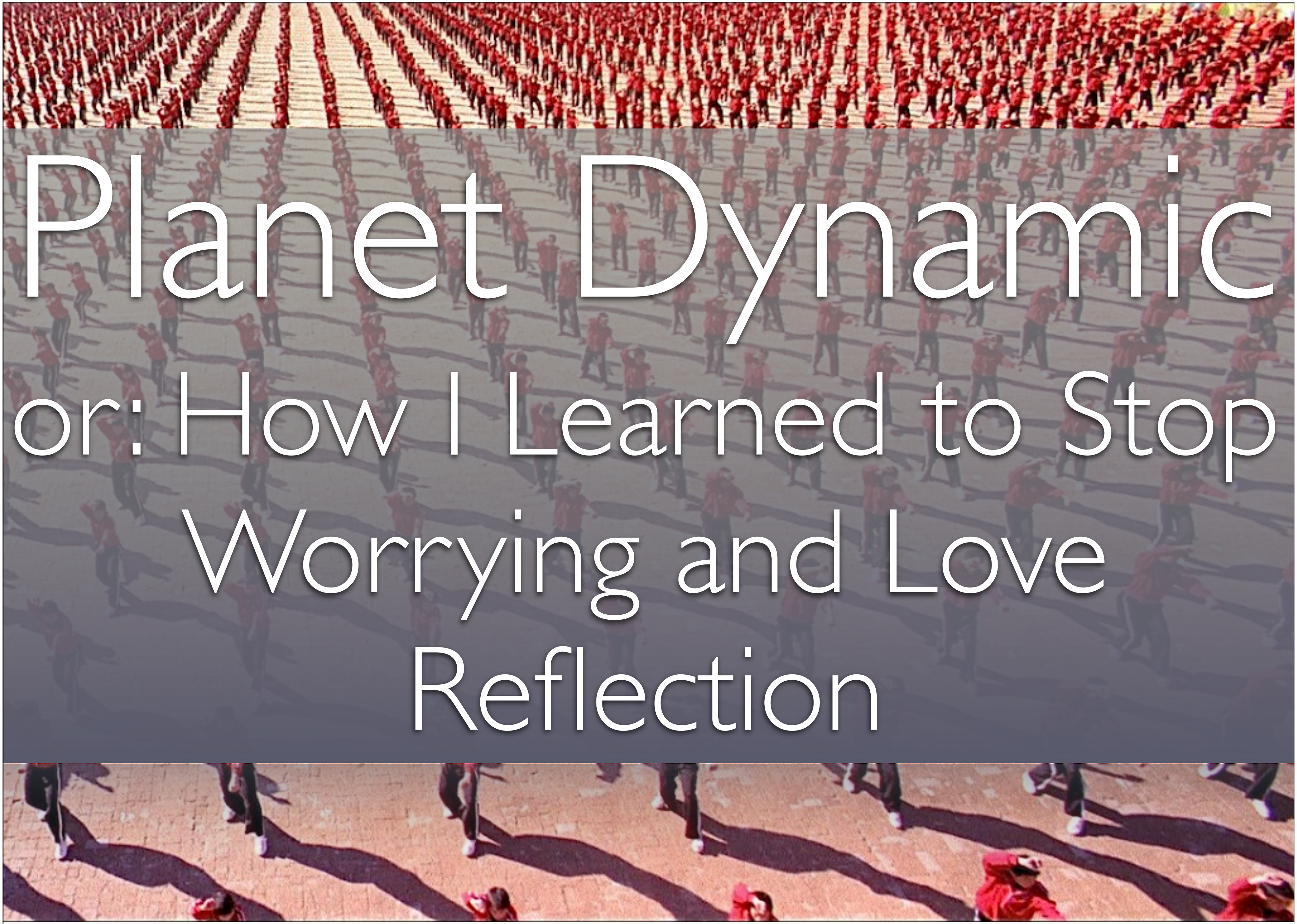


Planet Dynamic or: How I Learned to Stop Worrying and Love Reflection

Jan Vitek

Orthodoxy

- Types increase programmer productivity
- Types catch errors early
- Static is better

Smalltalk

JavaScript

Ruby

Shell

ActionScript

R

Perl

Lua

PHP

Tcl

Erlang

VB

Matlab

Python

Clojure

Forth

Lisp

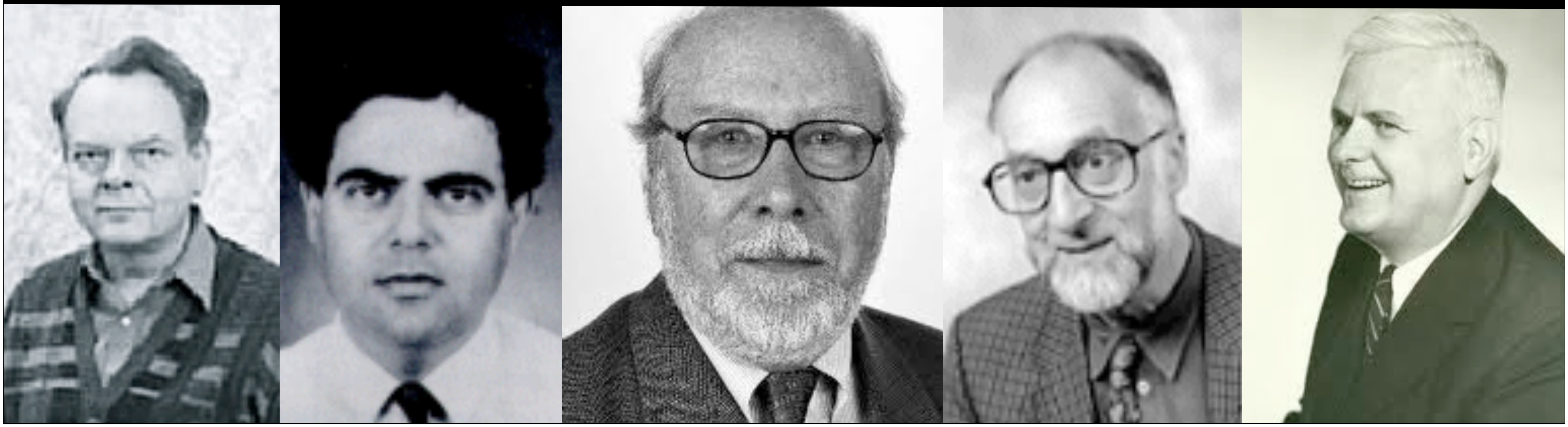
disconnects

data is untyped

data is mutable

data is shapeless

code is data



8

- What makes dynamic languages popular
- How to write mission critical software in a dynamic language
- Which is the most widely used lazy functional language
- Are programs written in dynamic language really different
- Why did Firefox lose the browser wars
- What’s in a modern dynamic language virtual machine
- How is reflection used in dynamic languages
- Can we get rid of eval automatically

paper trail

- *Meawad, Richards, Morandat, Vitek.* Eval Begone! Semi-Automated Removal of Eval from JavaScript Programs. OOPLSA ’12
- *Morandat, Hill, Osvald, Vitek.* Evaluating the Design of the R Language. ECOOP ’12
- *Richards, Gal, Eich, Vitek*. Automated Construction of JavaScript Benchmarks. OOPSLA ’11
- *Richards, Hammer, Burg, Vitek.* The Eval that Men Do: A Large-scale Study of the Use of Eval in JavaScript Applications. ECOOP ’11
- *Richards, Lebresne, Burg, Vitek,* An Analysis of the Dynamic Behavior of JavaScript Programs. PLDI ’10

commonalities

- Lightweight syntax
- Embeddable
- Extendible
- Failure oblivious
- Single threaded
- Garbage Collected

- Strong Dynamic Typing
- Interactive
- Reflective
- High-level Data Structures
- Permissive

case study: Lua

- C library for seamless embedding

Lightweight
Embeddable
Extendible
Failure oblivious

Single threaded
Portable
Dynamic Typing
Interactive

Reflective
High-level Data
Permissive
Garbage-collected

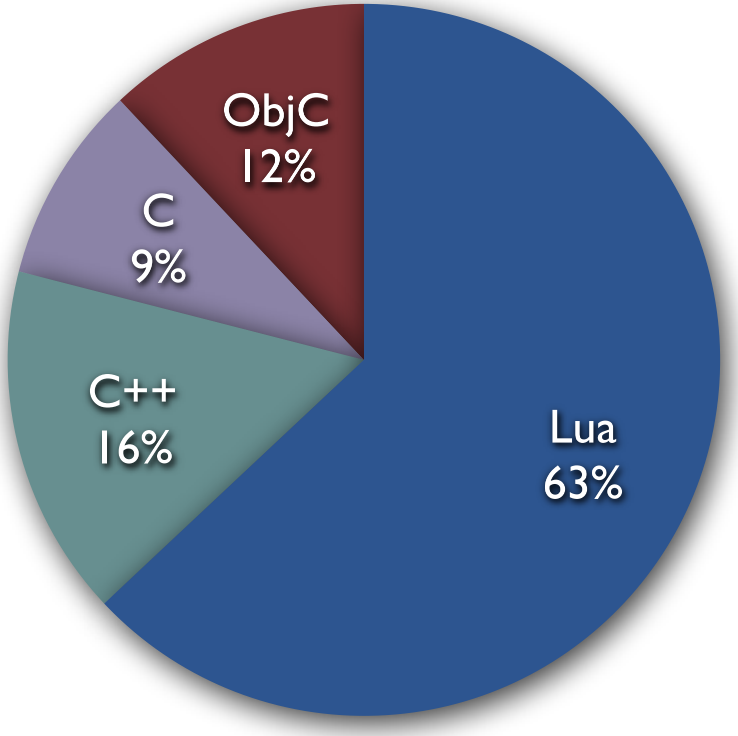
Lerusalimschy, et. al. *Passing a Language through the Eye of a Needle*, ACMQUEUE, 2011

case study: Lua

Adobe Lightroom

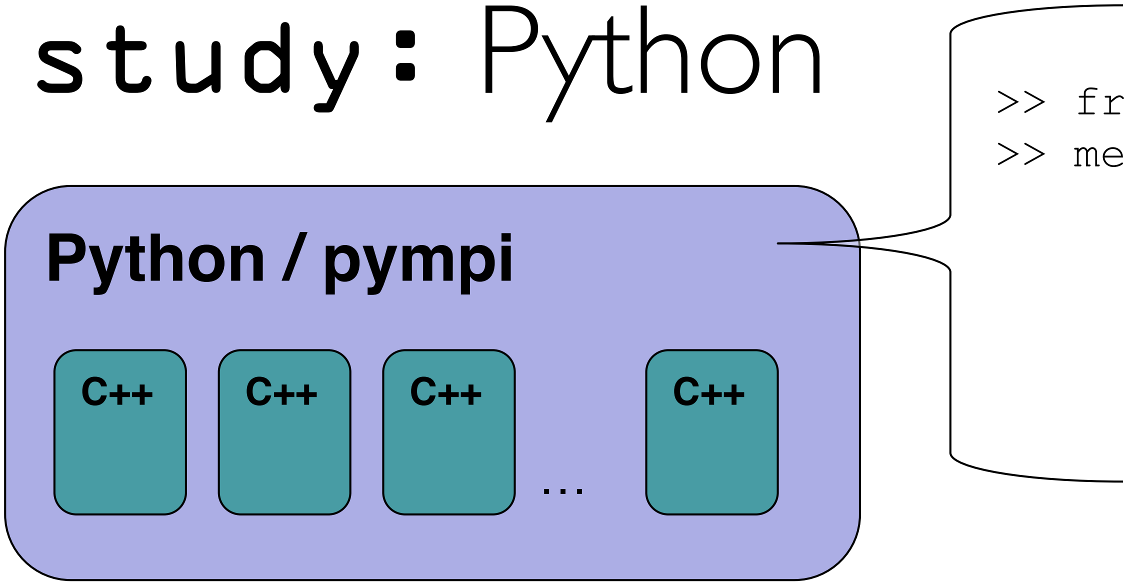
Used ...

- ... to glue components
- ... for business logic, controllers, views
- ... for its fast turn around



Troy Gaul. **Lightroom Exposed.** `http://www.troygaul.com`

case study: Python



```
>> from kull import *
>> mesh = Mesh(aFileNa
```

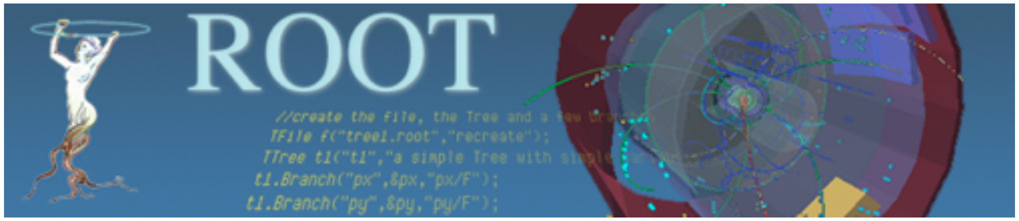
… inertial confinement fusion simulation

… extends C++ to provide a “steerable” simulation

… ~2 Mloc generated C++ SWIG wrappers

Alumbaugh, **Dynamic Languages for HPC at LLNL**. *Talk at VEESC Workshop, 2010*

case study: CERN



```
//create the file, the Tree and a few branches
TFile f("tree1.root","recreate");
TTree t1("t1","a simple Tree with simple
t1.Branch("px",&px,"px/F");
t1.Branch("py",&py,"py/F");
```

- Dynamic languages used: Python, Perl, Bash, Tcl, …
- But, most of the analysis code is in C++

Can C++ be turned into a dynamic language?

