

ADsafety

Type-based Verification of JavaScript Sandboxing

Joe Gibbs Politz

Spiridon Aristides Eliopoulos

Arjun Guha

Shriram Krishnamurthi



BROWN

August 6, 2011

THE HUFFINGTON POST

THE INTERNET NEWSPAPER: NEWS BLOGS VIDEO COMMUNITY

[Like](#) [Follow](#)

Search the Huffington Post



CONNECT [f](#) [t](#)

[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/>"

EDITION: U.S. ▾

For Profit

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

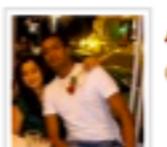
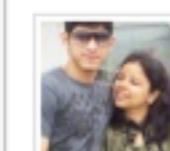
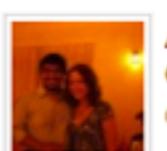
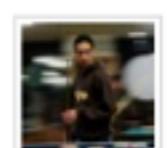
Filter Friends ▾

All

Selected (0)

Aaron T.
Myers

Brown

Adam Khalil
GrinnellAditi Mehta
ColumbiaAditya
Dhandhania
JWU Prov...Aditya Pai
Georgia ...Adityarup
Chakravorty
GrinnellAheli
Purkayastha
Universi...Akili
Thomas
GrinnellAlbert
Varma
UVA

Send HuffingtonPost Invitation

Cancel

FRONT PAGE | POLITICS | BUS

FEATURED ★ BLACKVOICES ★

FROM AP:

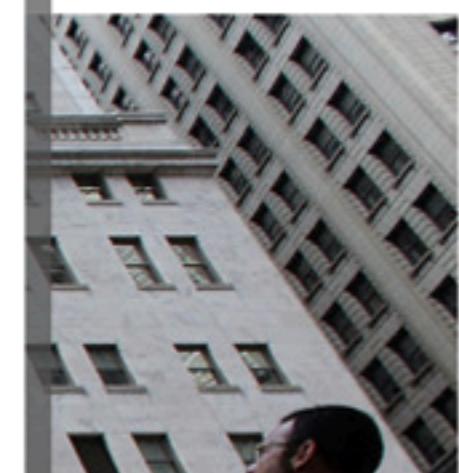
OUT
U.

CONNECT

NG ★ VOICES LOCAL MORE

WEIRD NEWS ARTS

Get Alerts

TIVE
rs

EDITION: U.S.

Megan Fox Music Movies Mad Men Smarter Ideas More

Arjun Guha | Sign Out

LET'S GO.

Find out more about
Shell's plans in Alaska.



News Web

Search News and Topics

SEARCH >

CONNECT



third-party ad

ENTERTAINMENT

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
about the state of mind we are in. Think about what we
can do to change it.



PHOTO GALLERIES

third-party ad

Web Inspector — http://www.huffingtonpost.com

The screenshot shows the Network tab of the Web Inspector for the URL http://www.huffingtonpost.com. The table lists various resources loaded by the page, categorized by their type (e.g., application/javascript, text/html) and status (e.g., 200 OK, 304 Not Modified). A red box highlights several JavaScript files, indicating they are being executed in the browser:

Name	Path	Method	Status	Type	Size Transfer
o.aolcdn.com	o.aolcdn.com	GET	304 Not Modif	application/javascript	54.18KB 223B
p-6fTutip1SMLM2.js	pixel.quantserve.com/seg	GET	200 OK	application/x-javascript	419B 870B
tw_dfp_adsonar.js	js.adsonar.com/js	GET	304 Not Modif	application/javascript	6.12KB 247B
beacon.js	b.scorecardresearch.com	GET	200 OK	application/x-javascript	3.37KB 1.90KB
quant.js	edge.quantserve.com	GET	304 Not Modif	application/x-javascript	5.14KB 454B
ad_html_wh.php	/ads	GET	200 OK	text/html	0B 217B
js.php	s.huffpost.com/assets	GET	200 OK	application/x-javascript	2.10KB 949B
js.php	s.huffpost.com/assets	GET	200 OK	application/x-javascript	3.56KB 1.56KB
aceUAC.js	uac.advertising.com/wrapper	GET	200 OK	application/x-javascript	14.81KB 5.54KB
dref=http%253A%252F%252Fwww.rl-ads.ace.advertising.com/site=80		GET	200 OK	application/x-javascript	536B 1.52KB
imb8Z50C5TH.js	s-static.ak.facebook.com/rsrsrc.php/v	GET	304 Not Modif	application/x-javascript	39.59KB 225B
documentwrite.js	img-cdn.mediaplex.com/0	GET	304 Not Modif	application/x-javascript	58B 202B

Who is
running
code in
your
browser?

Who is
running
code in
your
browser?

The screenshot shows a browser window with the Quantcast homepage loaded. A red arrow points from the left side of the screen towards the network requests listed in the developer tools' Network tab. The Network tab lists various JavaScript files (JS) and their paths, some of which are circled in red. One specific request, 'p-6fTutip1SMLM2.js' from pixel.quantserve.com, is highlighted with a large red rectangle. An overlay window titled 'Quantcast Measurement' is displayed over the main content area, containing promotional text about audience measurement.

Name	Path	Method	Status	Type	Size
o.aolcdn.com	o.aolcdn.com	GET	OK	text/html	217B
p-6fTutip1SMLM2.js	pixel.quantserve.com/seg...	GET	200 OK	application/x-javascript	2.10KB 949B
tw_dfp_adsonar.js	js.adsonar.com/js	GET	200 OK	application/x-javascript	3.56KB 1.56KB
beacon.js	b.scorecardresearch.com	GET	200 OK	application/x-javascript	14.81KB 5.54KB
quant.js	edge.quantserve.com	GET	200 OK	application/x-javascript	536B 1.52KB
ad_html_wh.php	/ads	GET	OK	text/html	
js.php	s.huffpost.com/assets	GET	200 OK	application/x-javascript	
js.php	s.huffpost.com/assets	GET	200 OK	application/x-javascript	
aceUAC.js	uac.advertising.com/wrapper	GET	200 OK	application/x-javascript	
dref=http%253A%252F%252Fwww...	r1-ads.ace.advertising.com/site=80...	GET	200 OK	application/x-javascript	
imb8Z50C5TH.js	s-static.ak.facebook.com/rsr...	GET	304 Not Modif...	application/x-javascript	39.59KB 225B
documentwrite.js	img-cdn.mediaplex.com/0	GET	304 Not Modif...	application/x-javascript	58B 202B

Who is running code in your browser?

The screenshot shows a web browser window with several tabs open. The main tab displays the Quantcast homepage, which offers free audience measurement services. A red arrow points from the Quantcast page to the browser's developer tools, specifically the Network tab, where a list of loaded JavaScript files is shown. Another red arrow points from the developer tools back to the Advertising.com page, highlighting the presence of multiple tracking scripts.

Home | Quantcast

http://www.quantcast.com/

NEWS New free publisher offering: Quantcast hosts Ad Choices Icon for Long-Tail Publishers. Learn more.

Sign In | Create Account | Opt-Out | Privacy Policy

quantcast

It's your audience. We just find it.™

HOME MEASUREMENT REACH AUDIENCES LEARNING CENTER HOW WE DO IT ABOUT BLOG

Measure your audience **for free**.

Reach them like never before.

