ADsafety

Type-based Verification of JavaScript Sandboxing

Joe Gibbs Politz
Spiridon Aristides Eliopoulos
Arjun Guha
Shriram Krishnamurthi

Breaking News and Opinion on The Huffington Post

http://www.huffingtonpost.com/

Google

EDITION: U.S. | For Profit Colleges | Silvio Berlusconi | Bernie Sanders | Smarter Ideas | More | Log in

August 6, 2011

THE HUFFINGTON POST

THE INTERNET NEWSPAPER: NEWS BLOGS VIDEO COMMUNITY

Like | Follow | Search the Huffington Post | CONNECT

FRONT PAGE | POLITICS | BUSINESS | ENTERTAINMENT | TECH | MEDIA | LIFE & STYLE | ★CULTURE | COMEDY | HEALTHY LIVING | ★VOICES | LOCAL | MORE

FEATURED | ★BLACKVOICES | ★PARENTS | CELEBRITY | SPORTS | CRIME | MOVIES | GREEN | RELIGION | GOOD NEWS | TRAVEL | WEIRD NEWS | ARTS

FROM AP: Copter shot down, killing 30 US troops, 7 Afghans... 2 minutes ago

Enter email address | Get Alerts

OUTLOOK: NEGATIVE

U.S. Credit Downgrade Stokes Recession Fears

Go to "http://www.huffingtonpost.com/news/good-news/"

Breaking News and Opinion on The Huffington Post

http://www.huffingtonpost.com/

Google

EDITION: U.S.

For Profit C

Log in

5 of your friends are already here!

Clare Patterson Kate MG Josh Rosenbluh

Jennifer B. Beinart

Ian Wolfcat Atha

Invite your friends to HuffPost

Cancel

Add up to 40 of your friends by clicking on their pictures below.

Find Friends: Start Typing a Name

Filter Friends All Selected (0)

Aaron T Myers Brown	Adam Khalil Grinnell	Aditi Mehta Columbia
Aditya Dhandhania JWU Prov...	Aditya Pai Georgia ...	Adityarup Chakravorty Grinnell
Aheli Purkayastha Universi...	Akili Thomas Grinnell	Albert Varma UVA

Send HuffingtonPost Invitation

Cancel

Like Follow

CONNECT

FRONT PAGE POLITICS BU NG ★VOICES LOCAL MORE

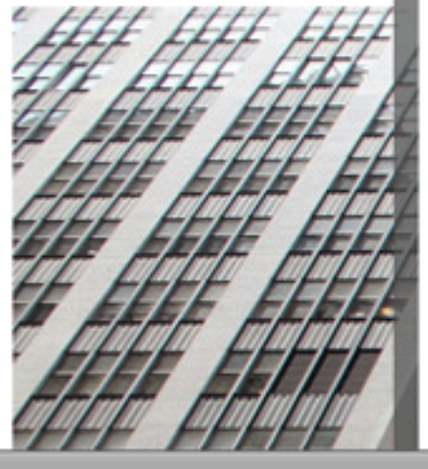
FEATURED ★BLACKVOICES WEIRD NEWS ARTS

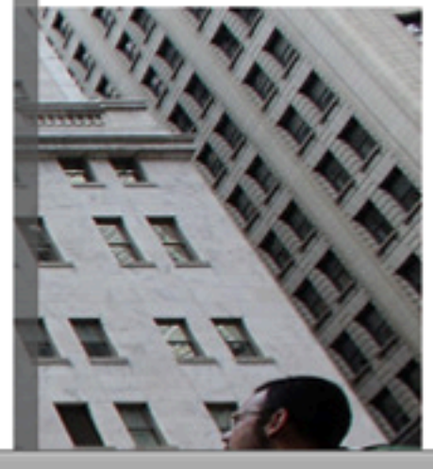
FROM AP:

Get Alerts

OUT TIVE

U.S rs





Will.i.am: Science Is Rockin' Roll and Technology Is Recession Proof

http://www.huffingtonpost.com/william/science-education_b_920125.html?ir=Entertai

Reader

Google

EDITION: U.S. | Megan Fox | Music | Movies | Mad Men | Smarter Ideas | More

Arjun Guha | Sign Out

third-party ad

OST ENTERTAINMENT

News | Web

Search News and Topics | SEARCH

CONNECT

POLITICS | BUSINESS | ENTERTAINMENT | TECH | MEDIA | LIFE & STYLE | *CULTURE | COMEDY | HEALTHY LIVING | *VOICES | LOCAL | MORE

ENTERTAINMENT | CELEBRITY | MEDIA | COMEDY | MUSIC | TV | CELEBRITY KIDS | MOVIEFONE

WHO Just Joined 'The Dark Knight Rises' Cast?

More In Entertainment: Star Goes Drag... Snow White's Parents... Guess Who?...

Gerard Butler's New Look

Will.i.am

Musician, technology advocate, and creator/executive producer of I.am FIRST: Science

GET UPDATES FROM WILL.I.AM

FAN | RSS | EMAIL | twitter | Like

Science Is Rockin' Roll and Technology Is Recession Proof

Posted: 8/6/11 12:40 PM ET

React > Amazing | Inspiring | Funny | Scary | Hot | Crazy | Important | Weird

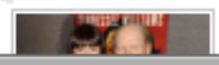
Follow > Job Creation , Science Education , Black Eyed Peas , Dean Kamen , Will.I.Am , First , U.S. Innovation , U.S. Jobs , Will.i.Am , Will.i.Am Science , Will.i.Am FIRST , Yes We Can Will.i.Am. , Entertainment News

SHARE THIS STORY

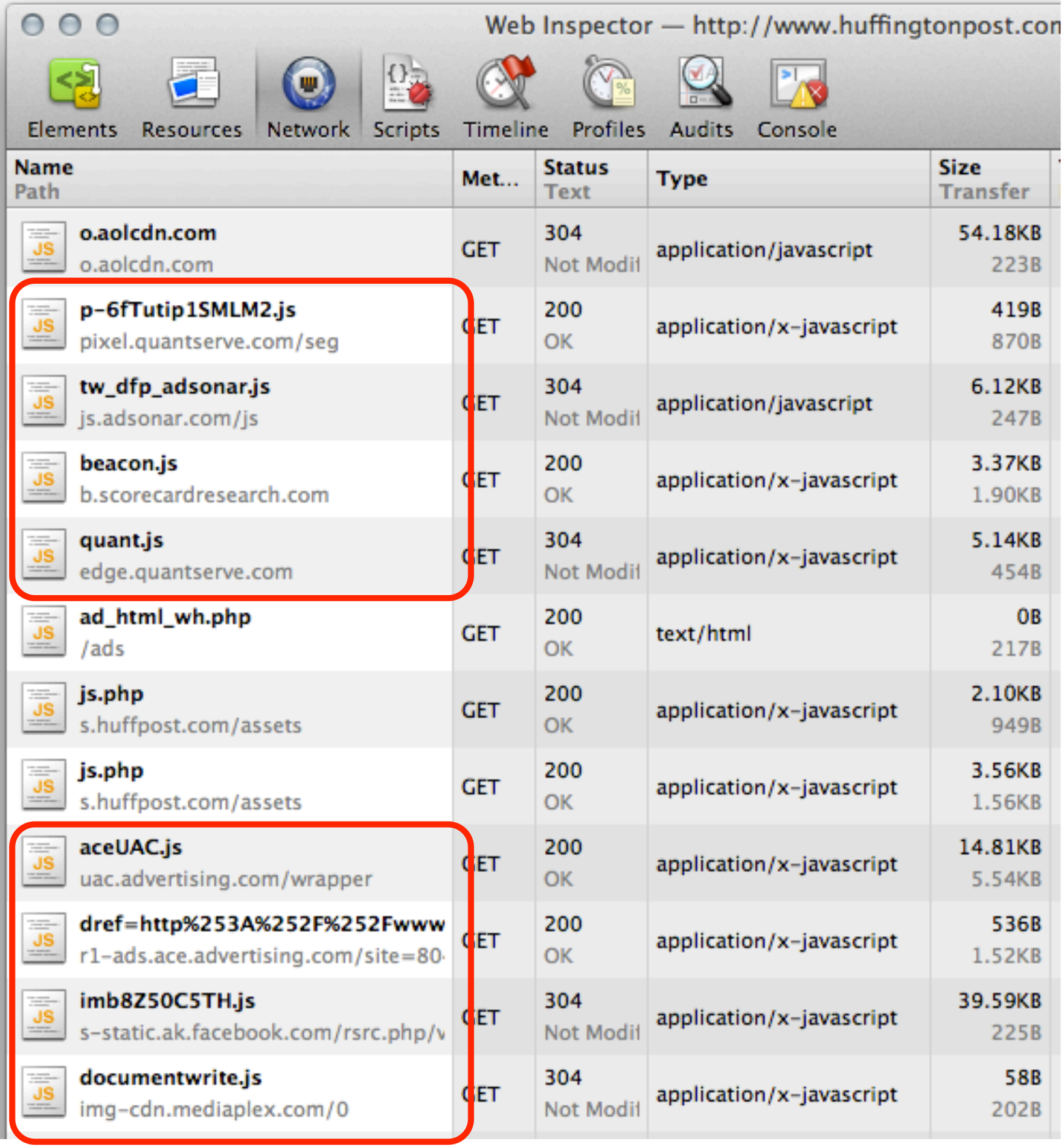
Like 54 | Send

Close your eyes and think about today. Think about the United States of America. Think about the state of education. Think about the state of job creation. Think

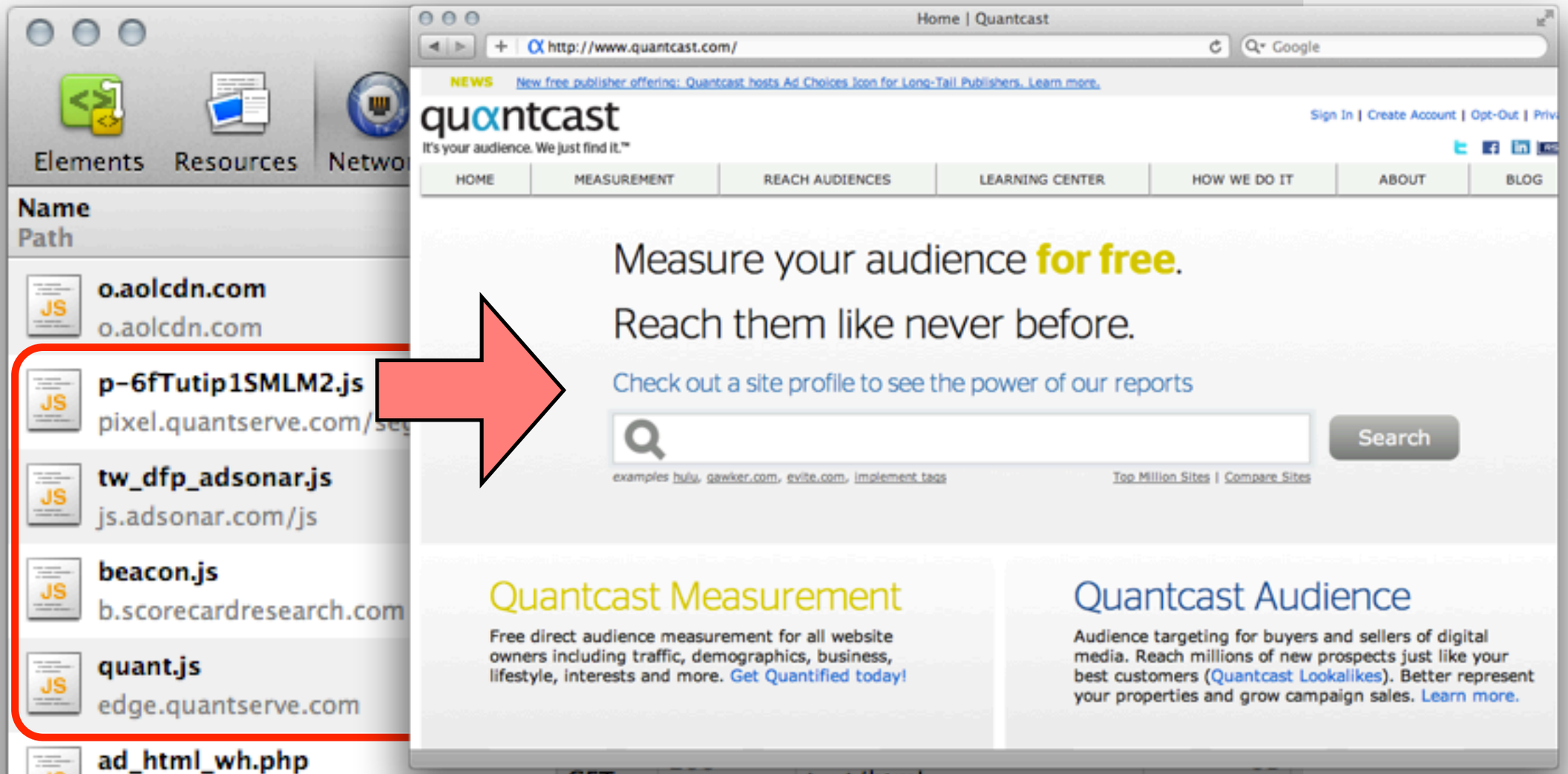
PHOTO GALLERIES



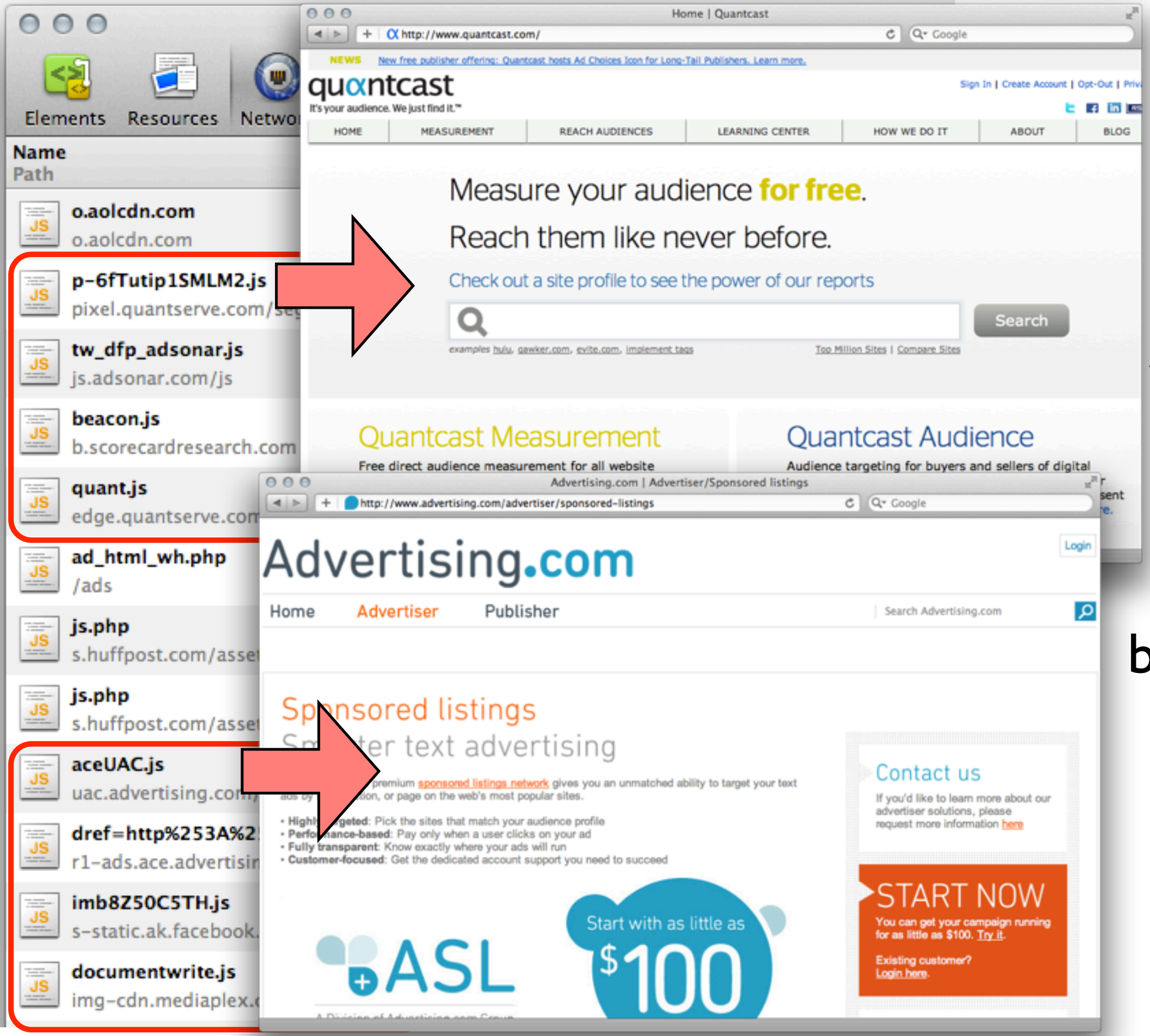
third-party ad



Who is running code in your browser?

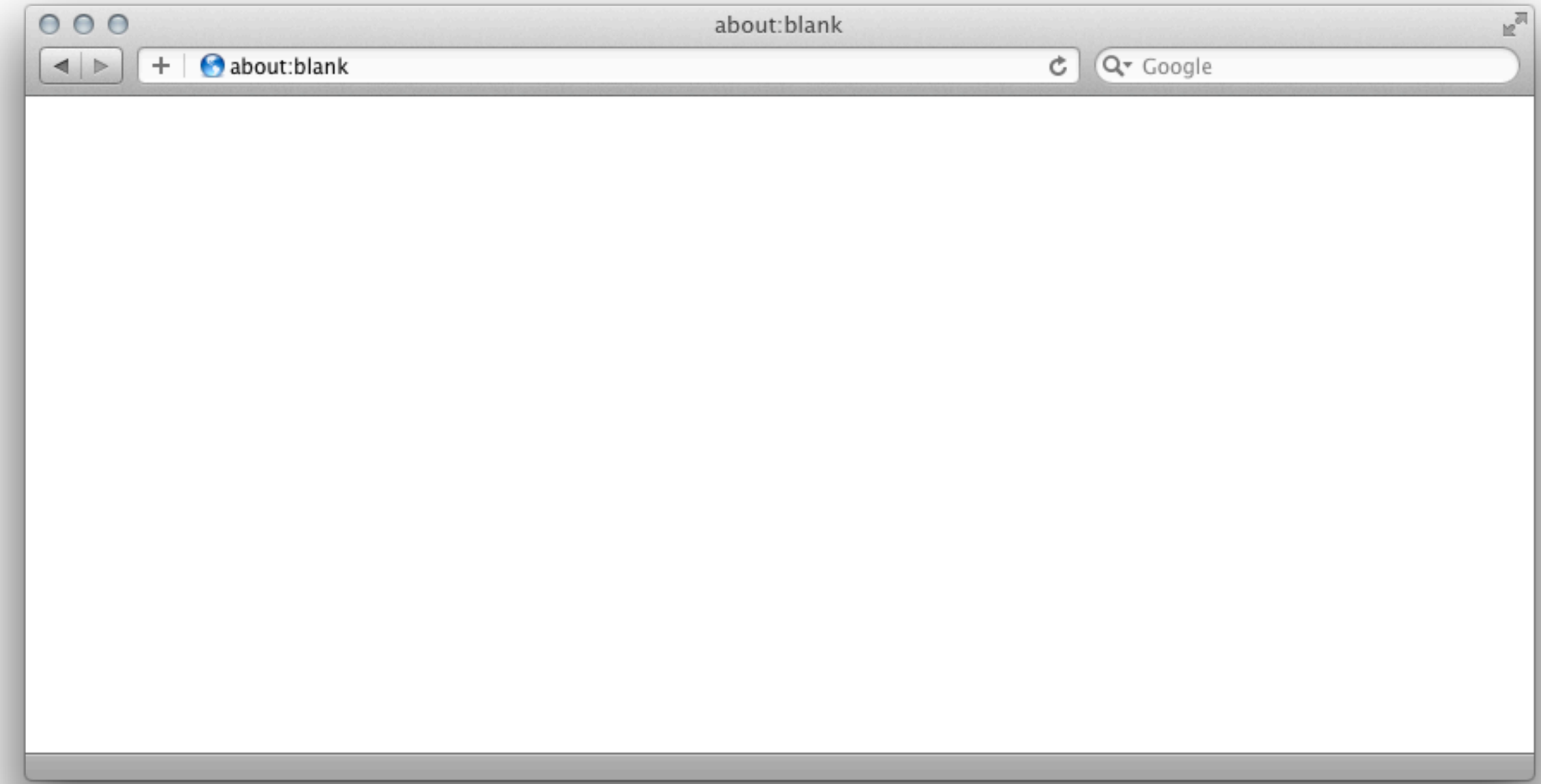


Who is running code in your browser?

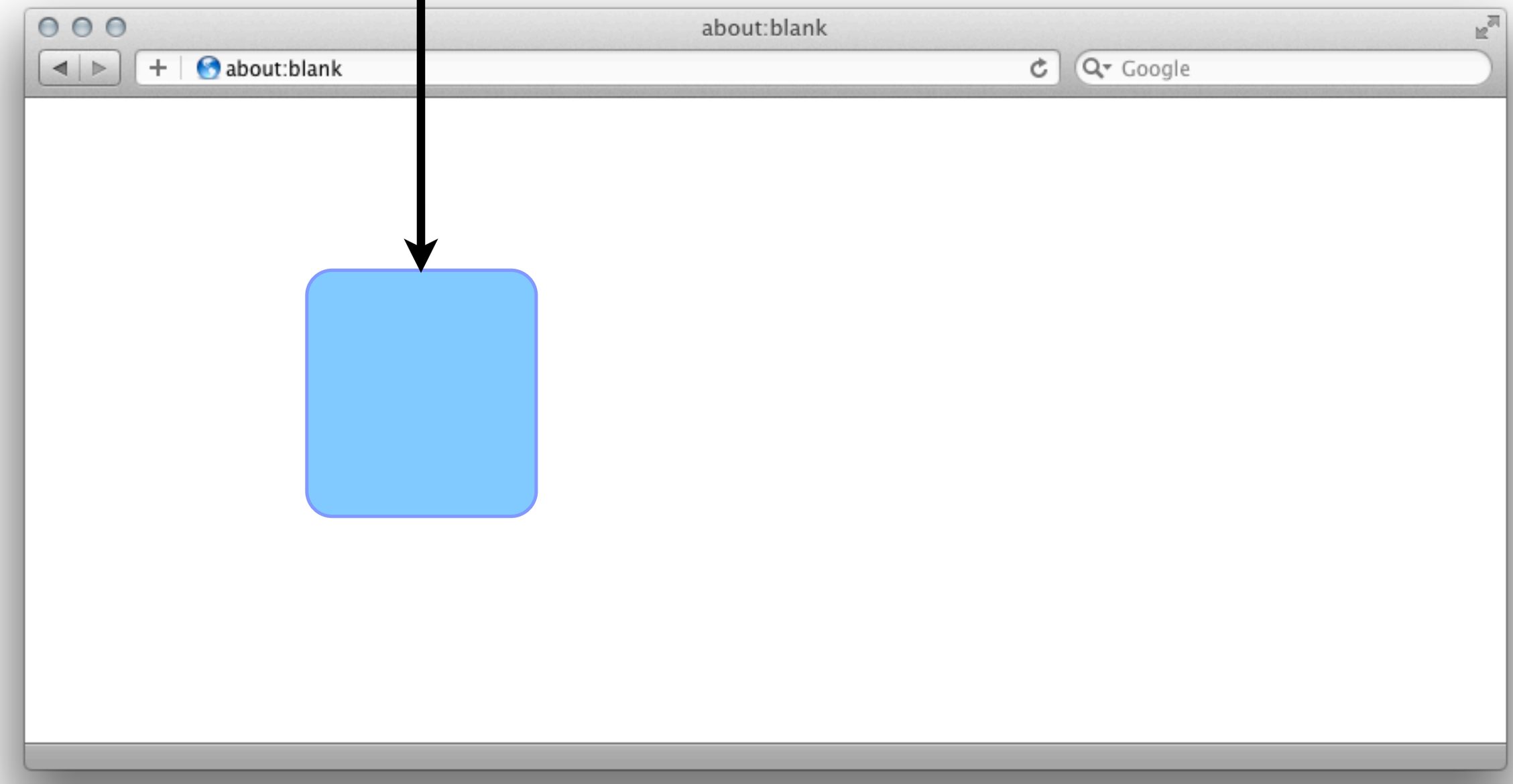


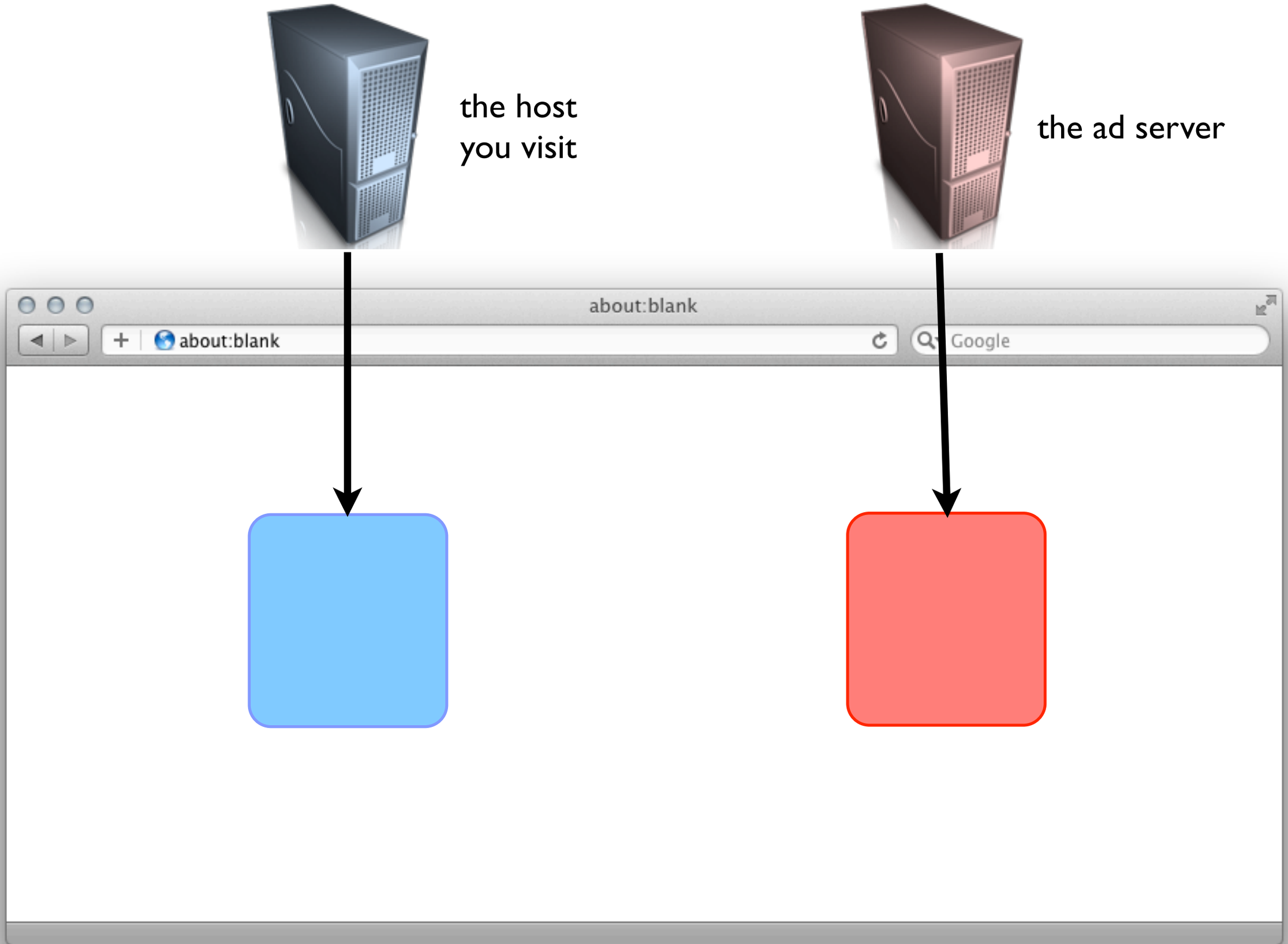
Who is running code in your browser?