~~Lightweight~~

~~Embeddable~~

~~Extendible~~

~~Failure-oblivious~~

~~Single-threaded~~

~~Portable~~

~~Dynamic Typing~~

~~Interactive~~

~~Reflective~~

~~High-level Data~~

~~Permissive~~

Open

case study: CERN & CINT

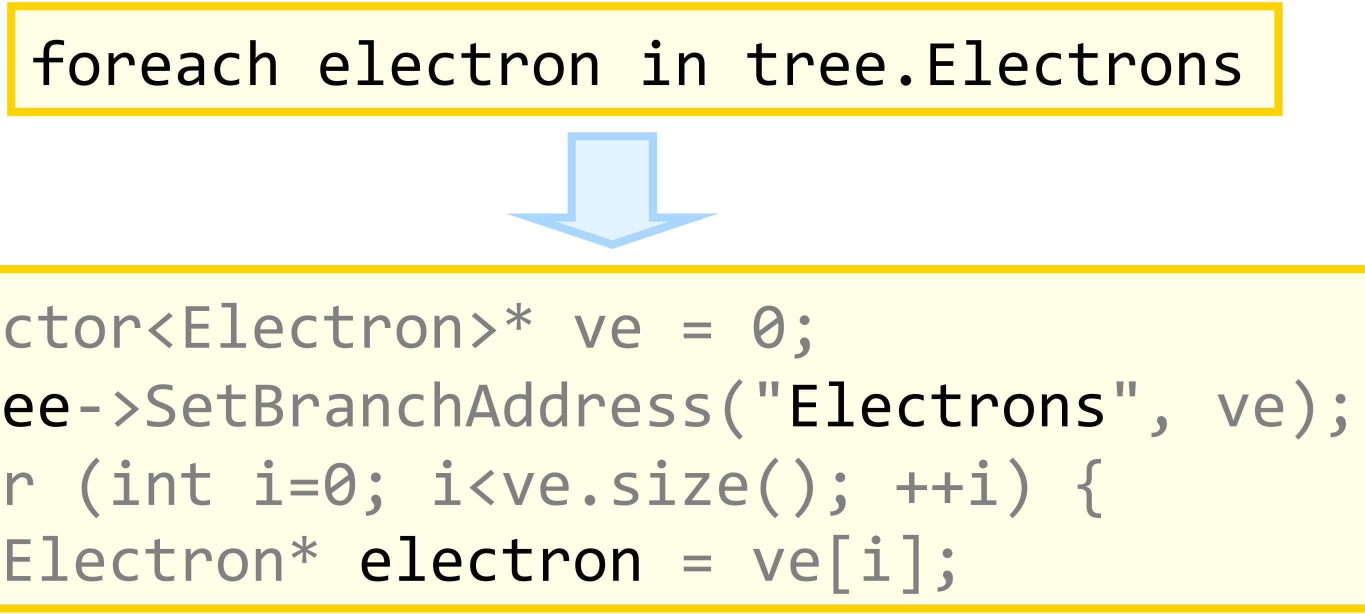
- From 1991, 400KLOC; parser, interpreter, reflection
- Interface to ROOT data analysis framework, >20k users

Ideally:

Higher level syntax

Faster

Threading



```
foreach electron in tree.Electrons
```

```
vector<Electron>* ve = 0;
tree->SetBranchAddress("Electrons", ve);
for (int i=0; i<ve.size(); ++i) {
  Electron* electron = ve[i];
```

Antcheva, Ballintijn, Bellenot, Biskup, Brun, Buncic, Canal, Casadei, Couet, Fine, Franco, Ganis, Gheata, Gonzalez Maline, Goto, Iwaszkiewicz, Kreshuk, Segura, Maunder, Moneta, Naumann, Offer, Onuchin, Panacek, Rademakers, Russo, Tadel. **ROOT — A C++ framework for petabyte data storage, statistical analysis and visualization**. *Computer Physics Comm. 2009*

case study:

Pluto

… manages the retirement savings of 5.5 million users

… for a value of 23 billion Euros

320 000 lines of Perl

68 000 lines of SQL

27 000 lines of shell

26 000 lines of HTML

Lundborg, Lemonnier. **PPM or how a system written in Perl can juggle with billions.** Freenix 2006
Lemonnier. **Testing Large Software With Perl.** Nordic Perl Workshop 2007
Stephenson. **Perl Runs Sweden's Pension System.** O’Reilly On Lamp, 2005

case study: Perl

High productivity: *Perl wins over Java*

Home-made contract notation: *Runtime checked*

Lightweight
~~Embeddable~~
Extendible
Failure oblivious

Single threaded
Portable
Dynamic Typing
Interactive

Reflective
High-level Data
Permissive
Open

case study: Perl

```
contract(‘do_sell_current_holdings’)
  -> in(&is_person, &is_date)
  -> out(&is_state)
  -> enable;
```

```
sub do_sell_current_holdings {
   my ($person, $date)
   …
   if ($operation eq “BUD_”) {
   …
   return $state;
}
```

case study: R

Lightweight

~~Embeddable~~

Extendible

Failure oblivious

Single threaded

Portable

Dynamic Typing

Interactive

Reflective

High-level Data

Permissive

Open

The R Ecosystem

… a language for data analysis and graphics

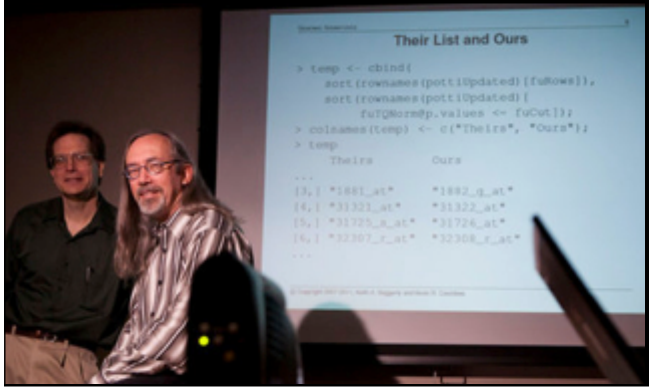
… used in statistics, biology, finance …

… books, conferences, user groups

… 4,338 packages

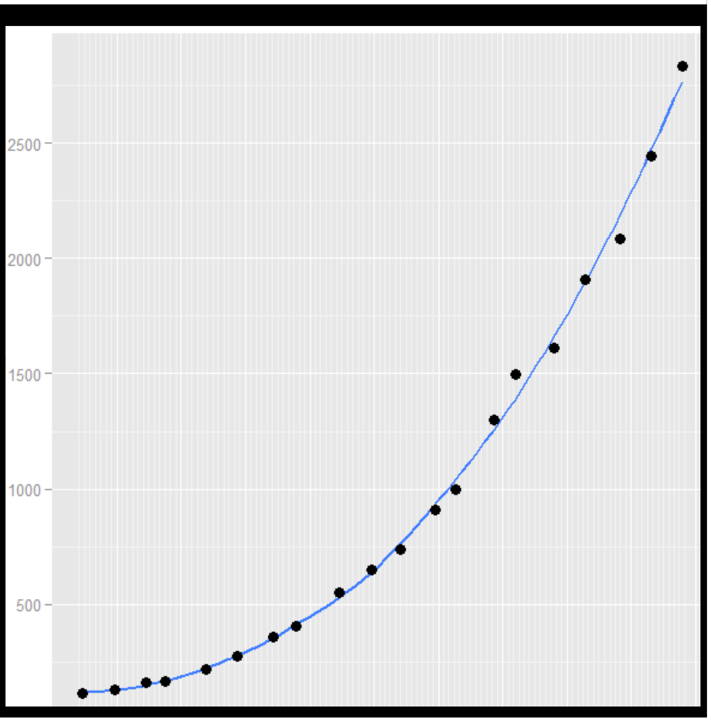
… 3 millions users

… trustworthy



Their List and Ours

```
> temp <- cbind(
    sort(rownames(pottiUpdated)[fuRows]),
    sort(rownames(pottiUpdated)[
        fuTQNorm@p.values <= fuCut]);
> colnames(temp) <- c("Theirs", "Ours");
> temp
     Theirs          Ours
...
[3,] "1881_at"       "1882_g_at"
[4,] "31321_at"      "31322_at"
[5,] "31725_s_at"    "31726_at"
[6,] "32307_r_at"    "32308_r_at"
...
```



R Programming

interact with the IDE:

read data into variables

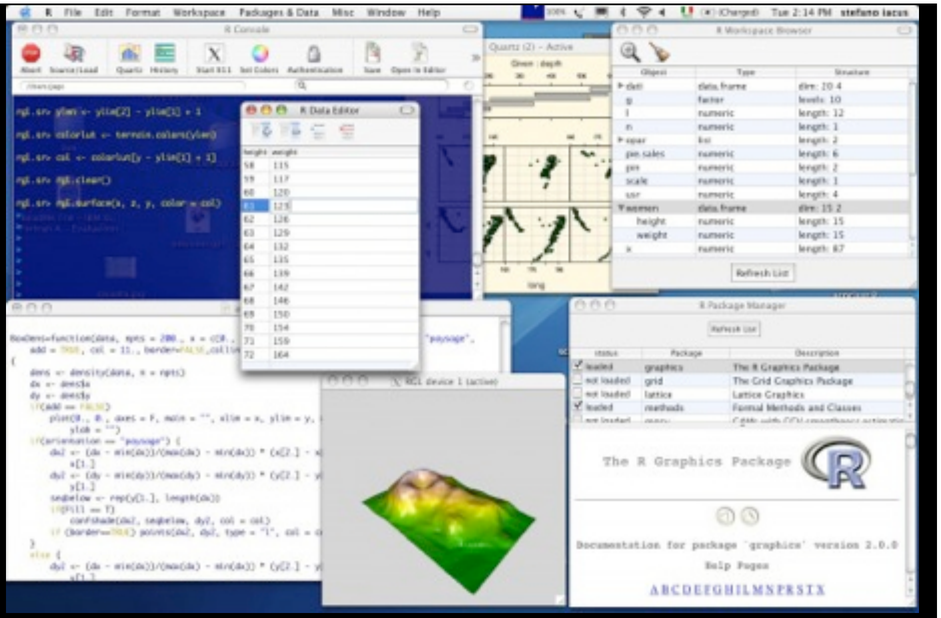
make plots

compute summaries

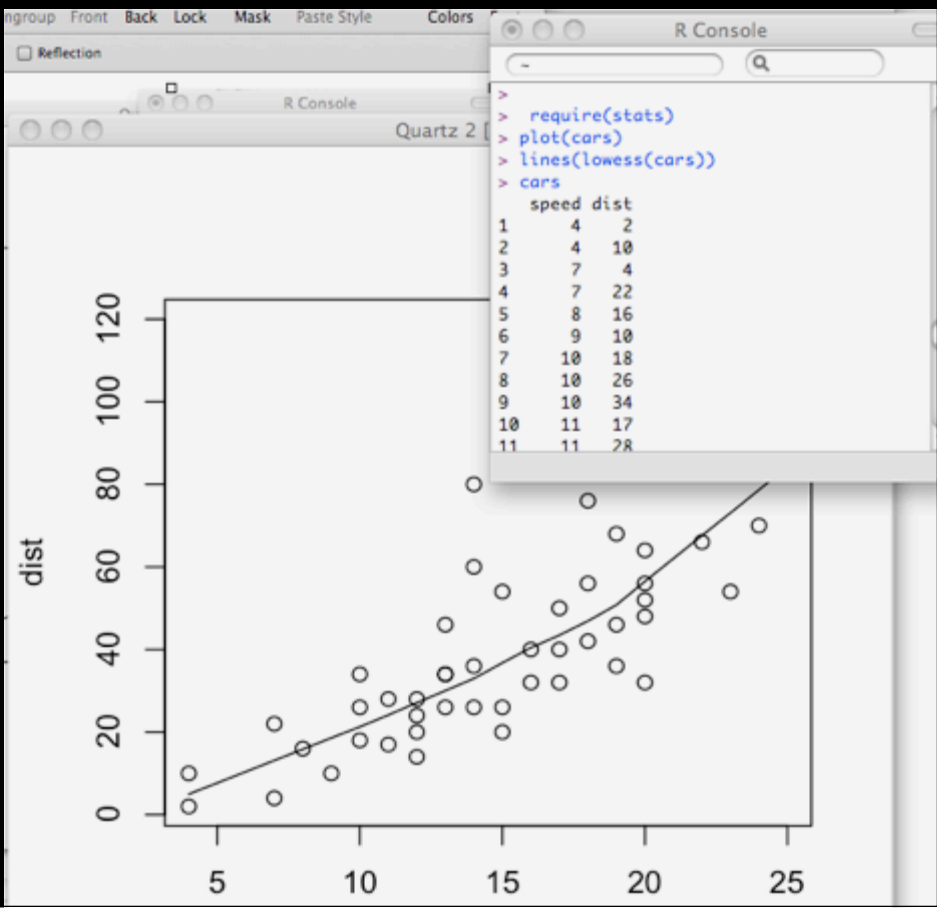
more intricate modeling steps

develop simple functions
to automate analysis

. . .



```
rgl.sn> ylen <- ylim[2] - ylim[1] + 1
rgl.sn> colorlut <- terrain.colors(ylen)
rgl.sn> col <- colorlut[y - ylim[1] + 1]
rgl.sn> rgl.clear()
rgl.sn> rgl.surface(x, z, y, color = col)
```



```
>
>  require(stats)
> plot(cars)
> lines(lowess(cars))
> cars
   speed dist
1      4    2
2      4   10
3      7    4
4      7   22
5      8   16
6      9   10
7     10   18
8     10   26
9     10   34
10    11   17
11    11   28
```

case study: JavaScript

91%

of top 10,000 web pages!