Check out a site profile to see the power of our reports

examples [hulu](#), [gawker.com](#), [evite.com](#), [implement tags](#)

Top Million Sites | Compare Sites

Quantcast Measurement

Free direct audience measurement for all website

Quantcast Audience

Audience targeting for buyers and sellers of digital advertising

Advertising.com | Advertiser/Sponsored listings

http://www.advertising.com/advertiser/sponsored-listings

Search

Login

Search Advertising.com

Advertising.com

Advertiser Publisher

Sponsored listings

Spontaneous text advertising

Start with as little as \$100

aceUAC.js

uac.advertising.com

dref=http%253A%252Brl-ads.ace.advertising.com

imb8Z50C5TH.js

s-static.ak.facebook.com

documentwrite.js

img-cdn.mediaplex.com

Elements Resources Network

Name Path

o.aolcdn.com o.aolcdn.com

p-6fTutip1SMLM2.js pixel.quantserve.com/seg

tw_dfp_adsonar.js js.adsonar.com/js

beacon.js b.scorecardresearch.com

quant.js edge.quantserve.com

ad_html_wh.php /ads

js.php s.huffpost.com/assets

js.php s.huffpost.com/assets

aceUAC.js uac.advertising.com

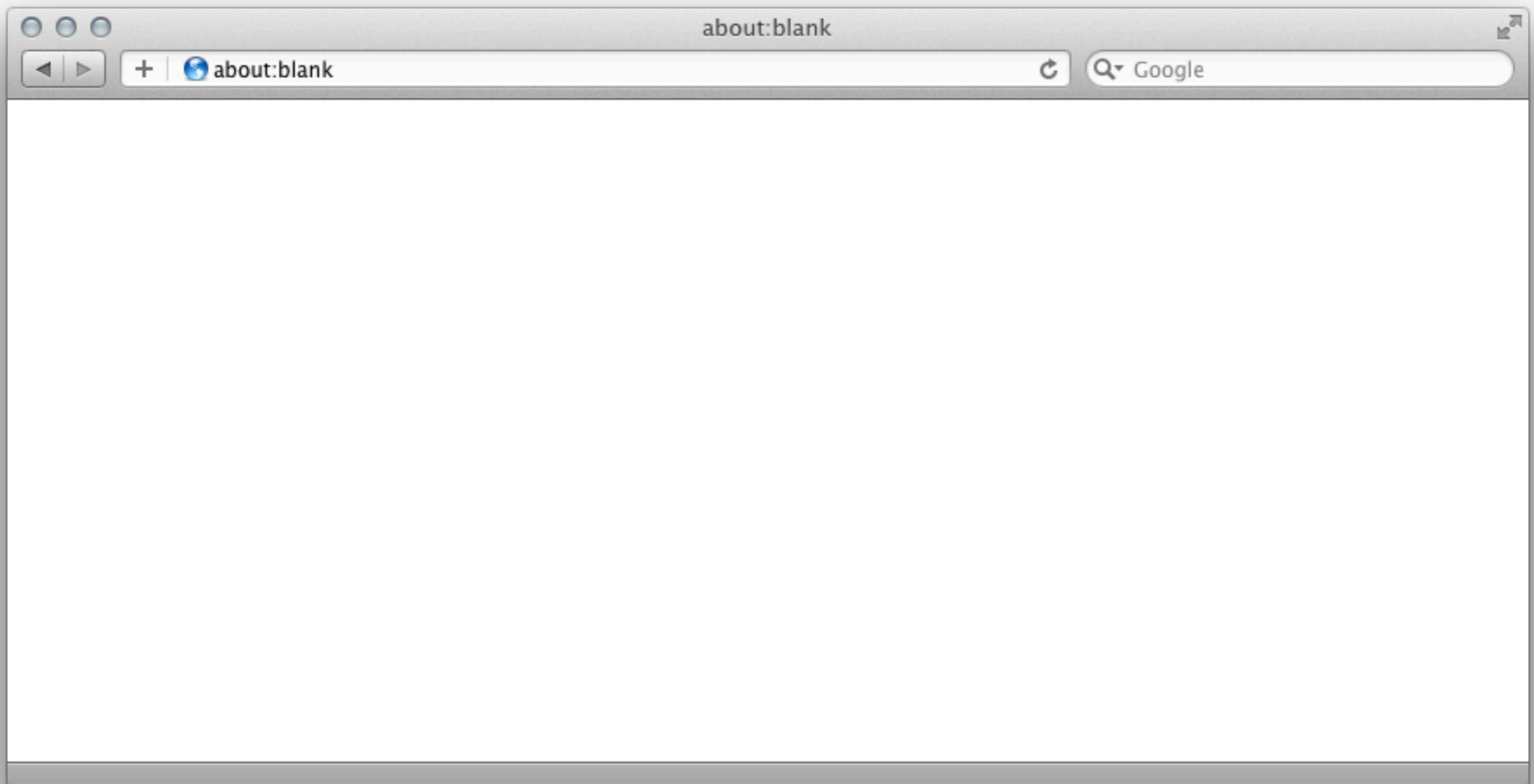
dref=http%253A%252Brl-ads.ace.advertising.com