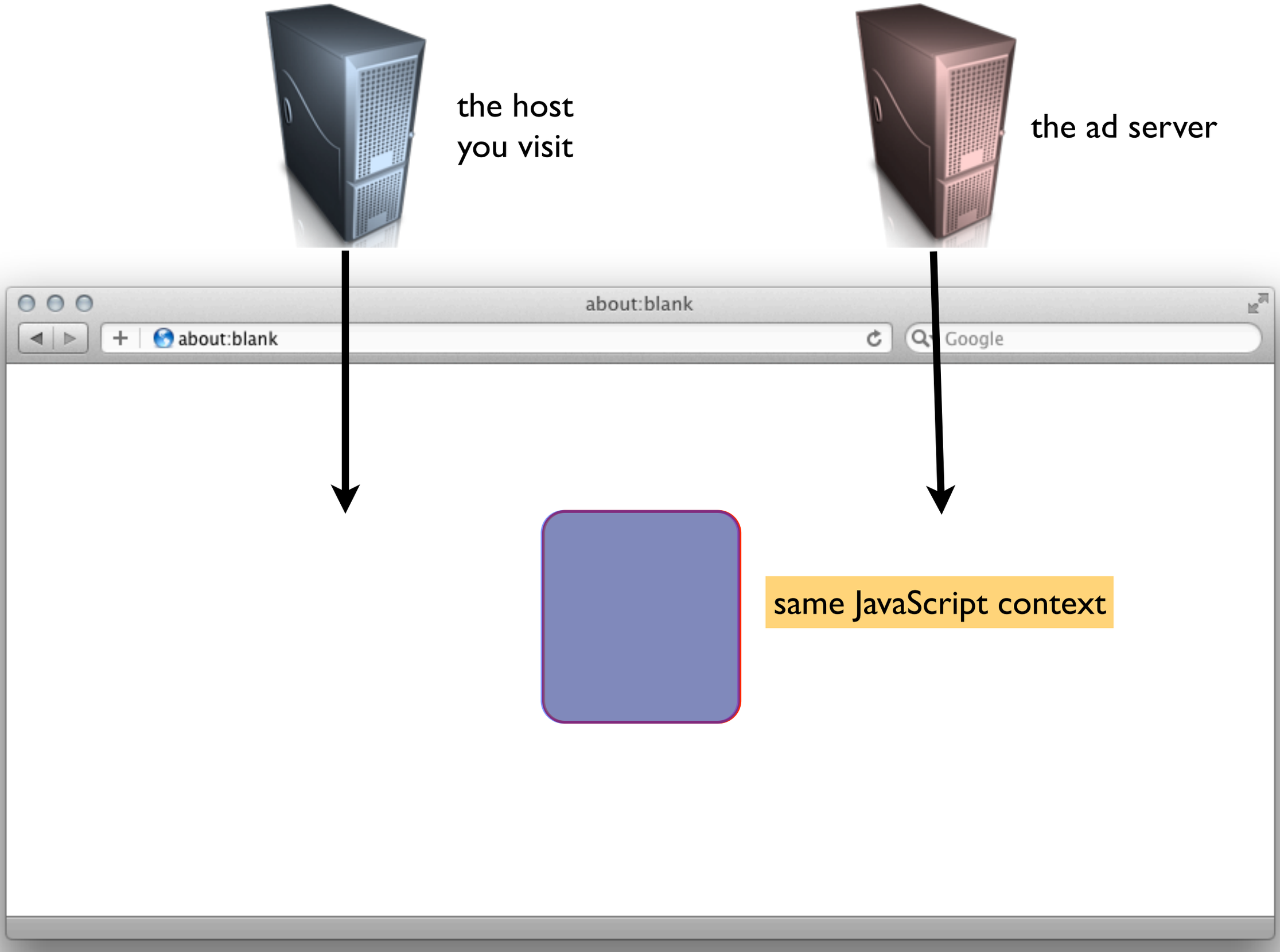
the host
you visit

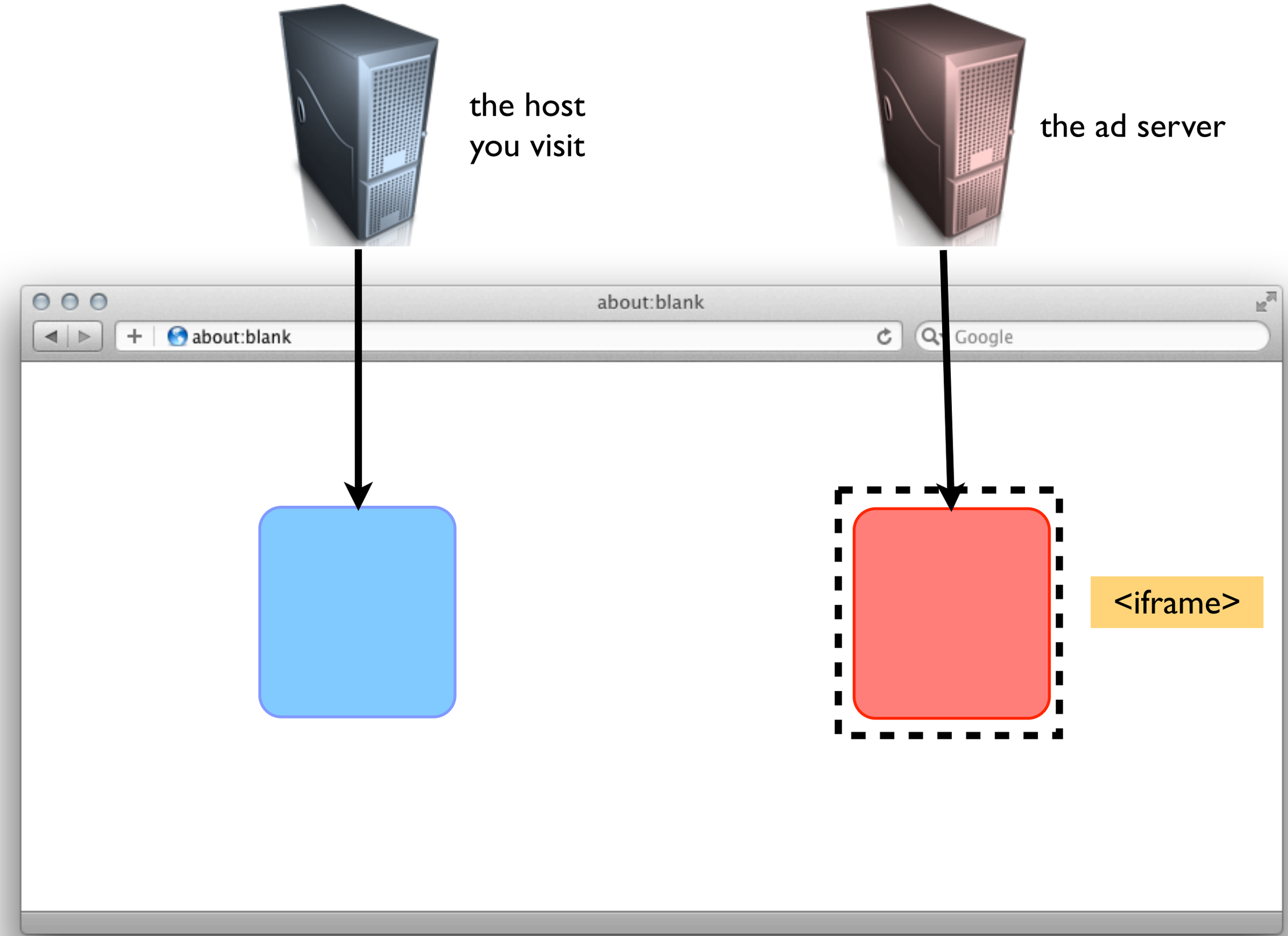


the host
you visit

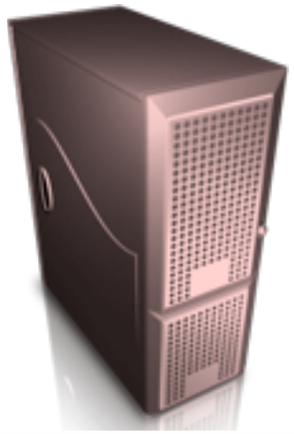



the host
you visit
the ad server
about:blank
about:blank
Google


the host
you visit
the ad server
about:blank
about:blank
Google
same JavaScript context


the host
you visit
the ad server
about:blank
about:blank
Google
<iframe>

the host
you visit



the ad server

about:blank

about:blank

Google

<iframe>

top.location.href

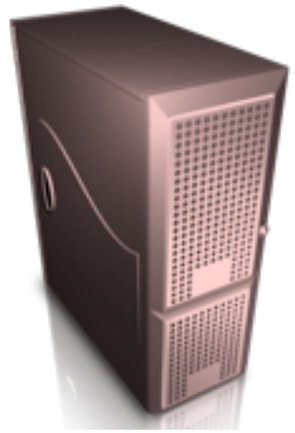
Facebook JavaScript
(FBJS)