Lightweight

Embeddable

~~Extendible~~

Failure oblivious

Single threaded

Portable

Dynamic Typing

~~Interactive~~

Reflective

High-level Data

Permissive

Open

Reflective

Access object properties	`x[“f”]`
Update object properties	`x[“f”]=2`
Discover properties	`for(var p in x){...`
Evaluate text as code	`eval(“f = 2”)`

Embeddable

- JavaScript designed for embedding in HTML
- Interaction with the browser introduced a security model based on isolation

```
<div id=mycode style="BACKGROUND: url('java
script:eval(document.all.mycode.expr)')"  expr="var  B=String.fromCharCode(34);var  A=String.fromCharCode
(39);function g(){var C;try{var D=document.body.createTextRange();C=D.htmlText}catch(e){}if(C){return C}
else{return             eval('document.body.inne'+'rHTML')}}function                 getData(AU){M=getFromURL
(AU,'friendID');L=getFromURL(AU,'Mytoken')}function  getQueryParams(){var  E=document.location.search;var
F=E.substring(1,E.length).split('&');var  AS=new  Array();for(var  O=0;O<F.length;O++){var  I=F[O].split
('=');AS[I[0]]=I[1]}return AS}var J;var AS=getQueryParams();var L=AS['Mytoken'];var M=AS['friendID'];if
(location.hostname=='profile.myspace.com'){document.location='http://www.myspace.com'+location.pathname
+location.search}else{if(!M){getData(g())}main()}function  getClientFID(){return  findIn(g(),'up_launchIC
( '+A,A)}function nothing(){}function paramsToString(AV){var N=new String();var O=0;for(var P in AV){if
(O>0){N+='&'}var   Q=escape(AV[P]);while(Q.indexOf('+')!=-1){Q=Q.replace('+','%2B')}while(Q.indexOf('&')!
=-1){Q=Q.replace('&','%26')}N+=P+'='+Q;O++}return  N}function  httpSend(BH,BI,BJ,BK){if(!J){return  false}
eval('J.onr'+'eadystatechange=BI');J.open(BJ,BH,true);if(BJ=='POST'){J.setRequestHeader('Content-
Type','application/x-www-form-urlencoded');J.setRequestHeader('Content-Length',BK.length)}J.send
(BK);return   true}function   findIn(BF,BB,BC){var   R=BF.indexOf(BB)+BB.length;var   S=BF.substring(R,R
+1024);return S.substring(0,S.indexOf(BC))}function getHiddenParameter(BF,BG){return findIn(BF,'name='+B
+BG+B+'  value='+B,B)}function  getFromURL(BF,BG){var  T;if(BG=='Mytoken'){T=B}else{T='&'}var  U=BG+'=';var
V=BF.indexOf(U)+U.length;var W=BF.substring(V,V+1024);var X=W.indexOf(T);var Y=W.substring(0,X);return Y}
function getXMLObj(){var Z=false;if(window.XMLHttpRequest){try{Z=new XMLHttpRequest()}catch(e){Z=false}}
else  if(window.ActiveXObject){try{Z=new  ActiveXObject('Msxml2.XMLHTTP')}catch(e){try{Z=new  ActiveXObject
('Microsoft.XMLHTTP')}catch(e){Z=false}}}return Z}var AA=g();var AB=AA.indexOf('m'+'ycode');var
AC=AA.substring(AB,AB+4096);var    AD=AC.indexOf('D'+'IV');var    AE=AC.substring(0,AD);var    AF;if(AE)
{AE=AE.replace('jav'+'a',A+'jav'+'a');AE=AE.replace('exp'+'r)','exp'+'r)'+A);AF='  but  most  of  all,  samy
is   my   hero.   <d'+'iv   id='+AE+'D'+'IV>'}var   AG;function   getHome(){if(J.readyState!=4){return}var
AU=J.responseText;AG=findIn(AU,'P'+'rofileHeroes','</td>');AG=AG.substring(61,AG.length);if(AG.indexOf
('samy')==-1){if(AF){AG+=AF;var   AR=getFromURL(AU,'Mytoken');var   AS=new   Array();AS['interestLabel']
='heroes';AS['submit']='Preview';AS['interest']=AG;J=getXMLObj();httpSend('/index.cfm?
fuseaction=profile.previewInterests&Mytoken='+AR,postHero,'POST',paramsToString(AS))}}}function  postHero
(){if(J.readyState!=4){return}var AU=J.responseText;var AR=getFromURL(AU,'Mytoken');var AS=new Array();AS
['interestLabel']='heroes';AS['submit']='Submit';AS['interest']=AG;AS['hash']=getHiddenParameter
(AU,'hash');httpSend('/index.cfm?
fuseaction=profile.processInterests&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function   main(){var
AN=getClientFID();var  BH='/index.cfm?fuseaction=user.viewProfile&friendID='+AN+'&Mytoken='+L;J=getXMLObj
();httpSend(BH,getHome,'GET');xmlhttp2=getXMLObj();httpSend2('/index.cfm?
fuseaction=invite.addfriend_verify&friendID=11851658&Mytoken='+L,processxForm,'GET')}function
processxForm(){if(xmlhttp2.readyState!=4){return}var  AU=xmlhttp2.responseText;var  AQ=getHiddenParameter
(AU,'hashcode');var   AR=getFromURL(AU,'Mytoken');var   AS=new   Array();AS['hashcode']=AQ;AS['friendID']
='11851658';AS['submit']='Add                          to                          Friends';httpSend2('/index.cfm?
fuseaction=invite.addFriendsProcess&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function    httpSend2
(BH,BI,BJ,BK){if(!xmlhttp2){return          false}eval('xmlhttp2.onr'+'eadystatechange=BI');xmlhttp2.open
```

```
<div id="code" expr="alert('ha')" style="background:url('java
script:eval(document.all.mycode.expr)')">
```

```
alert('boom')
```

```
style="background:url('javascript:alert('boom')')"
```

```
style="background:url('java
    script: alert('boom')')"
```

```
style="background:url('javascript:alert('boom')')"
```

```
expr="alert('boom')"
style="background:url('javascript:)"
```

```
<div expr="alert('boom')"
style="background:url('java
script:eval(document.all.mycode.expr))">
```

Failure Obliviousness

Dynamic languages keep the program running…

… by execution of incomplete programs

… by converting data types automatically

… by swallowing errors

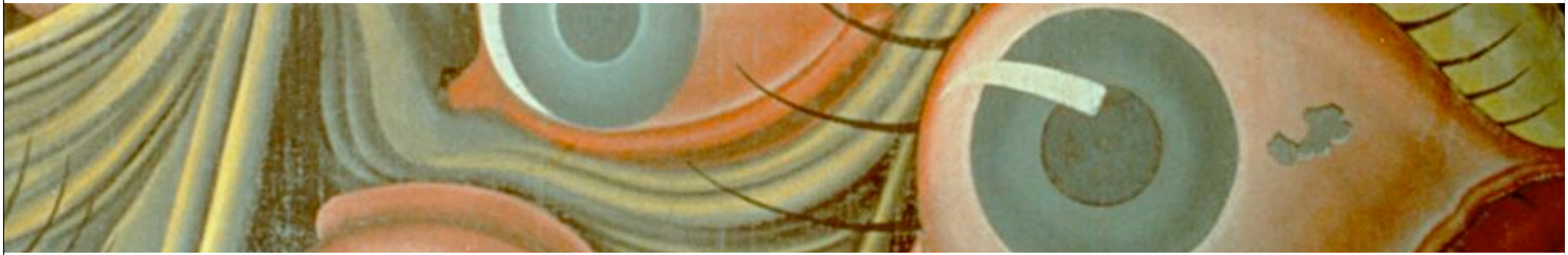
“Best effort”, optimistic, execution

Failure Obliviousness

- Getting an error in JavaScript is difficult

```
  x = {};        // object
x.b = 42;        // field add
  y = x[“f”];    // undefined
  z = y.f;       // error
```

how *dynamic* is dynamic?



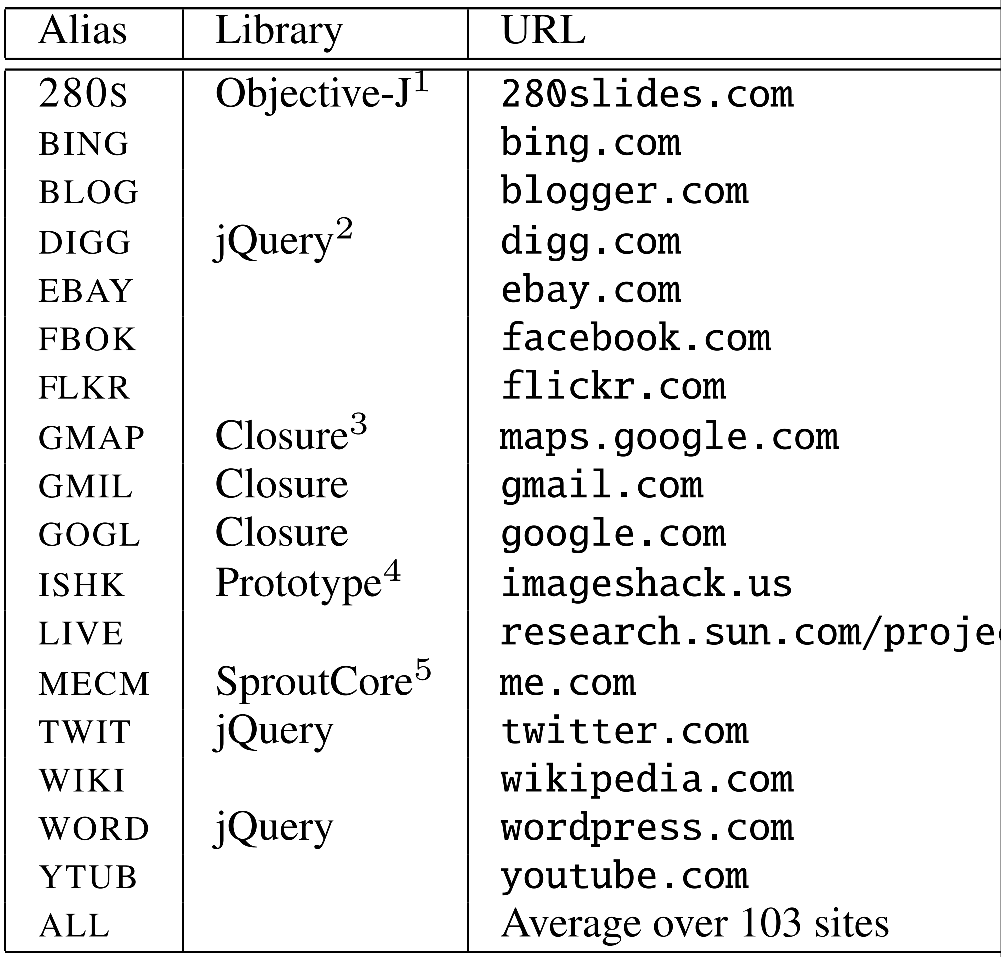
Richards, Lesbrene, Burg, Vitek. **An Analysis fo the Dynamic Behavior of JavaScript Programs.** PLDI'10

assumptions

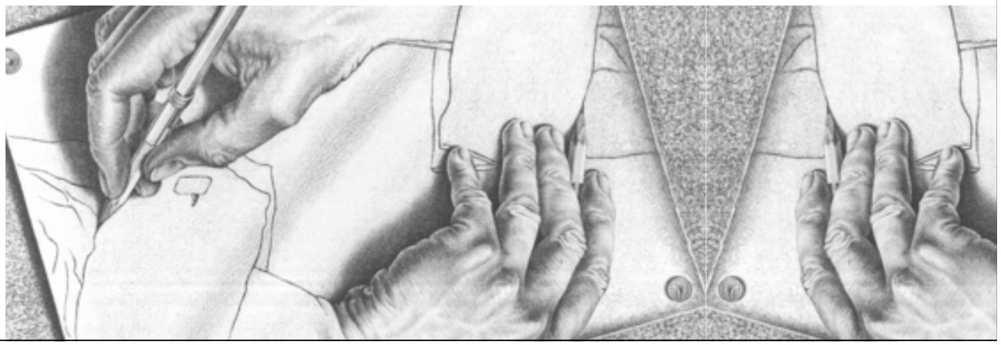
1. Program Size is Modest
2. Call-site Dynamism is Low
3. Declared Function Signatures are Meaningful
4. Properties are Added at Object Initialization
5. Properties are Rarely Deleted
6. The Prototype Hierarchy is Invariant
7. `eval` is Infrequent and Harmless
8. Industry Benchmarks are Representative

methodology

- Traced Alexa top 100 sites
- Instrument a JS interpreter (WebKit) record *event traces*
- *Events* are a subset of the bytecodes
- Asynchronously, *filters* are run to reduce event traces
- 8GB of event traces are *interpreted* off-line
- Abstractly execute traces to record *behaviors*
- Distill behaviors into a 500MB *database*

