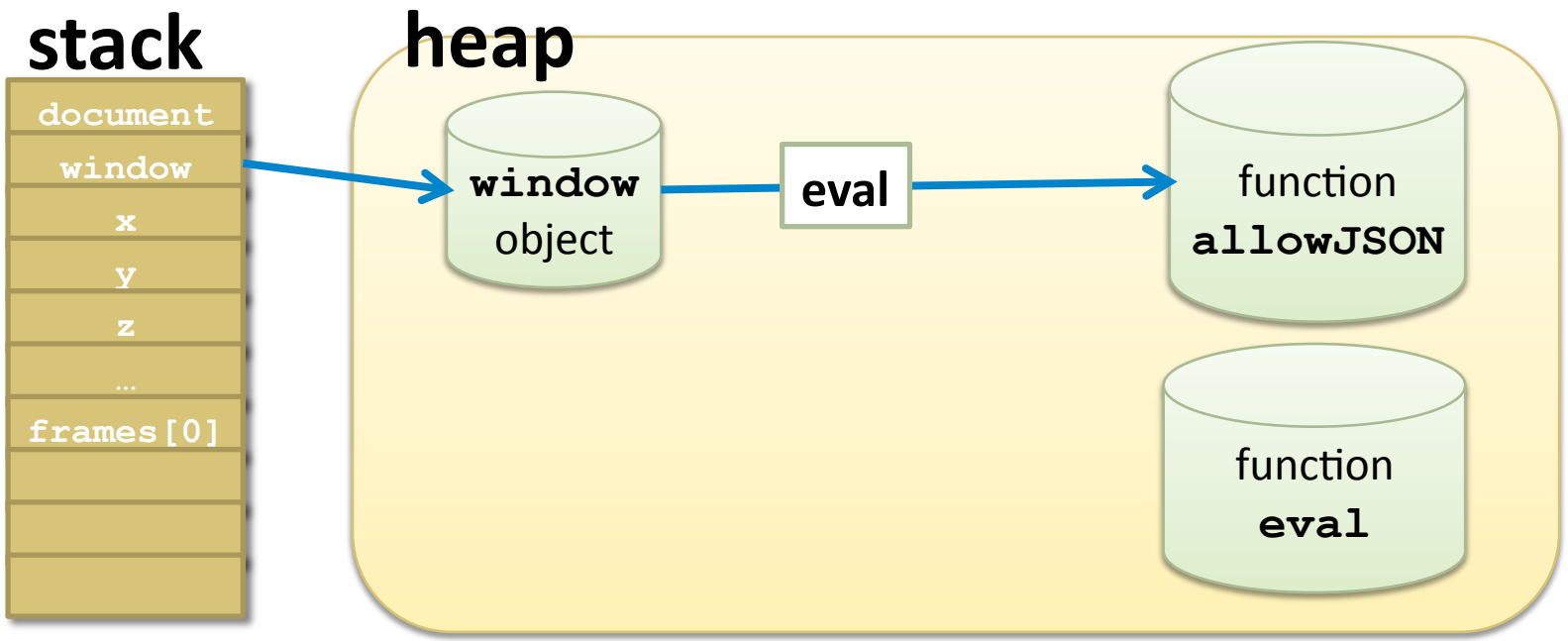
Advising Calls is Tricky

```
window.eval = function allowJSON() { … }
```



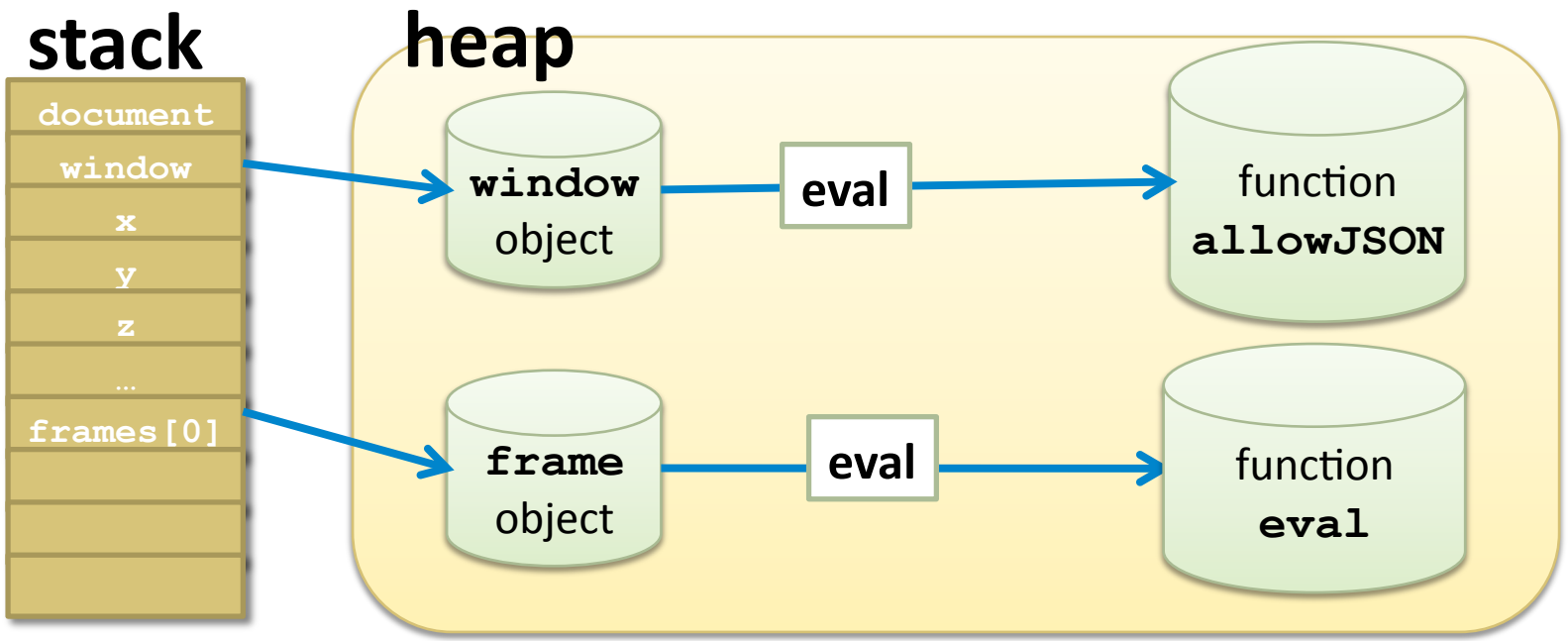
Advising Calls is Tricky

```
window.eval = function allowJSON() { … }
```