imb8Z50C5TH.js s-static.ak.facebook.com

documentwrite.js img-cdn.mediaplex.com

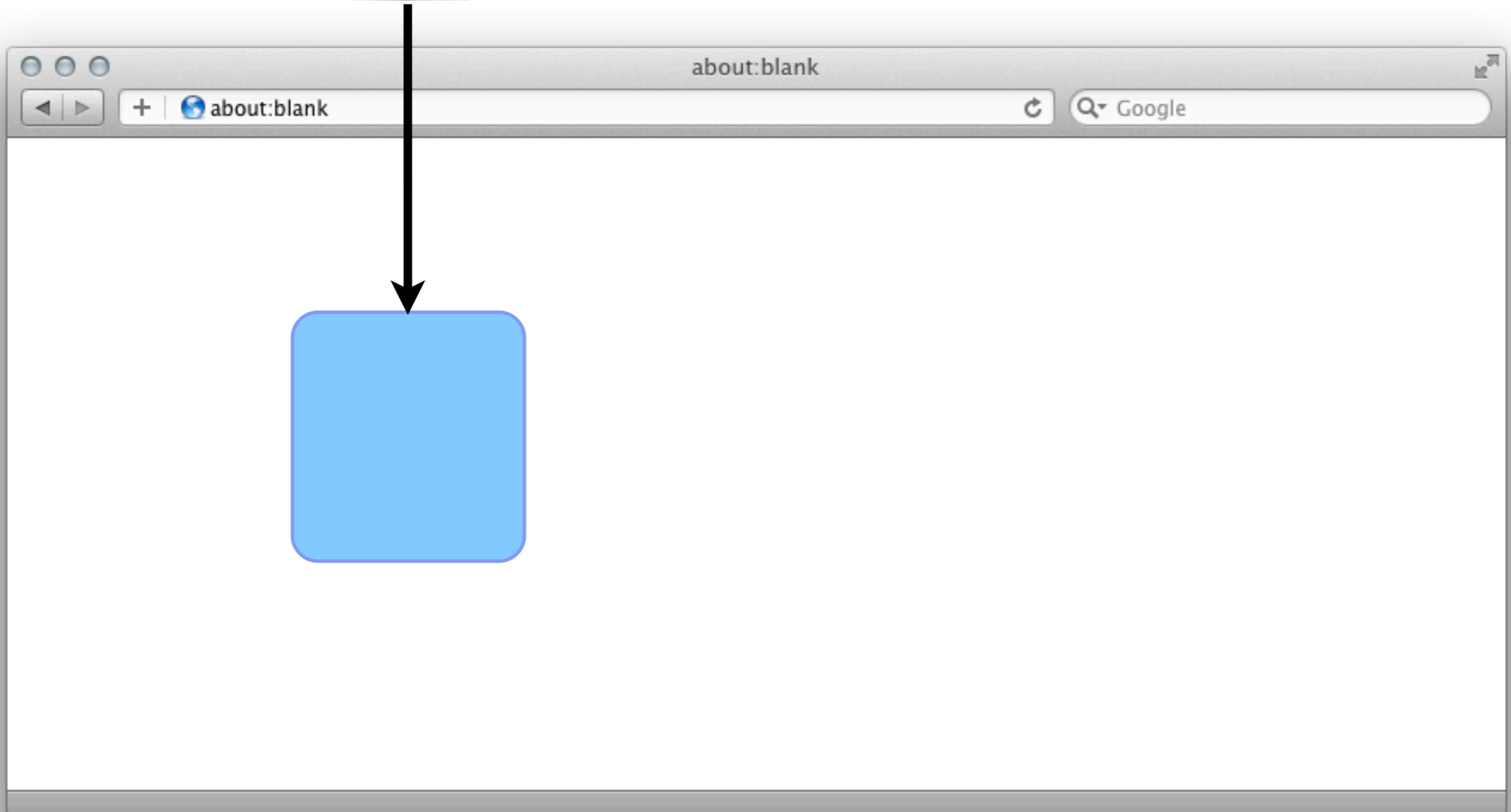


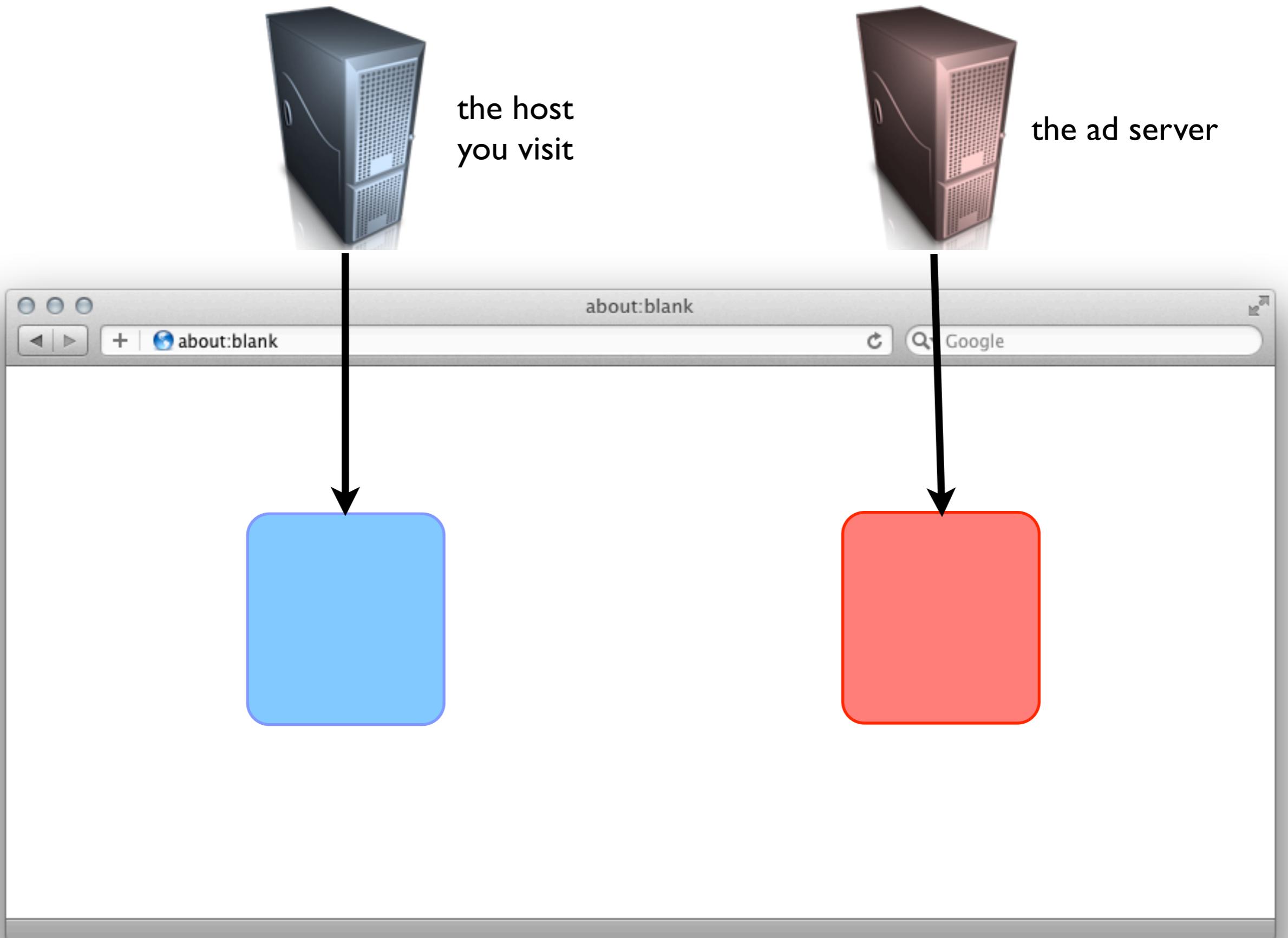
the host
you visit