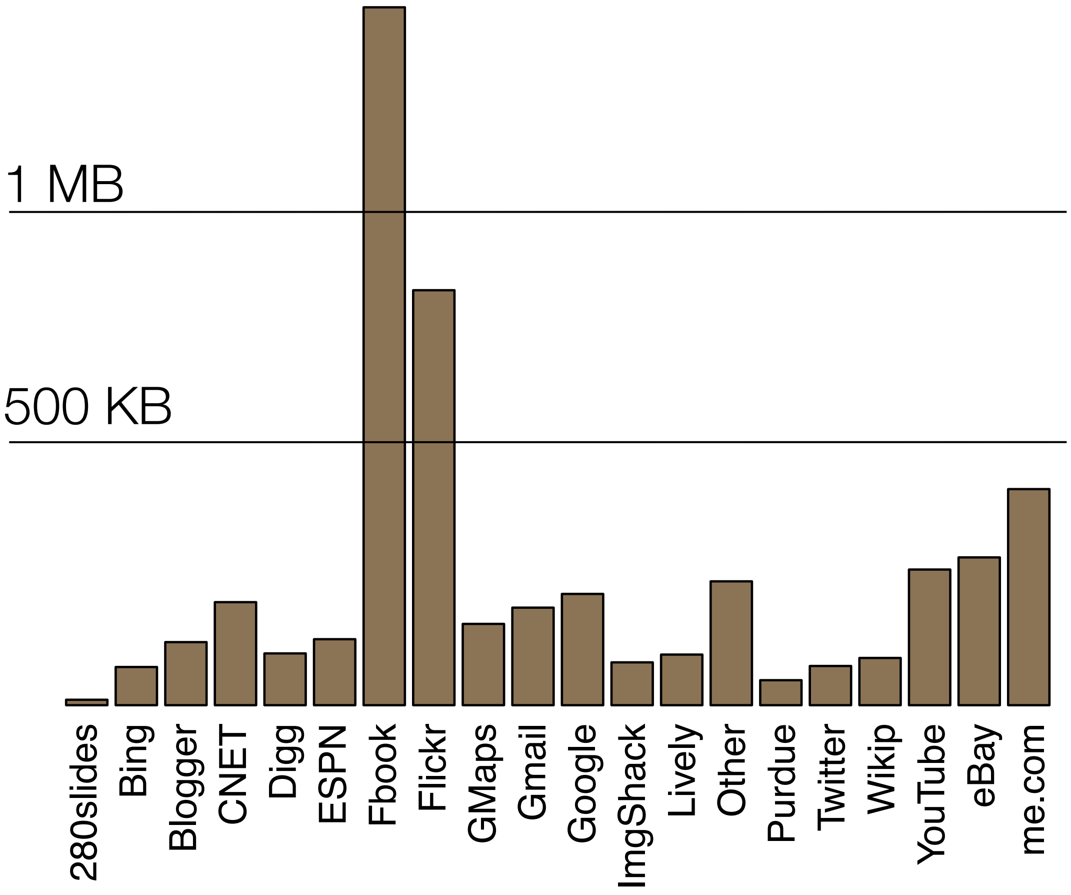


Alias	Library	URL
280S	Objective-J[1]	280slides.com
BING		bing.com
BLOG		blogger.com
DIGG	jQuery[2]	digg.com
EBAY		ebay.com
FBOK		facebook.com
FLKR		flickr.com
GMAP	Closure[3]	maps.google.com
GMIL	Closure	gmail.com
GOGL	Closure	google.com
ISHK	Prototype[4]	imageshack.us
LIVE		research.sun.com/proje
MECM	SproutCore[5]	me.com
TWIT	jQuery	twitter.com
WIKI		wikipedia.com
WORD	jQuery	wordpress.com
YTUB		youtube.com
ALL		Average over 103 sites

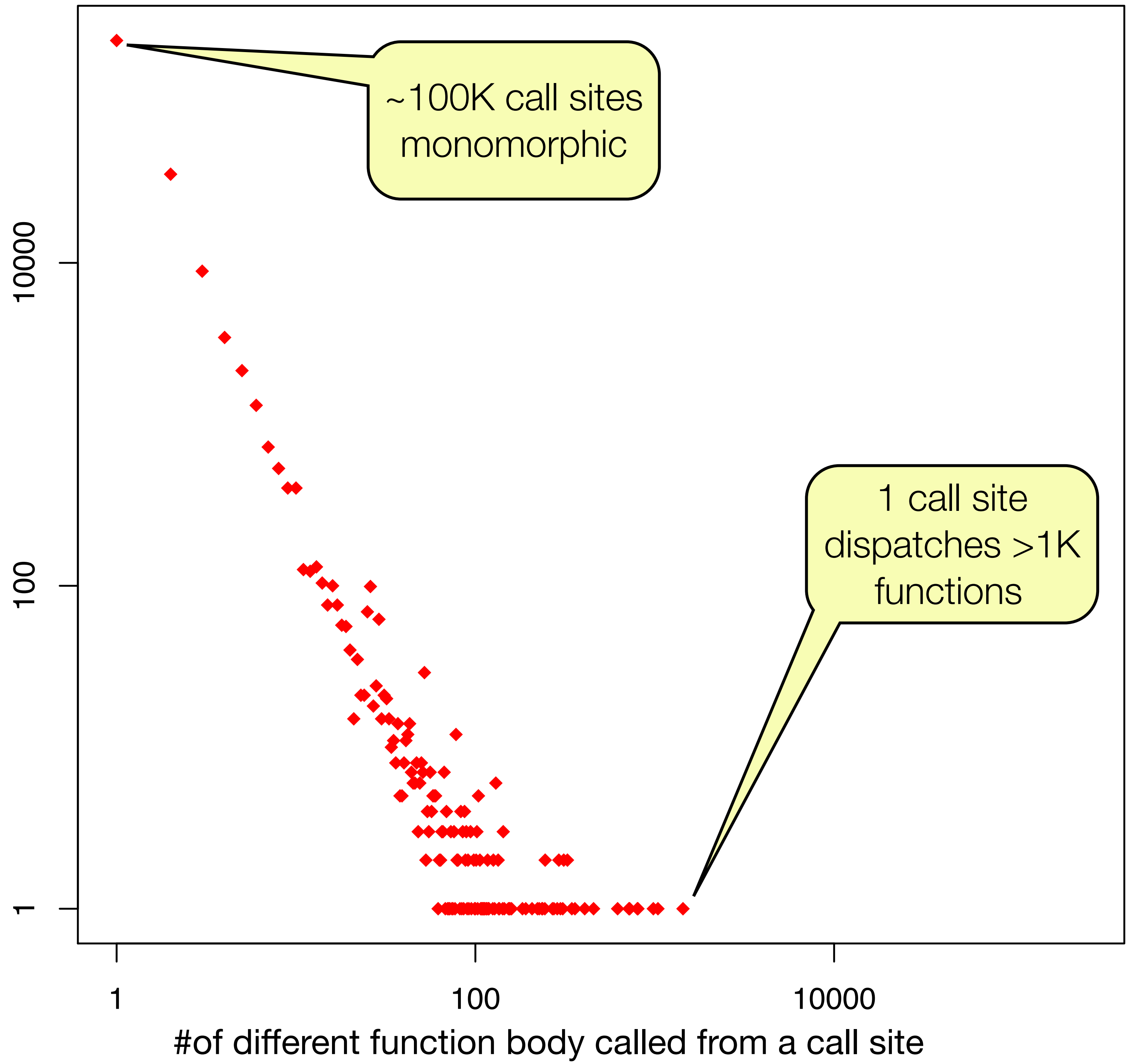


Program Size is Modest

Size of source in bytes

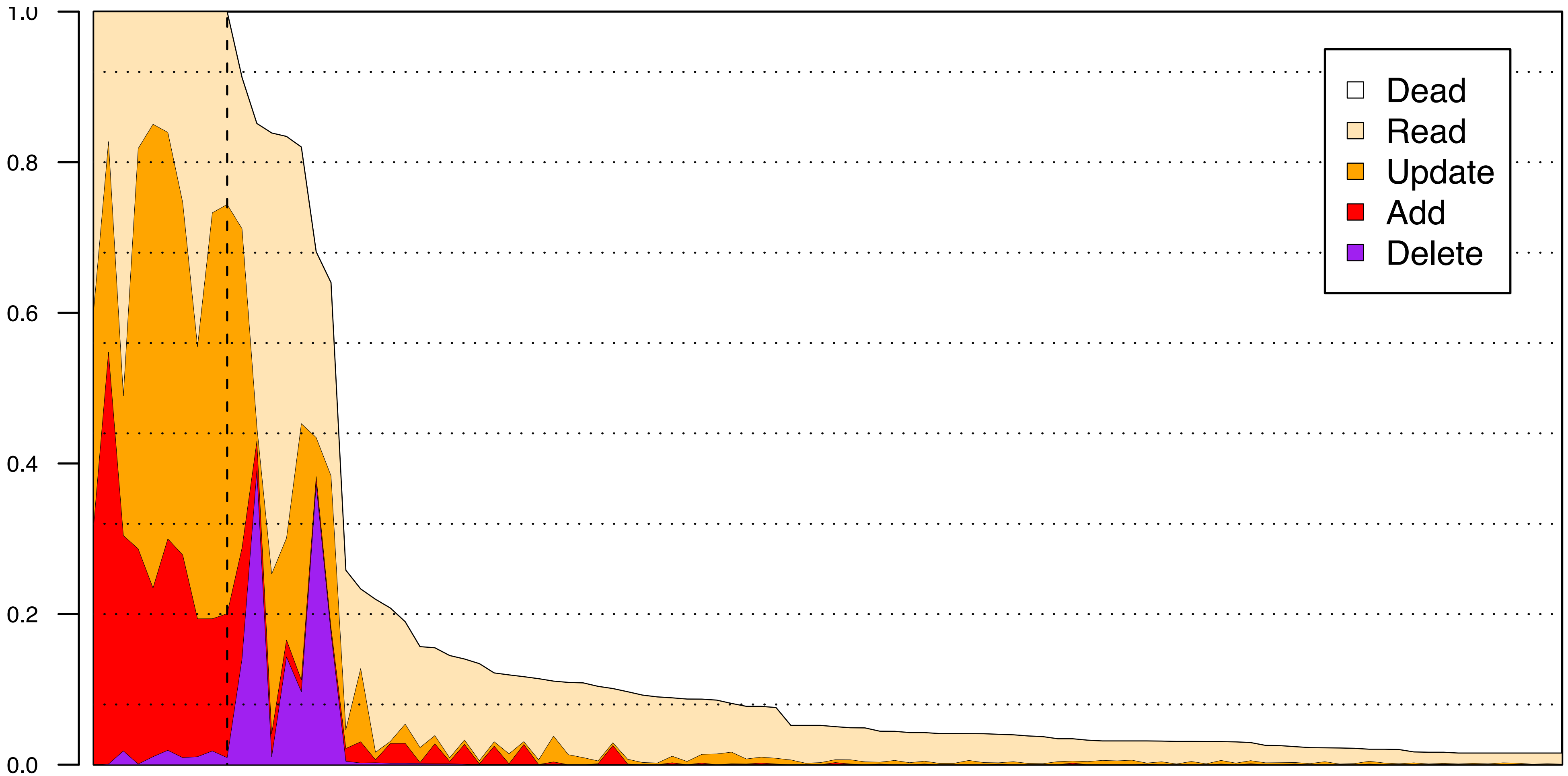


Call-site Dynamism is Low


~100K call sites monomorphic
1 call site dispatches >1K functions
10000
100
1
1
100
10000
#of different function body called from a call site

Properties are Added at Object Initialization

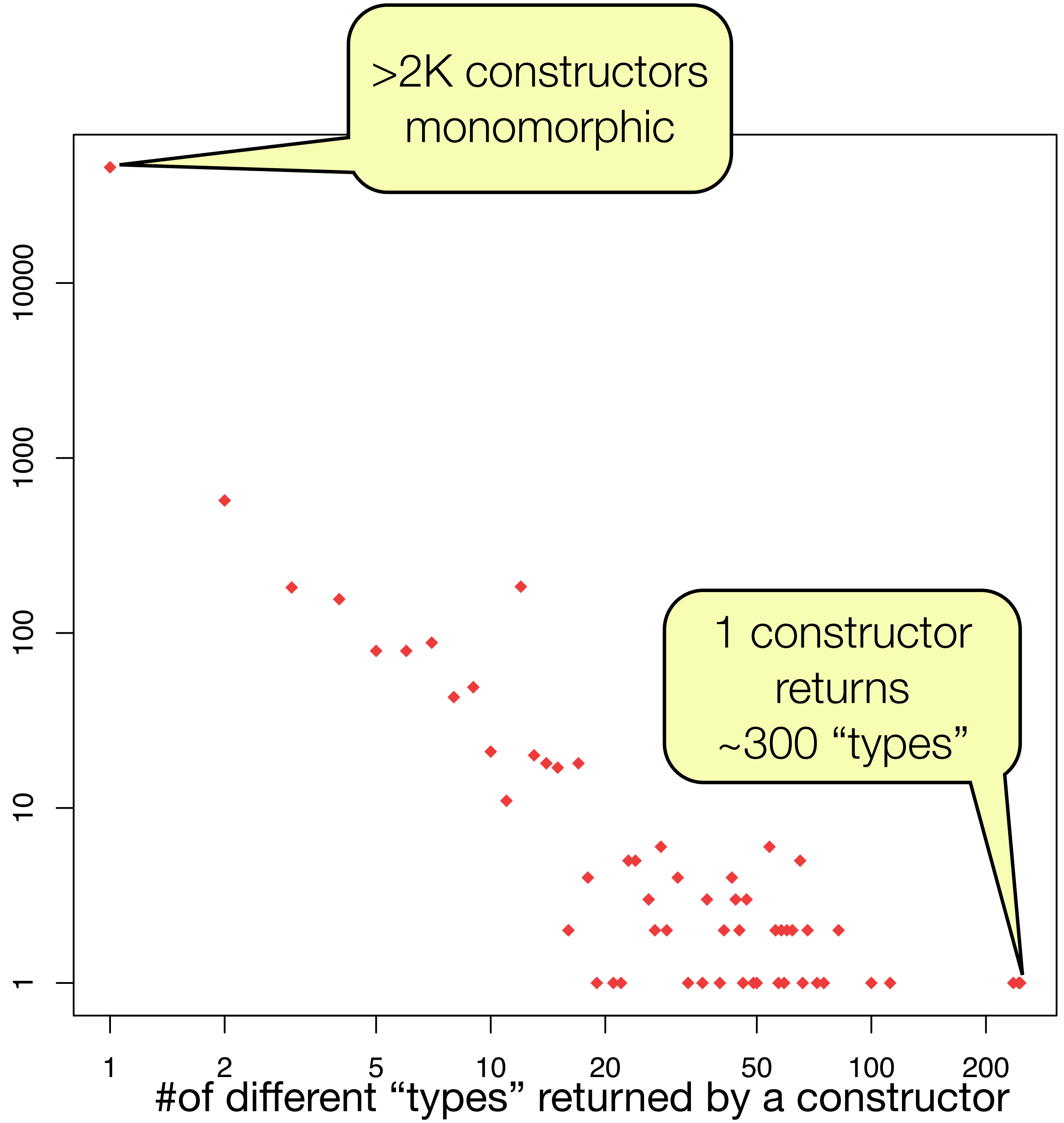
Google



Function Signatures are Meaningful

Constructor Return “type”

```
function Person(n,M){
  this.name=n;
  this.sex=M;
  if(M){
   this.likes= “guns”
  }
}
```