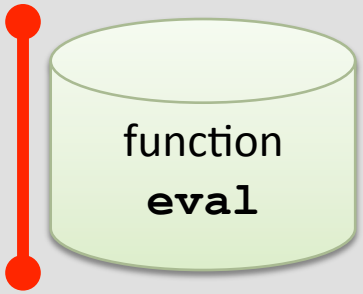
Advising Calls is Tricky

```
window.eval = function allowJSON() { … }
```



ConScript approach

- Deep advice for complete mediation
- Implemented within the browser for efficiency and reliability



Example of Applying Advice in ConScript

```
1. <SCRIPT SRC="facebook.js" POLICY="
2.     var substr = String.prototype.substring;
3.     var parse = JSON.parse;
4.     around(window.eval,
5.       function(oldEval, str) {
6.         var str2 = uCall(str, substr, 1,
7.                          str.length - 1);
8.         var res = parse(str2);
9.         if (res) return res;
10.        else throw "eval only for JSON";
11. } );">
```

Contributions of ConScript

A case for aspects in browser

- Protect benign users by giving control to hosting site
- ConScript approach: browser-supported aspects

Correctness checking

- Policies are easy to get wrong
- Type system to ensure policy correctness

Expressiveness

- Wide catalog of policies from literature and practice
- 17 concise hand-written policies
- Implemented 2 policy generators

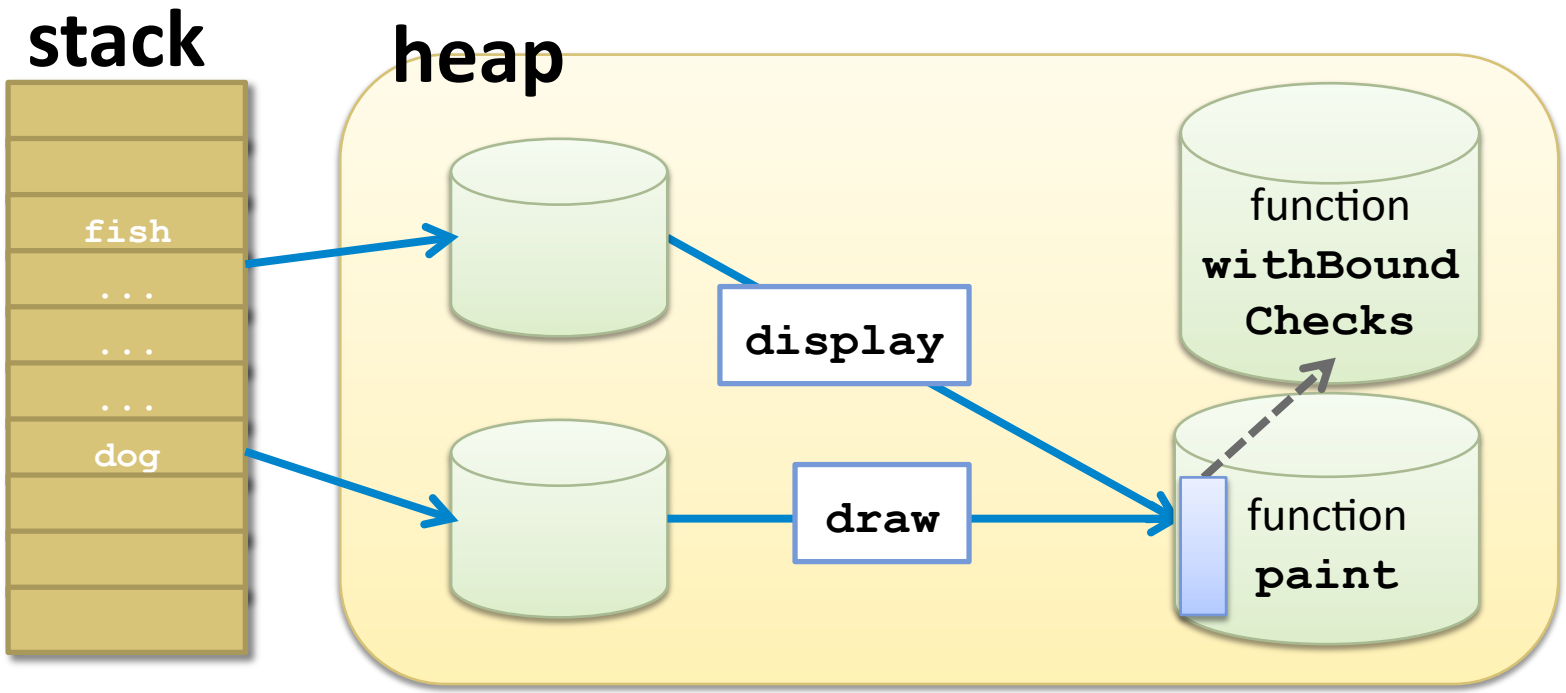
Real-world Evaluation

- Built into IE8 JavaScript interpreter
- Tested on real apps: Google Maps, Live Desktop, etc.
- Runtime and space overheads under 1% (vs. 30-550%)


A case for aspects in browser
Implementation
Correctness checking
Expressiveness
Real-world Evaluation