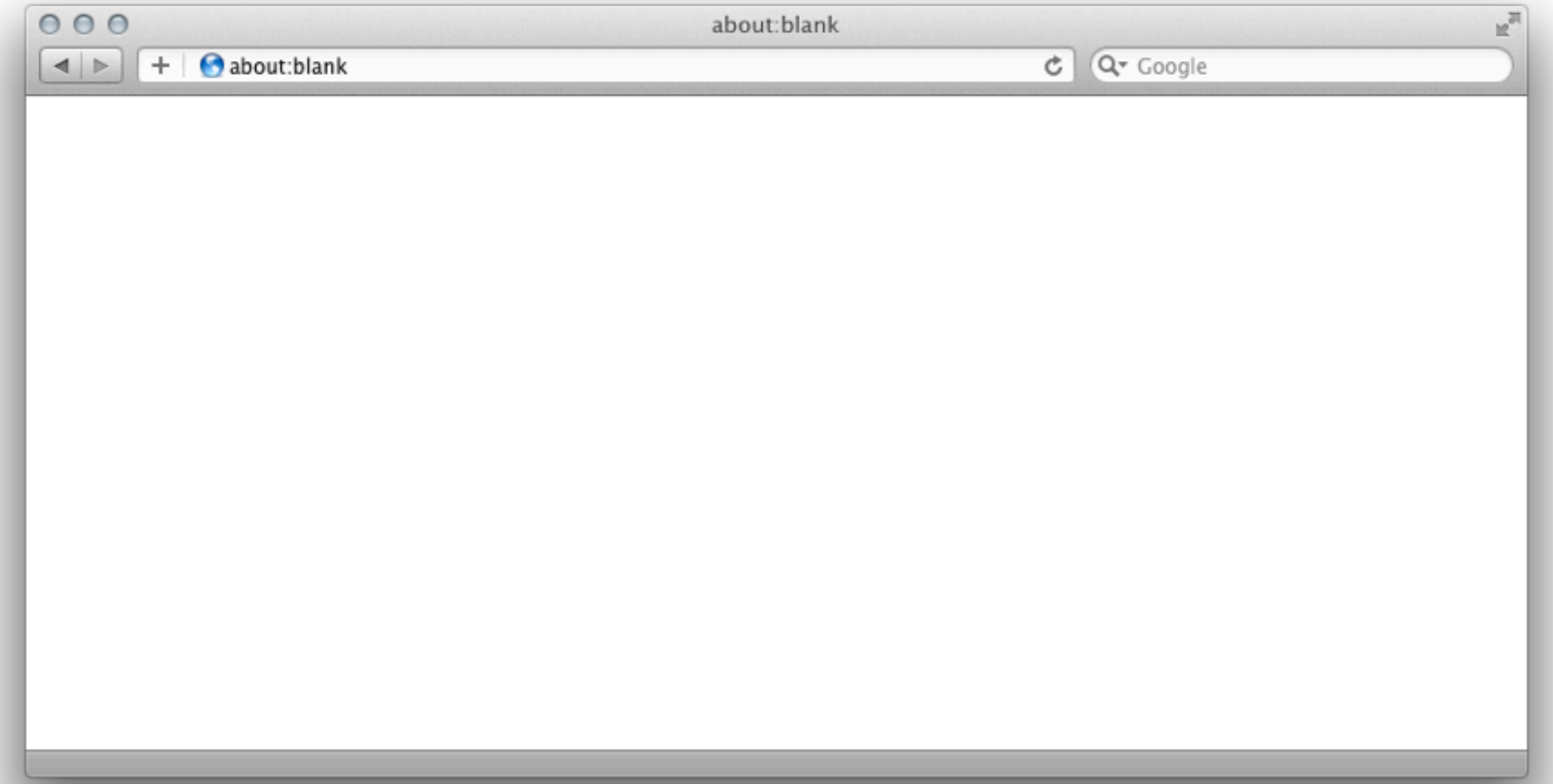
Microsoft Web Sandbox

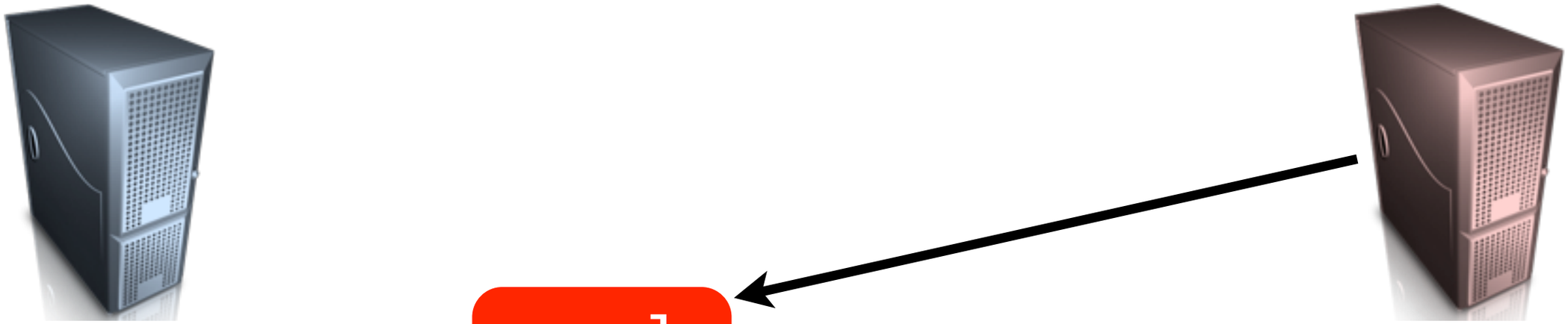
Google
Caja

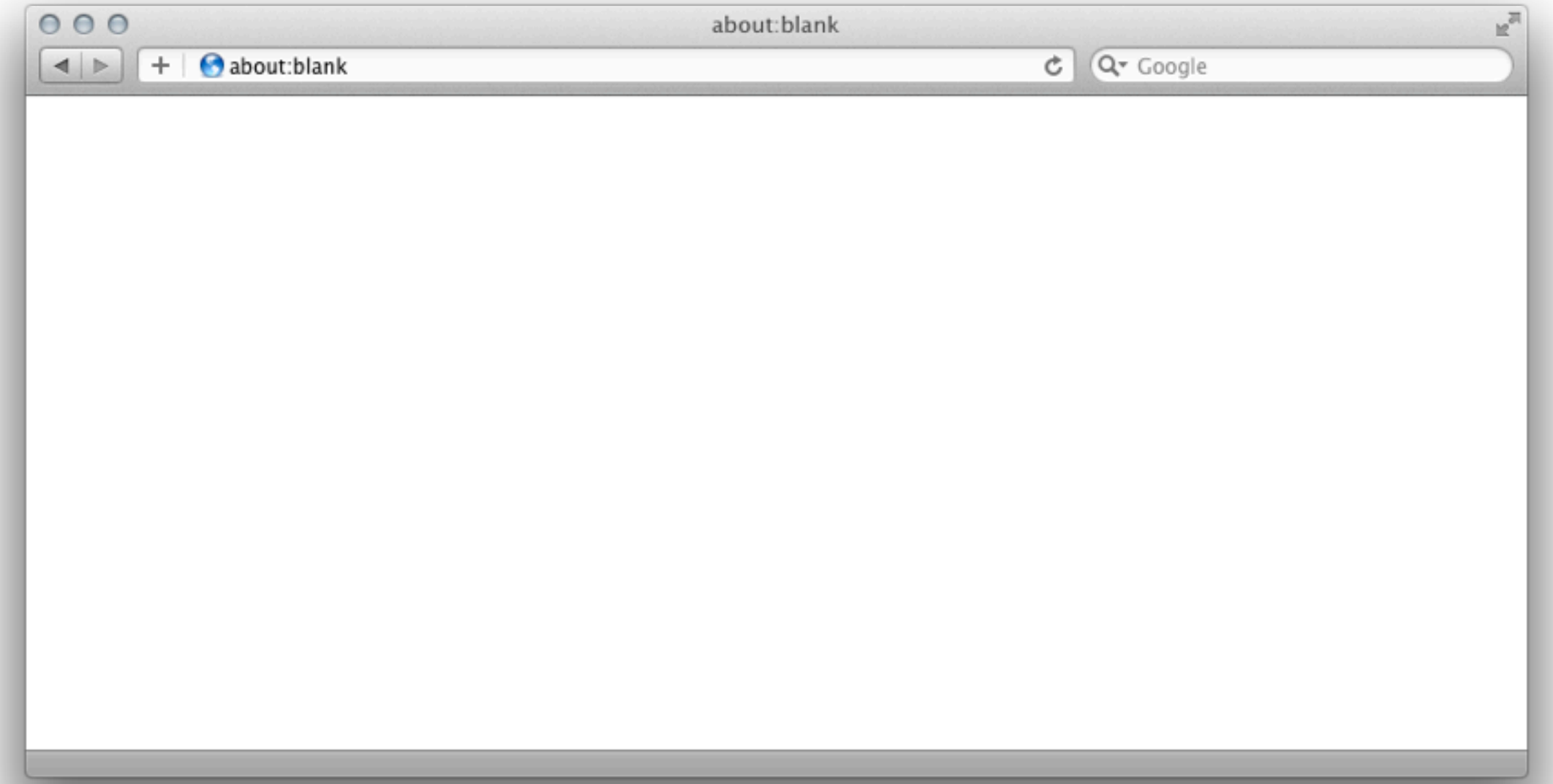
Yahoo!
ADsafe

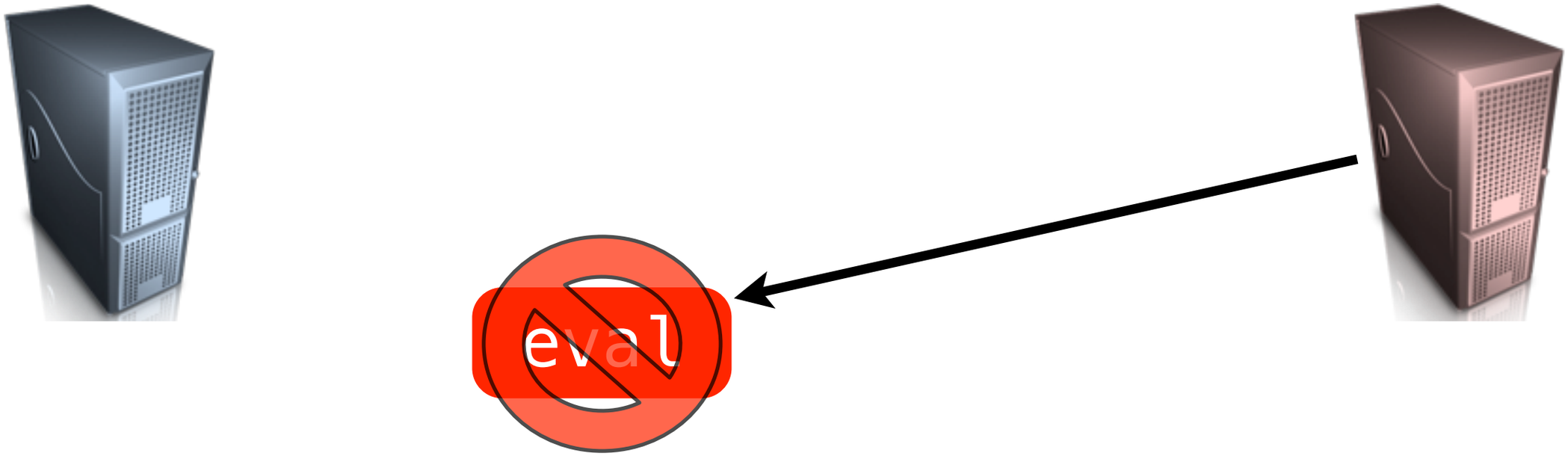
All are **defining safe sub-languages**

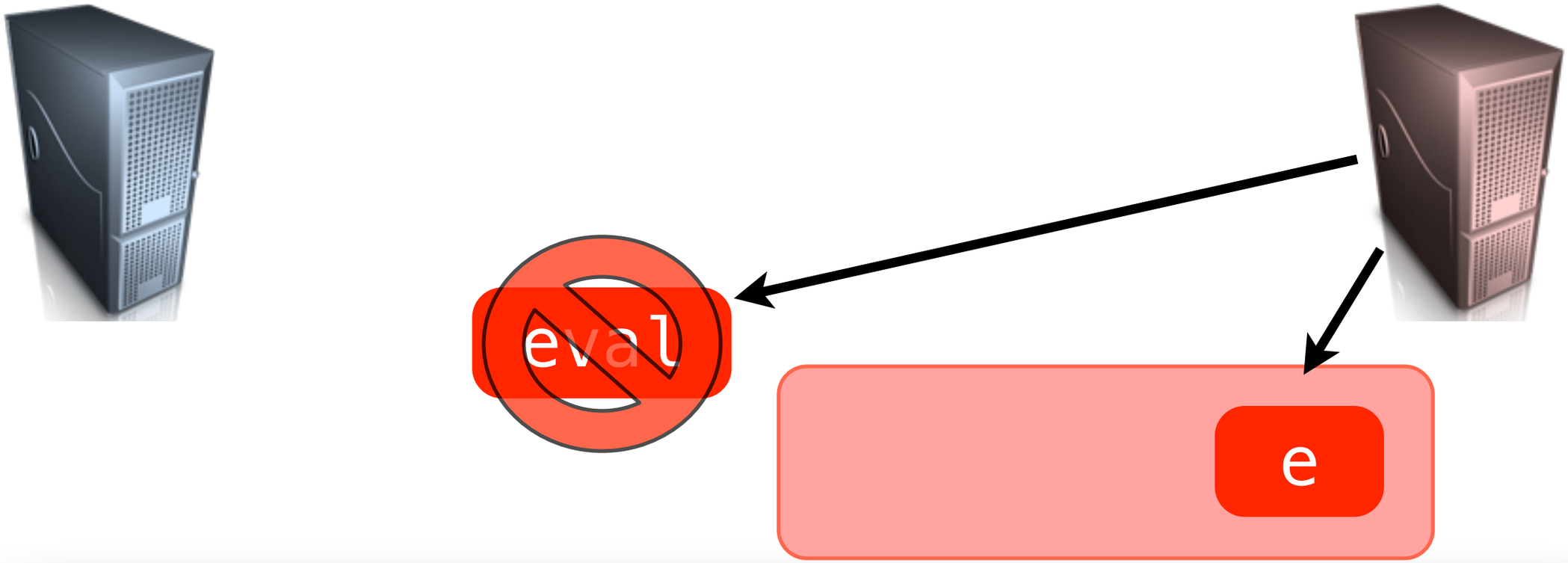


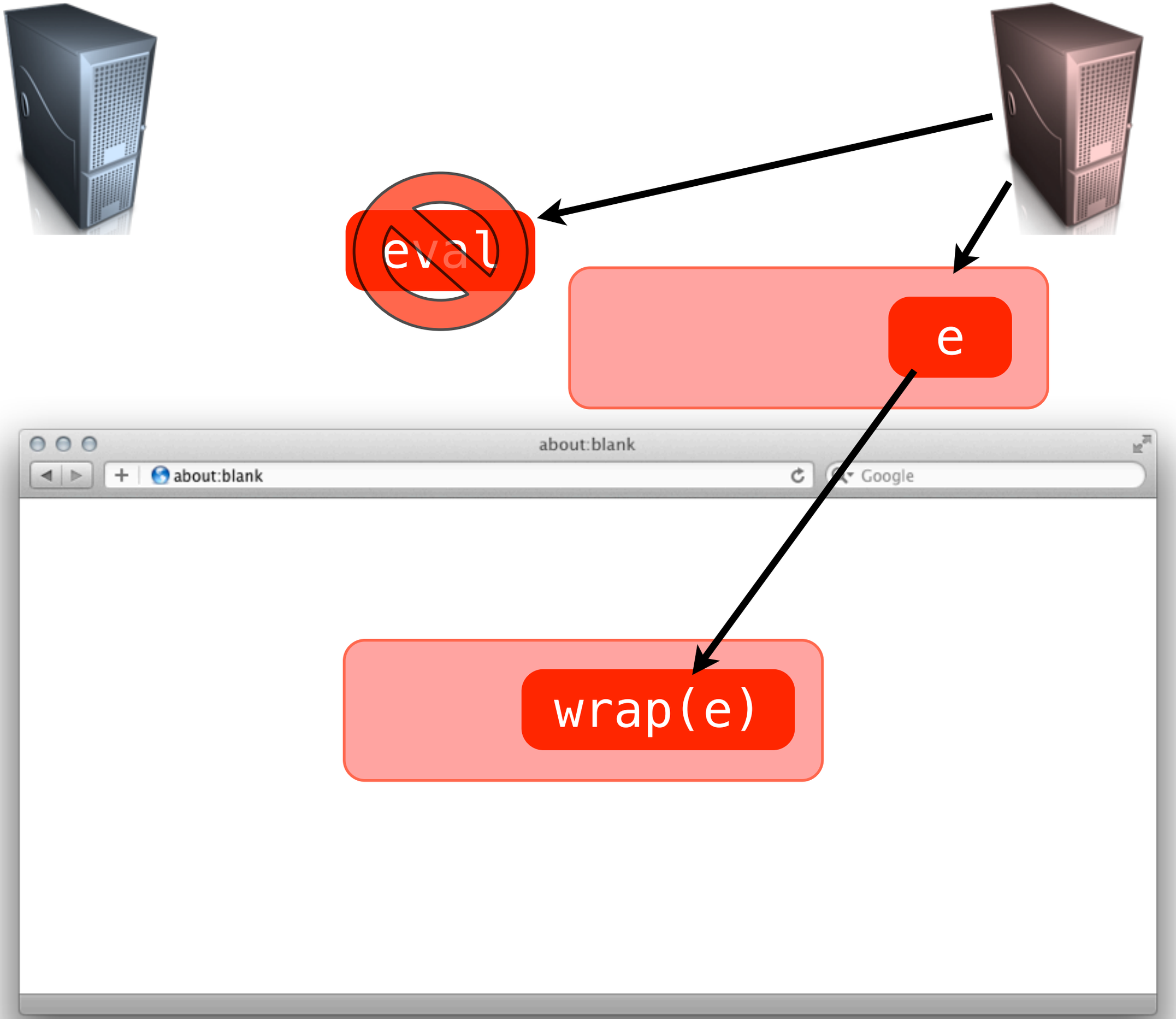

about:blank
about:blank
Google

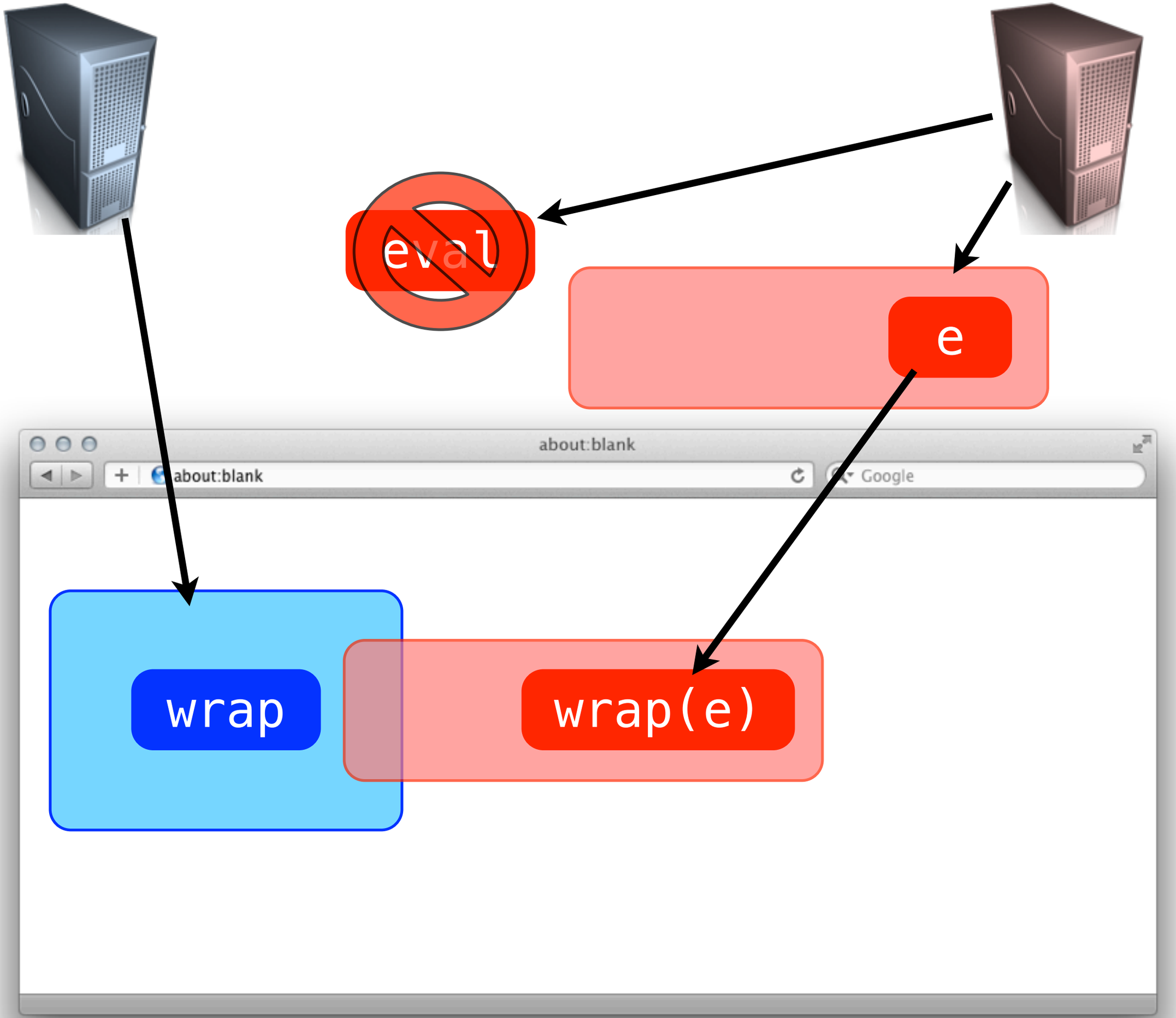

eval

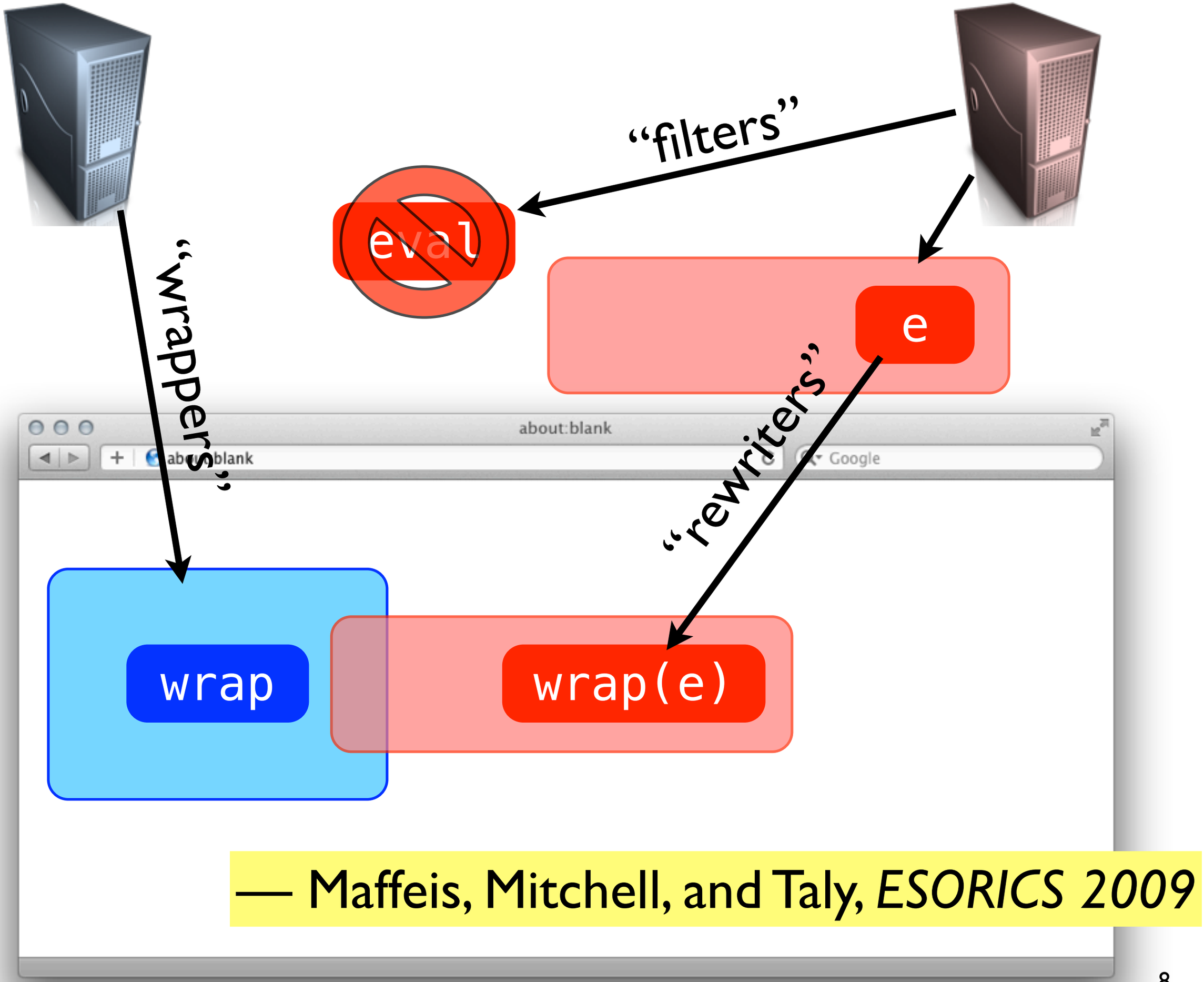

about:blank
about:blank
Google


eval
about:blank
about:blank
Google

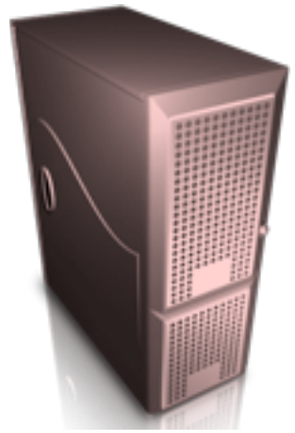

eval
e
about:blank
about:blank
Google

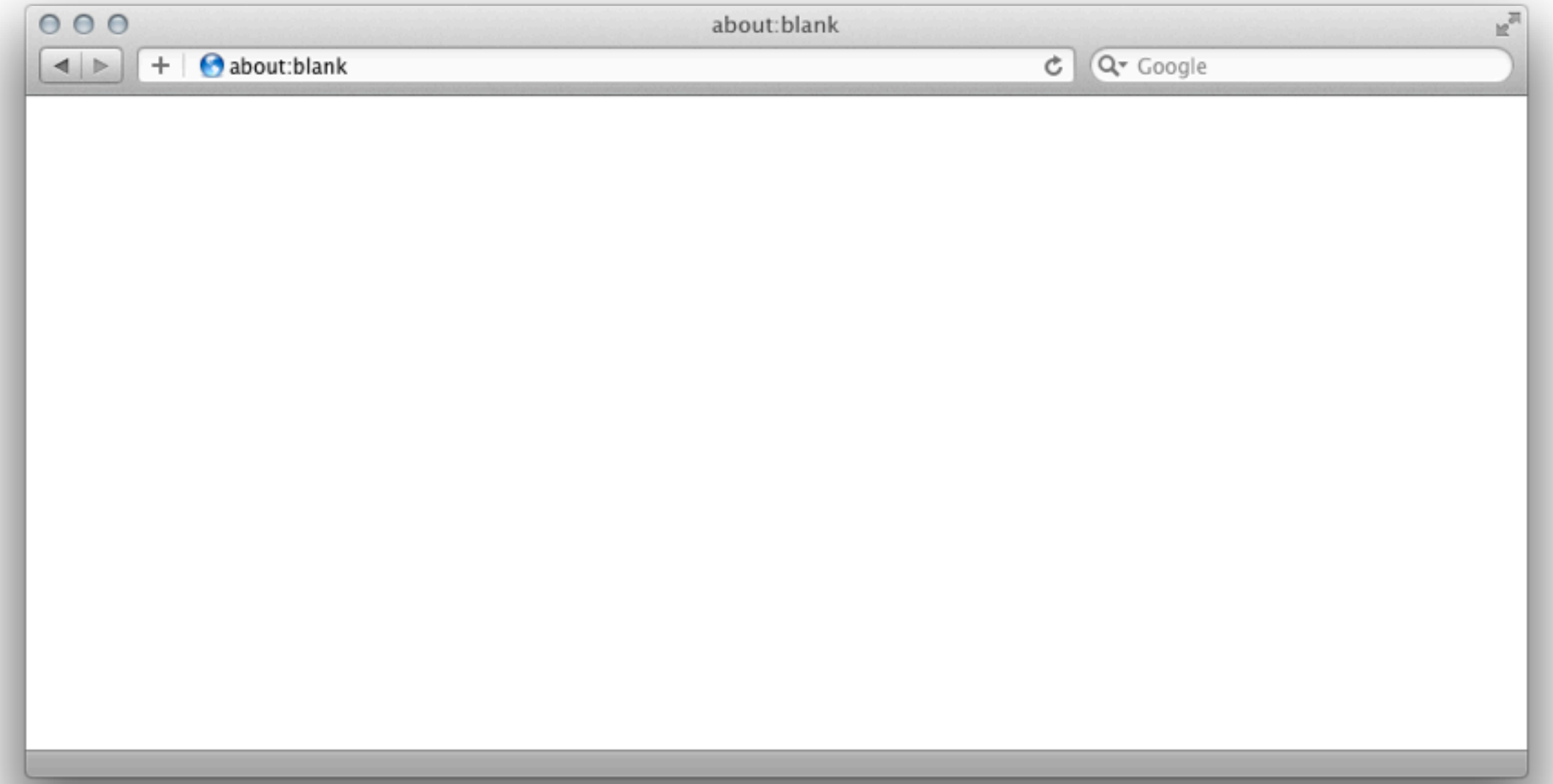

eval
e
about:blank
about:blank
Google
wrap(e)

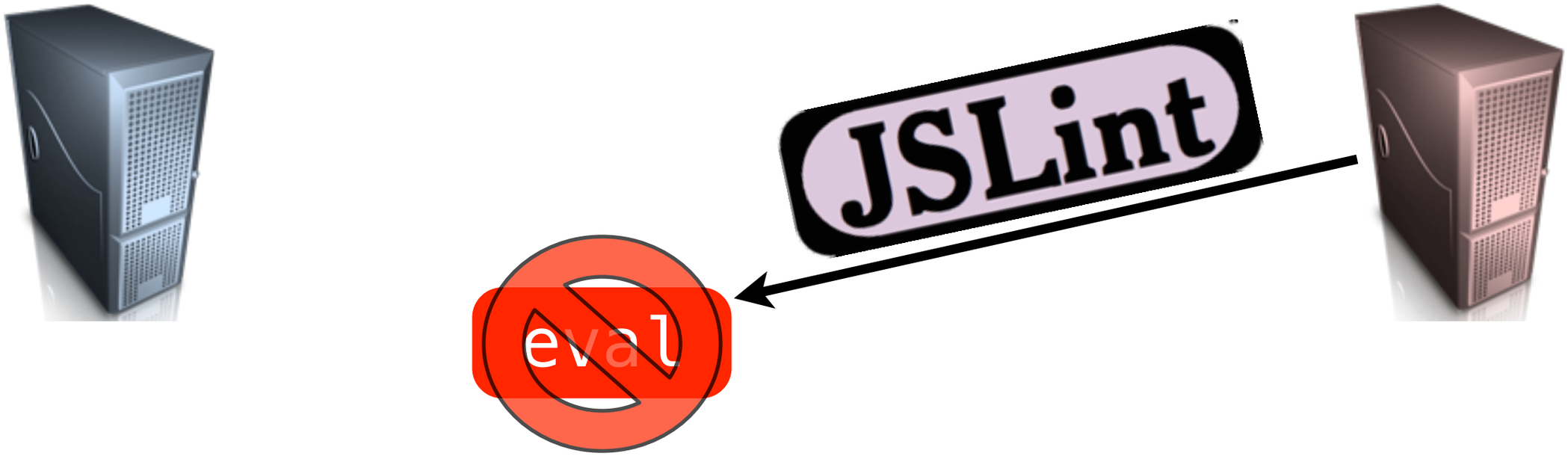

eval
e
about:blank
about:blank
Google
wrap
wrap(e)

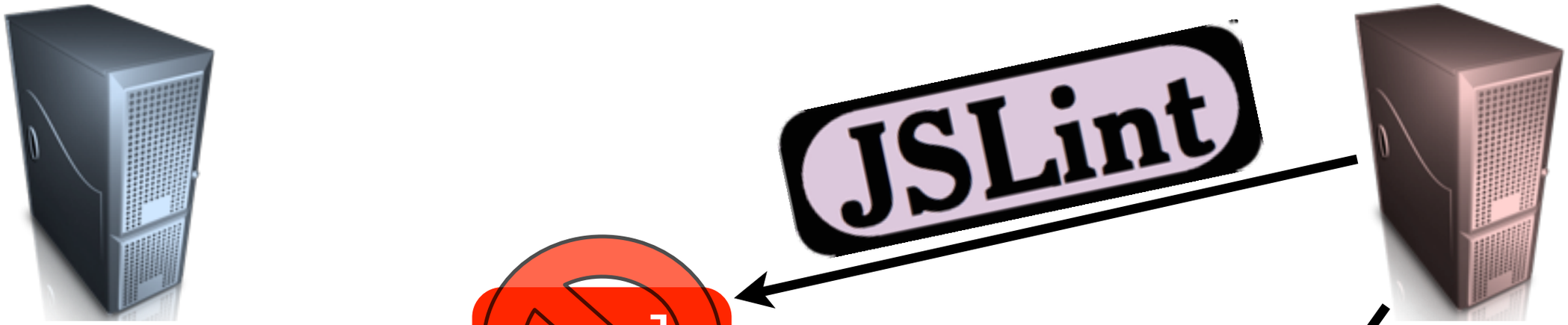


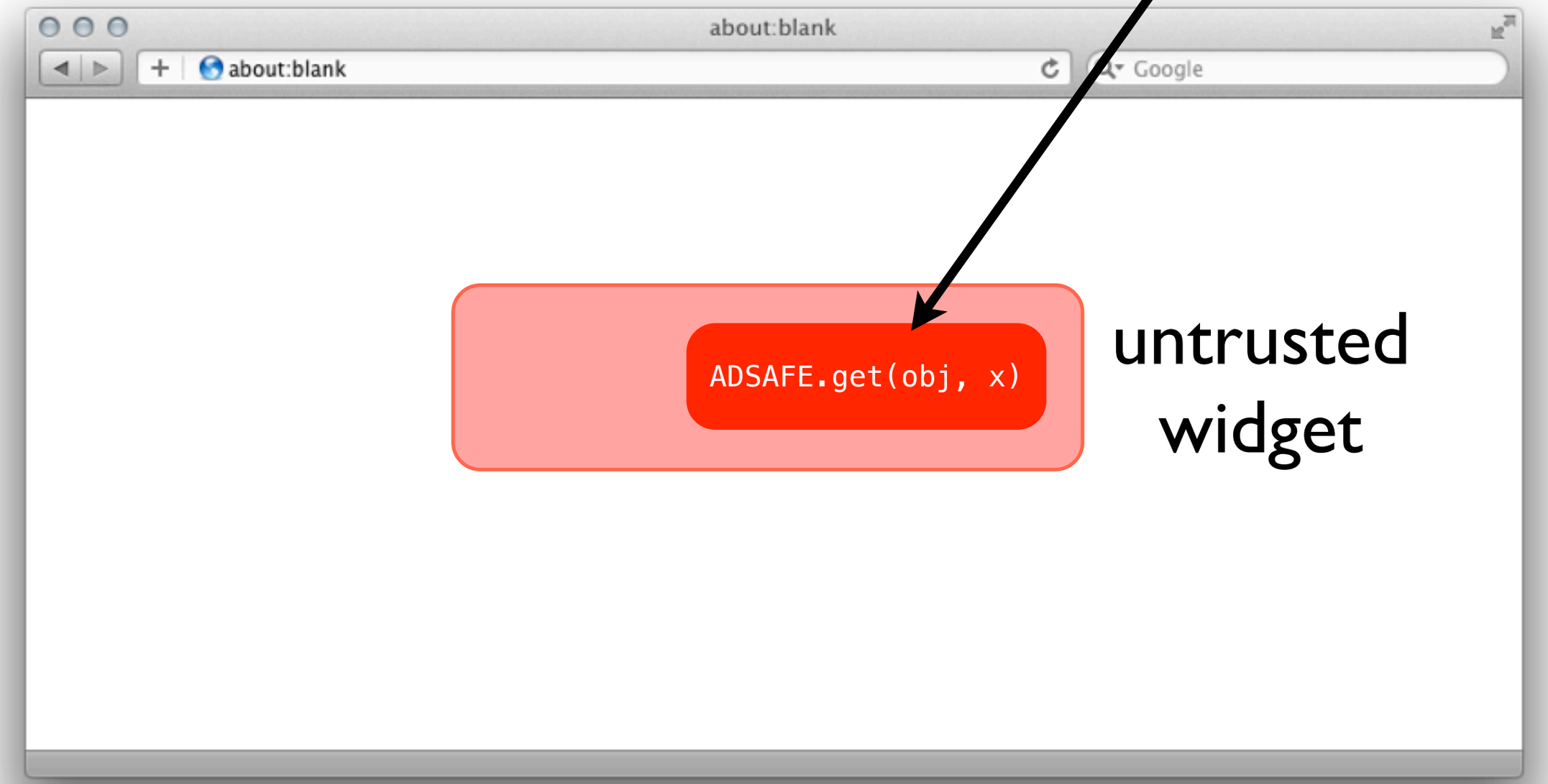
— Maffeis, Mitchell, and Taly, *ESORICS 2009*



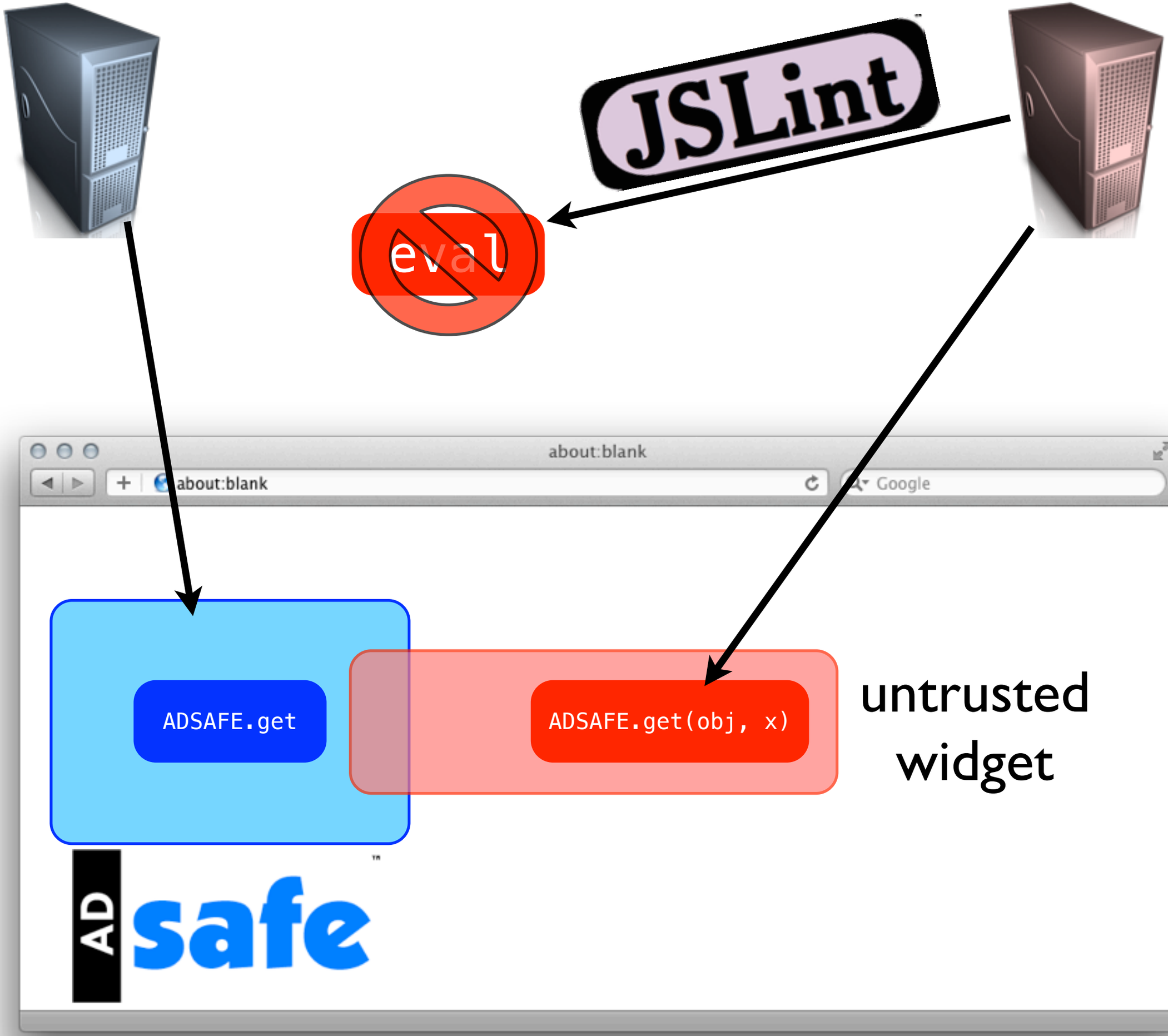

about:blank
about:blank
Google


JSLint
eval
about:blank
about:blank
Google





untrusted widget


JSLint
eval
about:blank
about:blank
Google
ADSAFE.get
ADSAFE.get(obj, x)
untrusted
widget
AD
safe

- 1,800 LOC adsafe.js library
- 50 calls to three kinds of assertions
- 40 type-tests
- 5 regular-expression based checks
- 60 privileged DOM method calls

- 1,800 LOC adsafe.js library
- 50 calls to three kinds of assertions
- 40 type-tests
- 5 regular-expression based checks
- 60 privileged DOM method calls

Type-based Verification of AD safe™

Definition 1 (ADsafety): If all embedded widgets pass JSLint, then:

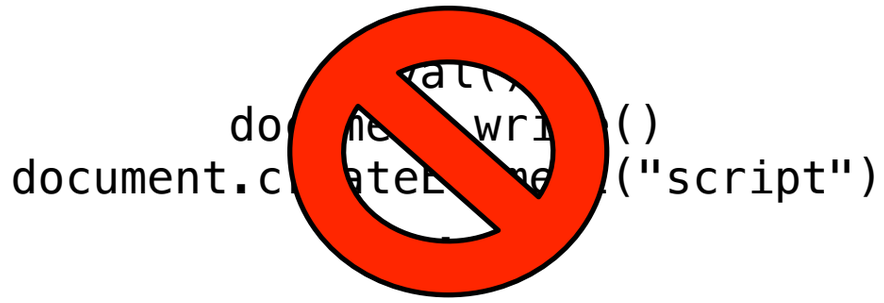
Definition 1 (ADsafety): If all embedded widgets pass JSLint, then:

1. Widgets cannot load new code at runtime, or cause ADsafe to load new code on their behalf;

```
eval()
document.write()
document.createElement("script")
...
```

Definition 1 (ADsafety): If all embedded widgets pass JSLint, then:

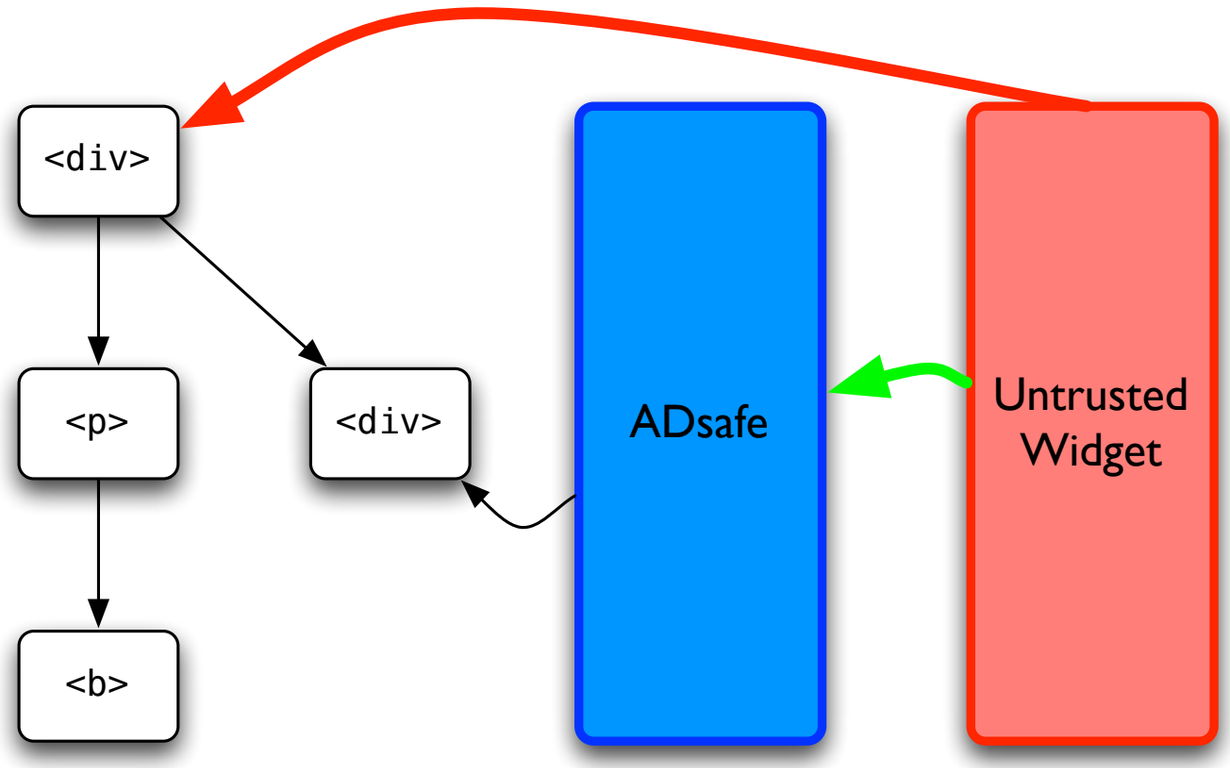
1. Widgets cannot load new code at runtime, or cause ADsafe to load new code on their behalf;



Definition 1 (ADsafety): If all embedded widgets pass JSLint, then:

1. Widgets cannot load new code at runtime, or cause ADsafe to load new code on their behalf;

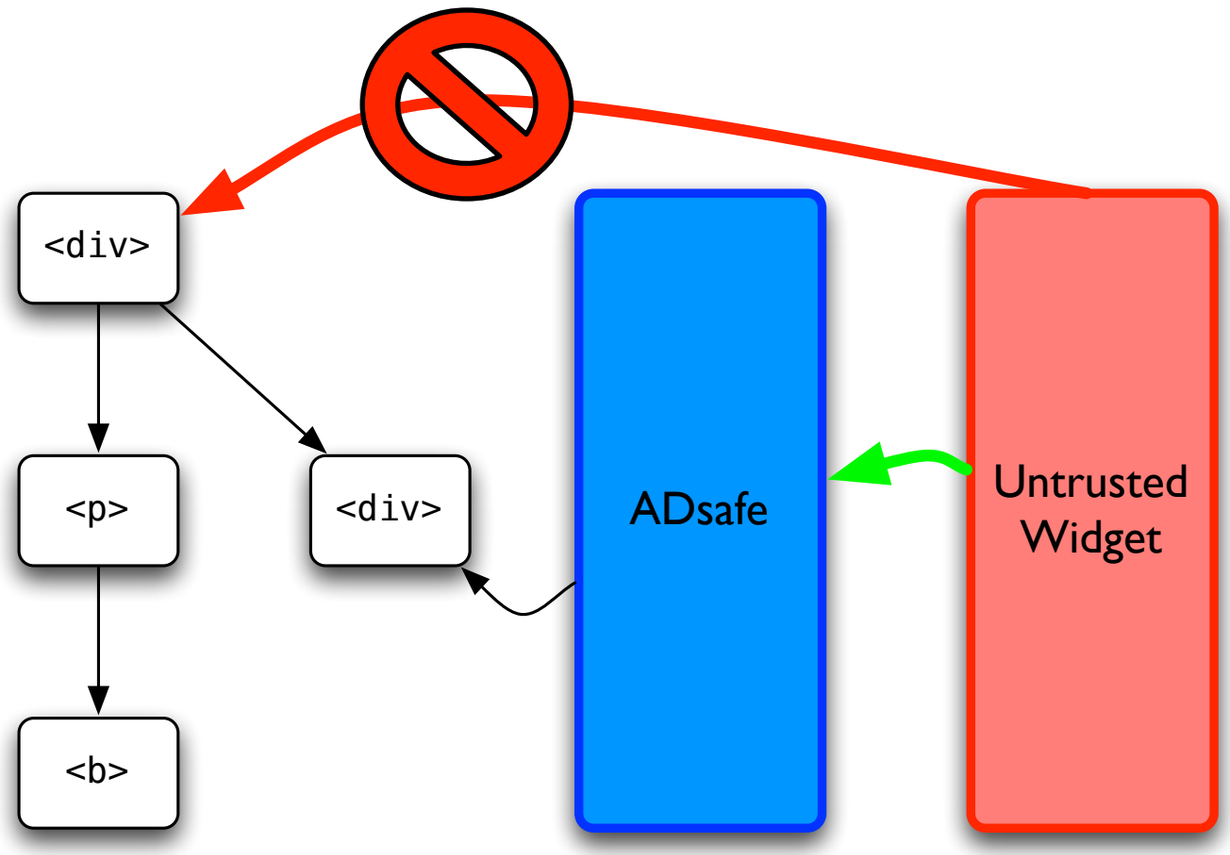
2. Widgets cannot obtain direct references to DOM nodes;



Definition 1 (ADsafety): If all embedded widgets pass JSLint, then:

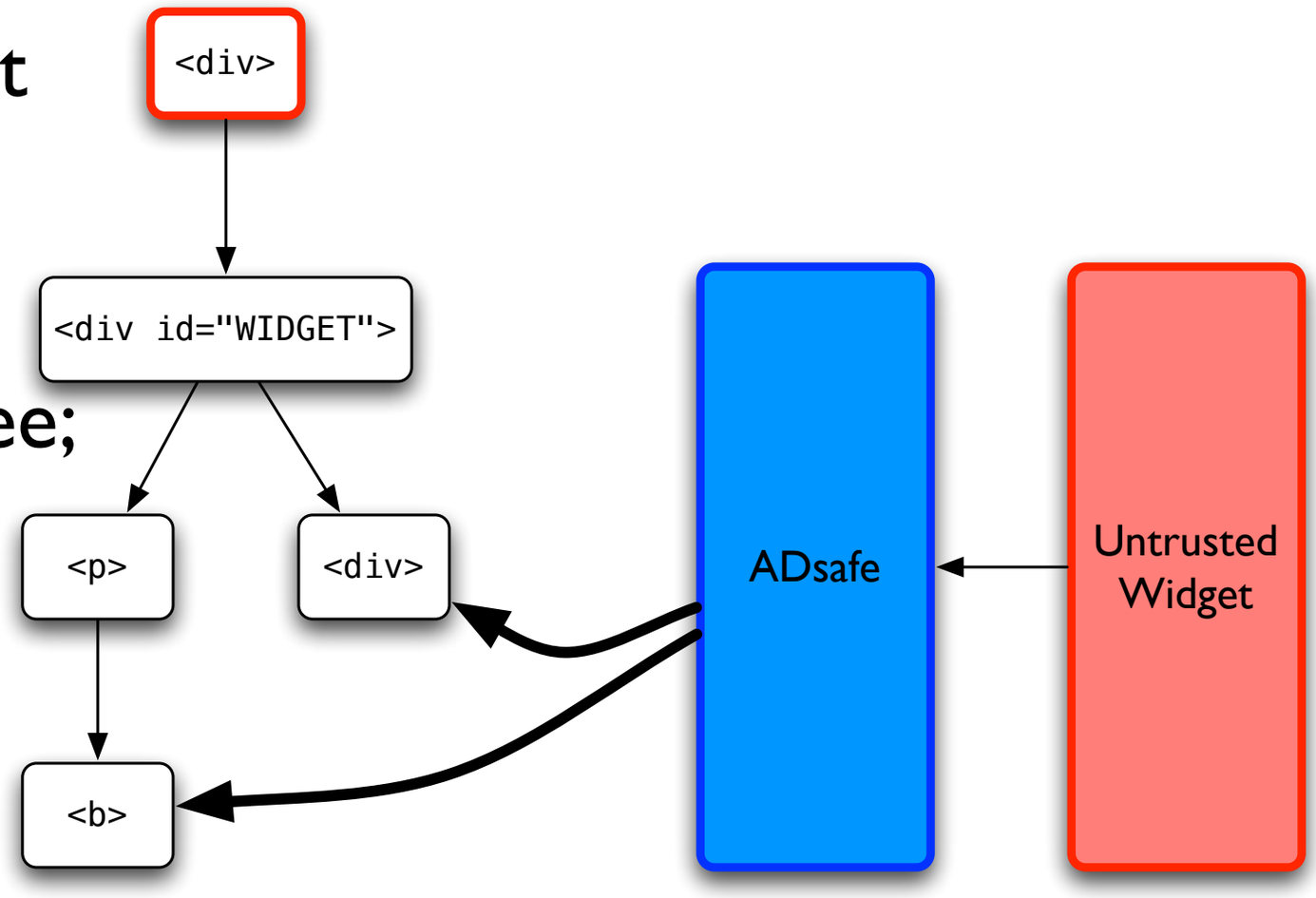
1. Widgets cannot load new code at runtime, or cause ADsafe to load new code on their behalf;