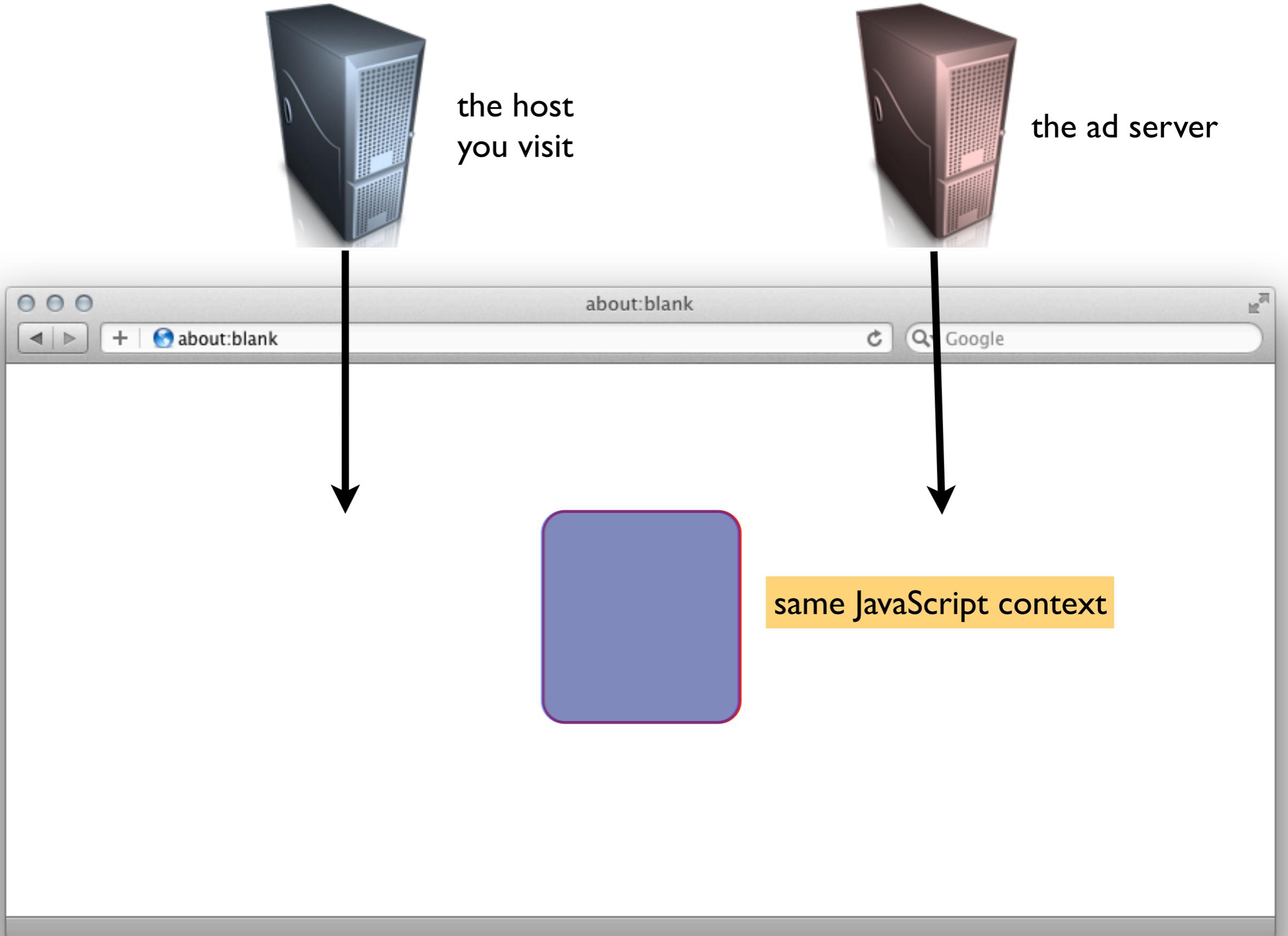


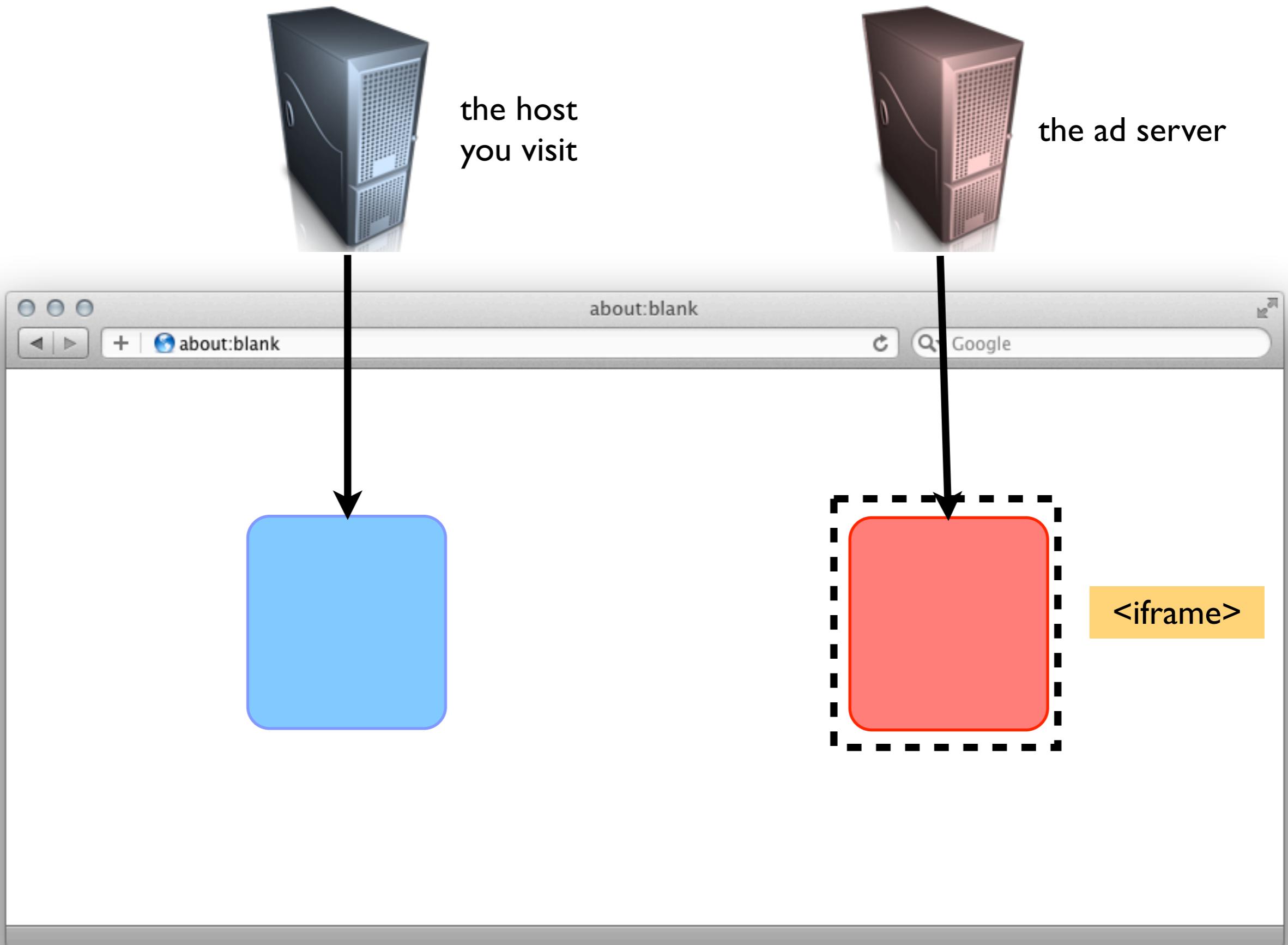


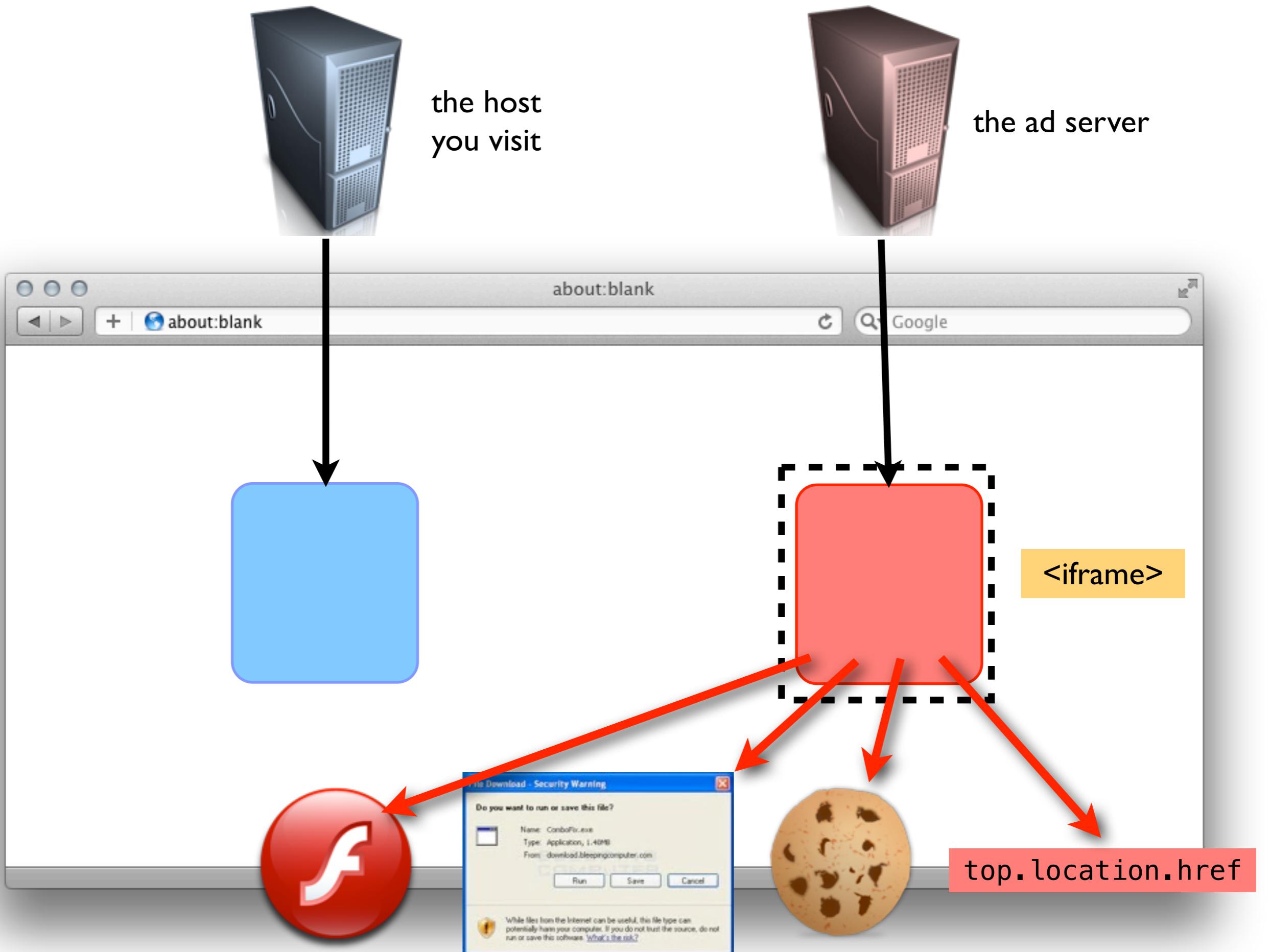
the host
you visit













Facebook JavaScript
(FBJS)