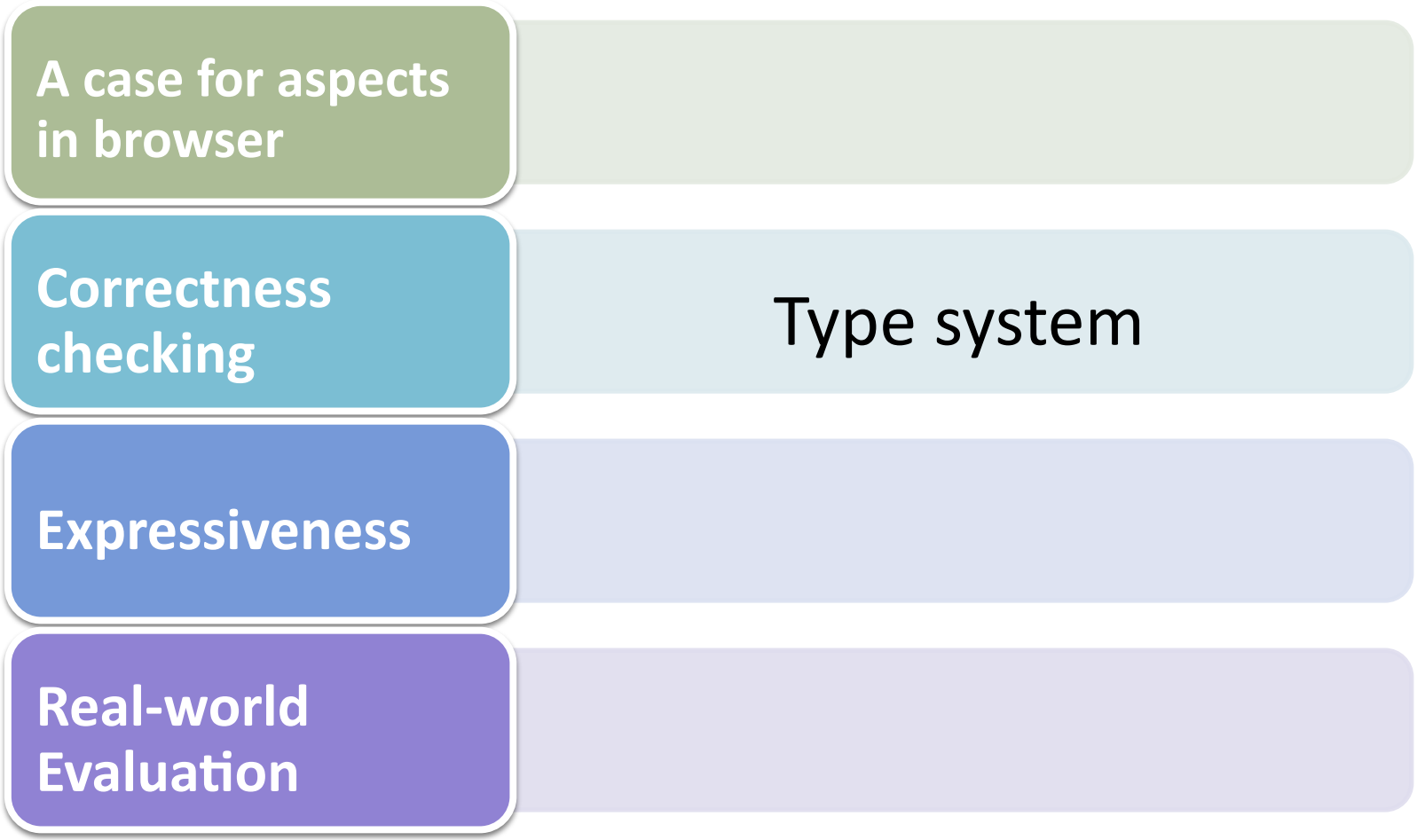
Advising JavaScript Functions in IE8

```
around(paint, withBoundChecks);
dog.draw();
fish.display();
```



This is Just the Beginning…

- Not just JavaScript functions
  - native JavaScript calls: **`Math.round,`** …
  - DOM calls: **`document.getElementById,`** …
- Not just functions…
  - script introduction
  - …
- Optimizations
  - Blessing
  - Auto-blessing


A case for aspects in browser
Correctness checking
Type system
Expressiveness
Real-world Evaluation

Policies are Easy to Get Wrong

```
1. var okOrigin={"http://www.google.com":true};
2. around(window.postMessage,
3.   function (post, msg, target) {
4.     if (!okOrigin[target]) {
5.       throw 'err';
6.     } else {
7.       return post.call(this, msg, target);
8.     }
9. });
```

Policies are Easy to Get Wrong

```
1.
2. around(windo      st
3.   function (post,
4.     if (!okOrigin[targ          {
5.       throw 'er      ';
6.     } else {
7.       return      t.call(this, msg, target);
8.
9
```

toString redefinition!

Function.prototype poisoning!

Object.prototype poisoning!

How Do We Enforce Policy Correctness?

Application code

- Unperturbed usage of legacy code
- Disallow **`arguments.caller`** to avoid stack inspection

  (disallowed by ES5's strict mode)

Policy code

- Modify the JavaScript interpreter
  - introduce **`uCall`**, **`hasProp`**, and **`toPrimitive`**
  - disable **`eval`**
- Propose a type system to enforce correct use of these primitives
  - disable **`with`**, …

Policy Type System

- ML-like type system
- Uses security labels to denote privilege levels
- Enforces *access path integrity* and *reference isolation*

$$\frac{\Gamma \vdash o : (f : T^L;\ r)^\circ \qquad \Gamma \vdash v : T^L}{\Gamma \vdash o.f = v : T^L} \quad \text{(k stat set)}$$

Policy Type System

Reference isolation

- *o* does not leak through poisoning if *f* is a field

- Enforc… *…th integrity* and *reference isolation*

$$\frac{\Gamma \vdash o : (f : T^L;\ r)^\circ \qquad \Gamma \vdash v : T^L}{\Gamma \vdash o.f = v : T^L} \text{ (k stat set)}$$