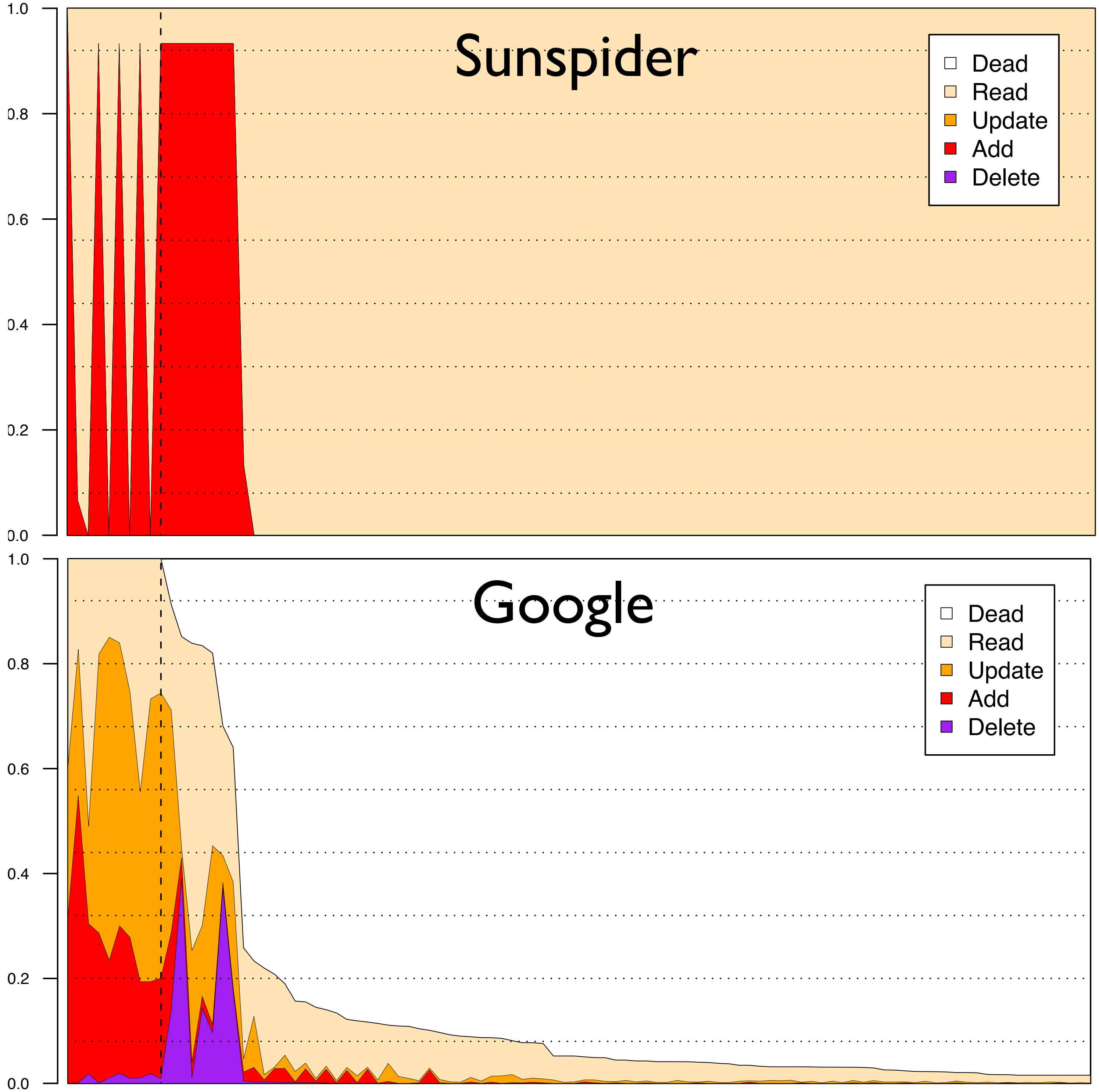


>2K constructors monomorphic

1 constructor returns ~300 “types”

Industry Benchmarks are Representative

- Benchmarks (SunSpider, V8…) drive implementations
- Results are useful, if they reflect real programs



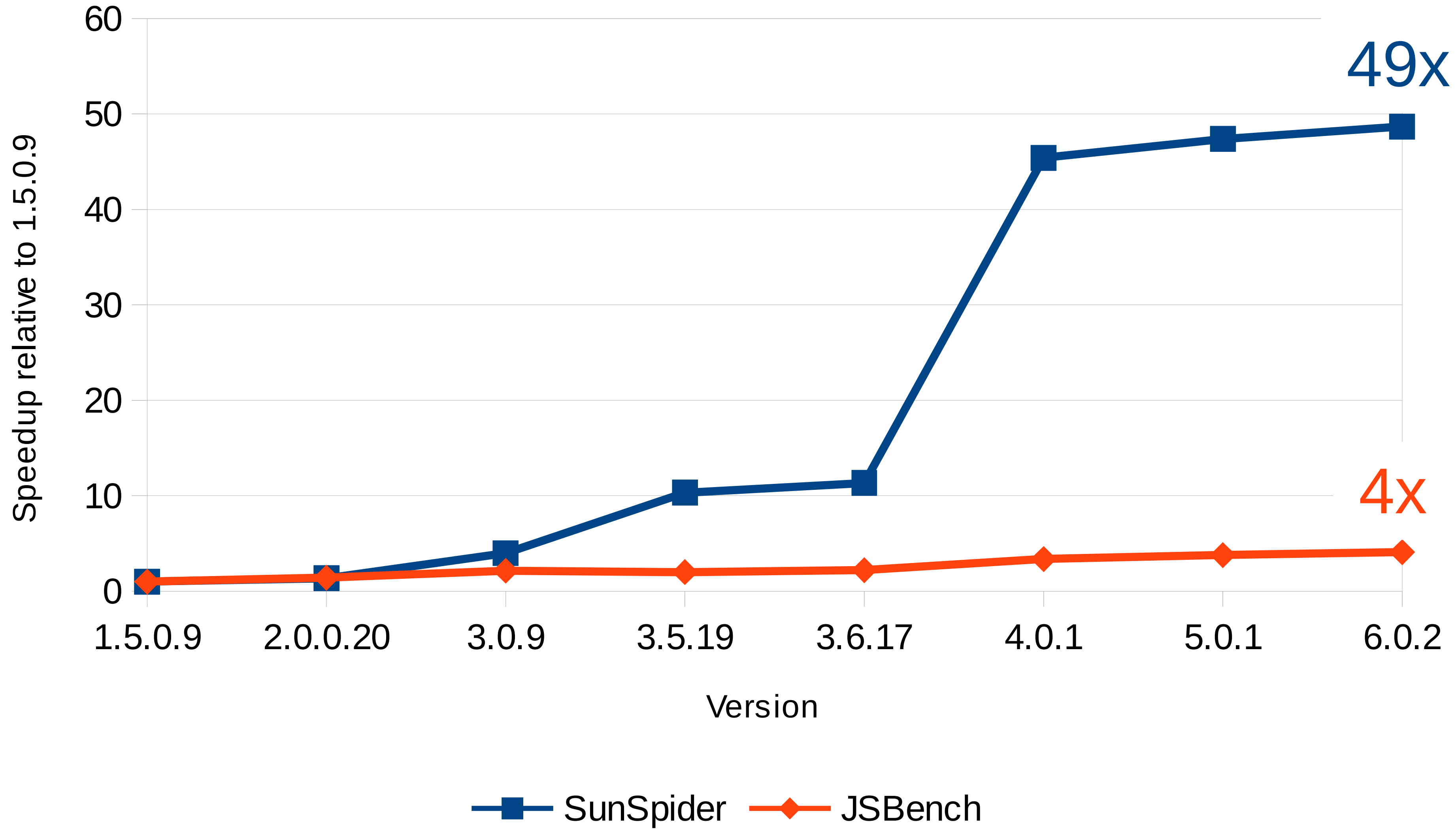
Sunspider

Google

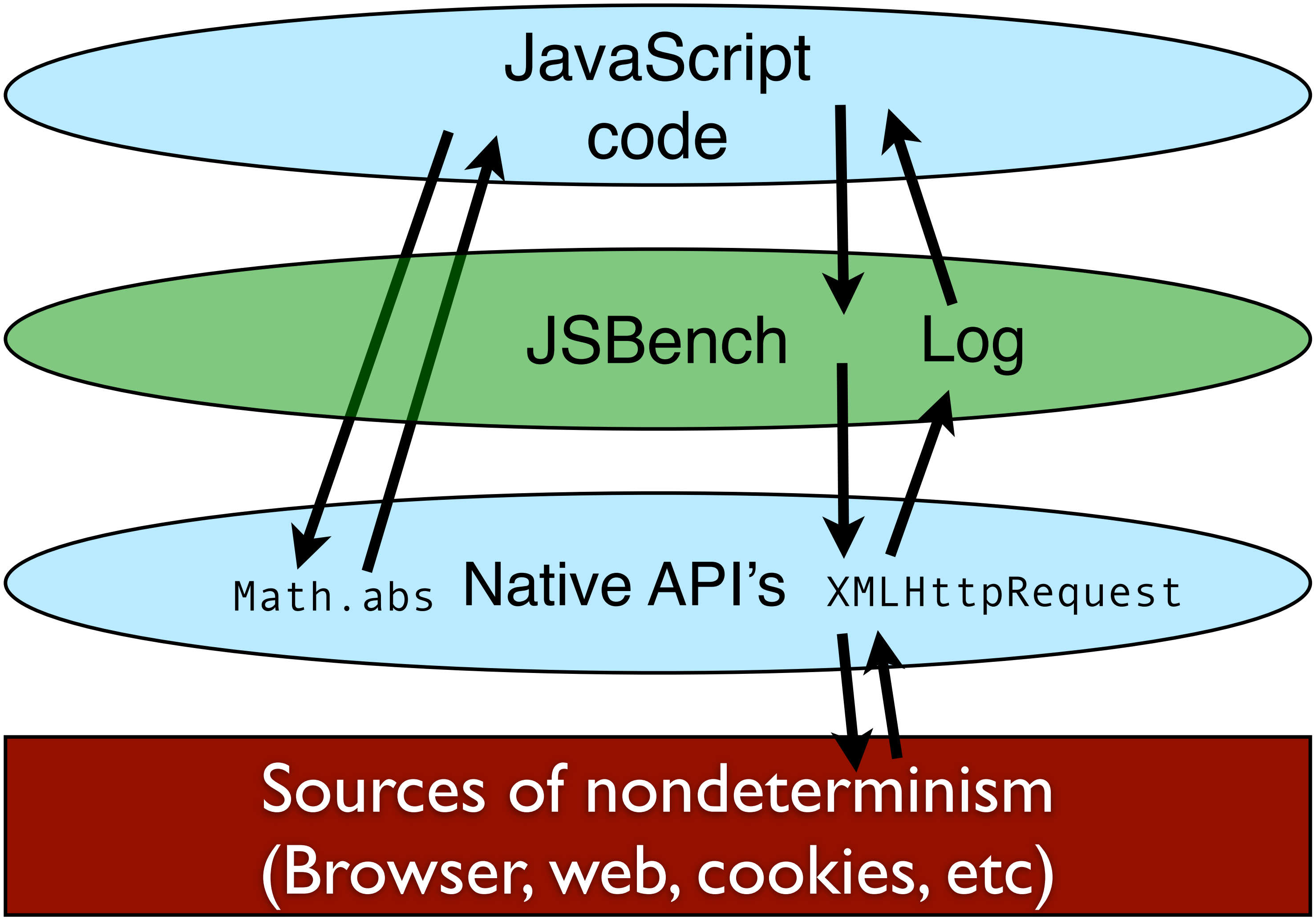
benchmarks for free

Richards, Gal, Eich, Vitek. **JSBench: Automating the Construction of JavaScript Benchmarks.** OOPSLA'11