Microsoft Web Sandbox

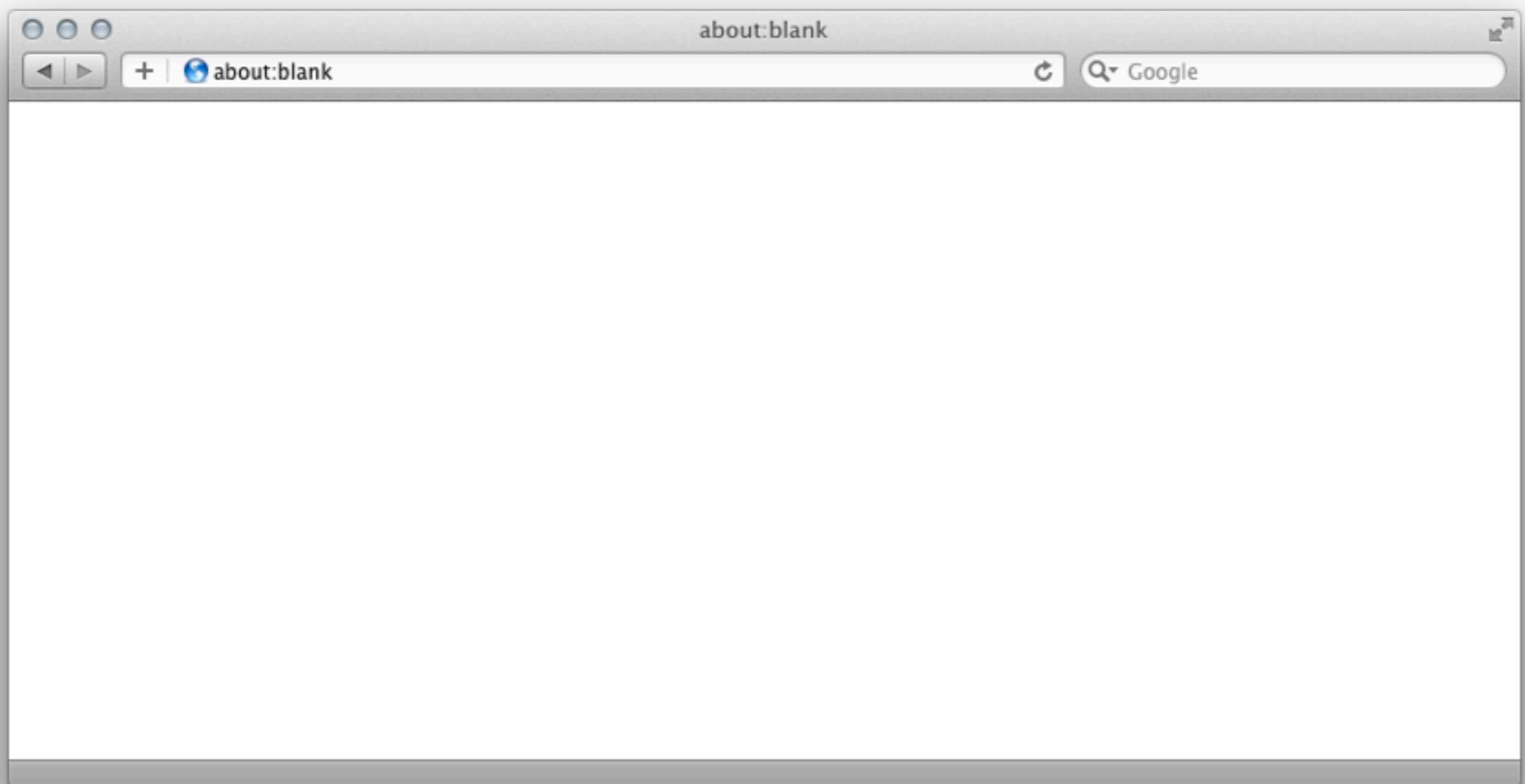


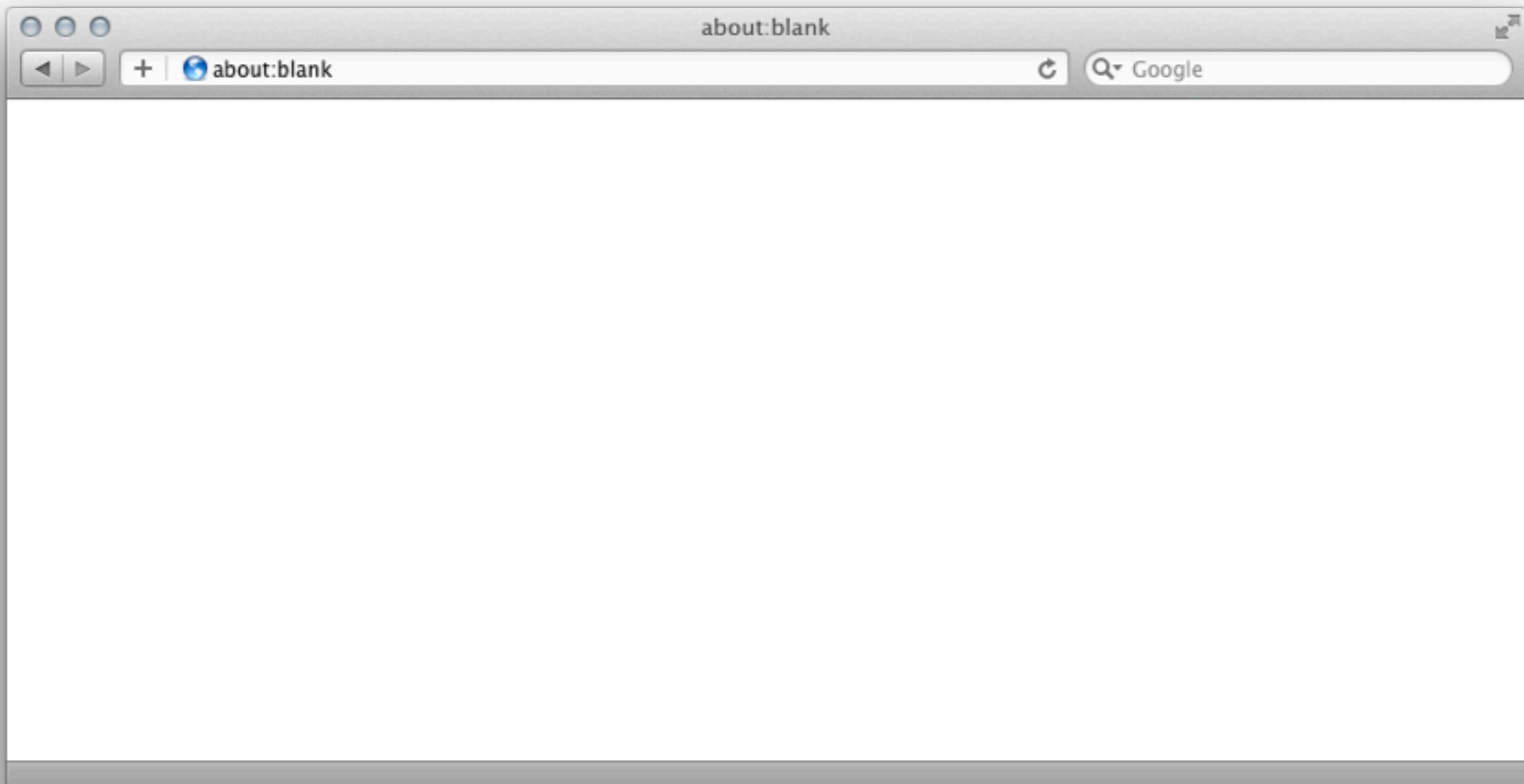
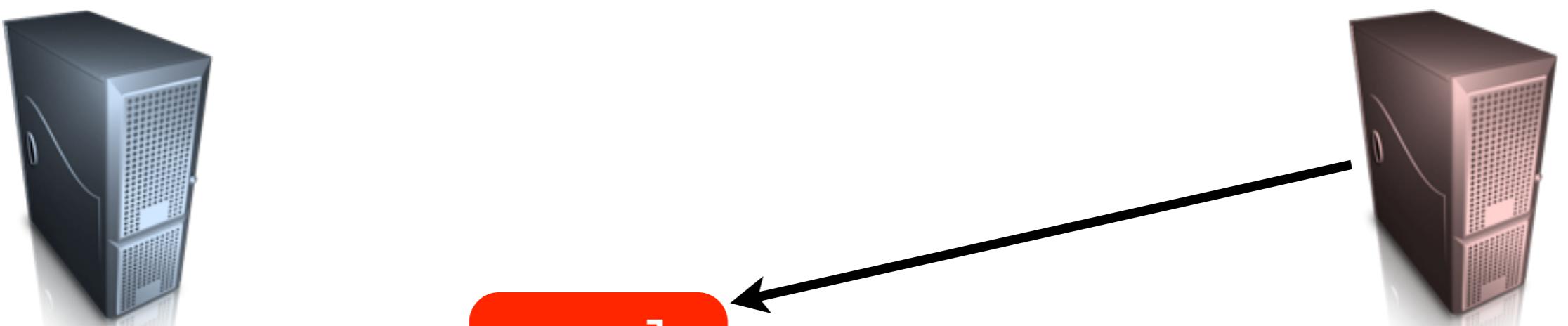
Google
Caja