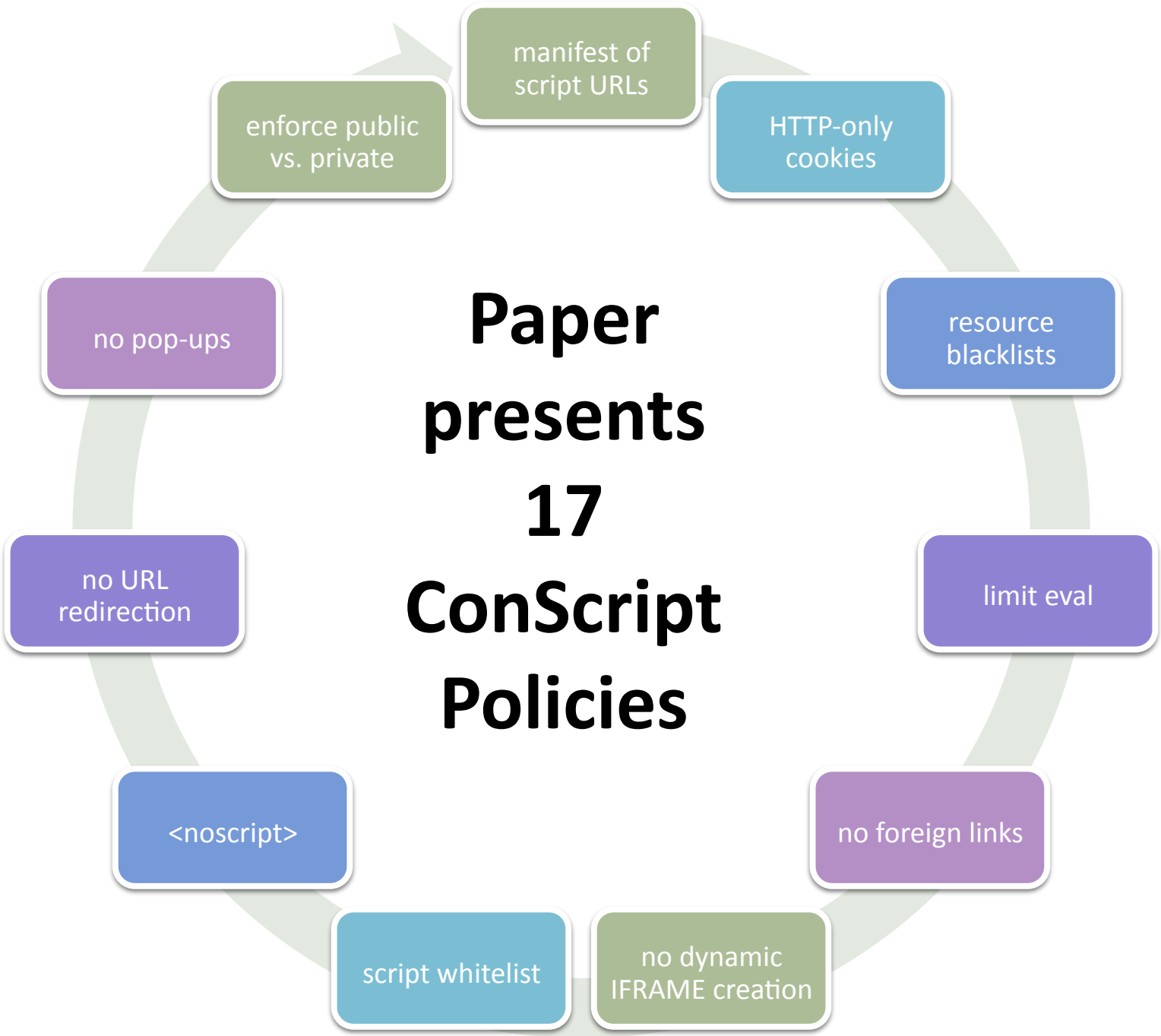
Access path integrity for function calls

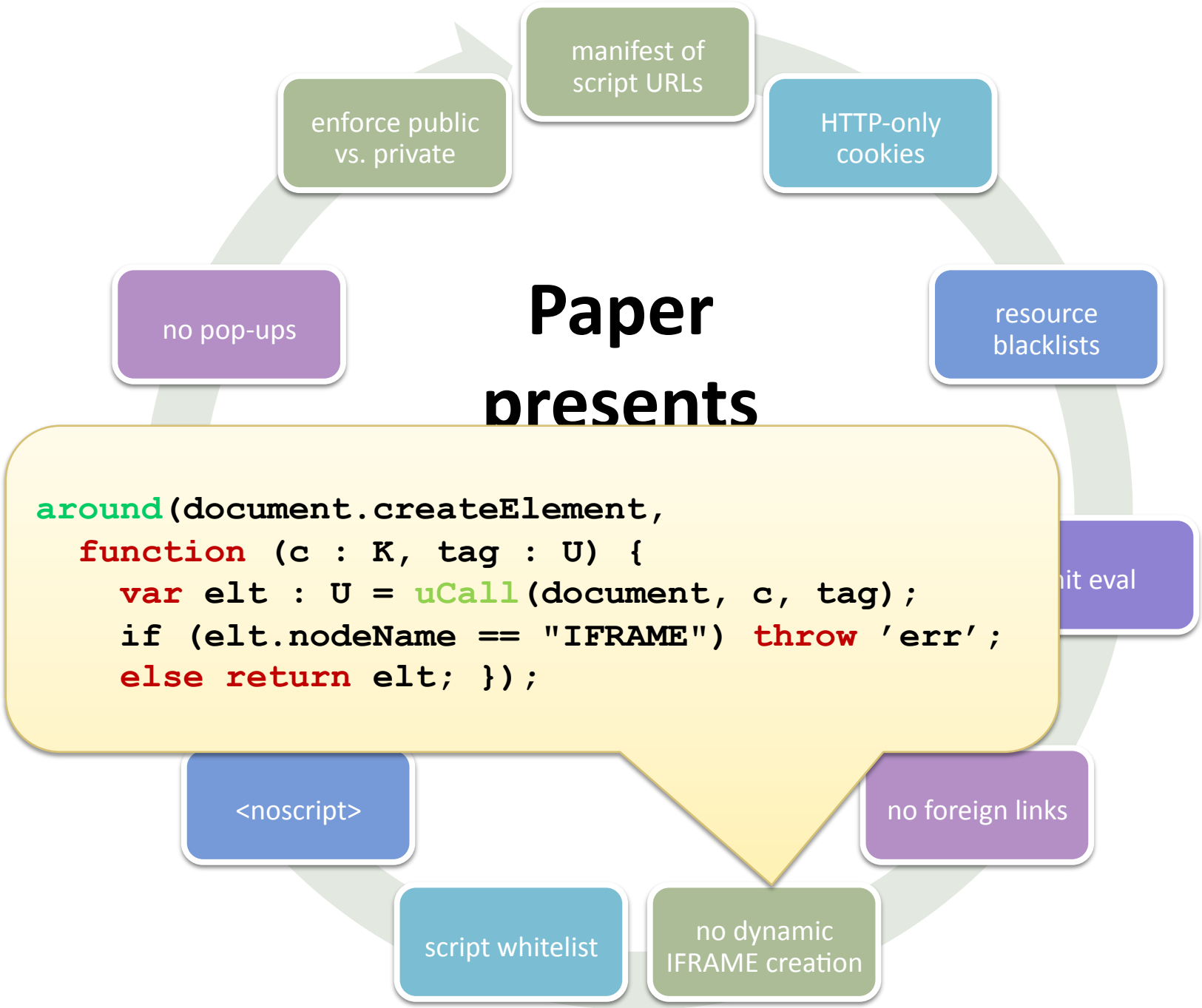
- *o.f* remains unpoisoned if *T* in *v : T* is not poisoned