2. Widgets cannot obtain direct references to DOM nodes;



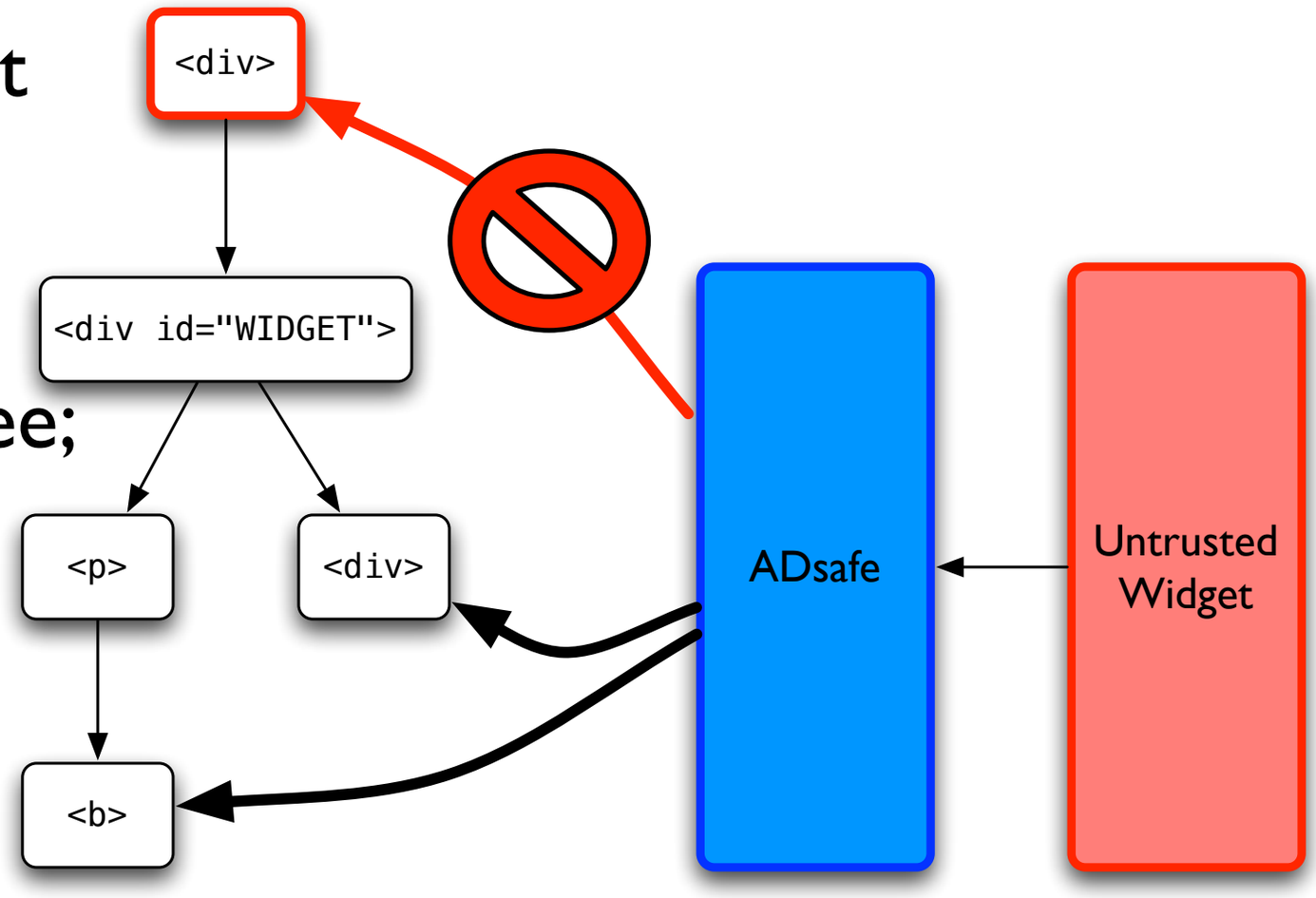
Definition 1 (ADsafety): If all embedded widgets pass JSLint, then:

1. Widgets cannot load new code at runtime, or cause ADsafe to load new code on their behalf;
2. Widgets cannot obtain direct references to DOM nodes;
3. Widgets cannot affect the DOM outside of their subtree; and



Definition 1 (ADsafety): If all embedded widgets pass JSLint, then:

1. Widgets cannot load new code at runtime, or cause ADsafe to load new code on their behalf;
2. Widgets cannot obtain direct references to DOM nodes;
3. Widgets cannot affect the DOM outside of their subtree; and



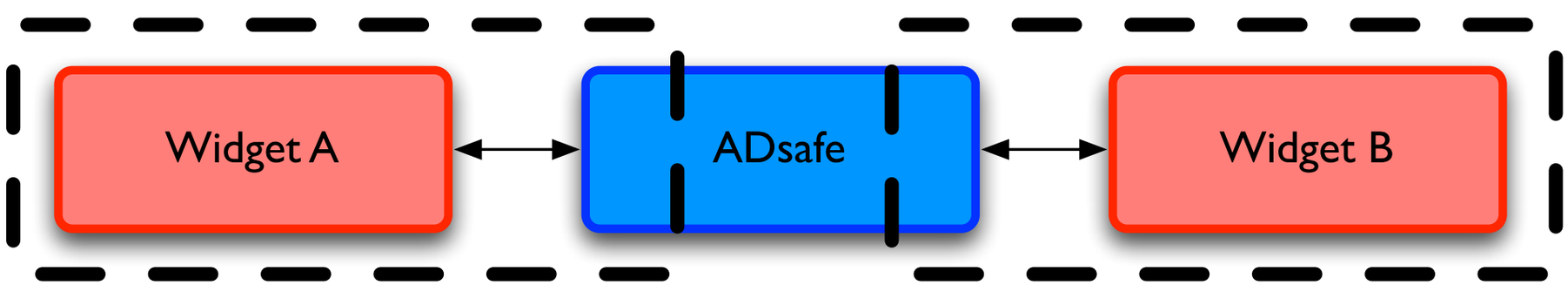
Definition 1 (ADsafety): If all embedded widgets pass JSLint, then:

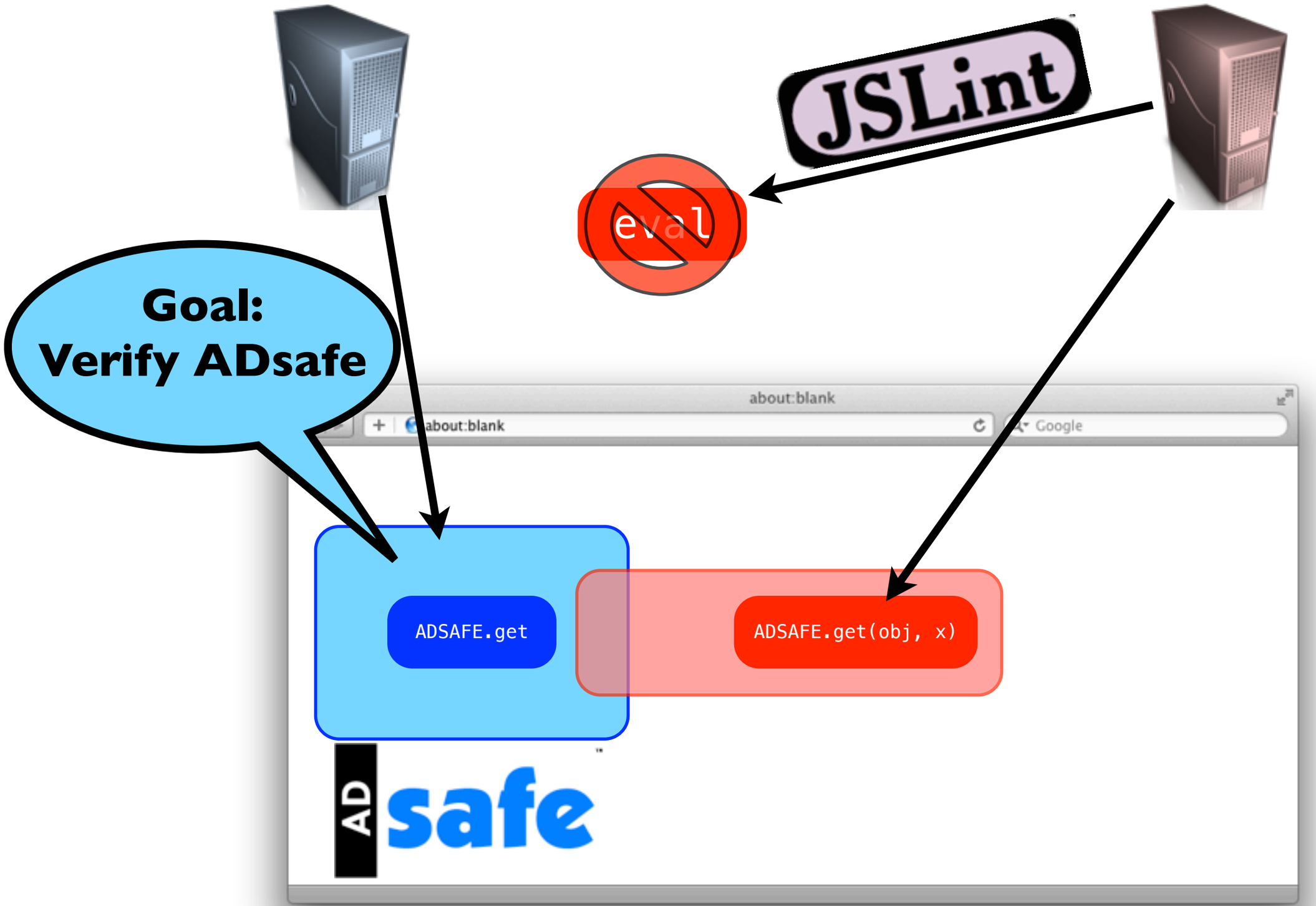
1. Widgets cannot load new code at runtime, or cause ADsafe to load new code on their behalf;

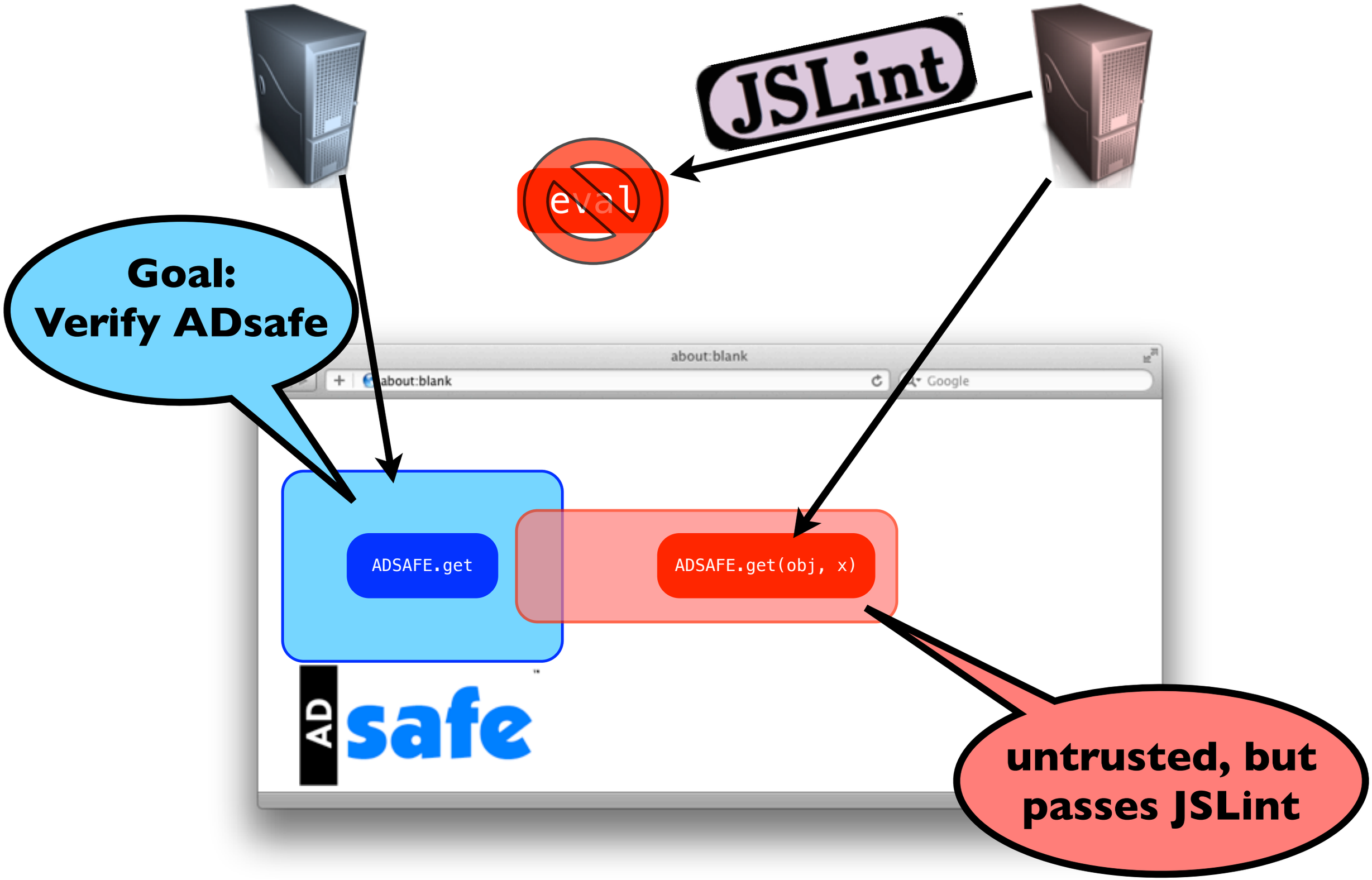
2. Widgets cannot obtain direct references to DOM nodes;

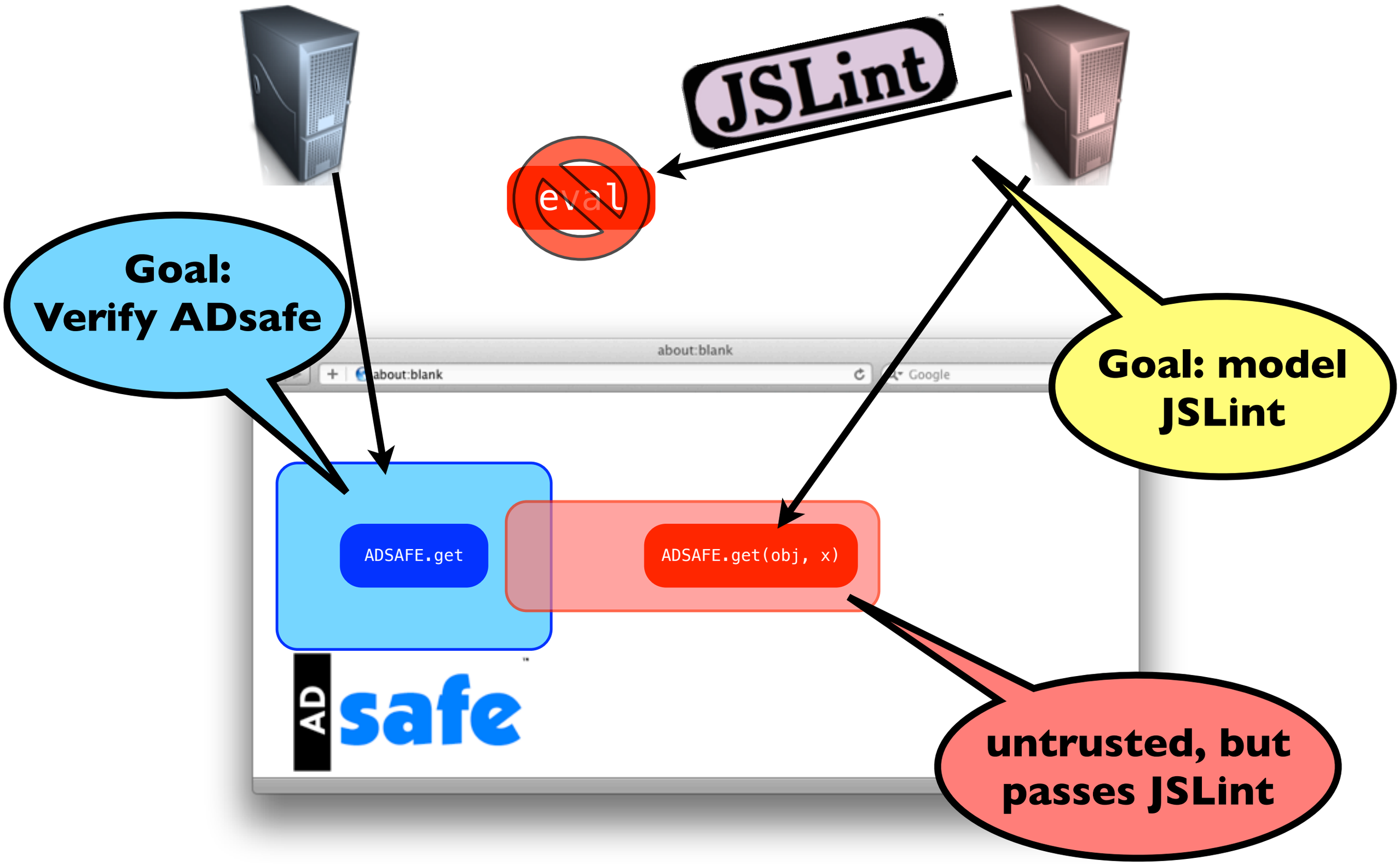
3. Widgets cannot affect the DOM outside of their subtree; and

4. Multiple widgets on the same page cannot communicate.



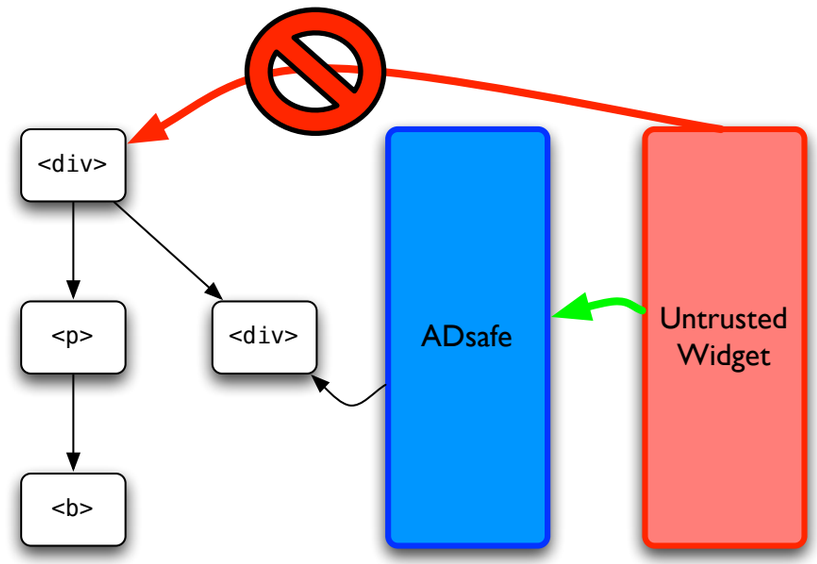

JSLint
eval
Goal:
Verify ADsafe
about:blank
about:blank
Google
ADSAFE.get
ADSAFE.get(obj, x)
AD
safe


JSLint
eval
Goal:
Verify ADsafe
about:blank
about:blank
Google
ADSAFE.get
ADSAFE.get(obj, x)
AD
safe
untrusted, but
passes JSLint


JSLint
eval
Goal:
Verify ADsafe
Goal: model
JSLint
about:blank
about:blank
Google
ADSAFE.get
ADSAFE.get(obj, x)
AD
safe
untrusted, but
passes JSLint

JSLint ensures:
no DOM
references

node



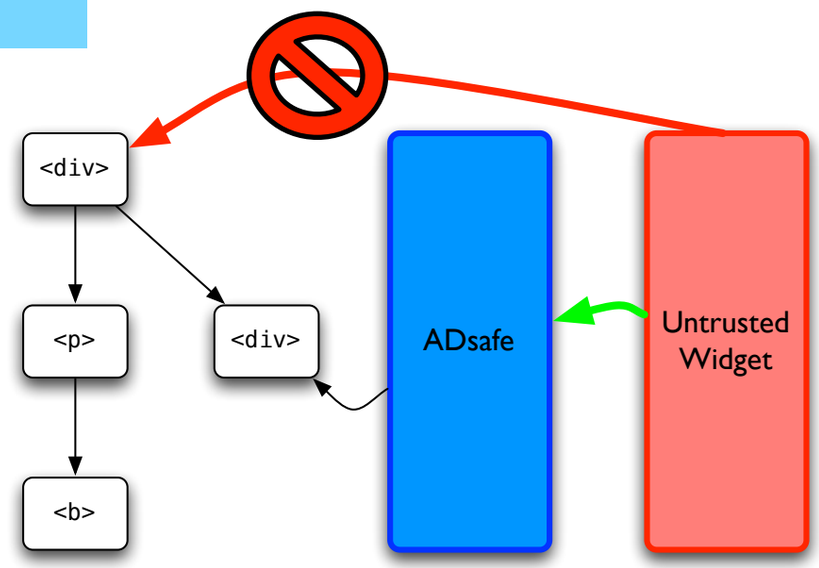
"Widgets cannot obtain direct references to DOM nodes."

JSLint ensures: no DOM references

```
node
```

ADsafe ensures: only "safe" methods on bunches

```
bunch = {
  __nodes__ : array of nodes,
  append: function ...,
  getText: function ...,
  ... 20 functions
}
```



"Widgets cannot obtain direct references to DOM nodes."

JSLint ensures: no DOM references

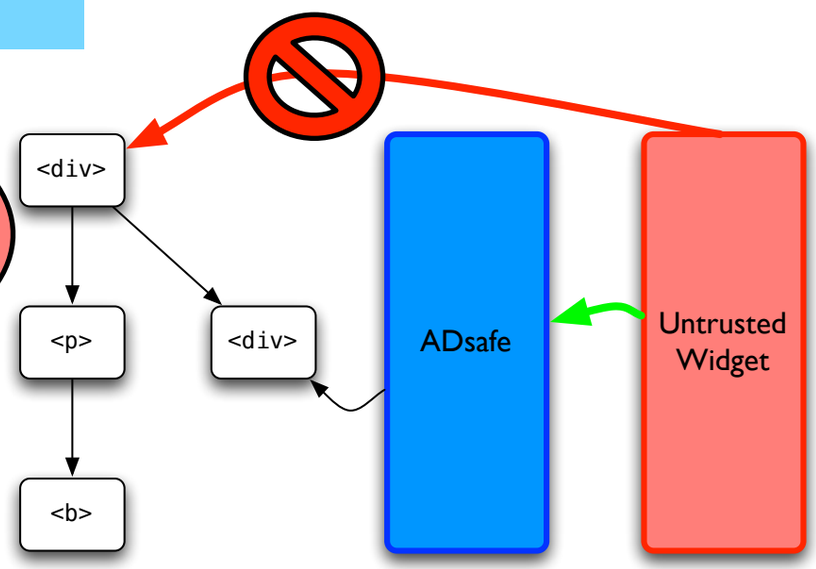
`node`

ADsafe ensures: only "safe" methods on bunches

```
bunch = {
  __nodes__ : array of nodes,
  append: function ...,
  getText: function ...,
  ... 20 functions
}
```

`bunch.__nodes__`

No private fields in JavaScript!



"Widgets cannot obtain direct references to DOM nodes."

JSLint ensures: no DOM references

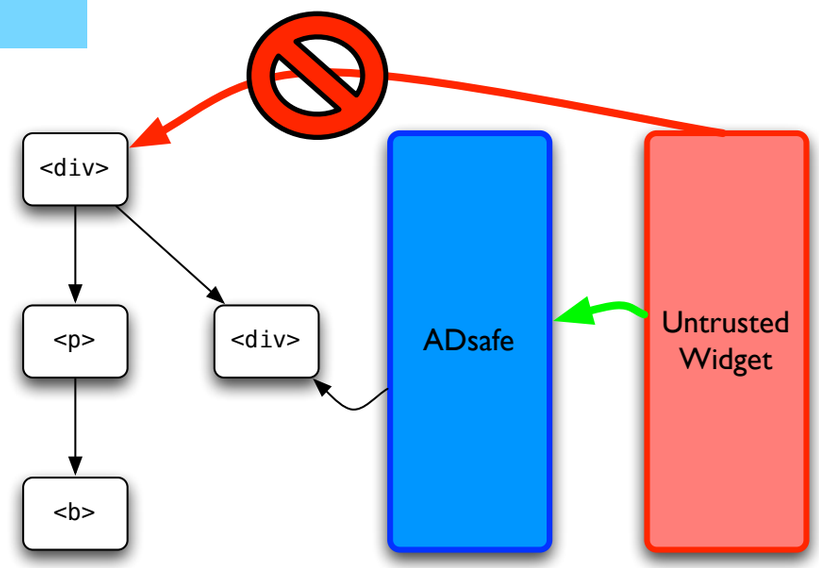
```
node
```

ADsafe ensures: only "safe" methods on bunches

```
bunch = {
  __nodes__ : array of nodes,
  append: function ...,
  getText: function ...,
  ... 20 functions
}
```

JSLint ensures: __nodes__ is "private"

```
bunch.__nodes__
```



"Widgets cannot obtain direct references to DOM nodes."

JSLint ensures: no DOM references

```
node
```

ADsafe ensures: only "safe" methods on bunches

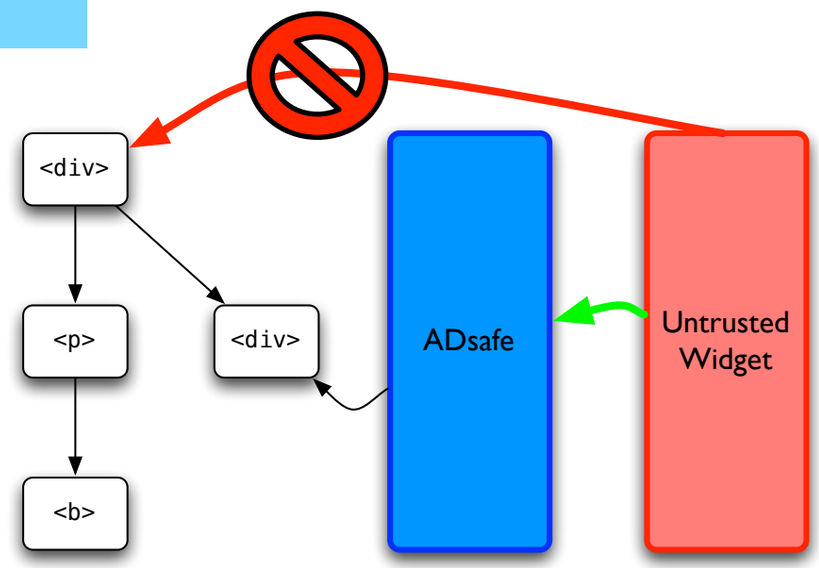
```
bunch = {
  __nodes__ : array of nodes,
  append: function ...,
  getText: function ...,
  ... 20 functions
}
```

JSLint ensures: __nodes__ is "private"

```
bunch.__nodes__
```

```
bunch.append(...)
```

Exploit append to return nodes?



"Widgets cannot obtain direct references to DOM nodes."