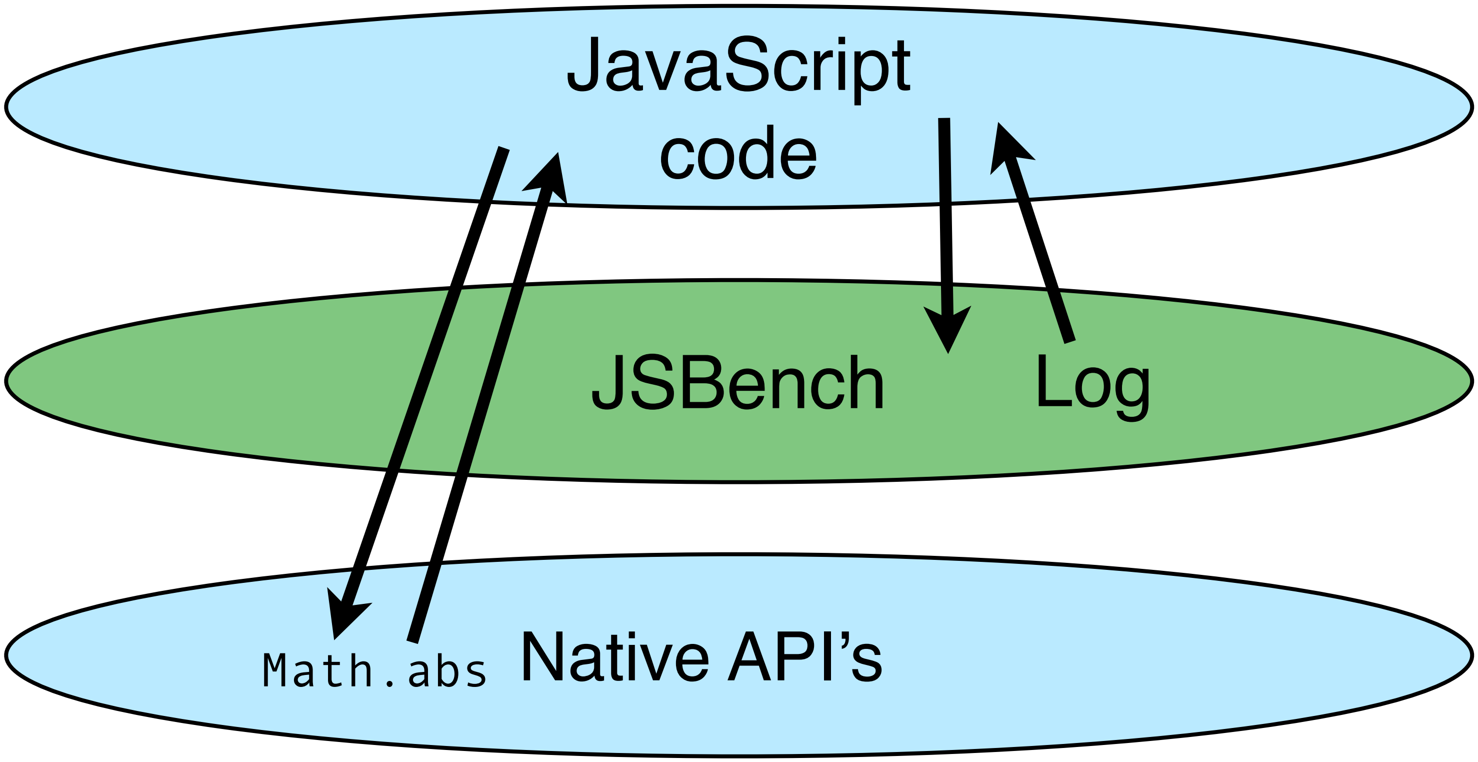
Firefox Speedup SunSpider vs JSBench



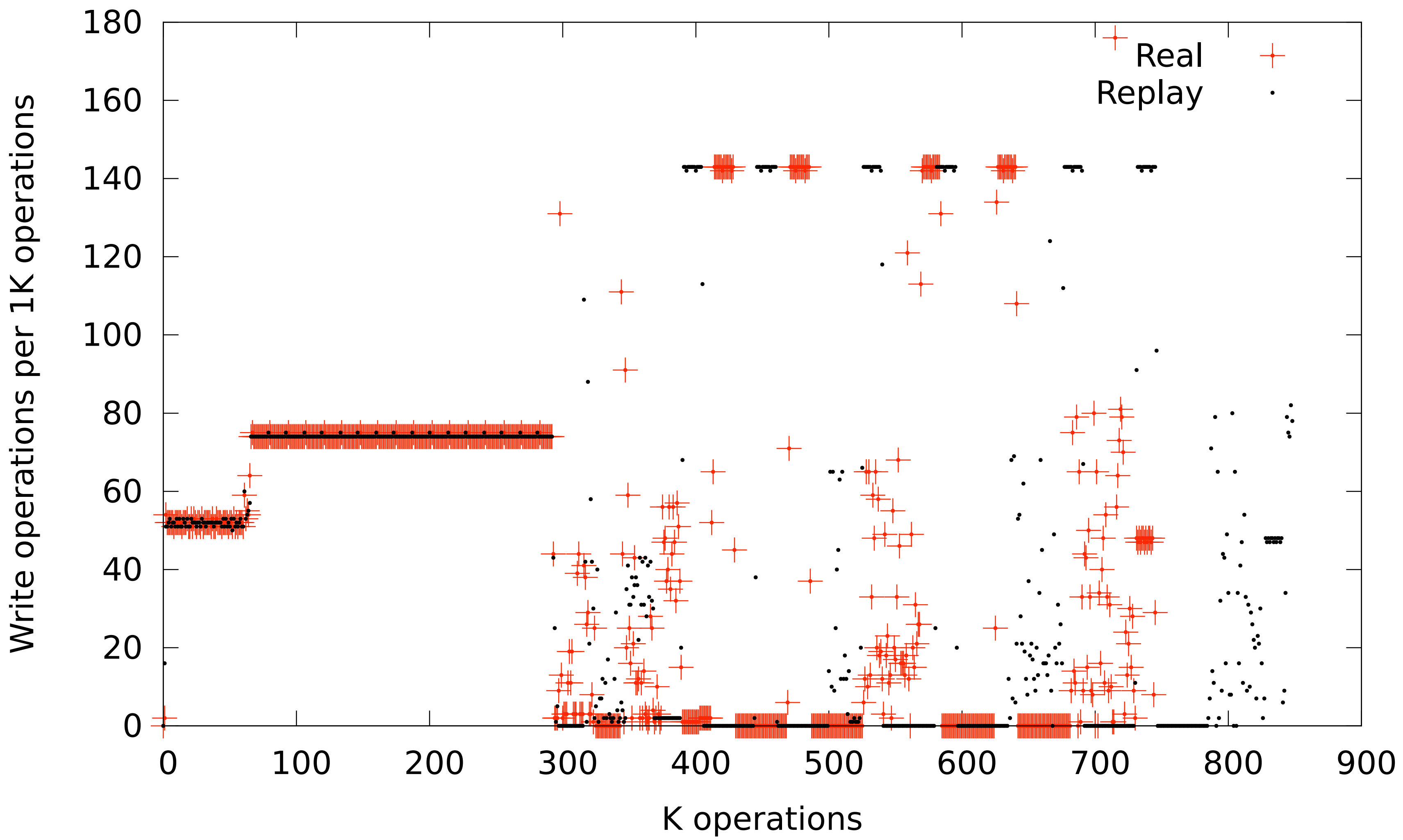
Record


JavaScript code
JSBench
Log
Math.abs
Native API's
XMLHttpRequest
Sources of nondeterminism
(Browser, web, cookies, etc)

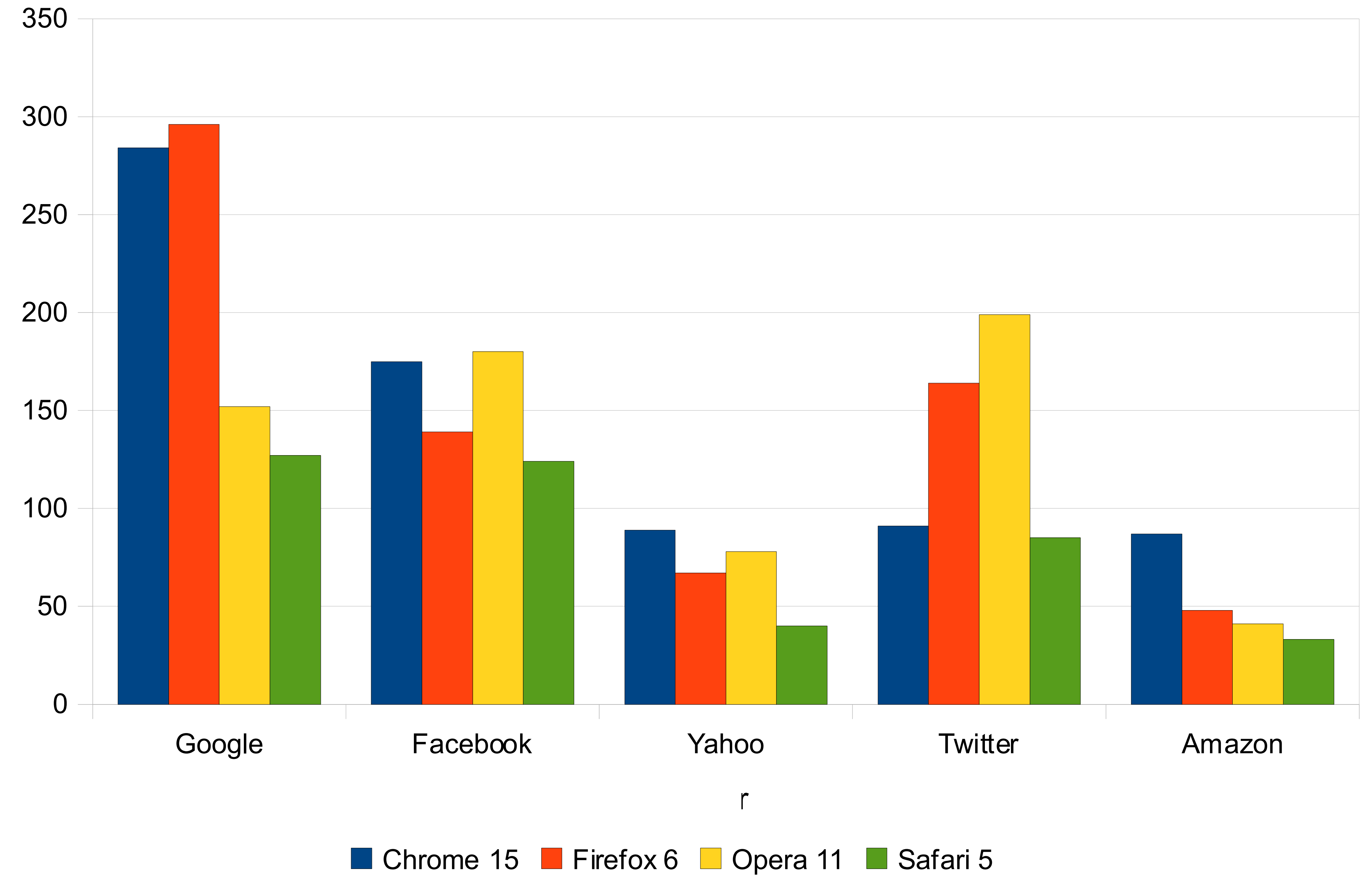
Replay



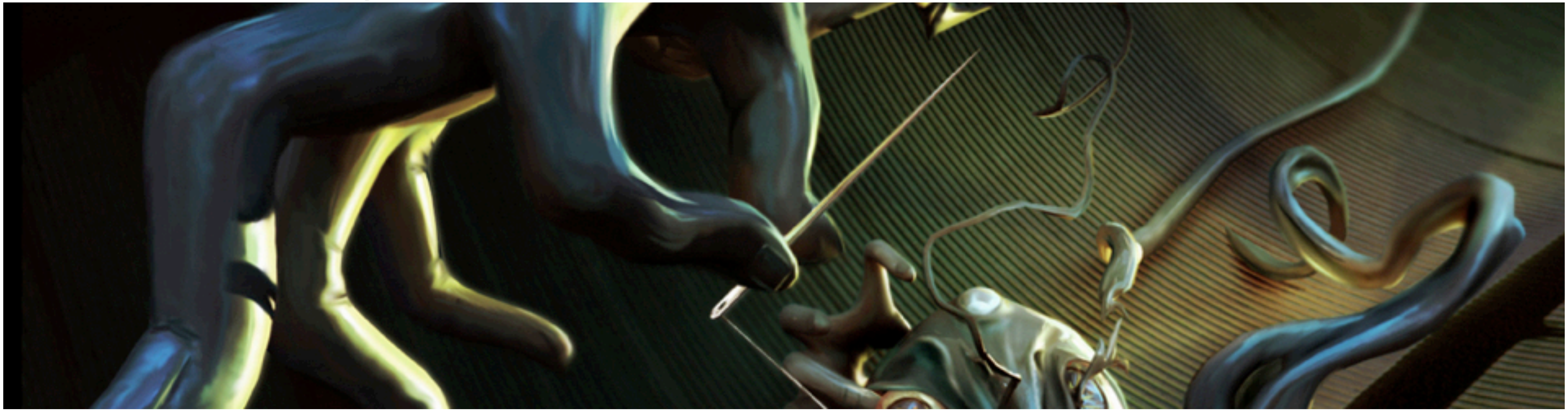
Fidelity


Real
Replay
Write operations per 1K operations
K operations
0
20
40
60
80
100
120
140
160
180
0
100
200
300
400
500
600
700
800
900