A case for aspects in browser
Correctness checking
Expressiveness
Real-world Evaluation
Policies

ConScript Policies

- **17 hand-written policies**
  - Diverse: based on literature, bugs, and anti-patterns
  - Short: wrote new HTML tags *with only a few lines of code*
- **2 automatic policy generators**
  - Using runtime analysis
  - Using static analysis

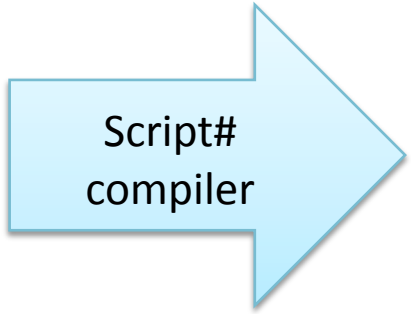

manifest of script URLs
HTTP-only cookies
resource blacklists
limit eval
no foreign links
no dynamic IFRAME creation
script whitelist
<noscript>
no URL redirection
no pop-ups
enforce public vs. private
Paper presents 17 ConScript Policies


manifest of script URLs
HTTP-only cookies
resource blacklists
it eval
no foreign links
no dynamic IFRAME creation
script whitelist
<noscript>
no pop-ups
enforce public vs. private
Paper presents
around(document.createElement,
function (c : K, tag : U) {
var elt : U = uCall(document, c, tag);
if (elt.nodeName == "IFRAME") throw 'err';
else return elt; });

Generating Intrusion Detection Policies

Enforcing C# Access Modifiers

```
class File {
    public File () { … }
    private open ()  { … }
    …
```




```
function File () { … }
File.construct = …
File.open = …
…
```

C#

JavaScript

Enforcing C# Access Modifiers

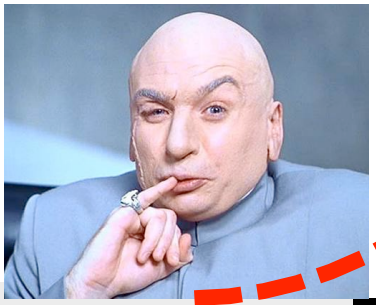
```
class File {
    public File () { … }
    private open ()  { … }
    …
```

Script# compiler

```
function File () { … }
File.construct = …
File.open = …
…
```

C#

JavaScript



flickr

jQuery write less, do more.

Enforcing C# Access Modifiers

```
class File {
    public File () { … }
    private open ()  { … }
    …
```

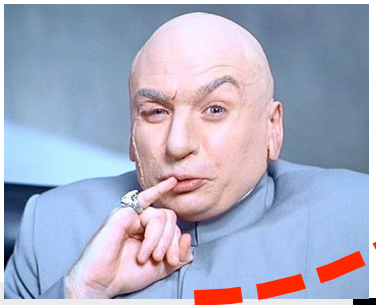
Script# compiler

```
function File () { … }
File.construct = …
File.open = …
…
```

C#

policy generator

JavaScript



```
around(File, pubEntryPoint);
around(File.construct, pubEntryPoint);
around(File.open, privCall);
```

flickr

jQuery write less, do more.

ConScript


A case for aspects in browser
Correctness checking
Expressiveness
Real-world Evaluation
Evaluation

Experimental Evaluation

Low adoption barrier

- Implemented on top of the IE 8 JavaScript interpreter
- TCB increase: under 1,000 lines added to IE8's JavaScript engine
- Changed a few language constructs to disallow `arguments.caller`