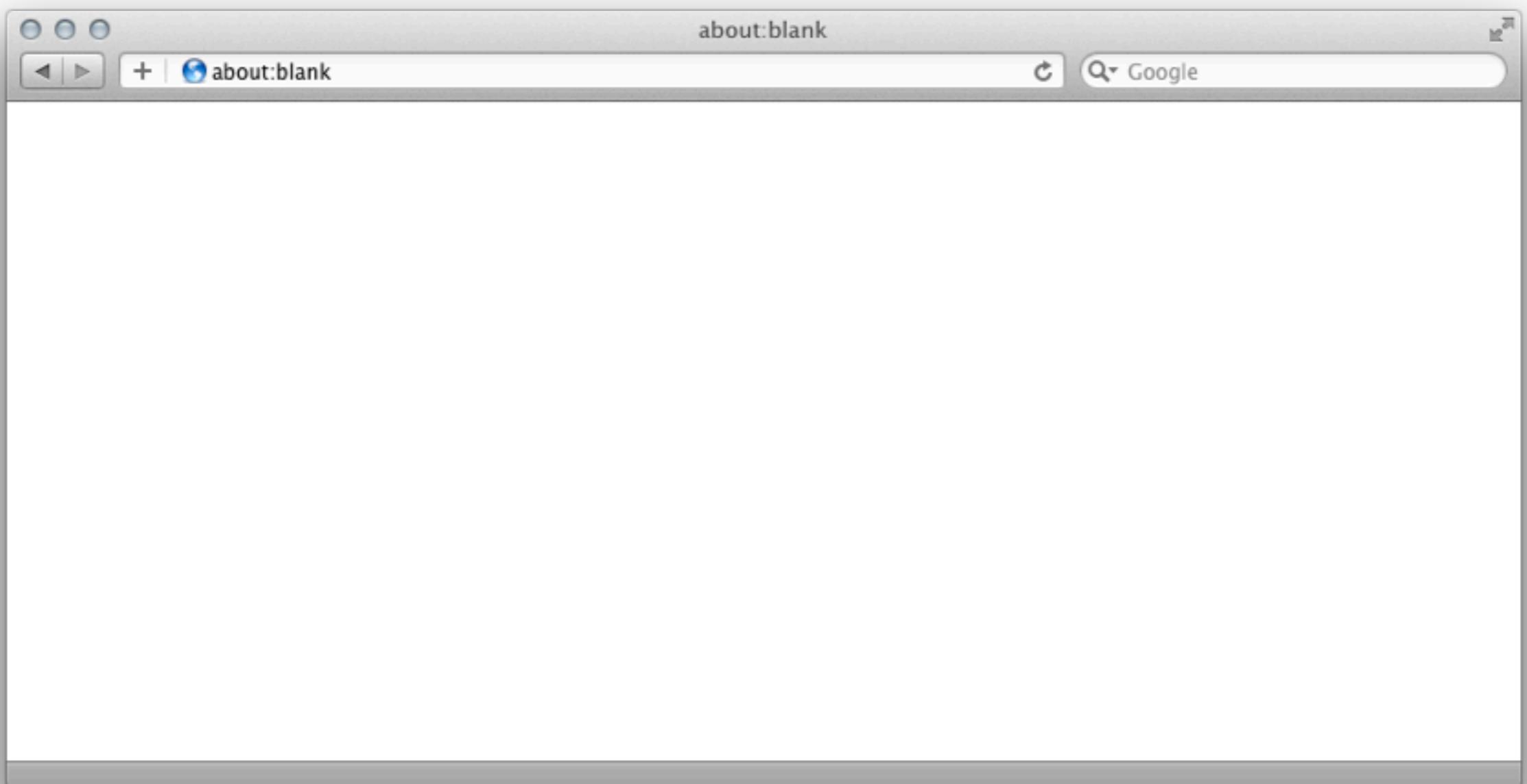
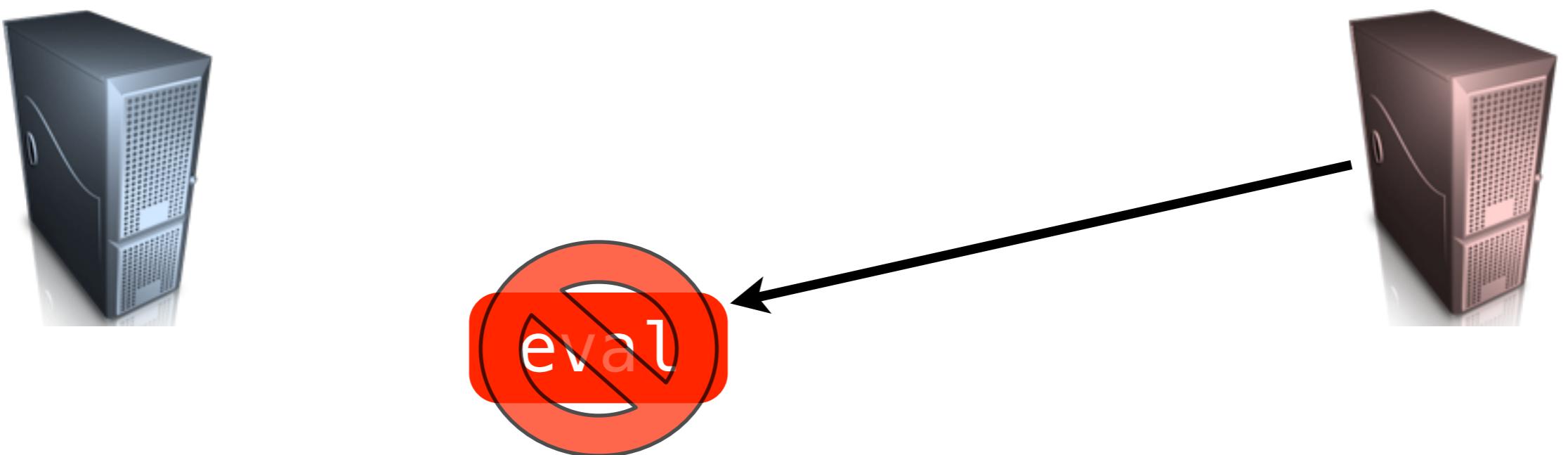