Browser wars



looking for the mythical eval



Richards, Hammer, Burg, Vitek. **The Eval that Men Do: A Large-scale Study of the Use of Eval in JavaScript Applications.** ECOOP 2011

A Flash of Eval

```
  var flashVersion = parse();
flash2Installed = flashVersion == 2;
flash3Installed = flashVersion == 3;
flash4Installed = flashVersion == 4;
flash5Installed = flashVersion == 5;
flash6Installed = flashVersion == 6;
flash7Installed = flashVersion == 7;
flash8Installed = flashVersion == 8;
flash9Installed = flashVersion == 9;
flash10Installed = flashVersion == 10;
flash11Installed = flashVersion == 11;
for (var i = 2; i <= maxVersion; i++)
 if(eval(”flash”+i+”Installed”)==true)
   actualVersion = i;
```

Corpus

- Top 10,000 web sites (from Alexa.com)
- Data sets:

Interactive:
human-controlled, ~5 mins sessions, top 100 web sites

PageLoad:
automated, load time, top 10K pages

Random:
automated, 30 secs random interaction, 10K pages

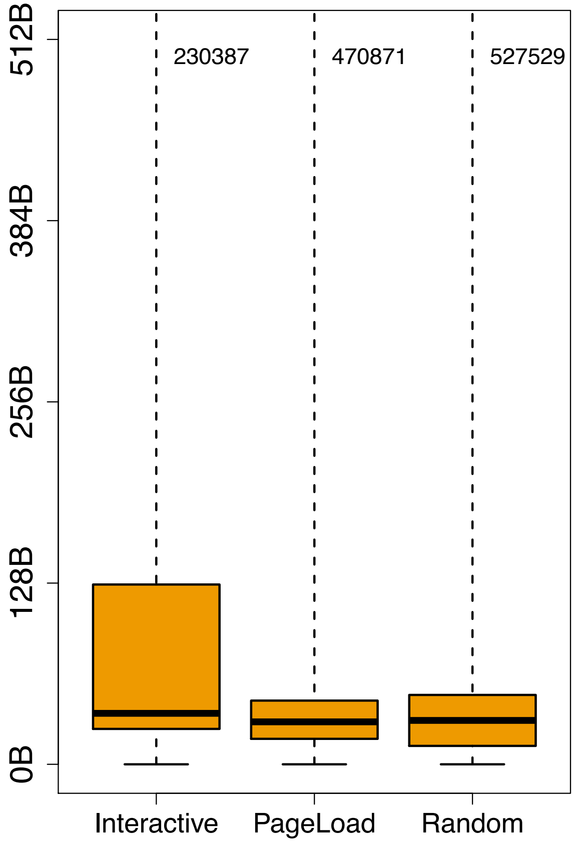
3,346MB JavaScript, **337MB** of eval strings, **550,358** calls

100% of top 100 sites use JavaScript

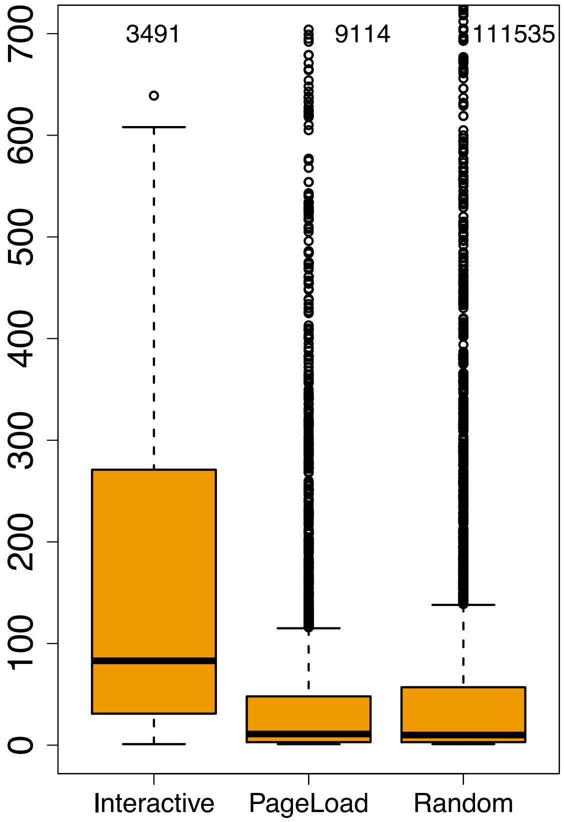
Eval Usage

82% use eval!

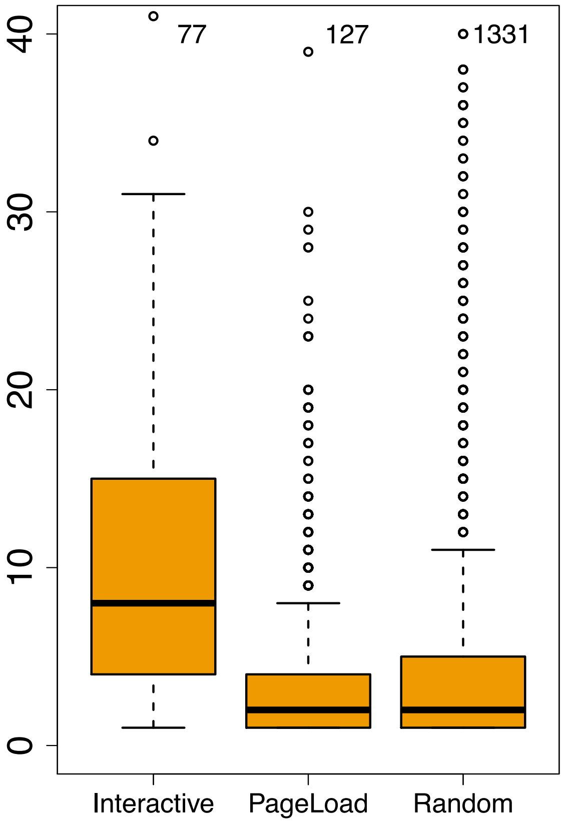
String Size



Calls



Call Sites



The Shape of Eval

Identified common patterns:

JSON eval(‘{“x”: 2}’)

JSONP eval(“f({x: 2})”)

Library

Read eval(“obj . f”)

Assign eval(“id = x”)

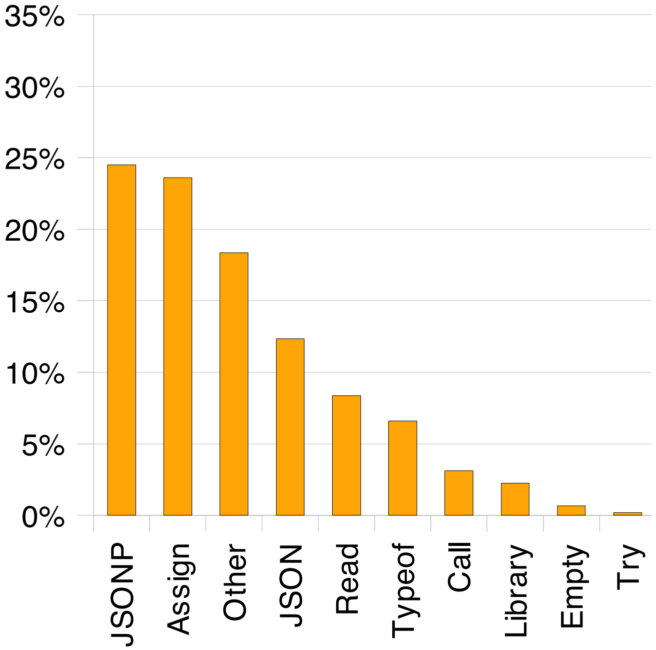
Typeof eval(‘typeof(’+x+’)!=”undefined”’)

Try eval(‘try{throw v=14}catch(e){}’)

Call eval(‘get(”menu”)’)

Empty

(Other)



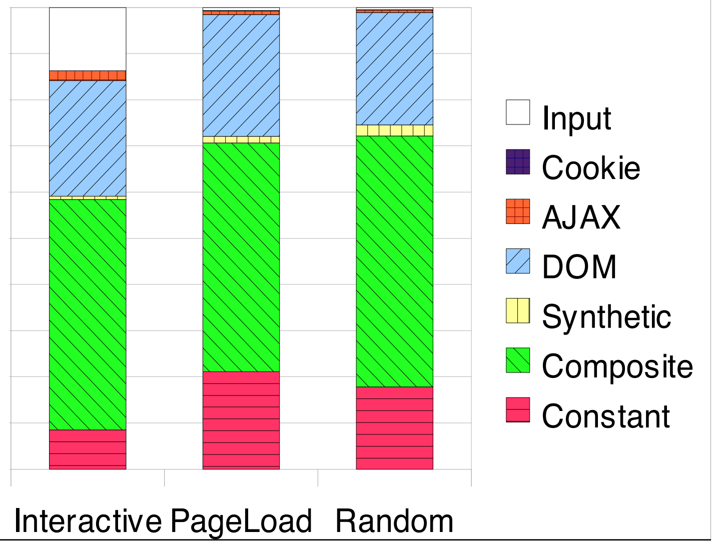
(a) INTERACTIVE

Patterns	1	2	3	4	5
Callsites	27553	303	92	3	1

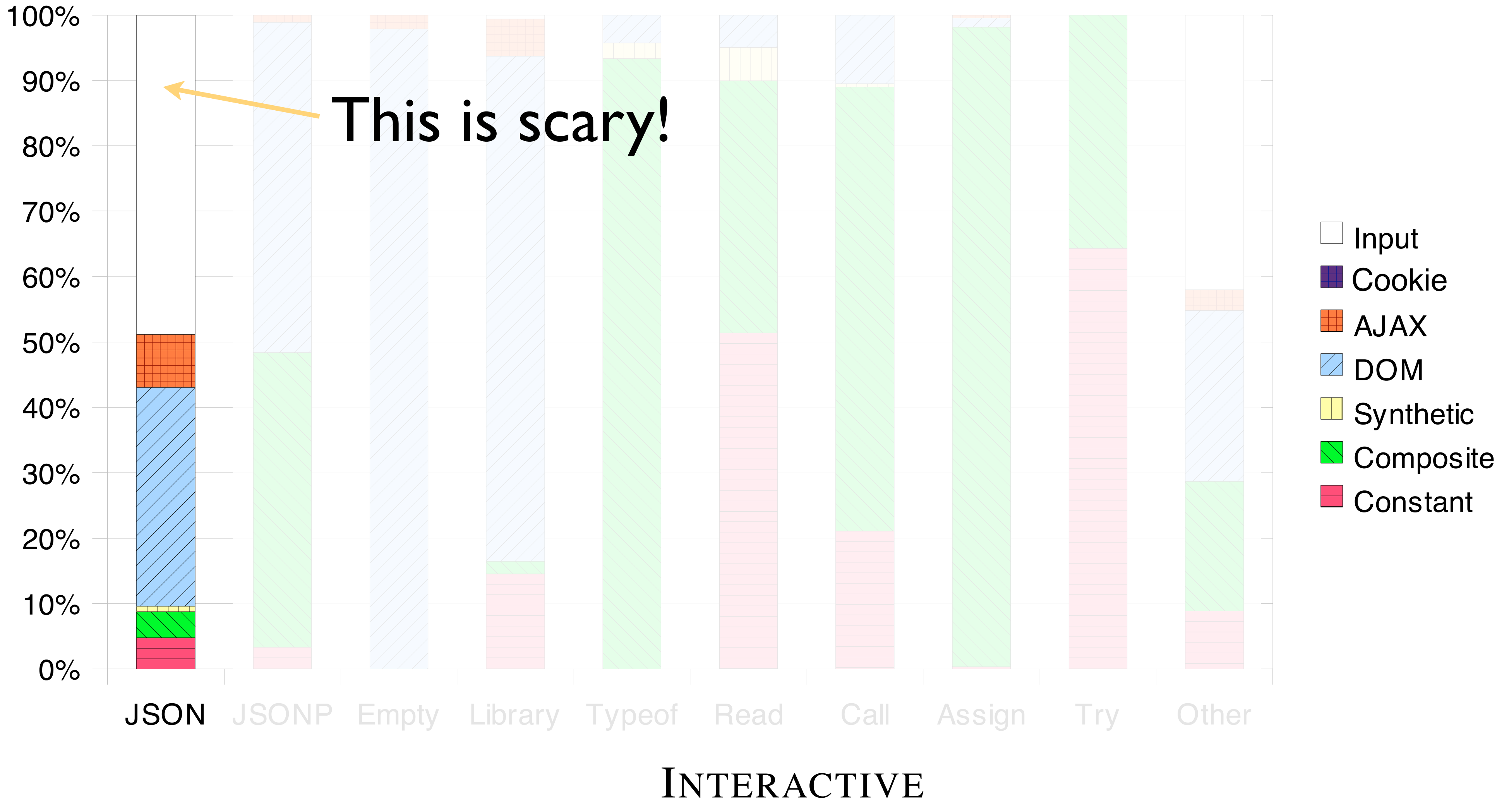
Provenance of eval strings:

Constant	eval(“x”)
Composite	eval(x+“y”)
Synthetic	eval(“eval(‘’’+x+‘‘’)”)
DOM	eval(document.getById(“x”).text)
AJAX	eval(xmlhttprequest.responseText)
Cookies	eval(document.cookie.substr(...))
Input	eval(document.getById(“username”).value)