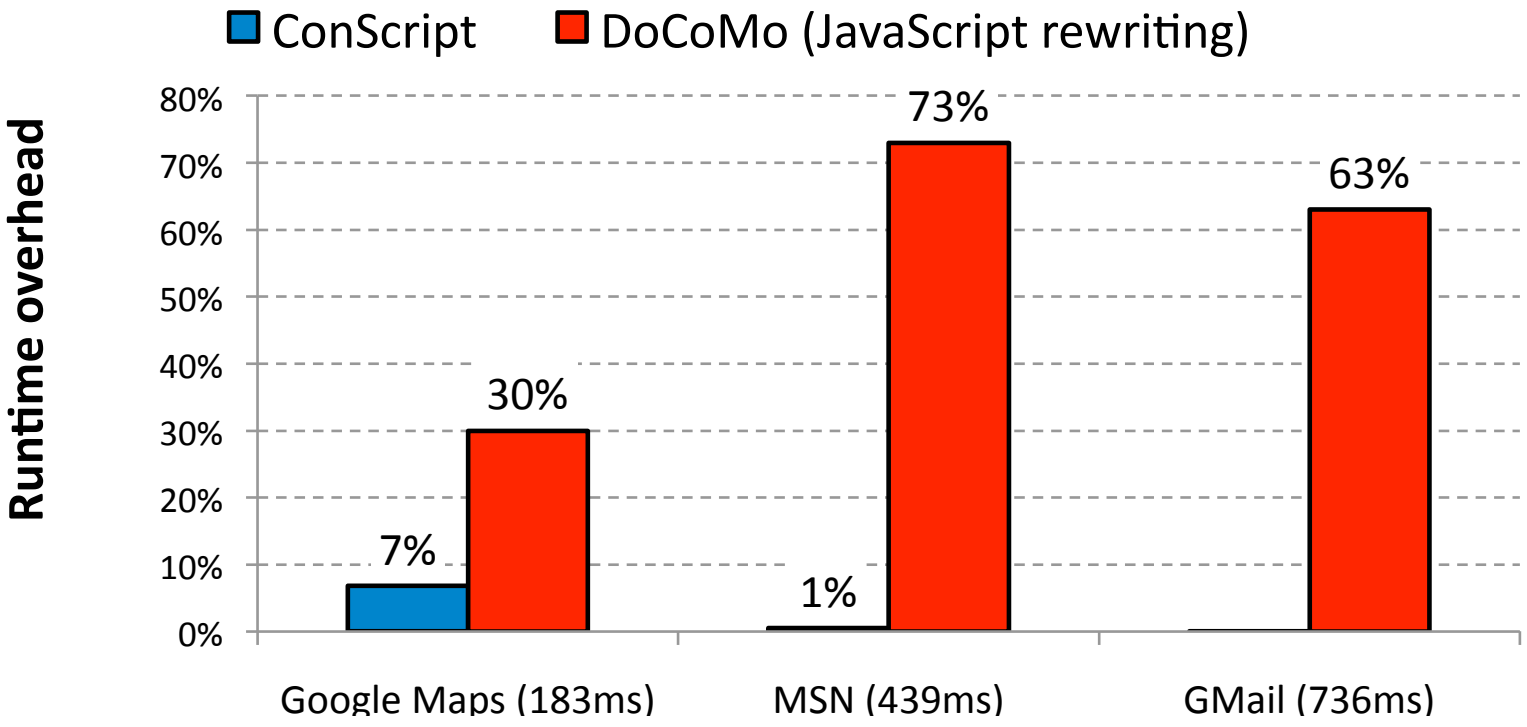
Micro-benchmarks

- Function, DOM call, eval overhead
- 2.5x faster than previously published source-level wrapping
- Advice optimizations make it faster still

Macro-benchmarks

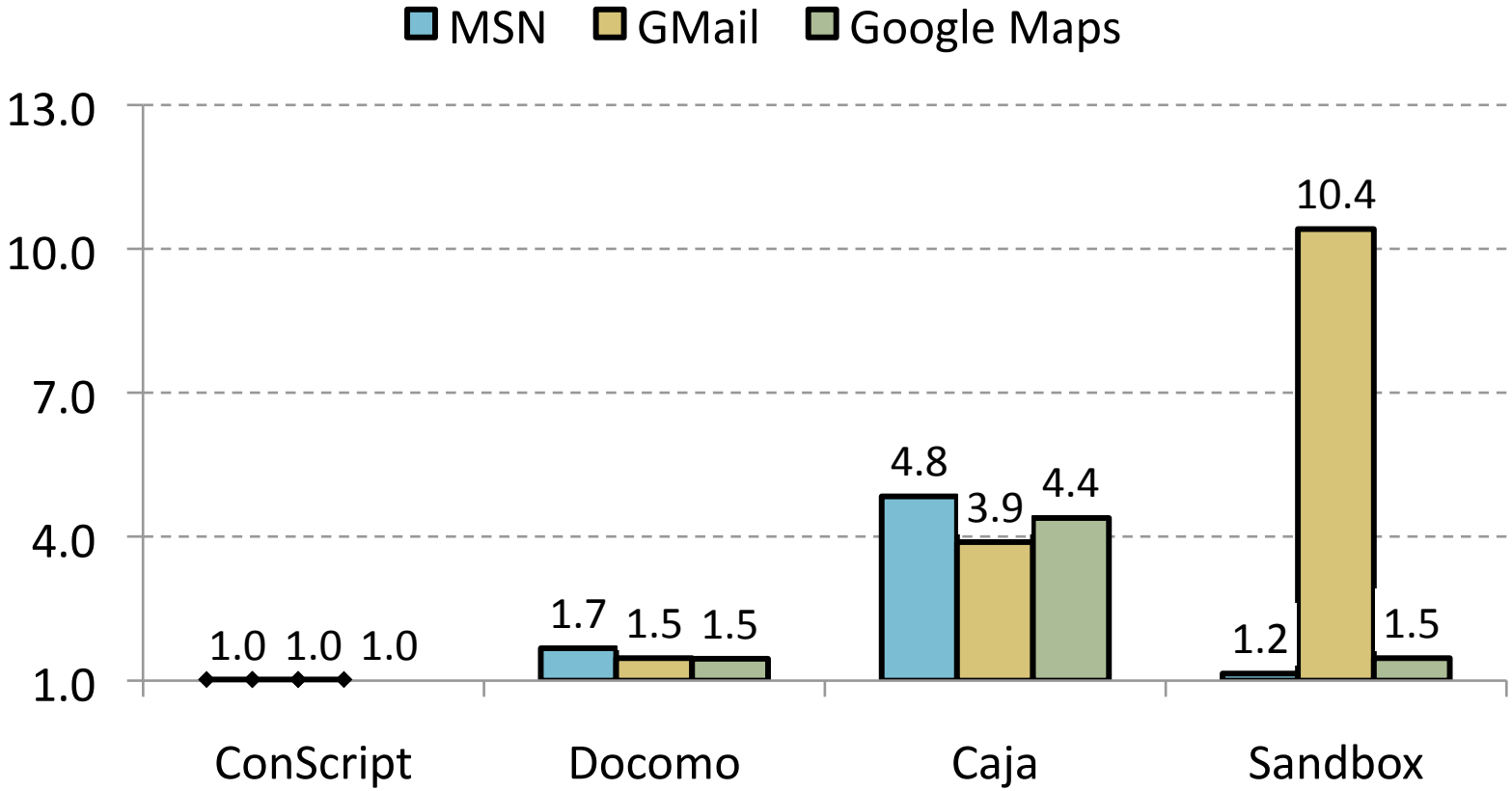
- Google Maps, MSN, Gmail, Live Desktop, Google Calendar
- Hundreds of KB of JavaScript code
- Runtime overhead due to ConScript advice: mostly under 1%
- File size increase due to ConScript advice: under 1%