JSLint ensures: no DOM references

```
node
```

ADsafe ensures: only "safe" methods on bunches

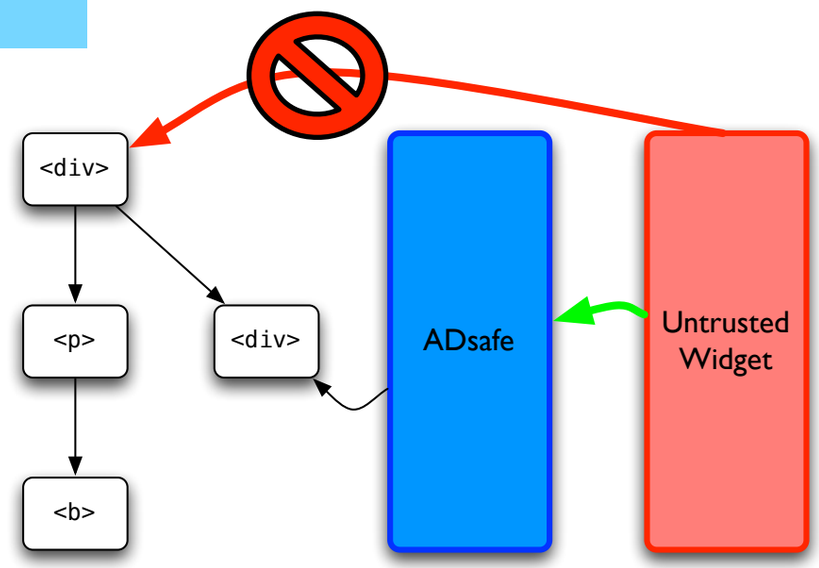
```
bunch = {
  __nodes__ : array of nodes,
  append: function ...,
  getText: function ...,
  ... 20 functions
}
```

JSLint ensures: __nodes__ is "private"

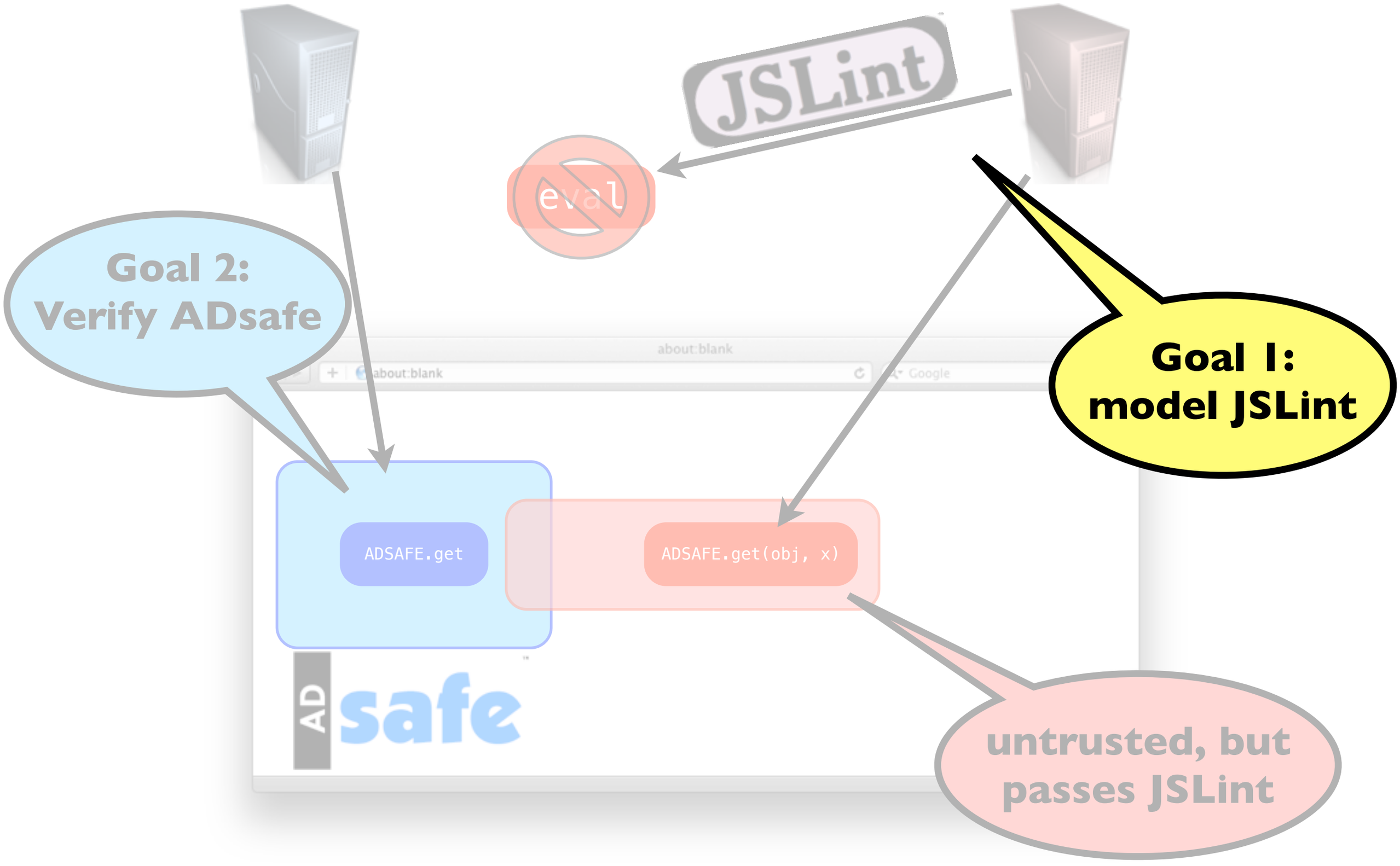
```
bunch.__nodes__
```

ADsafe ensures: DOM nodes are not returned

```
bunch.append(...)
```



"Widgets cannot obtain direct references to DOM nodes."


JSLint
eval
Goal 2:
Verify ADsafe
Goal 1:
model JSLint
about:blank
about:blank
Google
ADSAFE.get
ADSAFE.get(obj, x)
AD
safe
untrusted, but
passes JSLint

```
var n = 6
```

```
var s = "a string"
```

```
var b = true
```

✓ `var n = 6`

✓ `var s = "a string"`

✓ `var b = true`

```
Widget := Number + String + Boolean + Undefined + Null +
```

```
Widget := Number + String + Boolean + Undefined + Null +
```

✓ `{ x: 6, b: "car" }`

```
Widget := Number + String + Boolean + Undefined + Null +
```

✓ `{ x: 6, b: "car" }`

✓ `{ nested: { y: 10, b: false } }`

```
Widget := Number + String + Boolean + Undefined + Null +
          {  ★: Widget  }
```

✓ `{ x: 6, b: "car" }`

✓ `{ nested: { y: 10, b: false } }`

🚫 `{ __nodes__: 90 }`

🚫 `myObj.prototype = { };`

```
Widget := Number + String + Boolean + Undefined + Null +
          {
            ★: Widget
            __nodes__: Array<Node>
            caller: ☠
            prototype: ☠
            ...
          }
```

✓ `{ x: 6, b: "car" }`

✓ `{ nested: { y: 10, b: false } }`

🚫 `{ __nodes__: 90 }`

🚫 `myObj.prototype = { };`

✓ `function foo(x) { return x + 1; }`

✓ `foo(900)`

✓ `foo.w = "functions are objects"`

✓ `["array", "of", "strings"]`

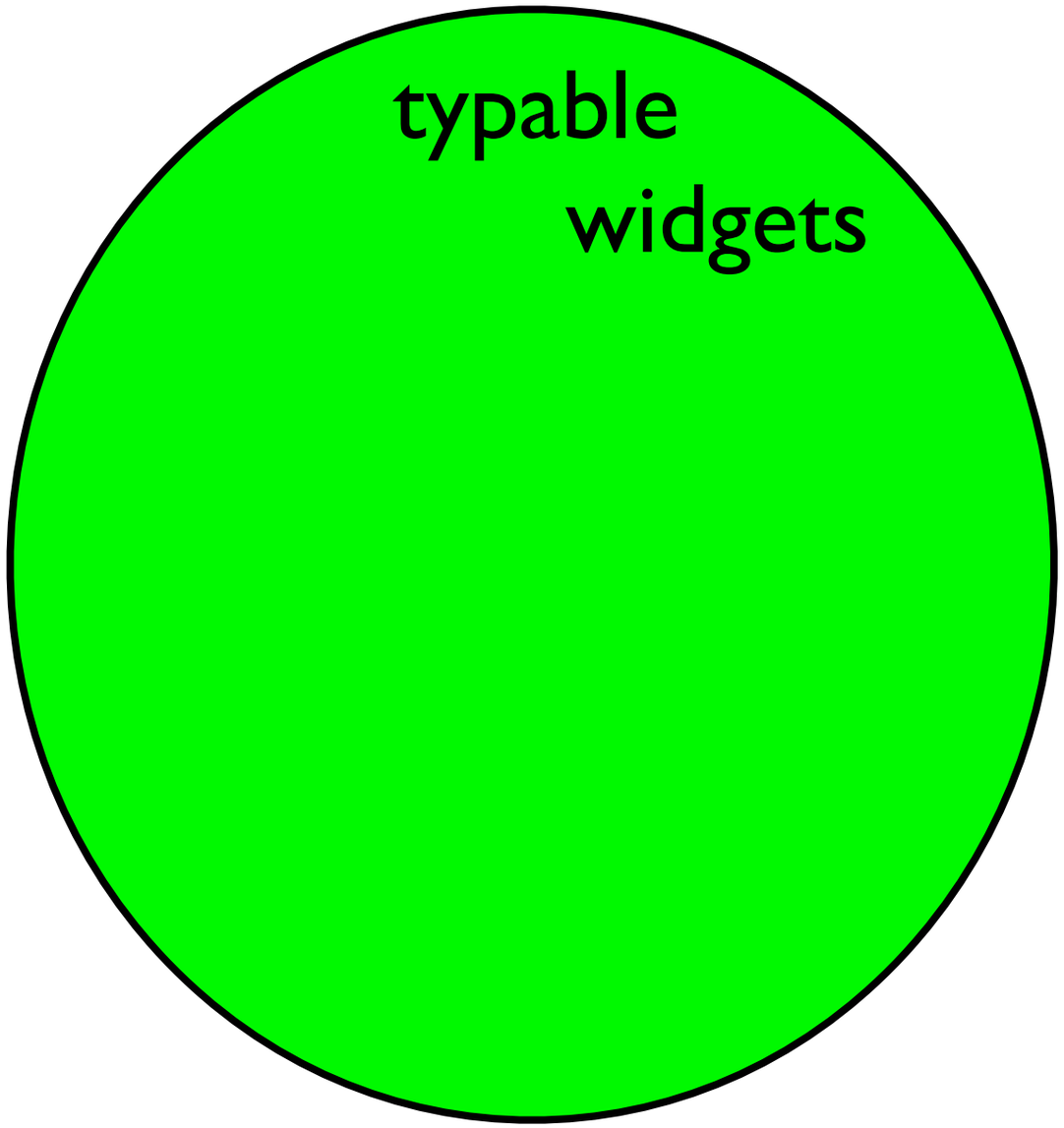
✓ `/regular[ \t]*expressions/`

`Widget := Number + String + Boolean + Undefined + Null +`

```
{ ★: Widget
  __nodes__: Array<Node>
  caller: ☠
  prototype: ☠
  ...
  code : Widget × ... → Widget
  __proto__: Object + Function + Array + ... }
```

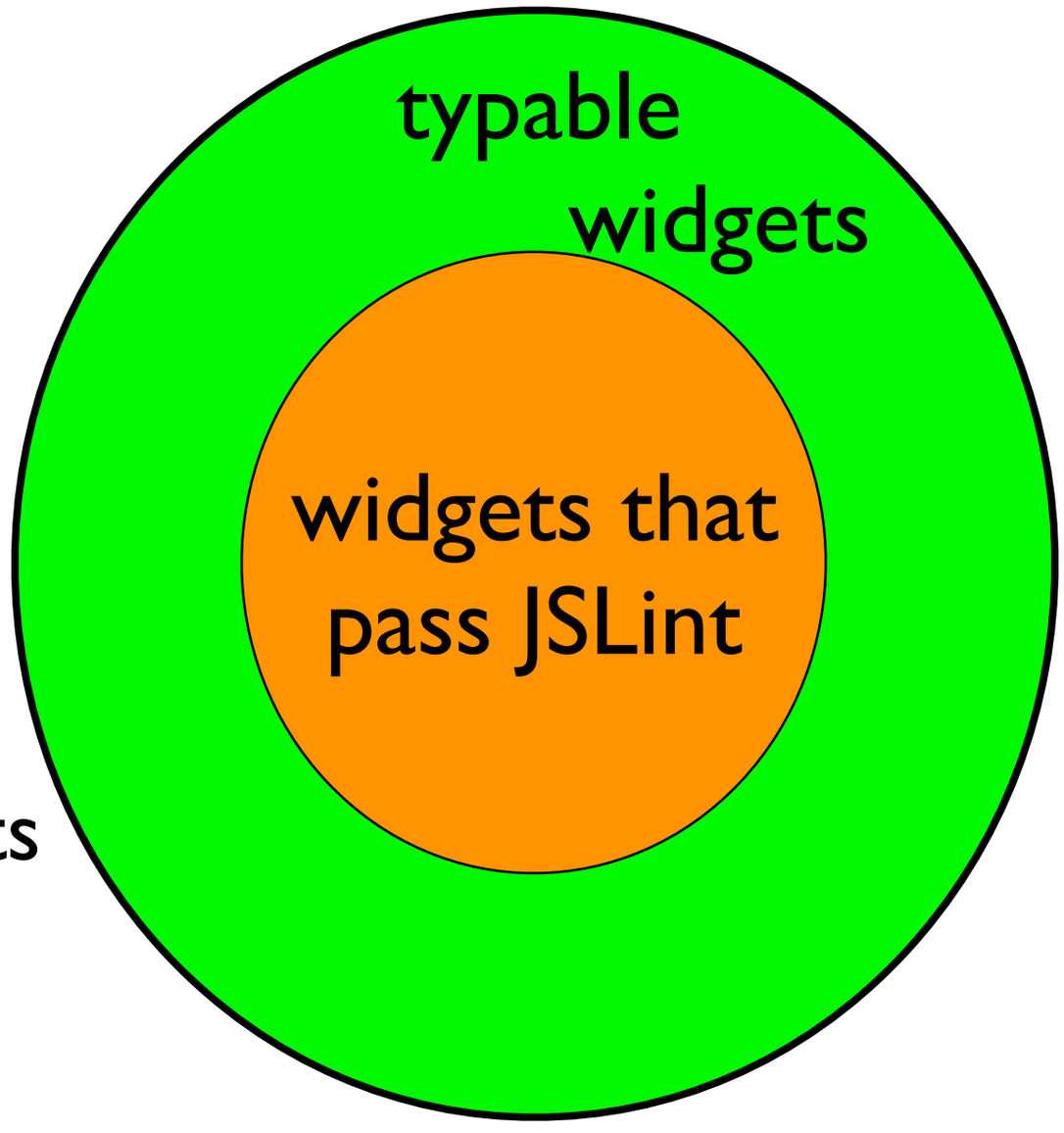
JSLint

Widget type-checker



JSLint

Widget type-checker



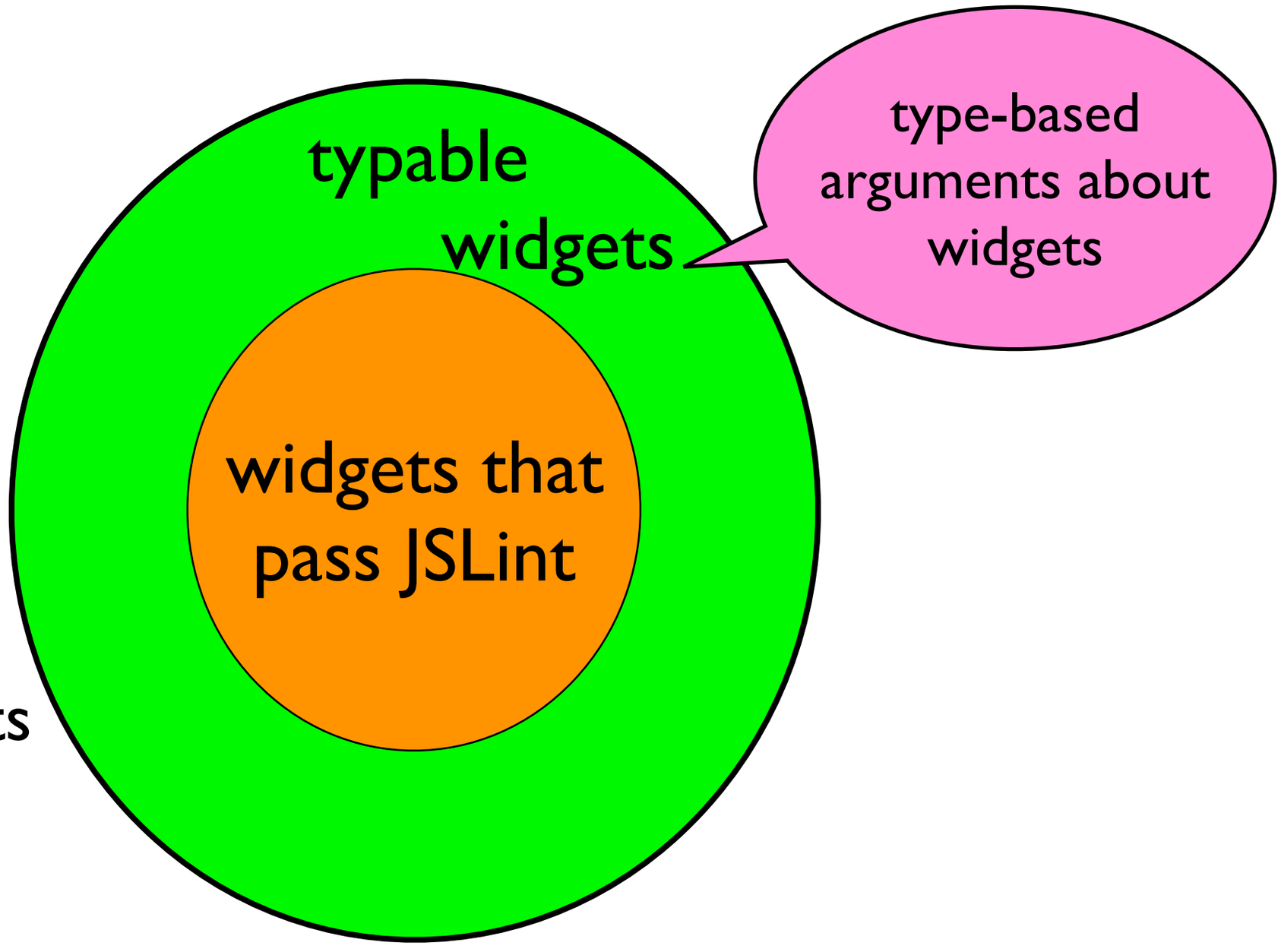
Claim:

evidence:
1,100 LOC of tests

or, passing JSLint ⇒Widget-typable

JSLint

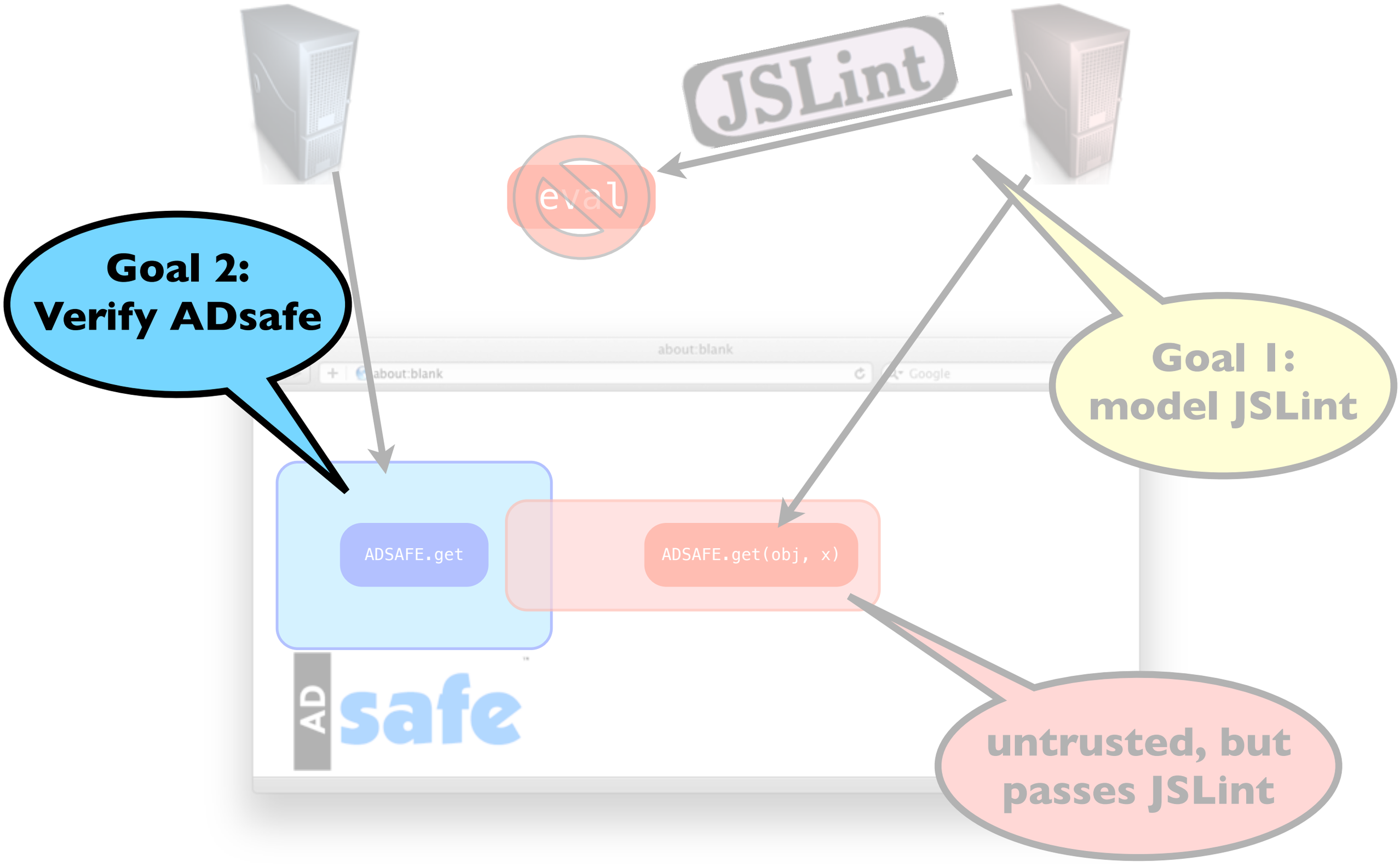
Widget type-checker



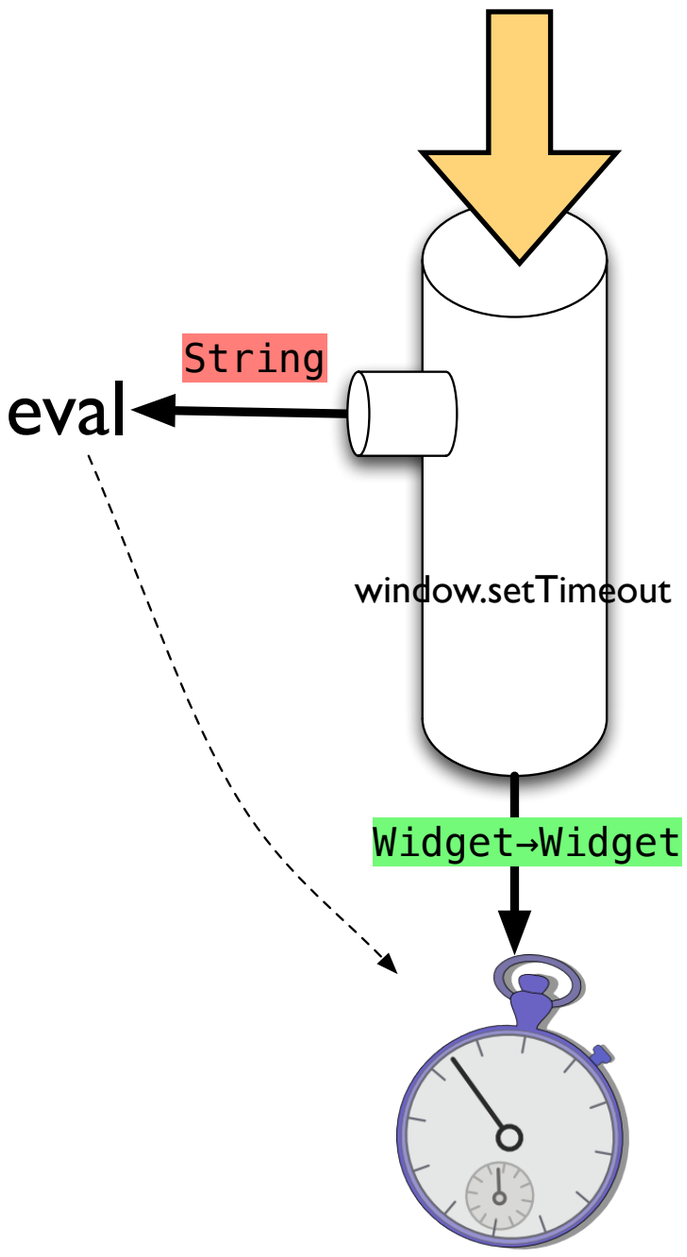
Claim:

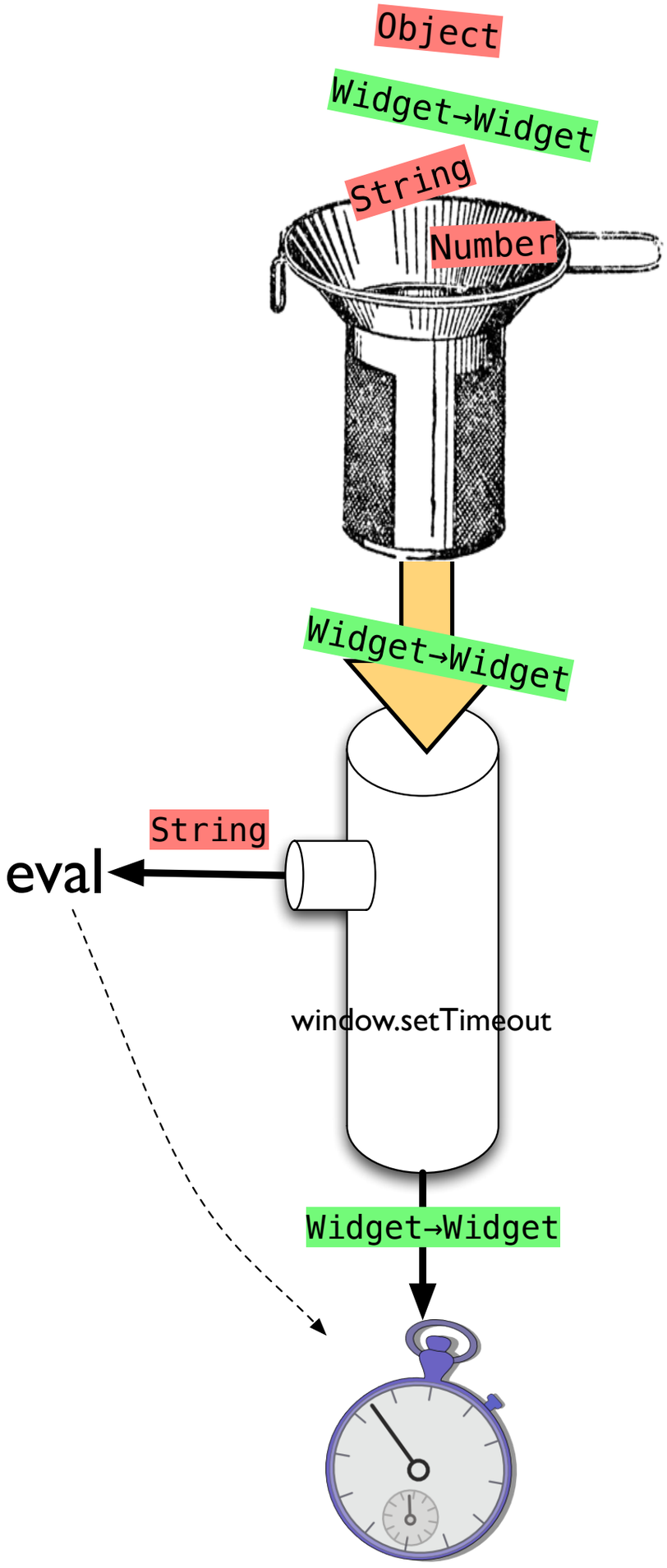
evidence:
1,100 LOC of tests

or, passing JSLint ⇒ Widget-typable


JSLint
eval
Goal 2:
Verify ADsafe
Goal 1:
model JSLint
about:blank
about:blank
Google
ADSAFE.get
ADSAFE.get(obj, x)
AD
safe
untrusted, but
passes JSLint

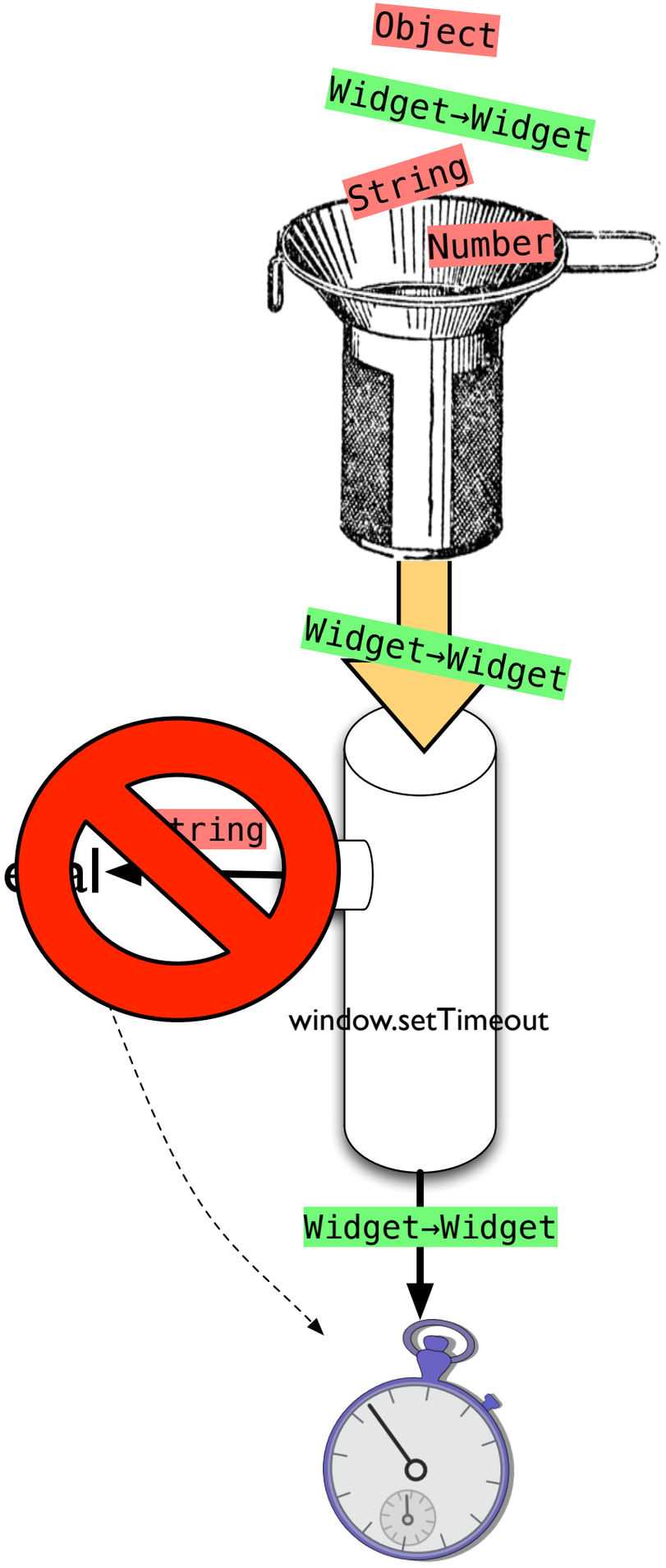
```
window.setTimeout(callback, delay);
```