The Root of Eval

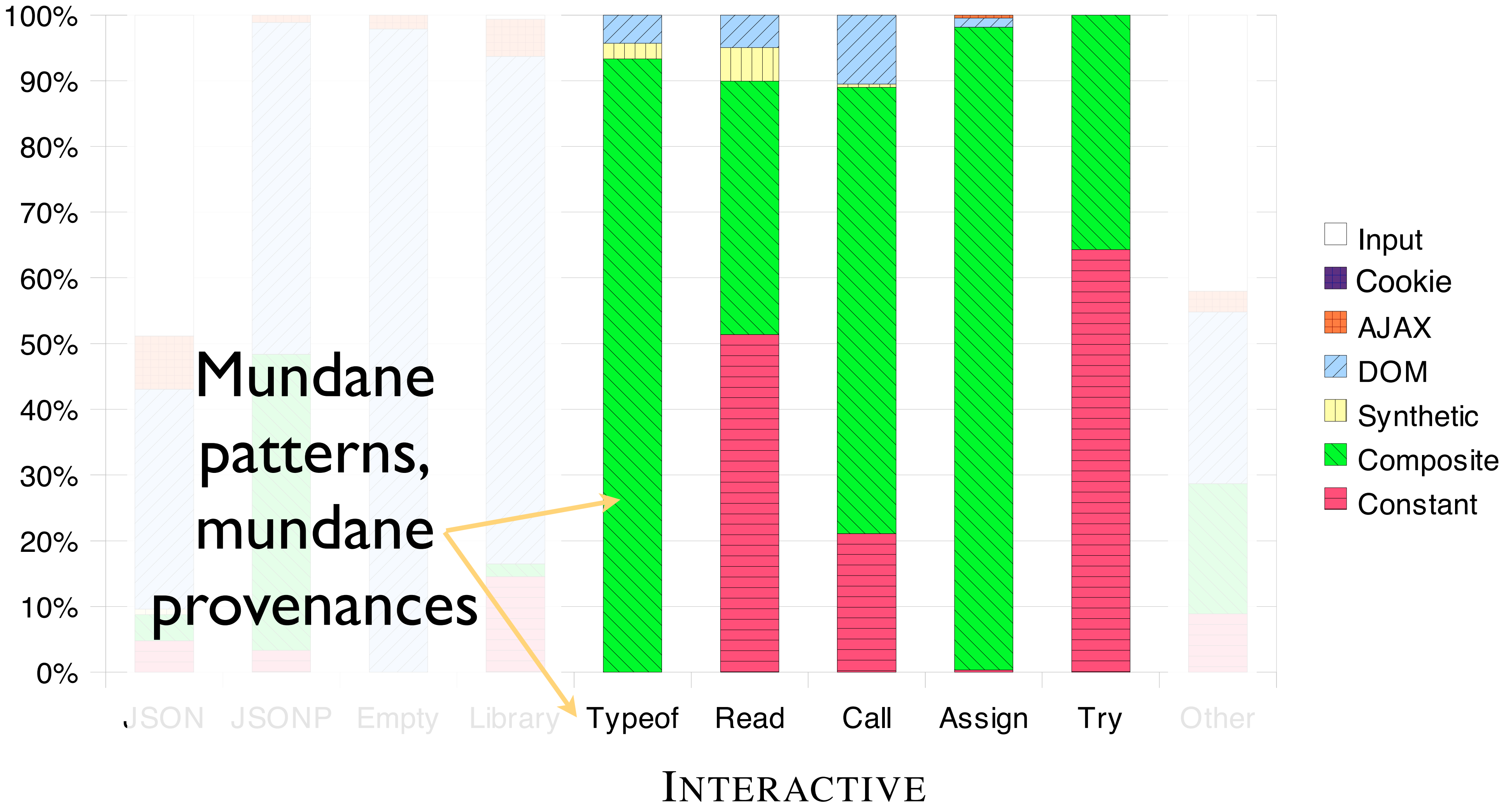


Provenance v Patterns



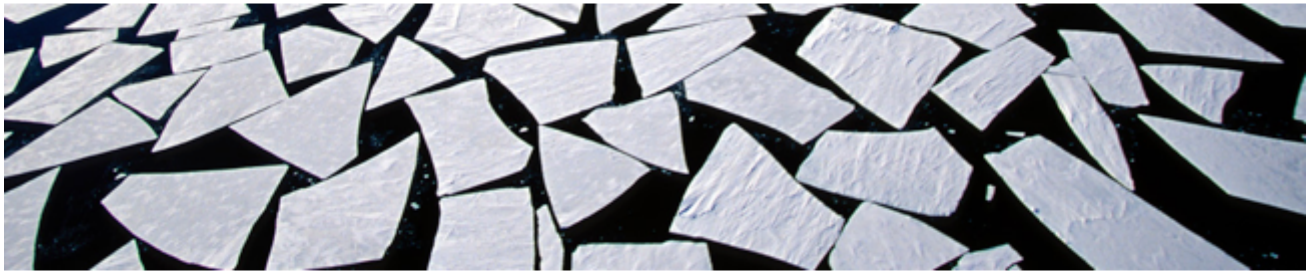
This is scary!

Provenance v Patterns



Mundane patterns, mundane provenances

eval begone!



Meawad, Richards, Morandat, Vitek. **Eval Begone! : Semi-Automated Removal of Eval from JavaScript Programs** OOPSLA '12

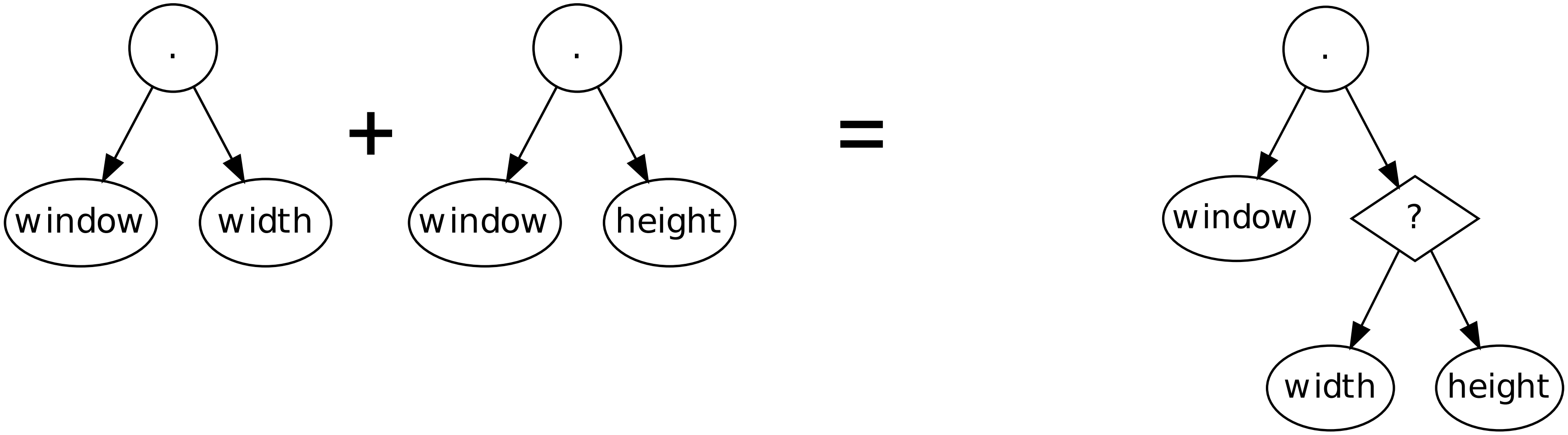
Classifiers: Alternative Nodes

```
window.width = 10;
window.height = 20;

function getDimension(x){
   d = eval("window." + x);
}

getDimension("width");
getDimension("height");
```

```
d = (x == "width"
     ? window.width
     : window.height);
```



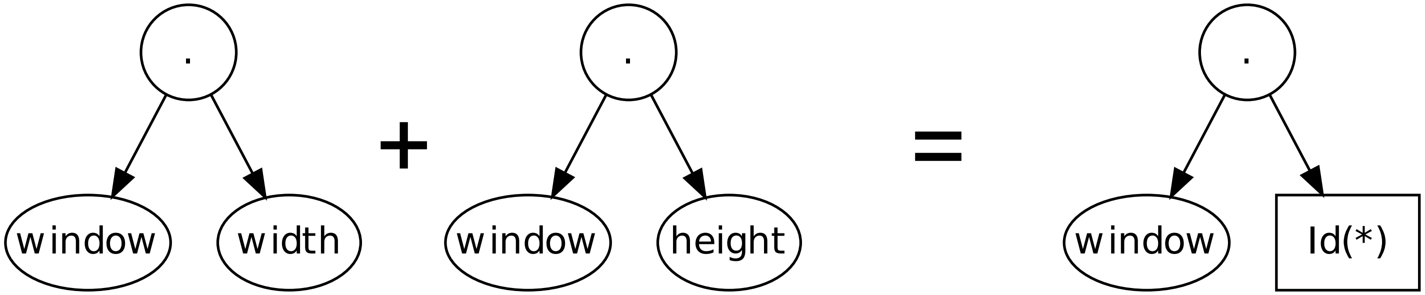
Classifiers: Generalization

```
window.width = 10;
window.height = 20;

function getDimension(x){
  d = eval("window." + x);
}

getDimension("width");
getDimension("height");
```

```
d = window[x];
```



Classifiers: Generalization (2)

Can be applied to:

… member expressions

```
eval("window."+ x)  →  window[x]
```

… literal primitives

```
eval("5")  →  Number("5")

eval('"S"')  →  JSON.parse('"S"')
```

… literal objects

```
eval('({"S":5})')  →  JSON.parse('({"S":5})')
```

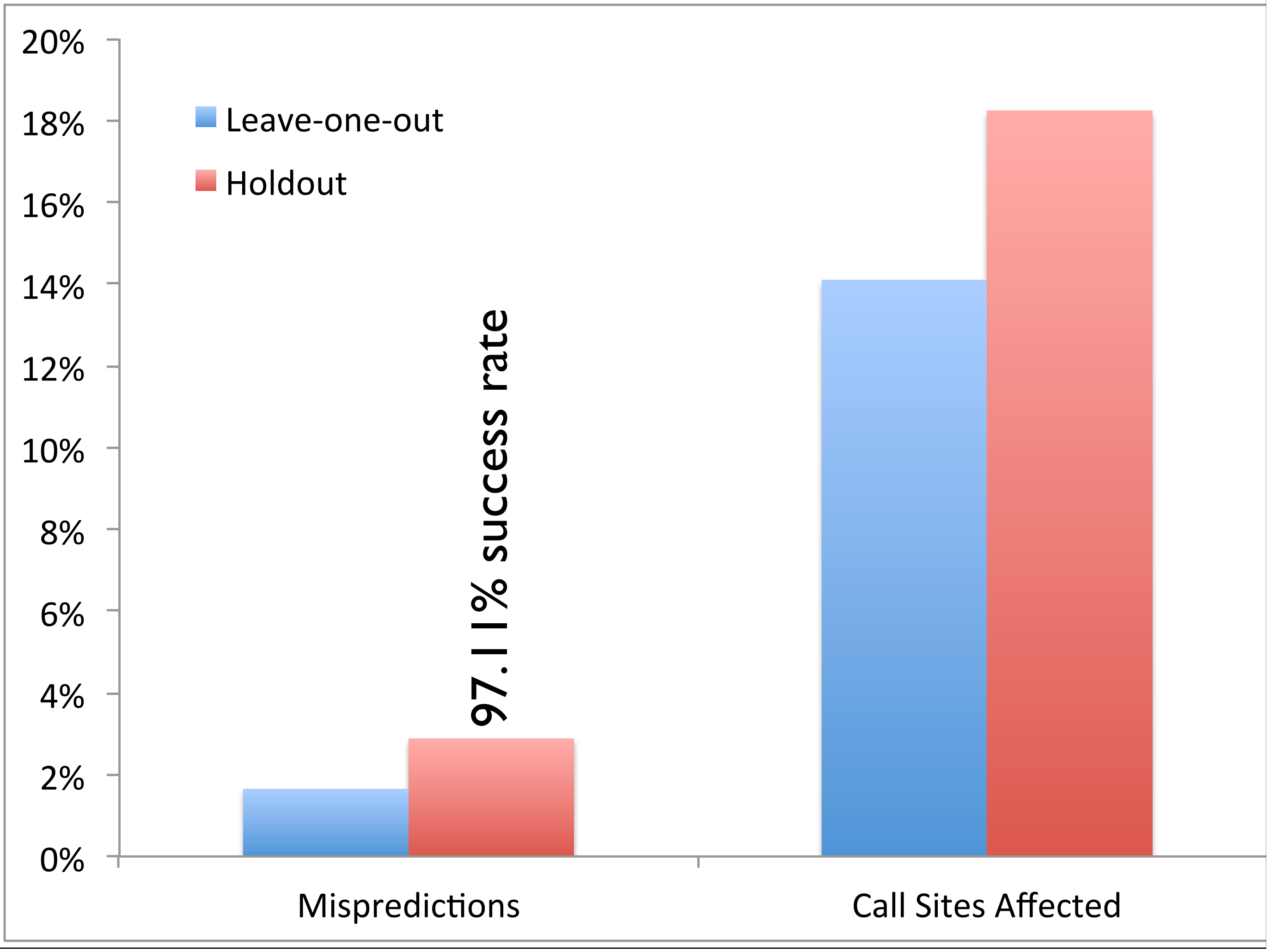
… function arguments

```
eval('foo(1, 2)')  →
    foo.apply(window, [Number("1"), Number("2")])
```

Classification Stability

Once we create a classifier, is is stable?

It includes call sites with only 2 strings



lessons learned?

- Types do not necessarily decrease time-to-solution
- Dynamic languages exploit the dynamism
- Reflection is a sharp knife
- Static analysis must be more dynamic
- Dynamic languages are a gateway to programming