DoCoMo Policy Enforcement Overhead



H. Kikuchi, D. Yu, A. Chander, H. Inamura, and I. Serikov, "JavaScript instrumentation in practice," 2008

File Size Increase for Blacklisting Policy

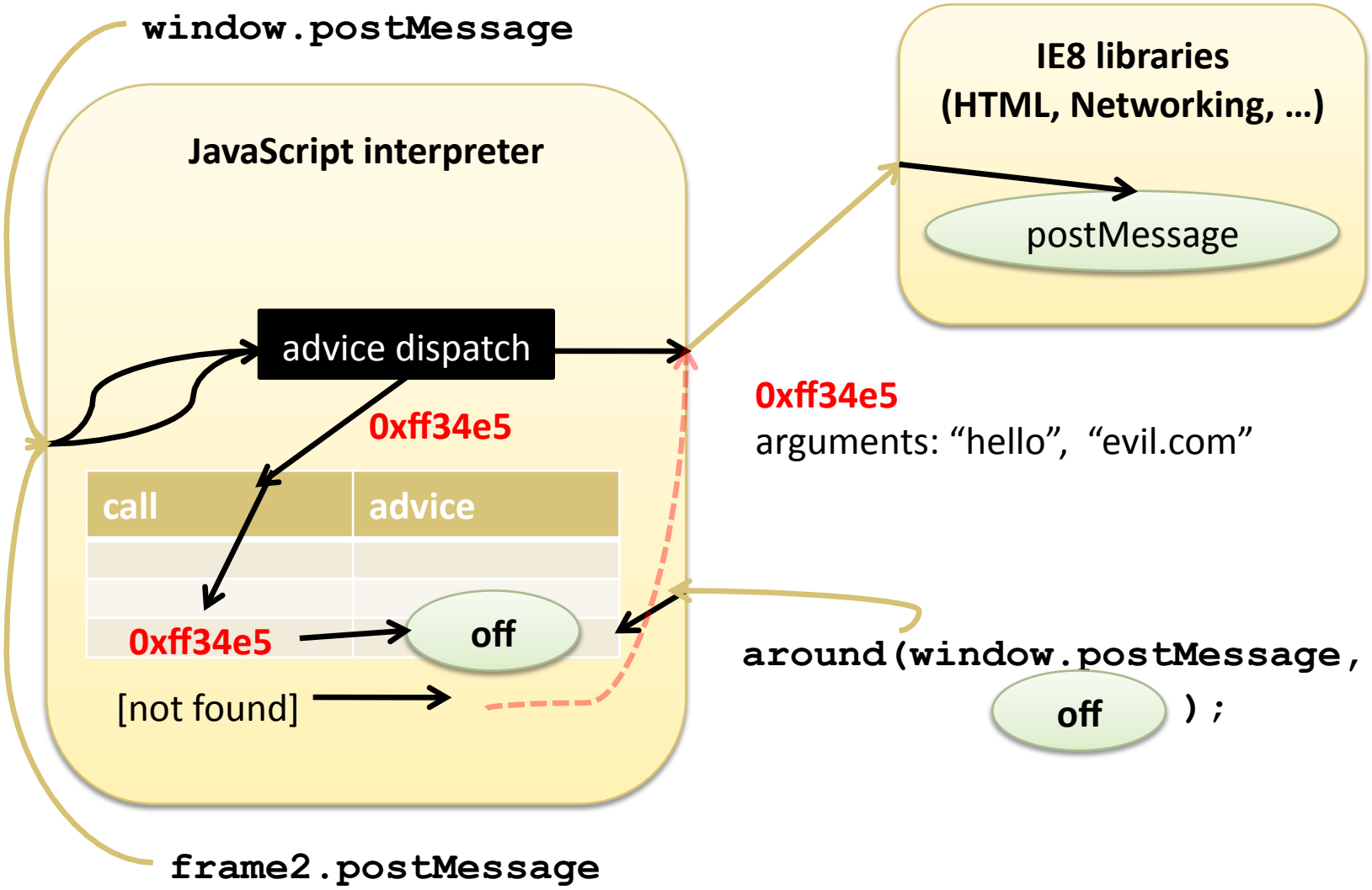


Conclusions

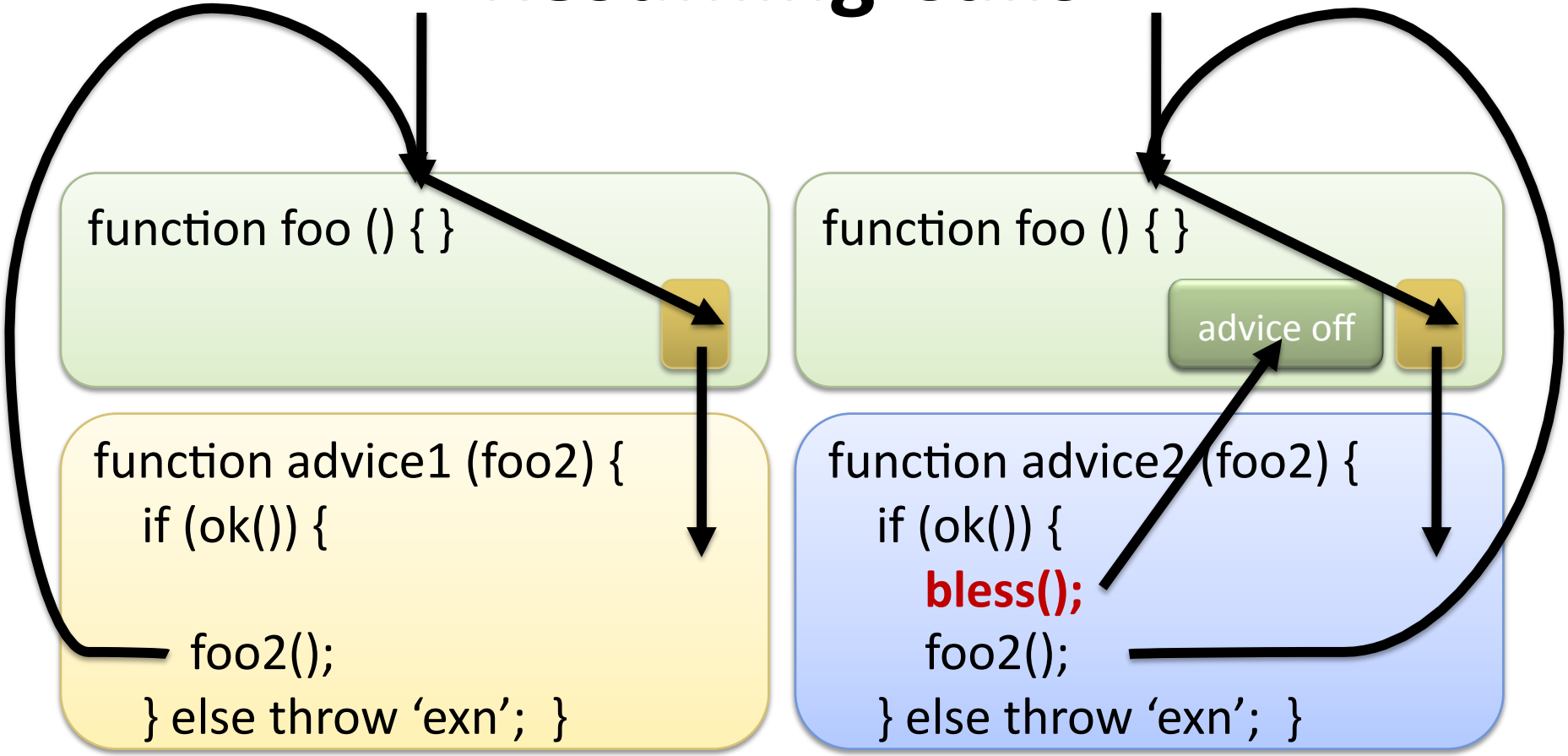
A case for aspects in browser	• To provide reliable enforcement, browser changes are required and can be minimal
Correctness checking	• Previous attempts illustrate that hand-written policies are buggy. ConScript addresses this with a type system without affecting legacy code