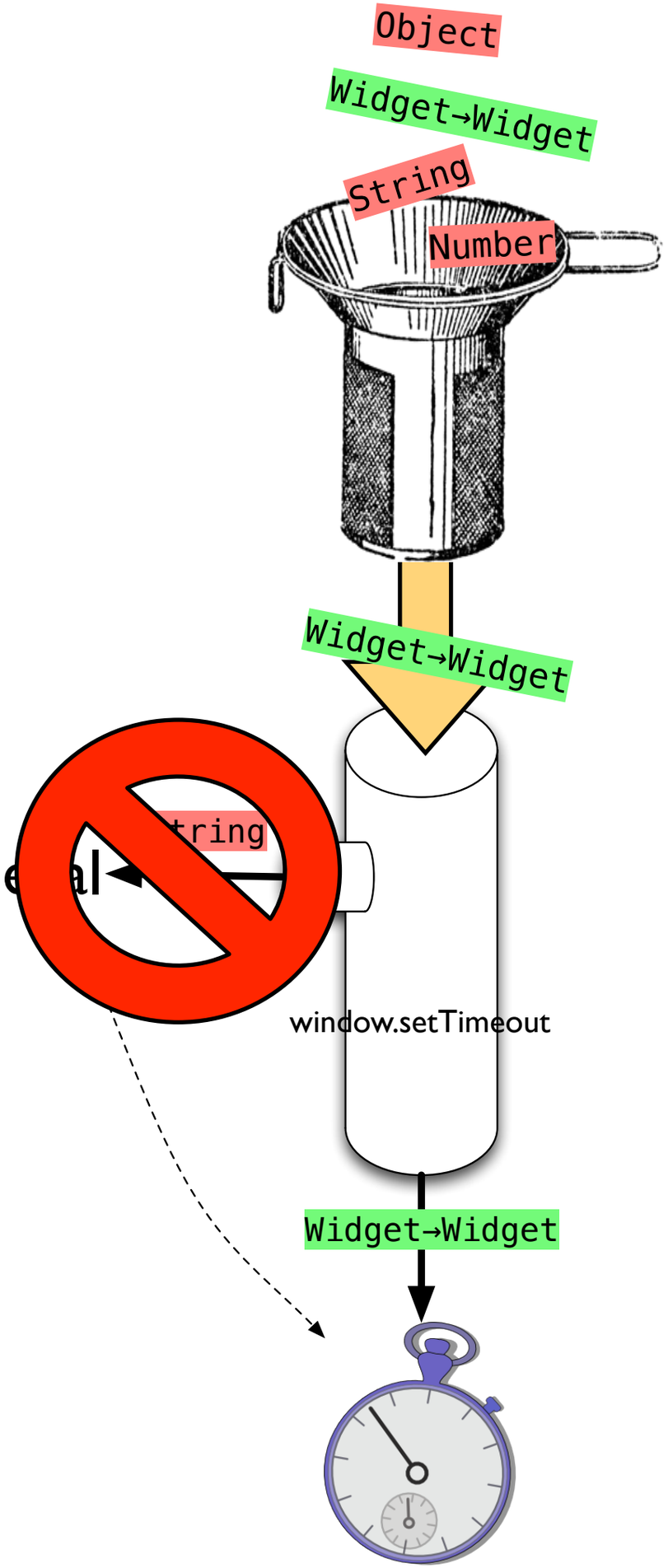

```
/*: Widget × Widget → Widget */
ADSAFE.later = function(callback, delay)
{
  if (typeof callback !== "function") {
    throw "expected function";
  }

  window.setTimeout(callback, delay);
}
```




```
window : {
  eval: ☠,
  setTimeout : (Widget × ... → Widget) × Widget → Undefined,
  …
}
```

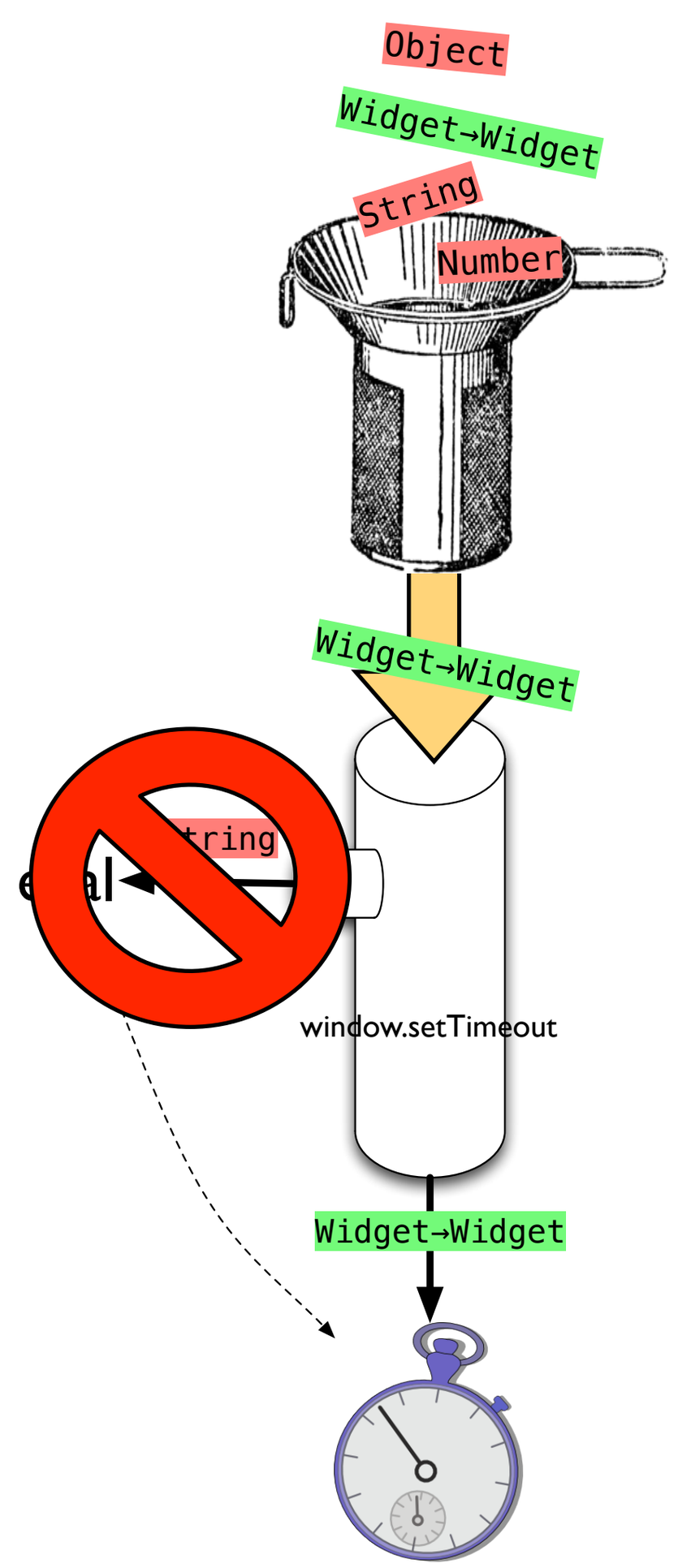
```
/*: Widget × Widget → Widget */
ADSAFE.later = function(callback, delay)
{
  if (typeof callback !== "function") {
    throw "expected function";
  }
  window.setTimeout(callback, delay);
}
```




```
window : {
  eval: ☠,
  setTimeout : (Widget × ... → Widget) × Widget → Undefined,
  …
}
```

Widget

```
/*: Widget × Widget → Widget */
ADSAFE.later = function(callback, delay)
{
  if (typeof callback !== "function") {
    throw "expected function";
  }
  window.setTimeout(callback, delay);
}
```

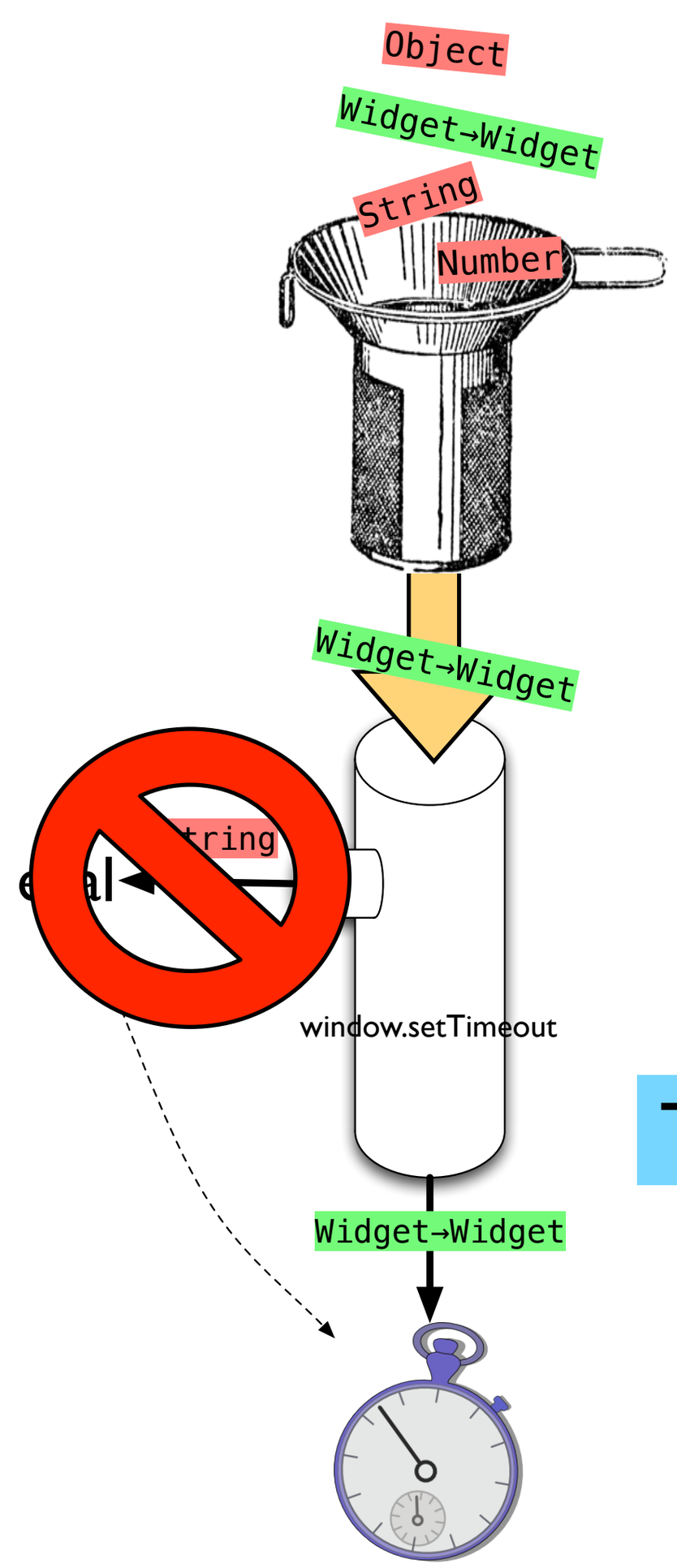



```
window : {
  eval: ☠,
  setTimeout : (Widget × ... → Widget) × Widget → Undefined,
  …
}
```

Widget

```
/*: Widget × Widget → Widget */
ADSAFE.later = function(callback, delay)
{
  if (typeof callback !== "function") {
    throw "expected function";
  }
  window.setTimeout(callback, delay);
}
```

Widget × ... → Widget




```
window : {
  eval: ☠,
  setTimeout : (Widget × ... → Widget) × Widget → Undefined,
  …
}
```

Widget

```
/*: Widget × Widget → Widget */
ADSAFE.later = function(callback, delay)
{
  if (typeof callback !== "function") {
    throw "expected function";
  }
  window.setTimeout(callback, delay);
}
```

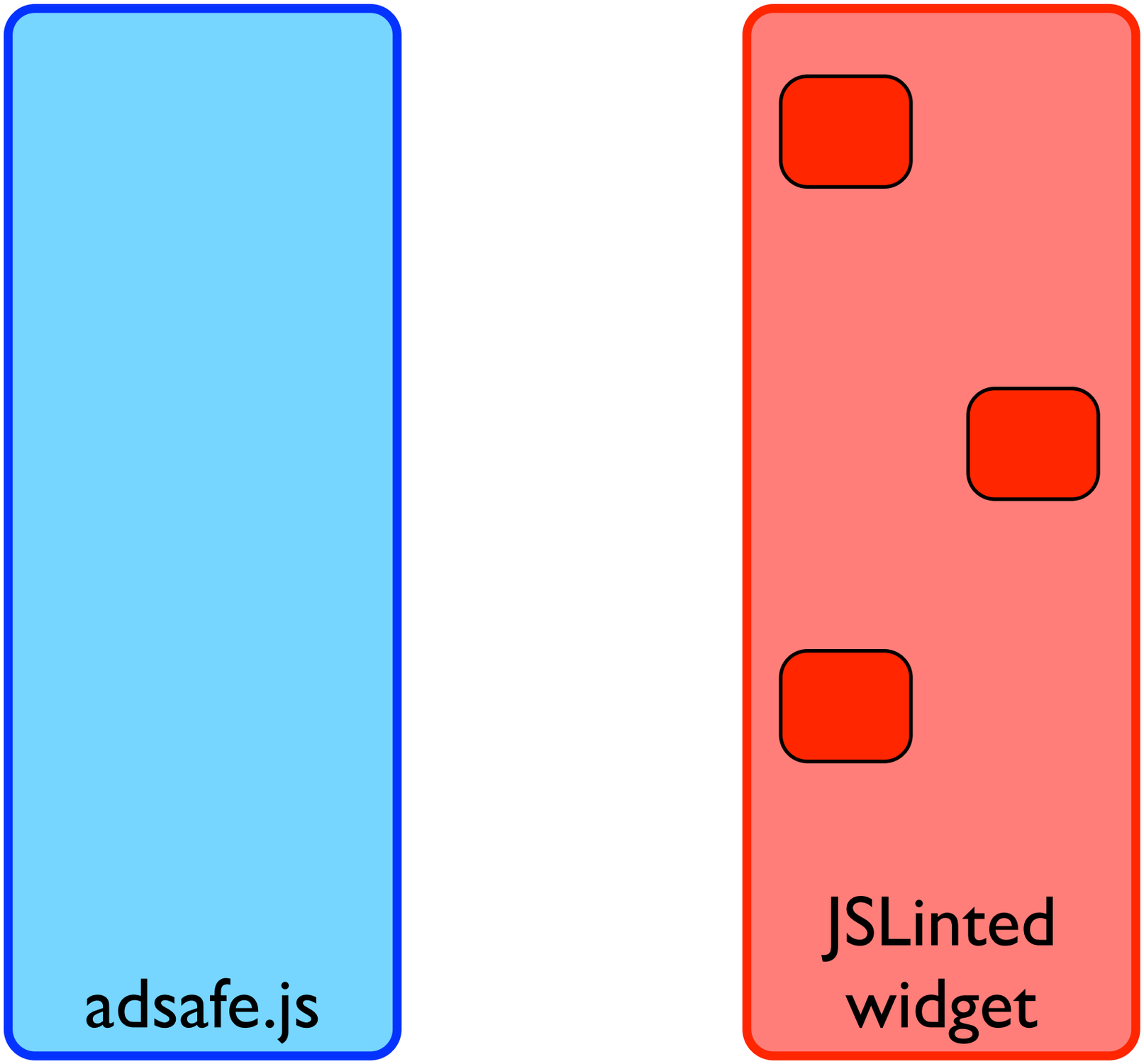
Widget × ... → Widget

This is just one kind of *if-split* we handle.

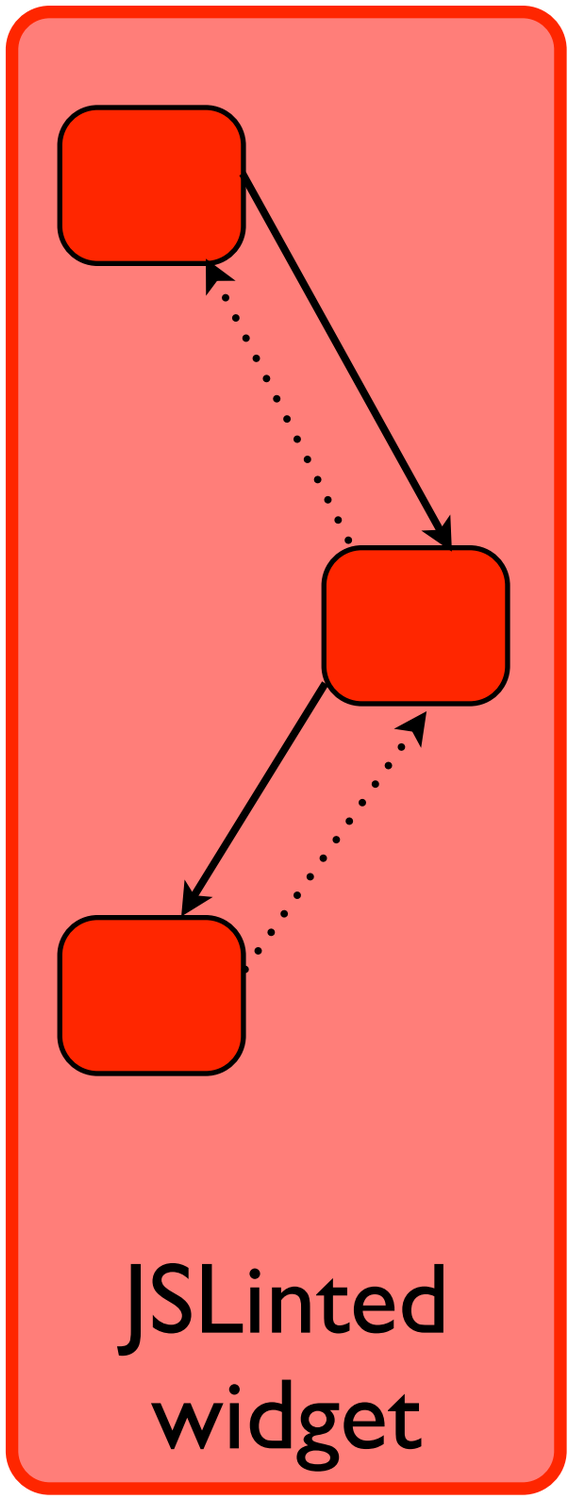
—Politz et al. *USENIX Security 2011* and Guha, Saftoiu, Krishnamurthi. *ESOP 2011*.