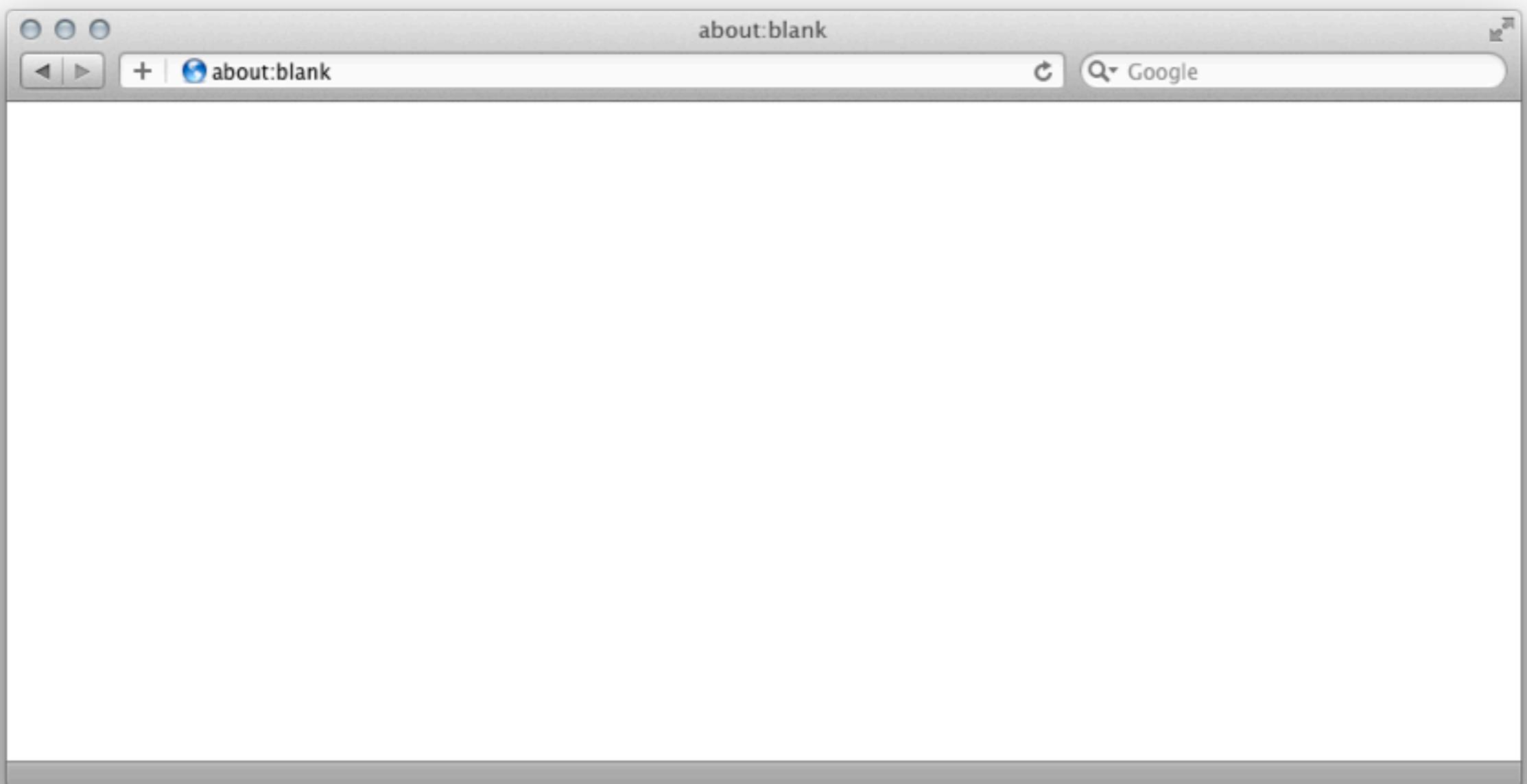
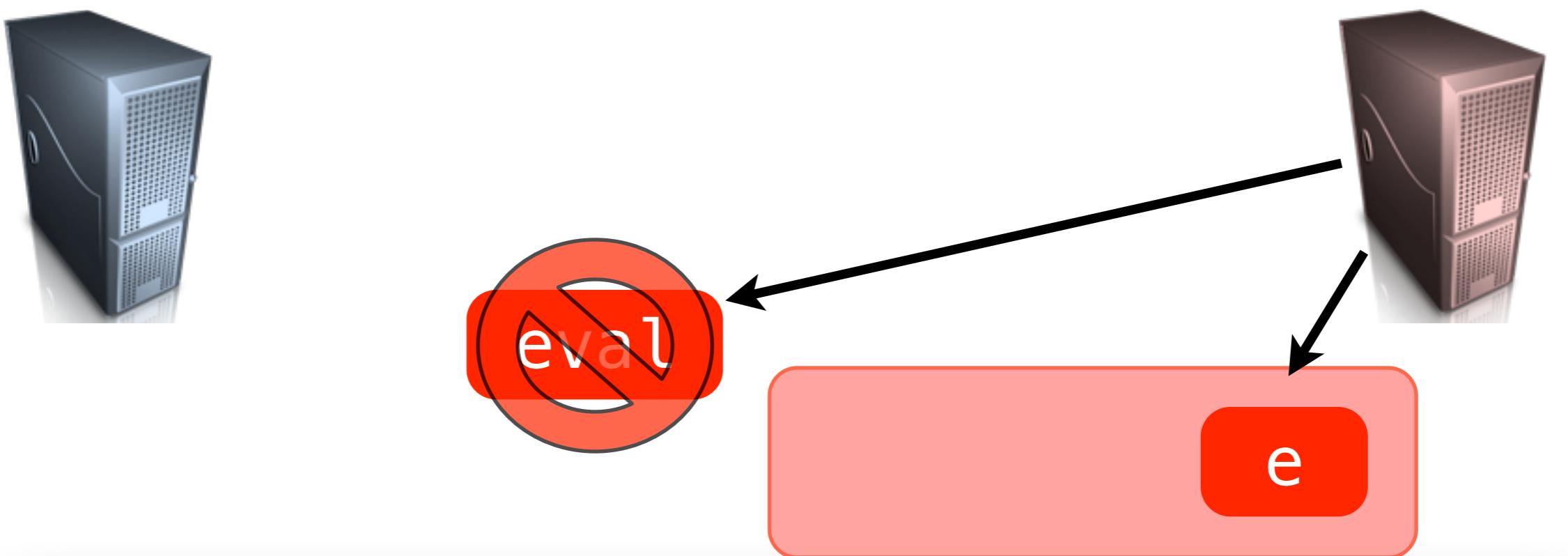
Yahoo!
ADsafe