Expressiveness	• Provide a catalog of 17 hand-written policies for other researchers to use and show how polices can be generated by translators like Script#
Real-world Evaluation	• Implementing policy enforcement in the browser and not at the source level has tremendous performance advantages

QUESTIONS?

Mediating DOM Functions

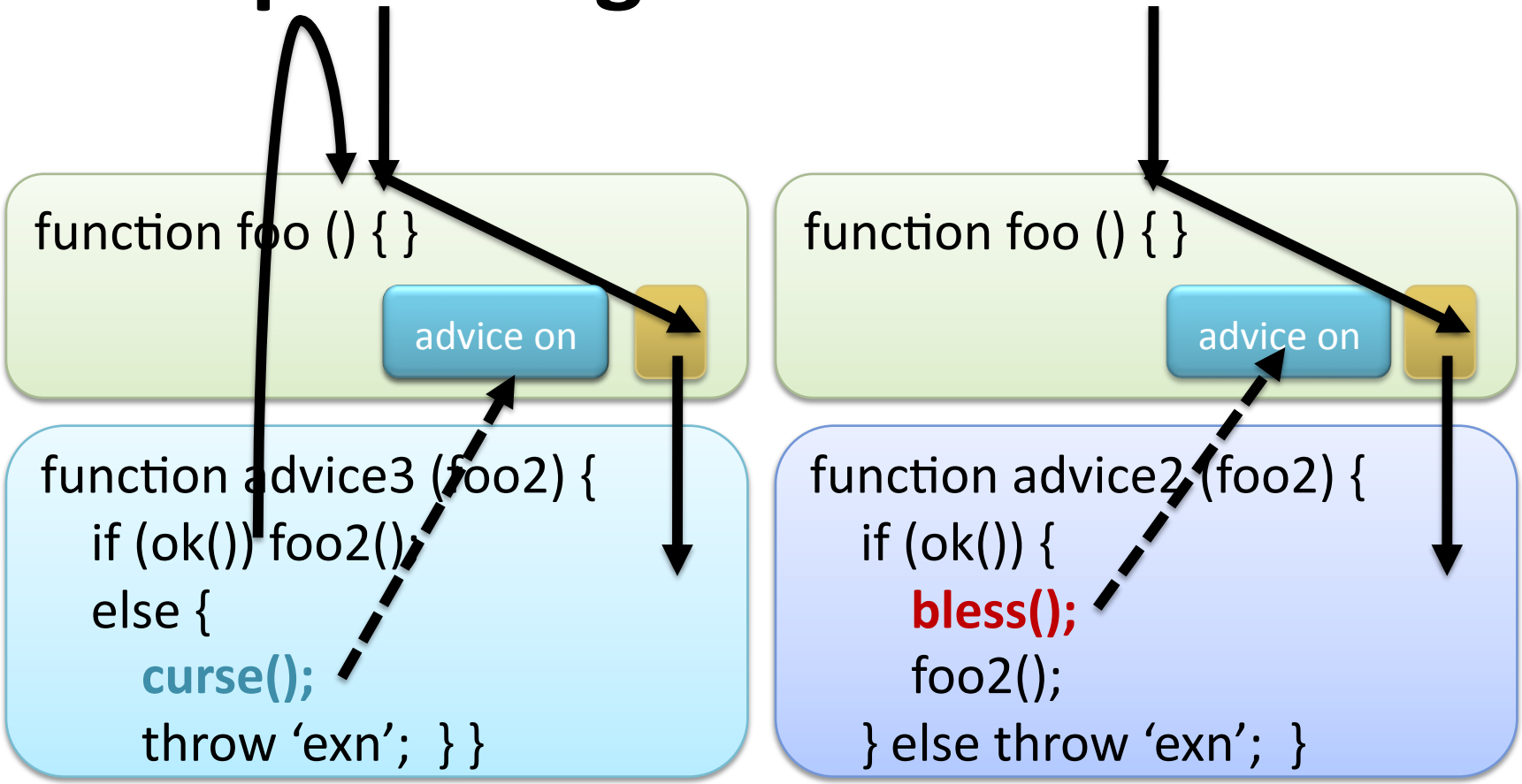


Resuming Calls



bless() temporarily disables advice for next call

Optimizing the Critical Path



- calling advice turns advice off for next call
- **curse()** enables advice for next call