adsafe.js

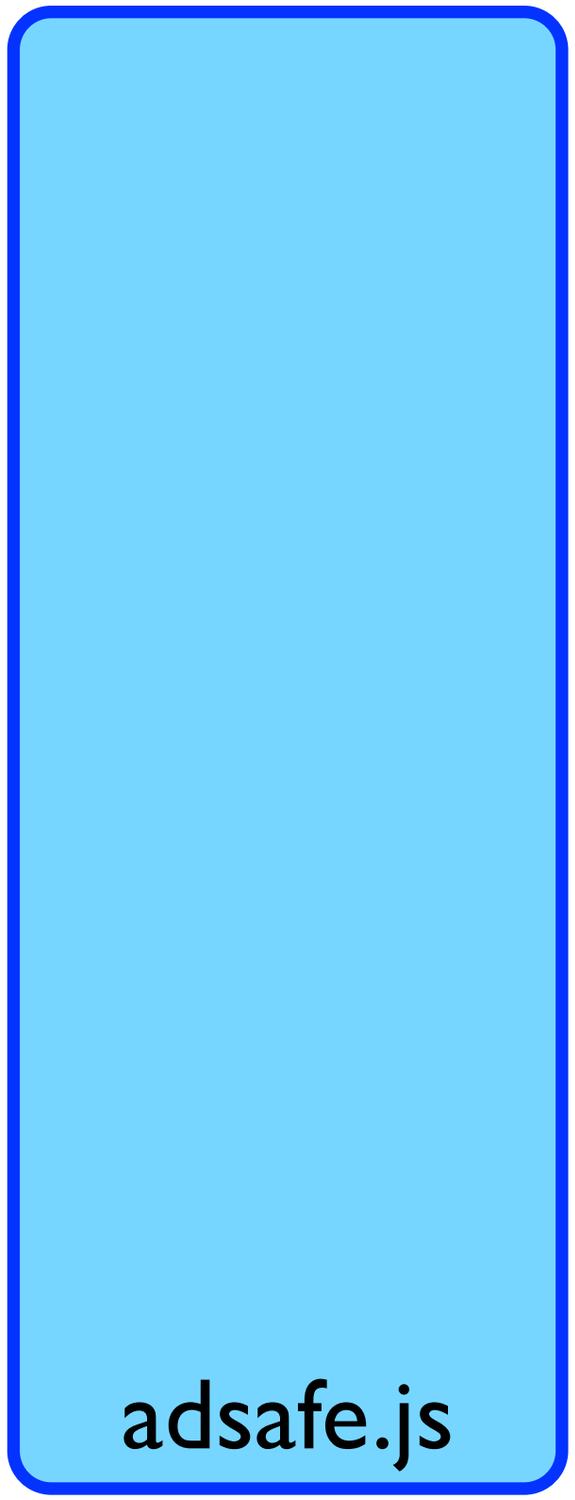
JSLinted
widget

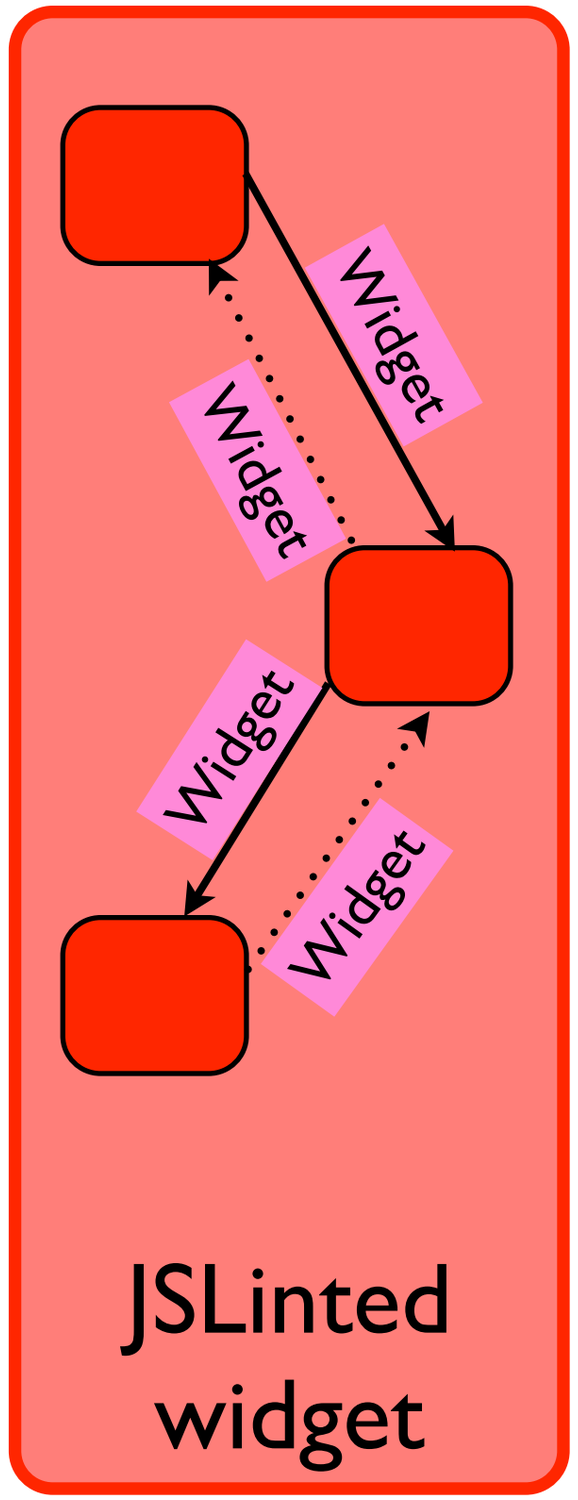

adsafe.js
JSLinted
widget

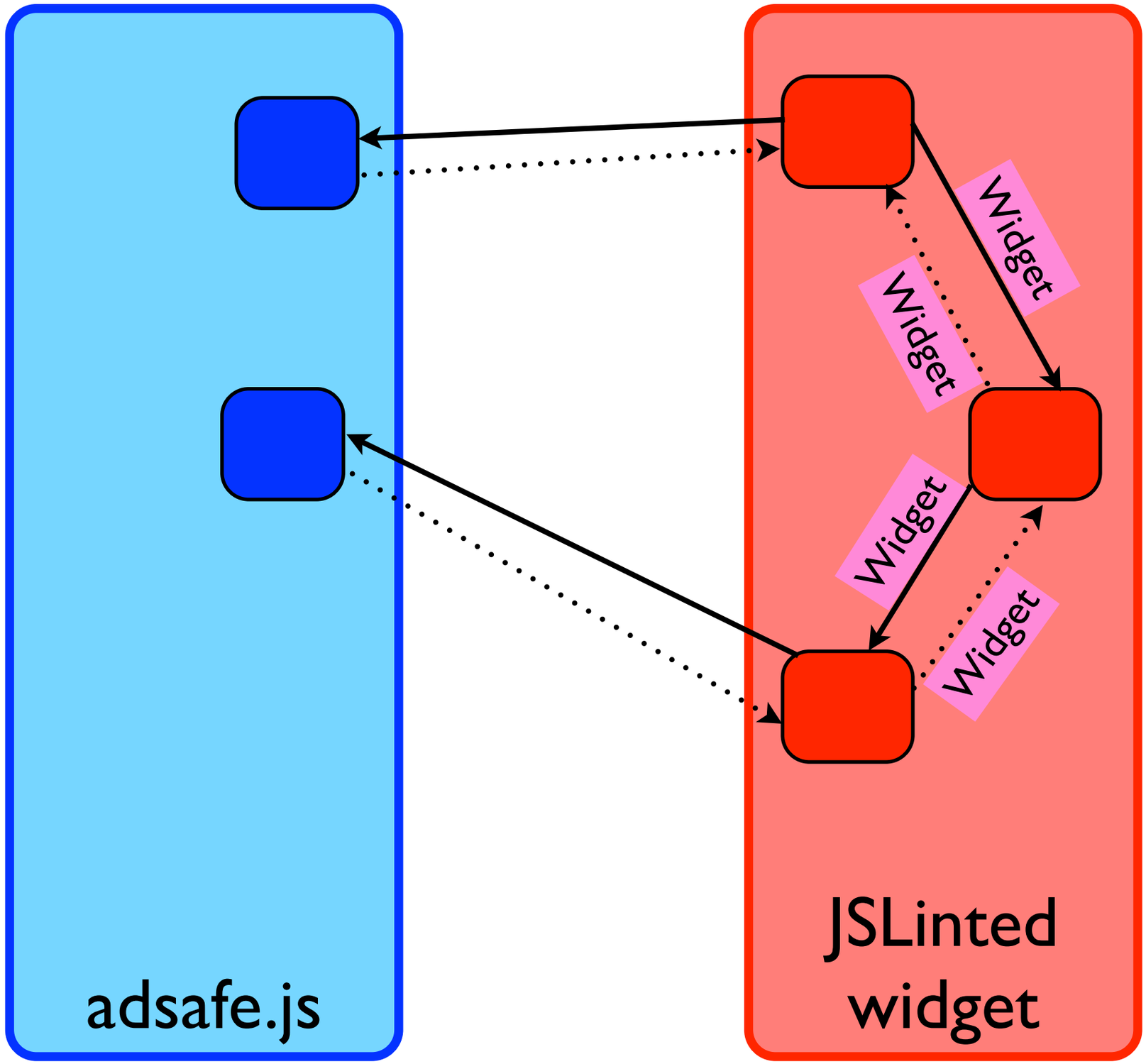
adsafe.js

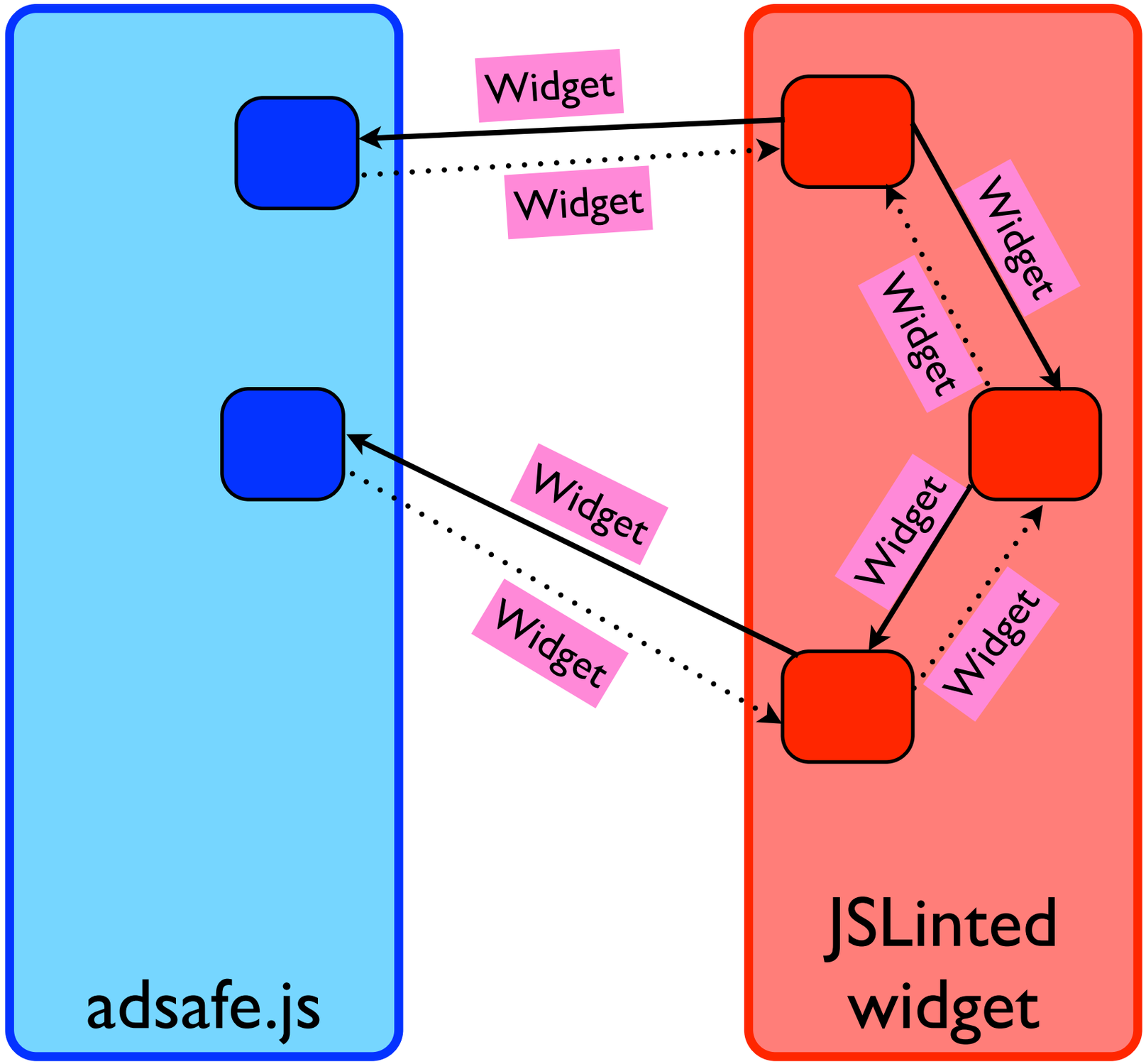


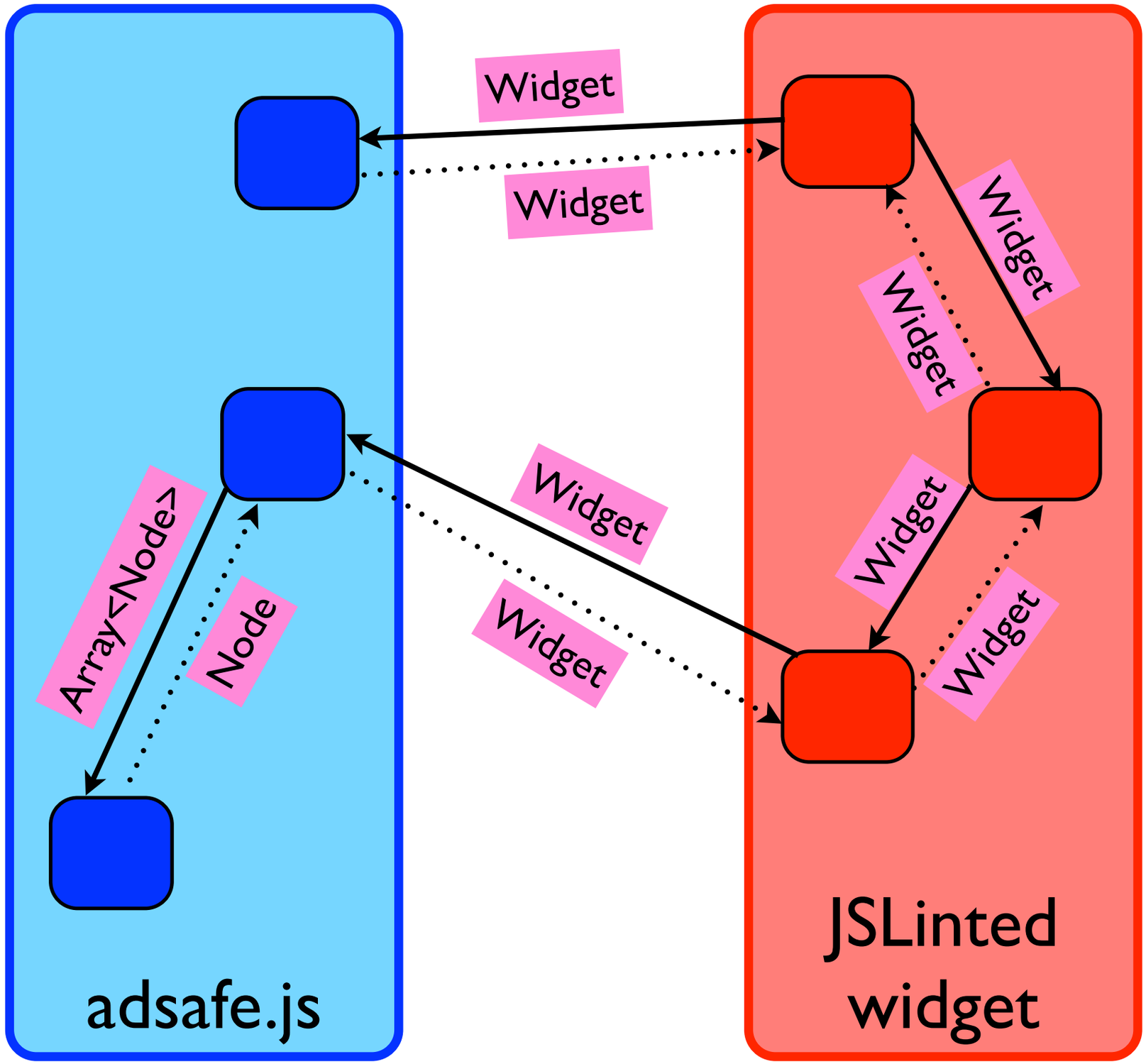
JSLinted
widget

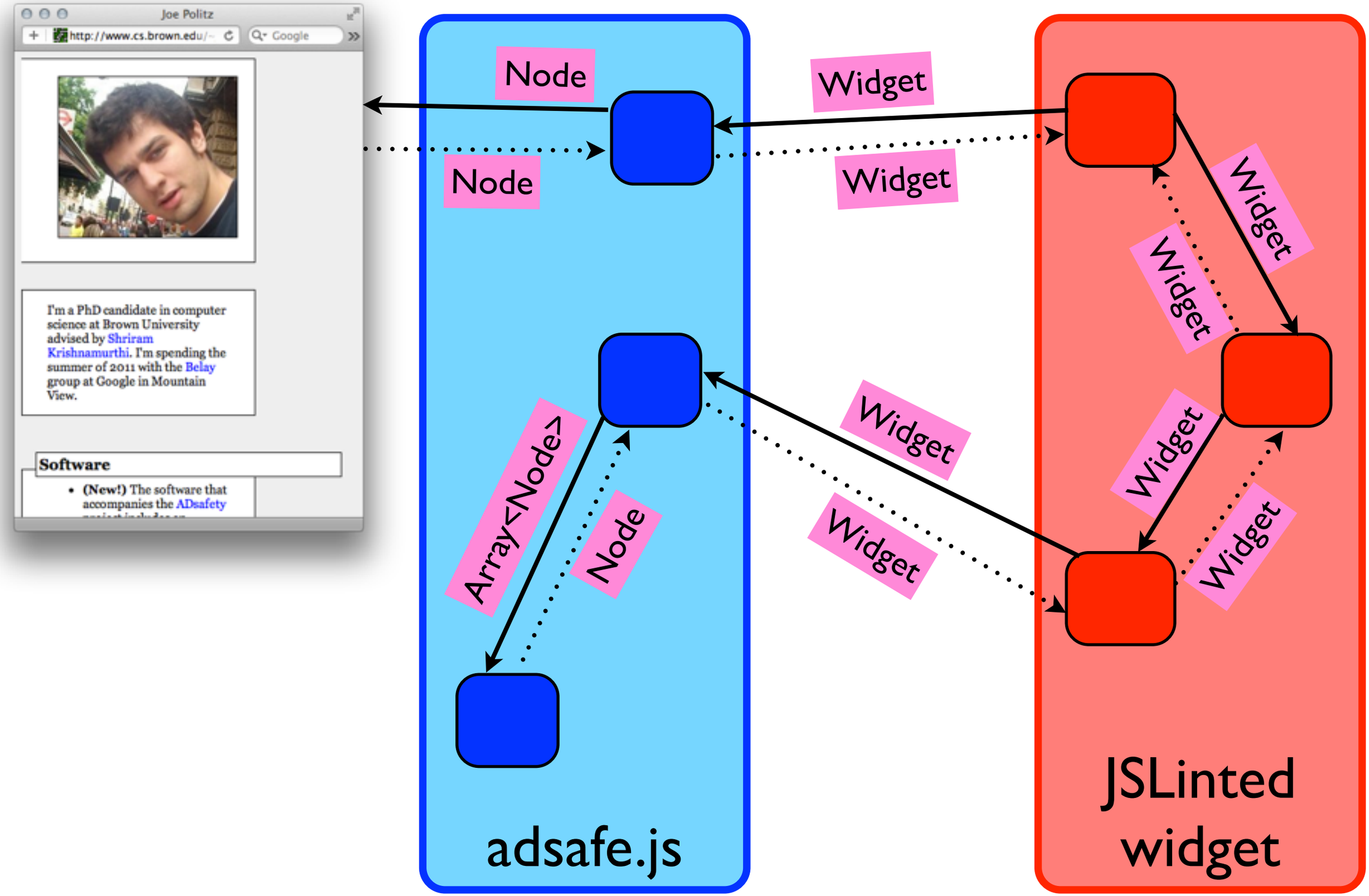

adsafe.js

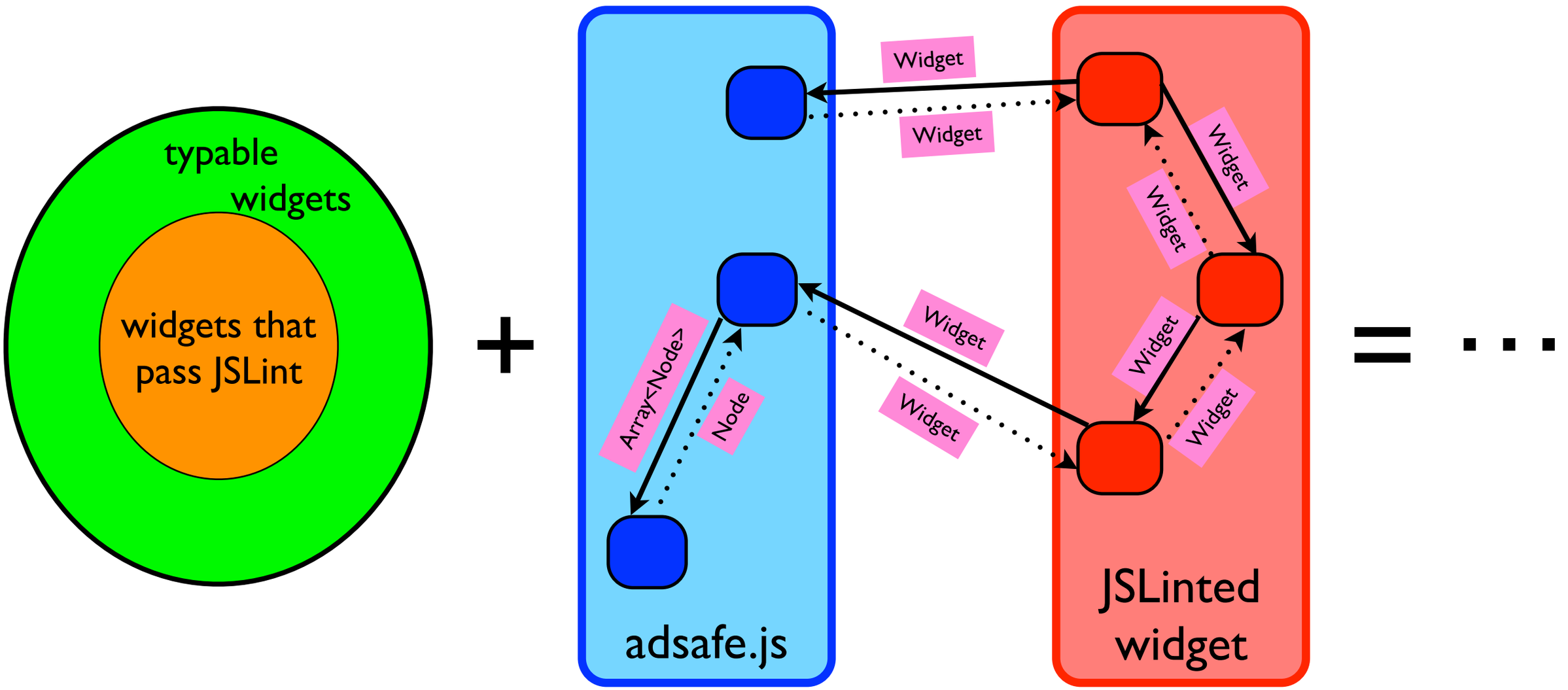

Widget
Widget
Widget
Widget
JSLinted
widget


Widget
Widget
Widget
Widget
adsafe.js
JSLinted
widget


Widget
Widget
Widget
Widget
Widget
Widget
Widget
Widget
Widget
adsafe.js
JSLinted
widget

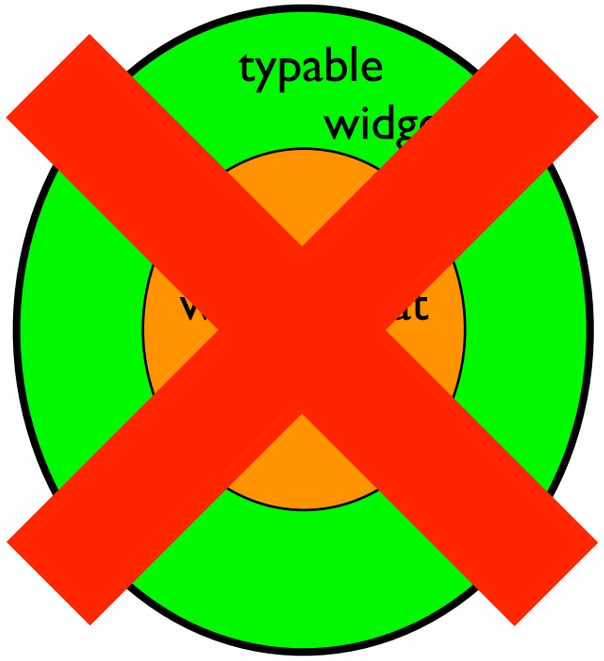

Widget
Widget
Widget
Widget
Widget
Widget
Widget
Widget
Widget
Widget
Array<Node>
Node
adsafe.js
JSLinted
widget


Joe Politz
http://www.cs.brown.edu/~
Google
I'm a PhD candidate in computer science at Brown University advised by Shriram Krishnamurthi. I'm spending the summer of 2011 with the Belay group at Google in Mountain View.
Software
(New!) The software that accompanies the ADsafety
Node
Node
Widget
Widget
Widget
Widget
Widget
Widget
Widget
Widget
Widget
Array<Node>
Node
adsafe.js
JSLinted widget

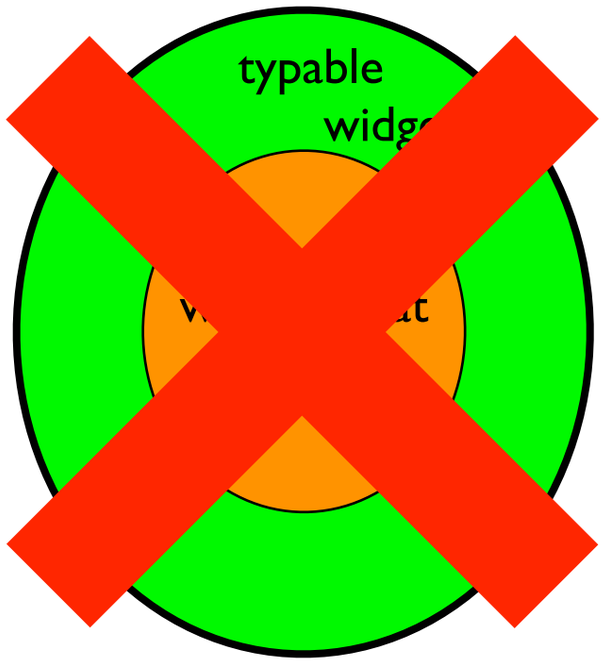


JSLint model

Type-checked ADsafe

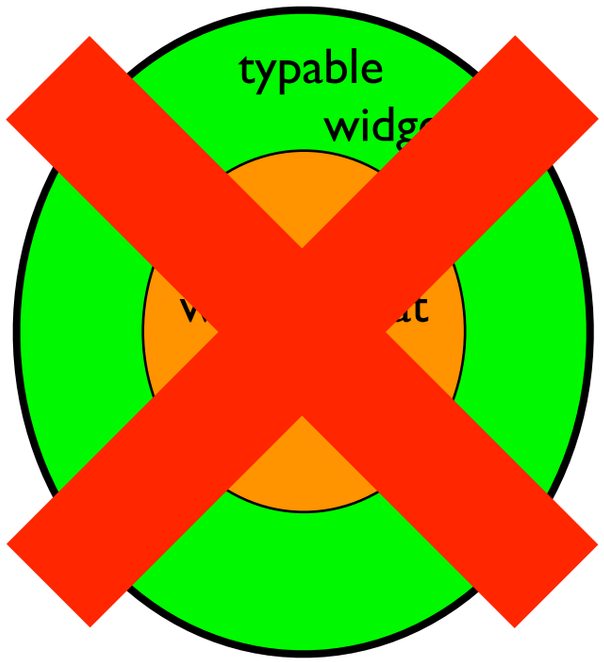



```
var fakeNode = {
  tagName: "div",
  appendChild: function(elt) {
    var win = elt.ownerDocument.defaultView;
    win.eval("alert('hacked')");
  }
};
```




```
var fakeNode = {
  tagName: "div",
  appendChild: function(elt) {
    var win = elt.ownerDocument.defaultView;
    win.eval("alert('hacked')");
  }
};
var fakeBunch = { __nodes__: [fakeNode] };
```

Rejected by JSLint




```
var fakeNode = {
  tagName: "div",
  appendChild: function(elt) {
    var win = elt.ownerDocument.defaultView;
    win.eval("alert('hacked')");
  }
};
```

```
var fakeBunch = { __nodes__: [fakeNode] };
```

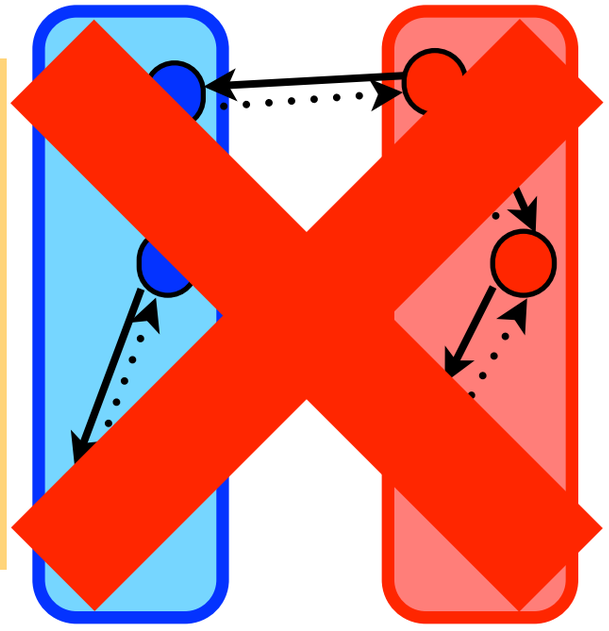
```
var fakeBunch = { '__nodes__': [fakeNode] };
```

Rejected by JSLint

Accepted by JSLint

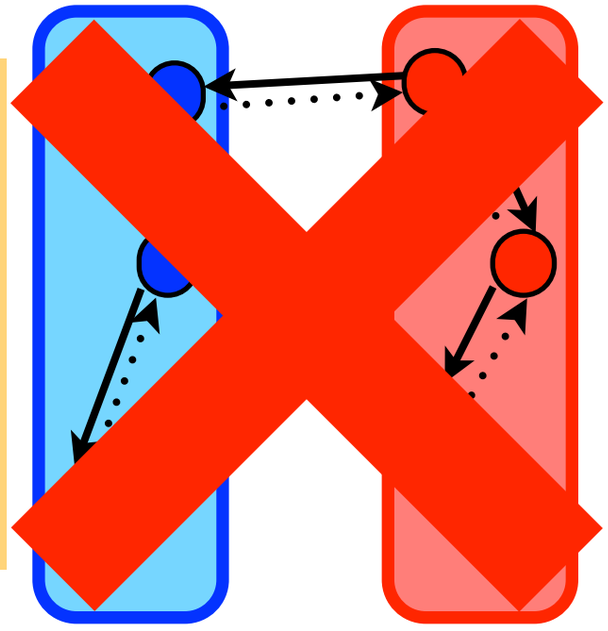
```
type error: expected Array<HTML>,
received Array<Widget>
```

```
/*: Widget × Widget → Widget */
WrappedElt.prototype.style = function(name, val) {
  var regexp = new Regexp("url");
  if (regexp.test(val)) {
    return error();
  }
  ... this.__node__.style[name] = val ...
}
```



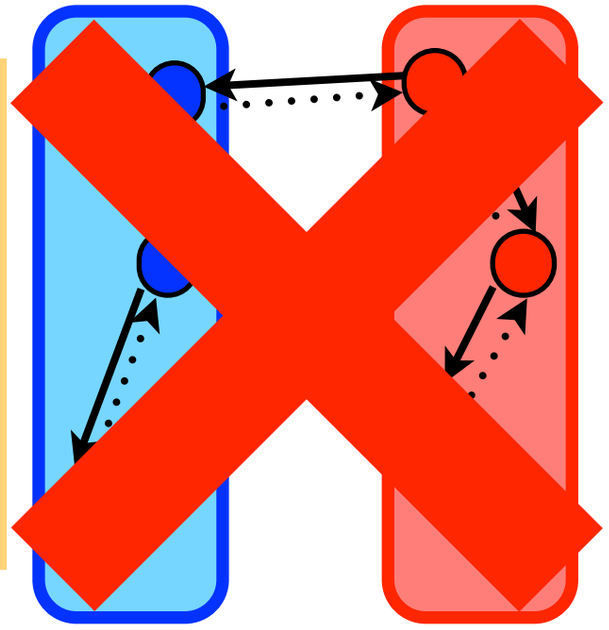
```
/*: Widget × Widget → Widget */
WrappedElt.prototype.style = function(name, val) {
  var regexp = new Regexp("url");
  if (regexp.test(val)) {
    return error();
  }
  ... this.__node__.style[name] = val ...
}
```

expected
String, received
Widget



```
/*: Widget × Widget → Widget */
WrappedElt.prototype.style = function(name, val) {
  var regexp = new Regexp("url");
  if (regexp.test(val)) {
    return error();
  }
  ... this.__node__.style[name] = val ...
}
```

expected String, received Widget



```
var firstCall = true;
var badName = {
  toString: function() {
    if (firstCall) {
      firstCall = false;
      return "font";
    }
    else {
      return "url('/evil.xml')";
    }
  }
};
```

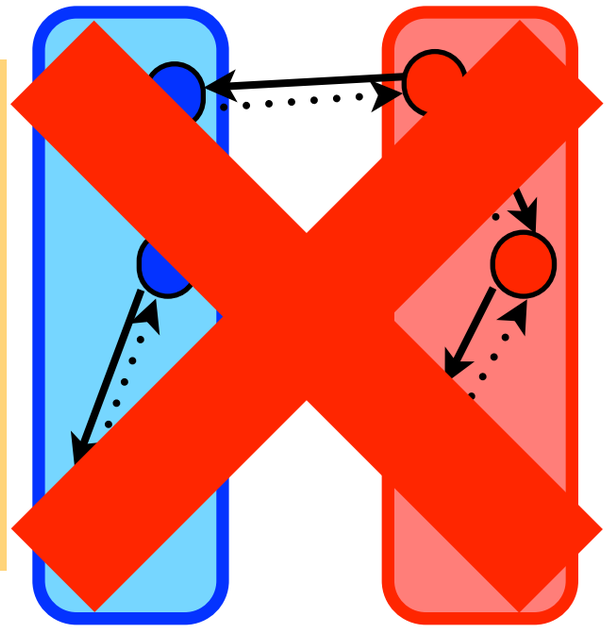
passes safety check

returns bad value

```
/*: Widget × Widget → Widget */
WrappedElt.prototype.style = function(name, val) {
  var regexp = new Regexp("url");
  if (regexp.test(val)) {
    return error();
  }
  ... this.__node__.style[name] = val ...
};
```

Fix:
check_string assertion inserted here, and in 16 other places

expected String, received Widget



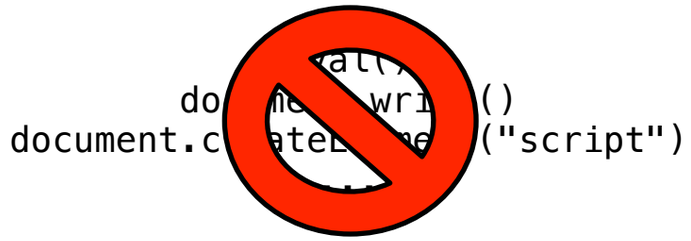
```
var firstCall = true;
var badName = {
  toString: function() {
    if (firstCall) {
      firstCall = false;
      return "font";
    }
    else {
      return "url('/evil.xml')";
    }
  }
};
```

passes safety check

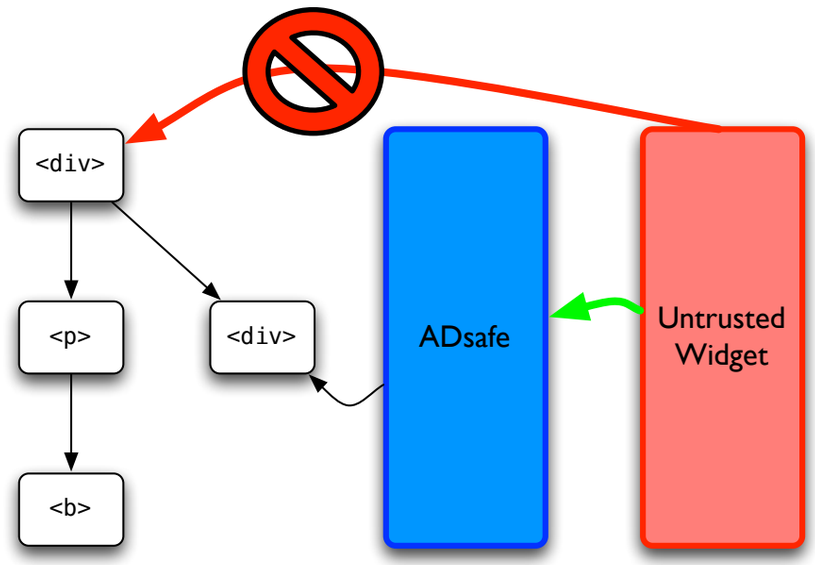
returns bad value

Definition 1 (ADsafety): If all embedded widgets pass JSLint, then:

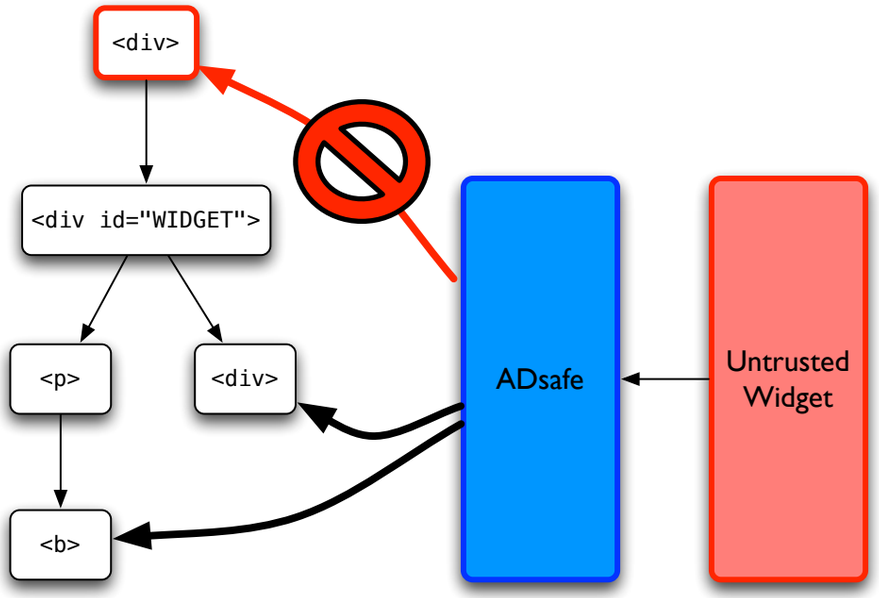
1. Widgets cannot load new code at runtime, or cause ADsafe to load new code on their behalf;



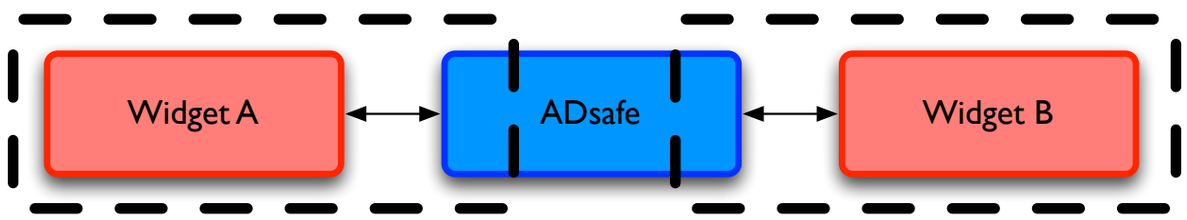
2. Widgets cannot obtain direct references to DOM nodes;



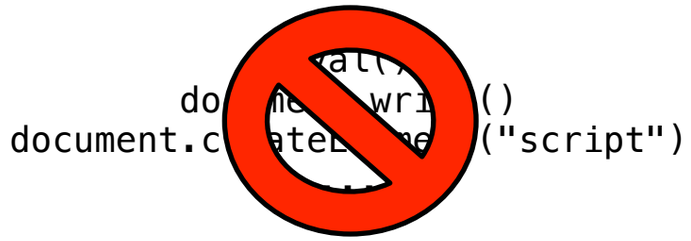
3. Widgets cannot affect the DOM outside of their subtree; and



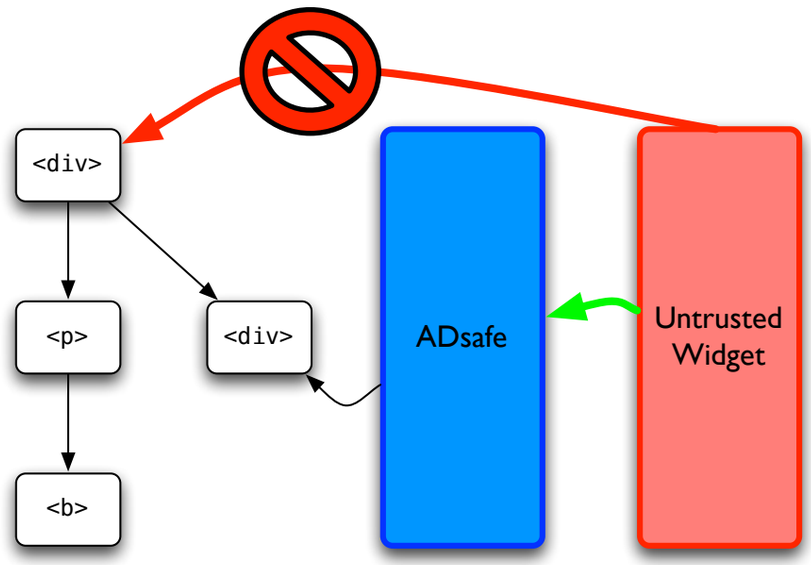
4. Multiple widgets on the same page cannot communicate.



Definition 1 (ADsafety): If all embedded widgets pass JSLint, then:

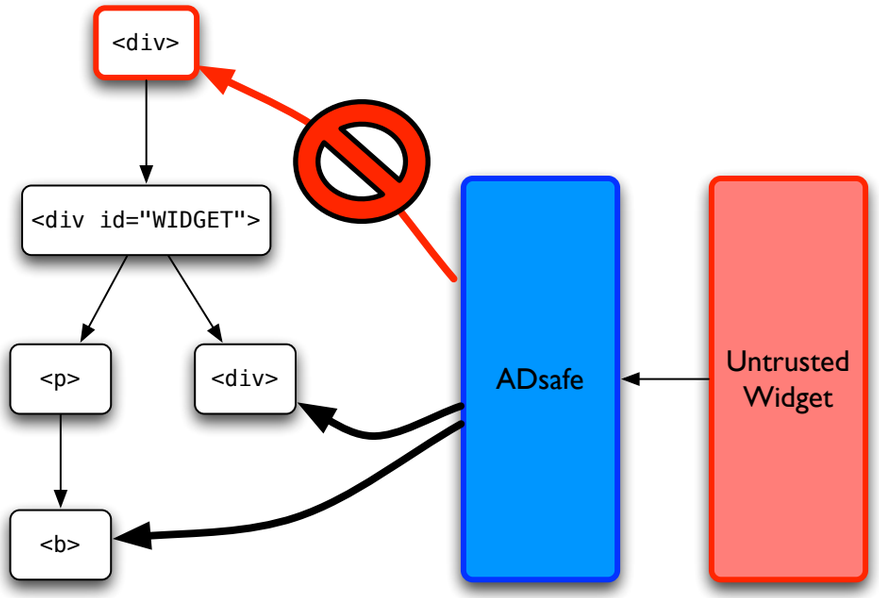


✓ 1. Widgets cannot load new code at runtime, or cause ADsafe to load new code on their behalf;

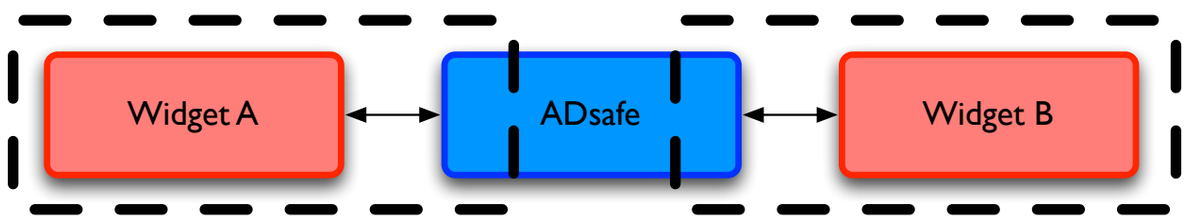


2. Widgets cannot obtain direct references to DOM nodes;

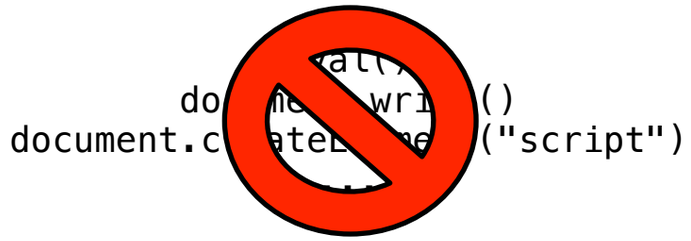
3. Widgets cannot affect the DOM outside of their subtree; and



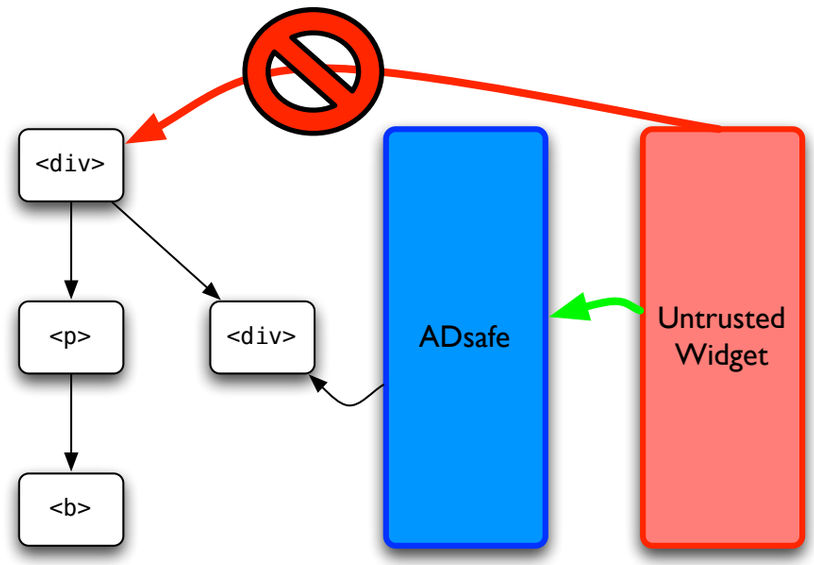
4. Multiple widgets on the same page cannot communicate.



Definition 1 (ADsafety): If all embedded widgets pass JSLint, then:

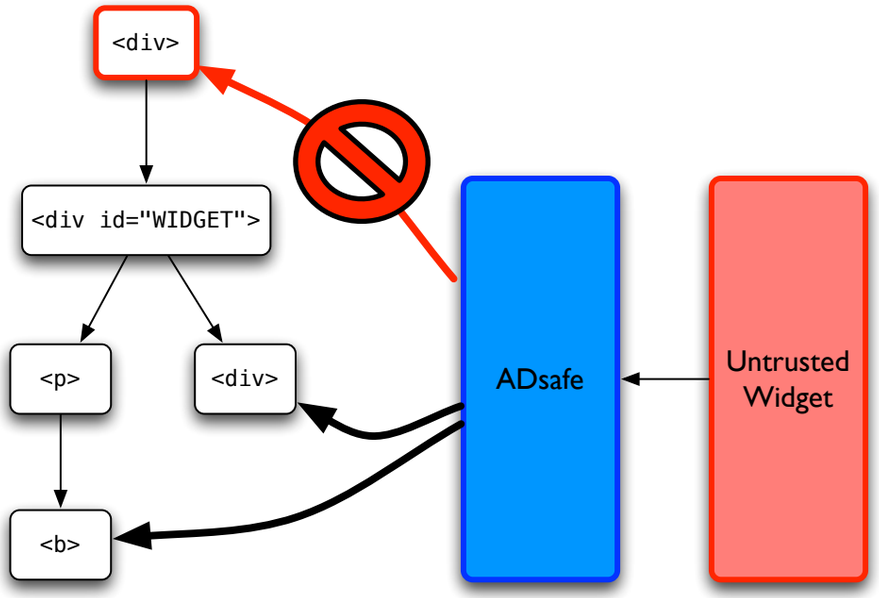


✓ 1. Widgets cannot load new code at runtime, or cause ADsafe to load new code on their behalf;

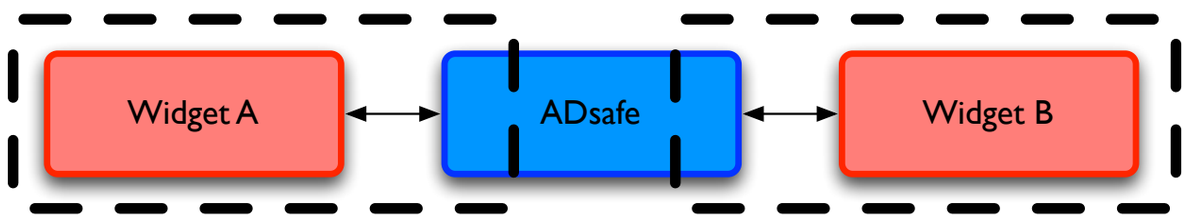


✓ 2. Widgets cannot obtain direct references to DOM nodes;

3. Widgets cannot affect the DOM outside of their subtree; and

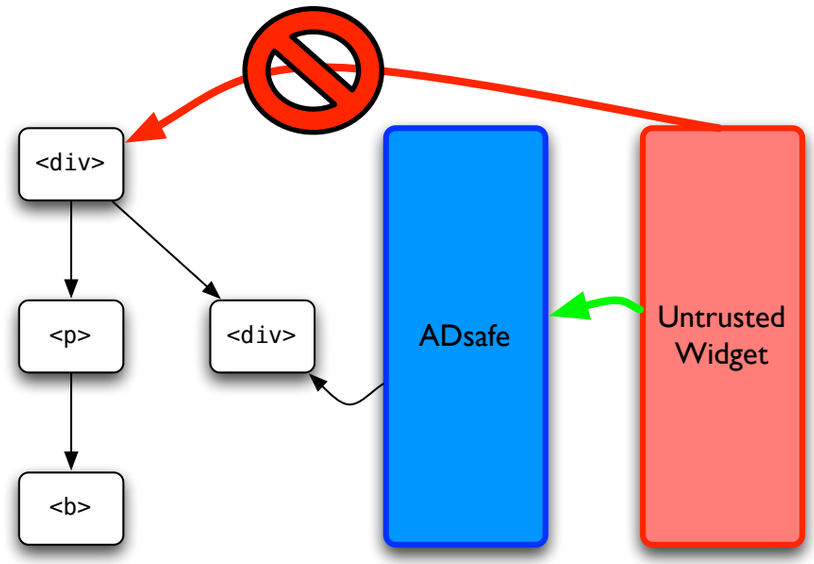


4. Multiple widgets on the same page cannot communicate.



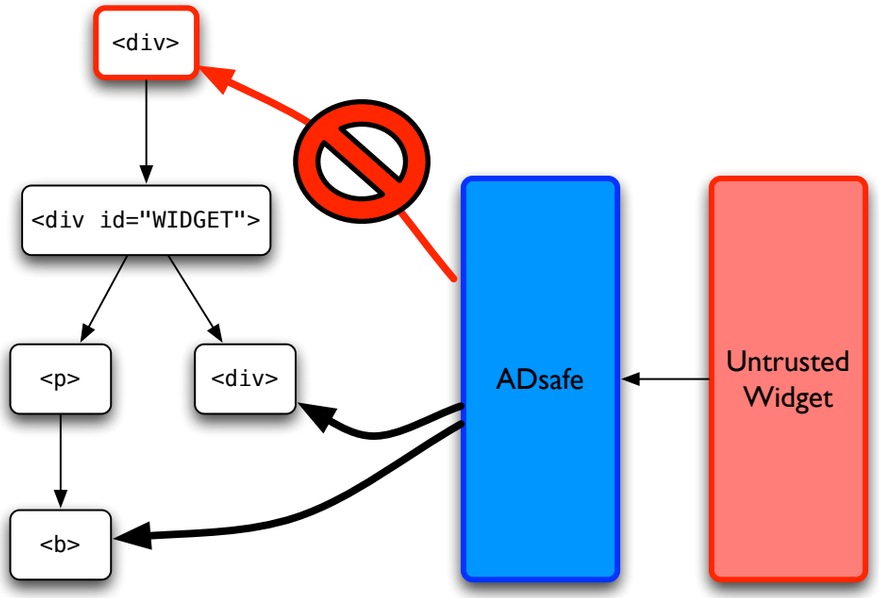
Definition 1 (ADsafety): If all embedded widgets pass JSLint, then:

✓ 1. Widgets cannot load new code at runtime, or cause ADsafe to load new code on their behalf;

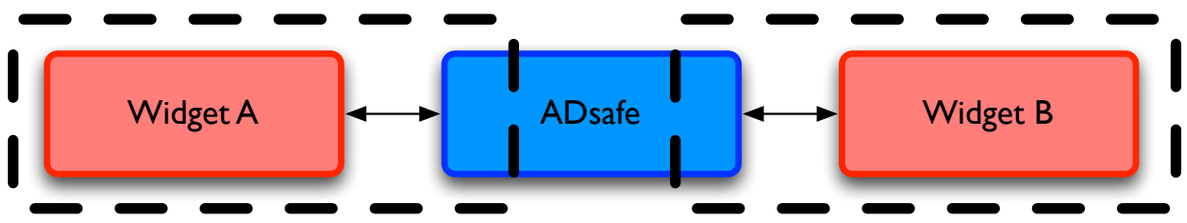


✓ 2. Widgets cannot obtain direct references to DOM nodes;

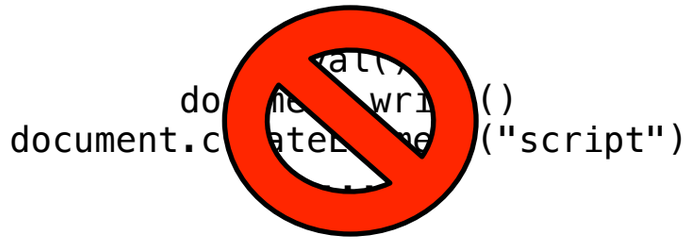
3. Widgets cannot affect the DOM outside of their subtree; and



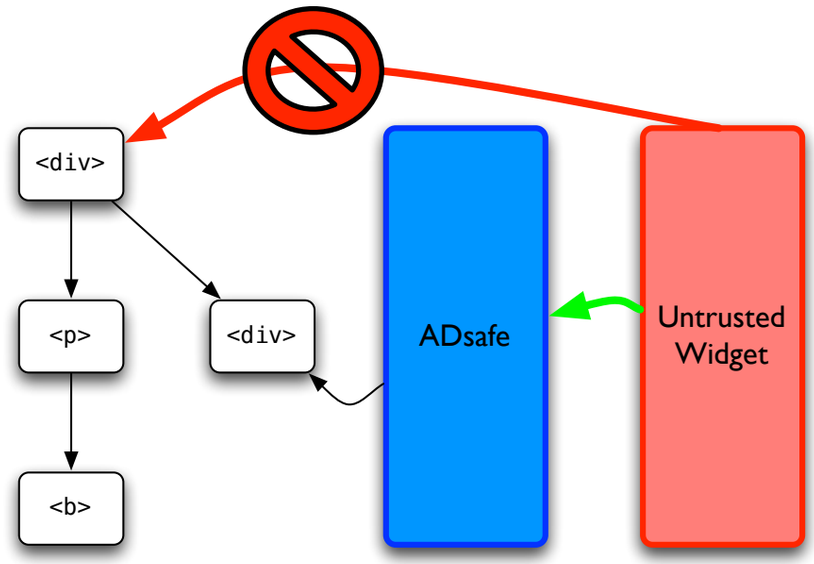
4. Multiple widgets on the same page cannot communicate.



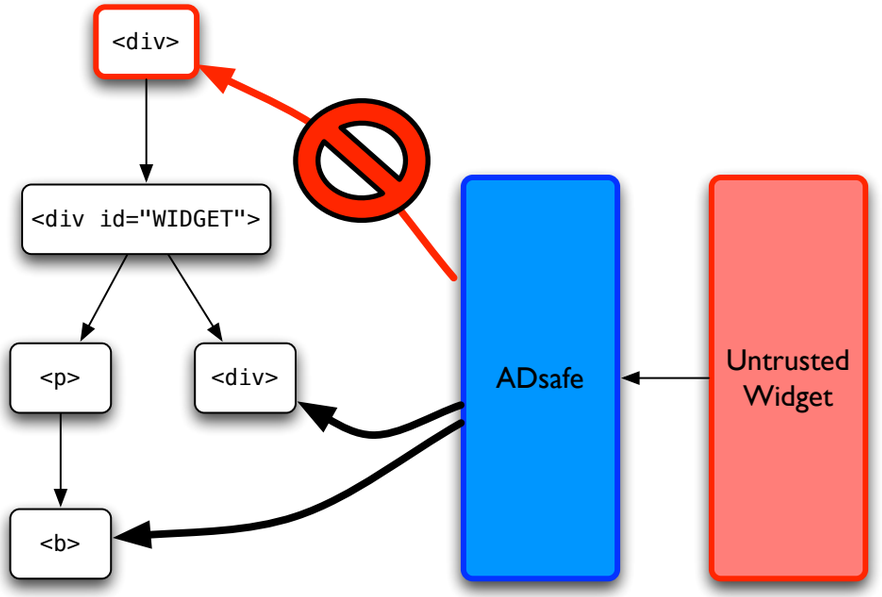
Definition 1 (ADsafety): If all embedded widgets pass JSLint, then:



✓ 1. Widgets cannot load new code at runtime, or cause ADsafe to load new code on their behalf;



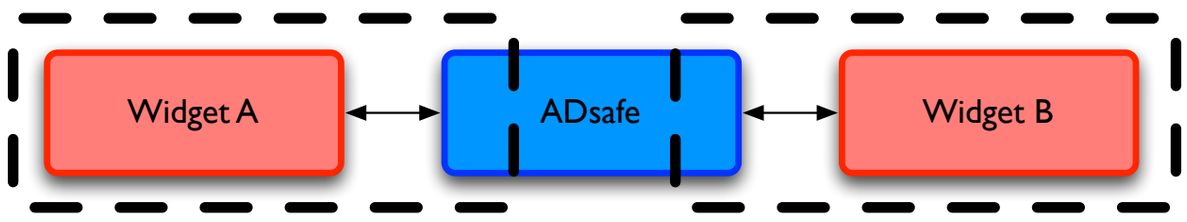
✓ 2. Widgets cannot obtain direct references to DOM nodes;



3. Widgets cannot affect the DOM outside of their subtree; and

Retracted

4. Multiple widgets on the same page cannot communicate.

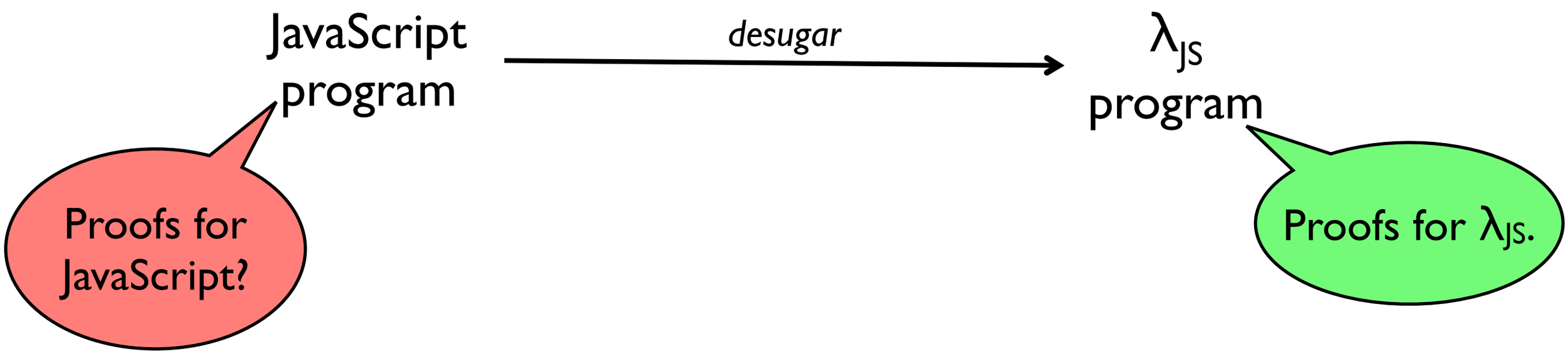


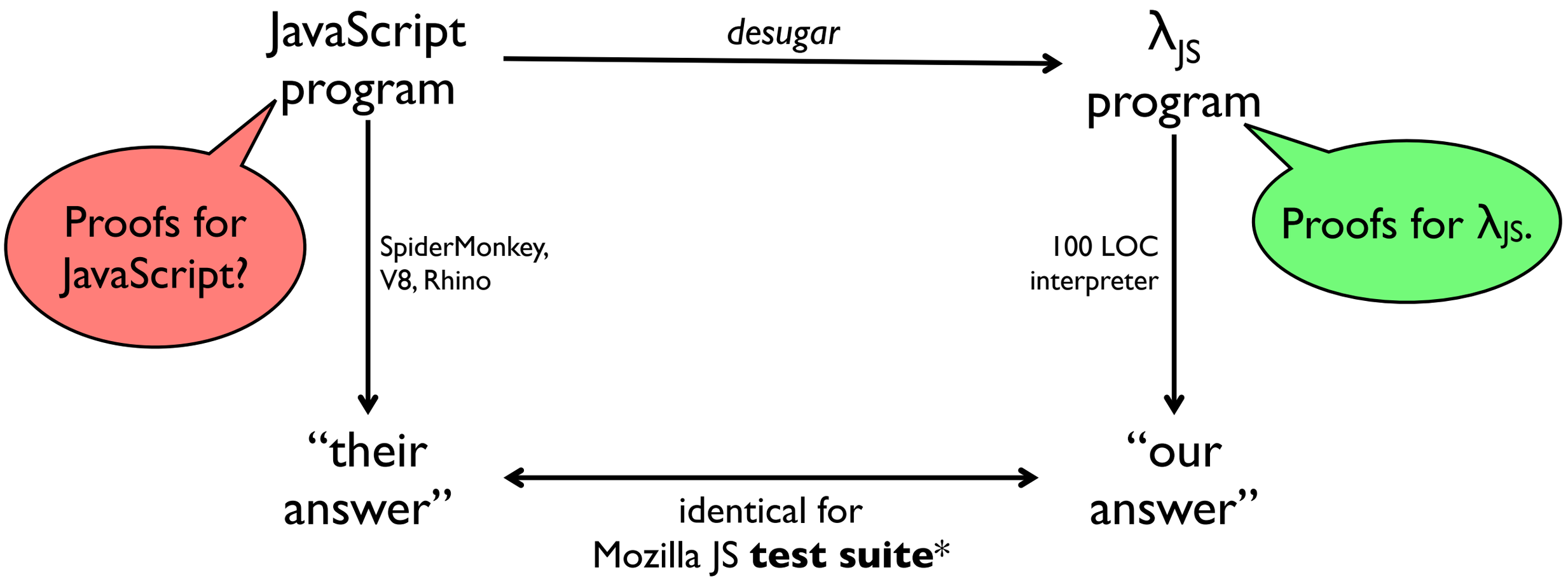
Caveats:

- 11 LOC unverified
- subtree property unverified

JavaScript program

Proofs for JavaScript?


JavaScript program
desugar
λJS program
Proofs for JavaScript?
Proofs for λJS.



— Guha, Saftoiu, Krishnamurthi. *ECOOP 2010.*

```
banned = {
  'arguments' : true,
  callee      : true,
  caller      : true,
  constructor : true,
  'eval'      : true,
  prototype   : true,
  stack       : true,
  unwatch     : true,
  valueOf     : true,
  watch       : true
}
```

+

```
function reject_global(that) {
  if (that.window) {
    error();
  }
}
```

+

```
if (/url/i.test(string_check(value[i]))) {
  error('ADsafe error.');
}
```

and other patterns...

```
banned = {
  'arguments' : true,
  callee      : true,
  caller      : true,
  constructor : true,
  'eval'      : true,
  prototype   : true,
  stack       : true,
  unwatch     : true,
  valueOf     : true,
  watch       : true
}
```

+

```
function reject_global(that) {
  if (that.window) {
    error();
  }
}
```

+

```
if (/url/i.test(string_check(value[i]))) {
  error('ADsafe error.');
}
```

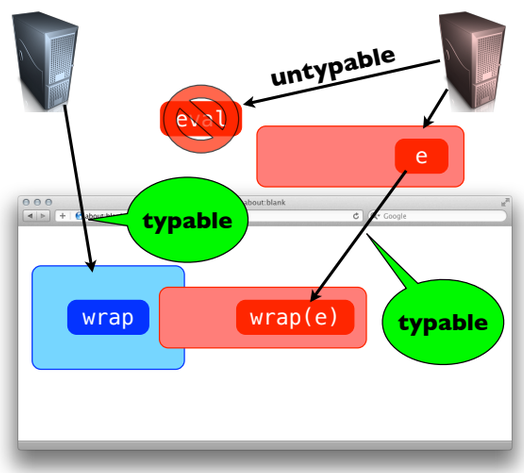
and other patterns...

... can be succinctly expressed with types

```
Widget := Number + String + Boolean + Undefined + Null +
          {
          ★: Widget
          __nodes__: Array<Node>
          caller: ☠
          prototype: ☠
          ...
          code : Widget × ... → Widget
          __proto__: Object + Function + Array + ...
          }
```

Conclusion

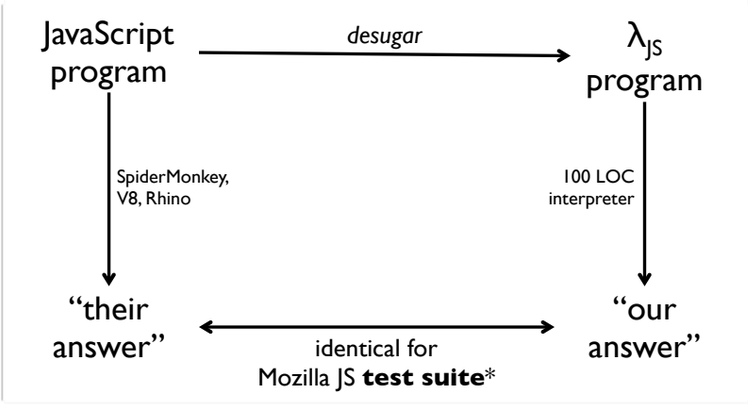
1. Model sandbox as a type system



2. Object types for JavaScript (★ and ☠)

```
Widget := Number + String + Boolean + Undefined + Null +
          {★: Widget
           __proto__: Object + Function + Array + ...
           code : Widget × ... → Widget
           arguments: ☠
           caller: ☠
           ...}
```

3. Proofs over tractable JavaScript semantics



Spiridon Aristides Eliopoulos

Extra Slides

Unverified Code

```
    function F() {};

ADSAFE.create = typeof Object.create === 'function' ? Object.create : function (o) {
  F.prototype = typeof o === 'object' && o ? o : Object.prototype;
  return new F();
};
```

```
/*: (banned → True) & (not_banned → False) */
function reject_name(name) {
  return banned[name] ||
    ((typeof name !== 'number' || name < 0) &&
    (typeof name !== 'string'  || name.charAt(0) === '_' ||
    name.slice(-1) === '_'     || name.charAt(0) === '-'));
}
```

Theorems

Lemma 1 (Type Preservation) *If, for an expression* e*, type* T*, environment* Γ *and abstract heap* Σ*,*

1. $\Sigma \vdash \sigma$,
2. $\Sigma; \Gamma \vdash e : T$, *and*
3. $\sigma e \rightarrow \sigma' e'$;

then there exists a Σ' *with* $\Sigma' \vdash \sigma'$ *and* $\Sigma'; \Gamma \vdash e' : T$.

Theorem 1 (ADsafety) *For all widgets* p*, if*

1. *all subexpressions of* p *are* Widget*-typable,*
2. `adsafe.js` *is typable,*
3. `adsafe.js` *runs before* p*, and*
4. $\sigma p \rightarrow \sigma' p'$ *(single-step reduction),*

then at every step p'*,* p' *also has the type* Widget*.*

Full Widget Type

$$\text{Widget} = \mu\alpha.\ \text{Str} \cup \text{Num} \cup \text{Null} \cup \text{Bool} \cup \text{Undef} \cup \text{Ref}\left\{\begin{array}{l} \text{Object} \cup \text{Function} \\ \textit{proto}: \cup\text{Bunch} \cup \text{Array} \cup \text{RegExp} \\ \qquad \cup\text{String} \cup \text{Number} \cup \text{Boolean}, \\ \star : \alpha, \\ \textit{code} : [\text{Global} \cup \alpha]\alpha \cdots \to \alpha, \\ \texttt{"\_\_\_nodes\_\_\_"} : \text{Array}\langle\text{HTML}\rangle \cup \text{Undef}, \\ \texttt{"\_\_\_star\_\_\_"} : \text{Bool} \cup \text{Undef}, \\ \texttt{"caller"} : \text{☠}, \texttt{"callee"} : \text{☠}, \\ \texttt{"eval"} : \text{☠}, \texttt{"prototype"} : \text{☠}, \\ \texttt{"watch"} : \text{☠}, \texttt{"constructor"} : \text{☠}, \\ \texttt{"\_\_proto\_\_"} : \text{☠}, \texttt{"unwatch"} : \text{☠}, \\ \texttt{"arguments"} : \text{☠}, \texttt{"valueOf"} : \text{Absent}, \\ \texttt{"toString"} : \text{Absent} \end{array}\right\}$$