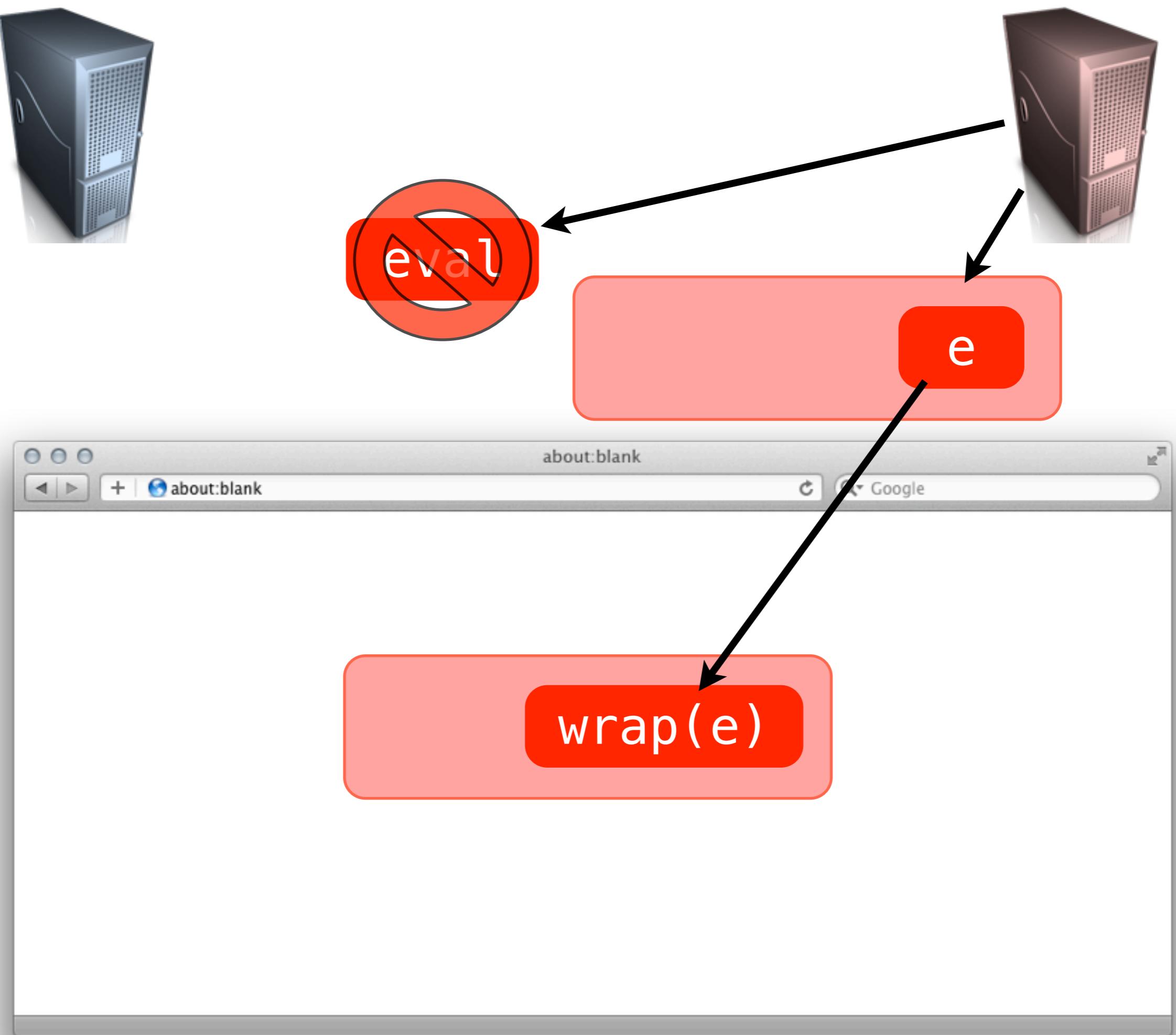
All are **defining safe sub-languages**

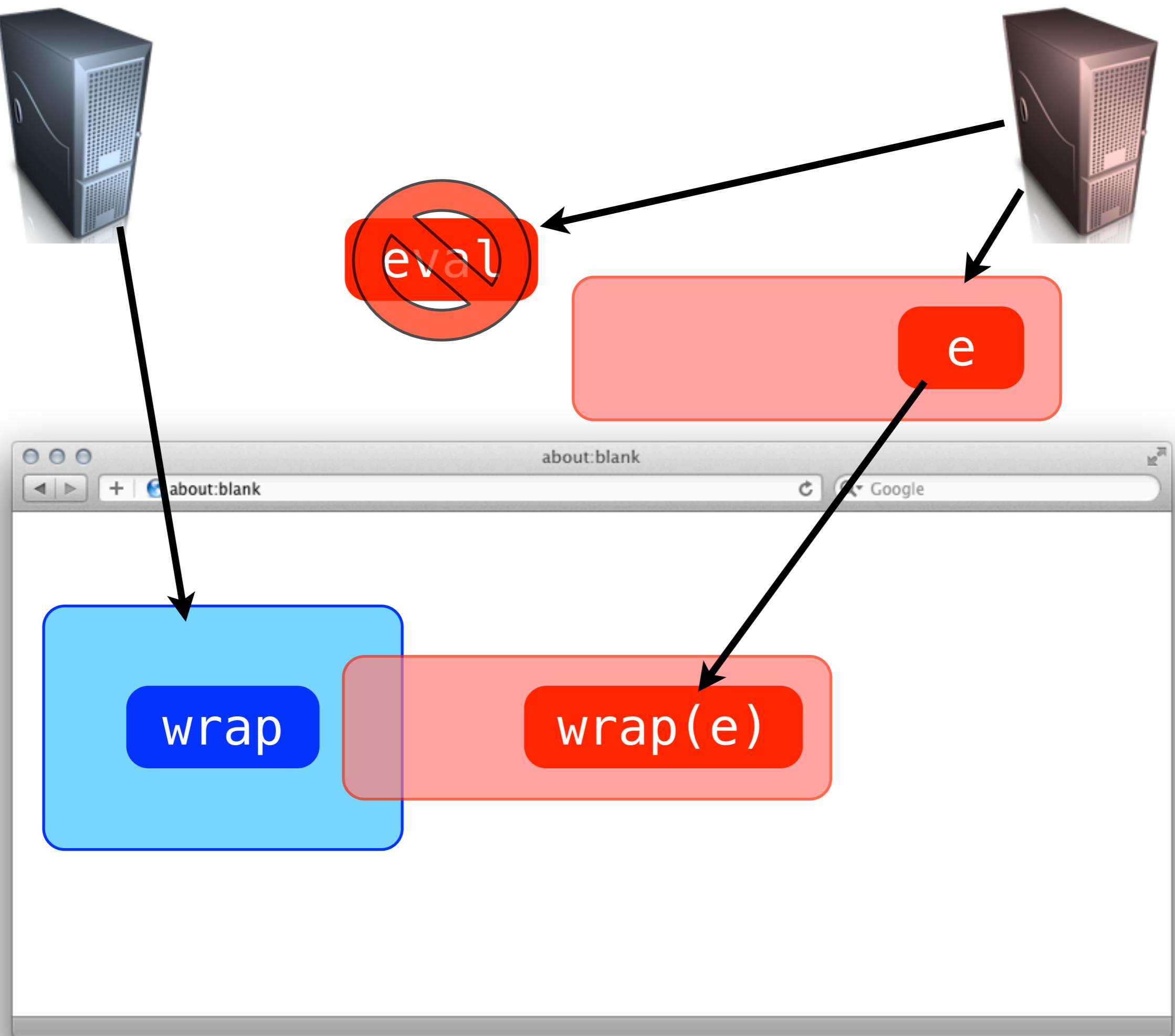


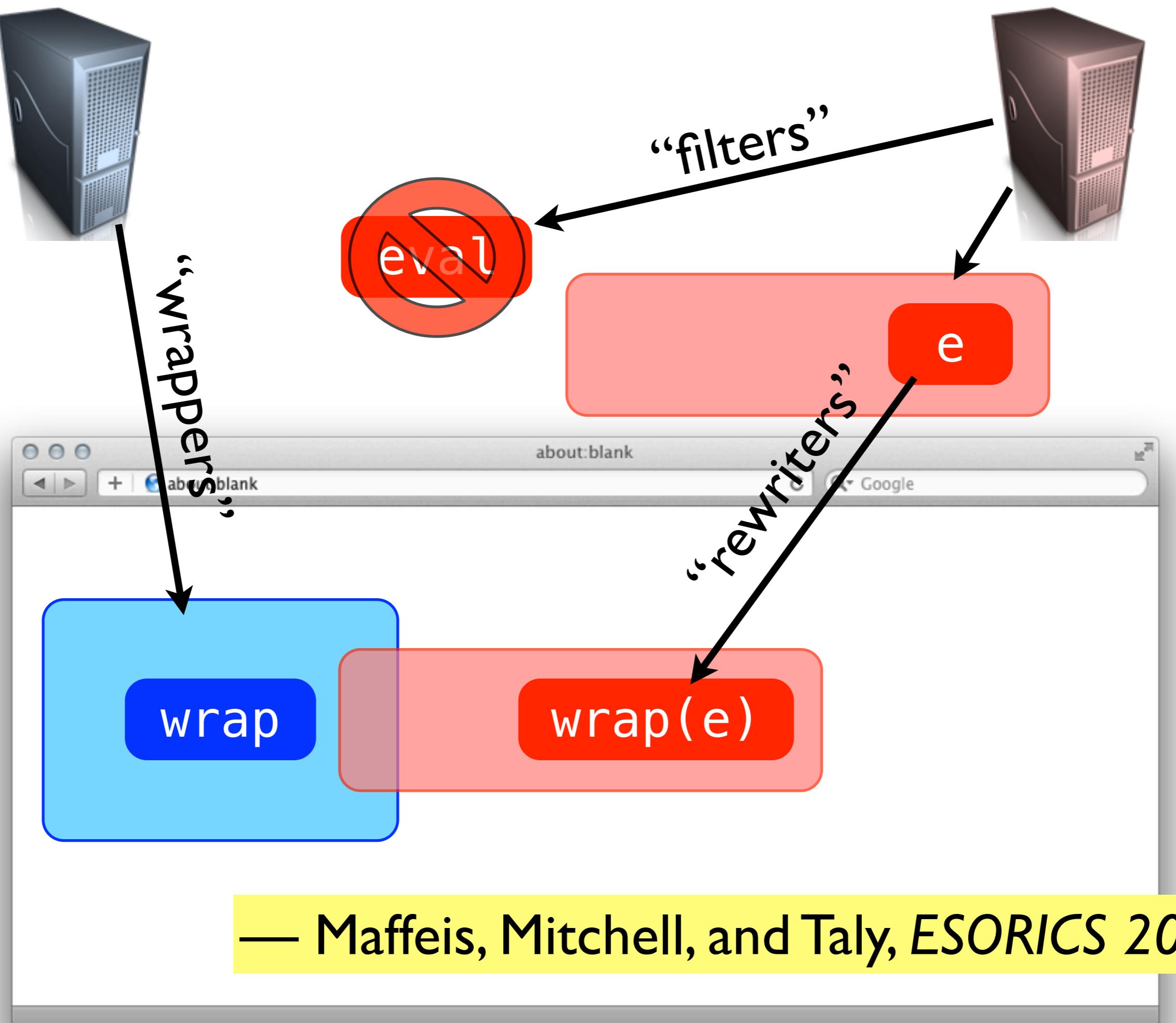


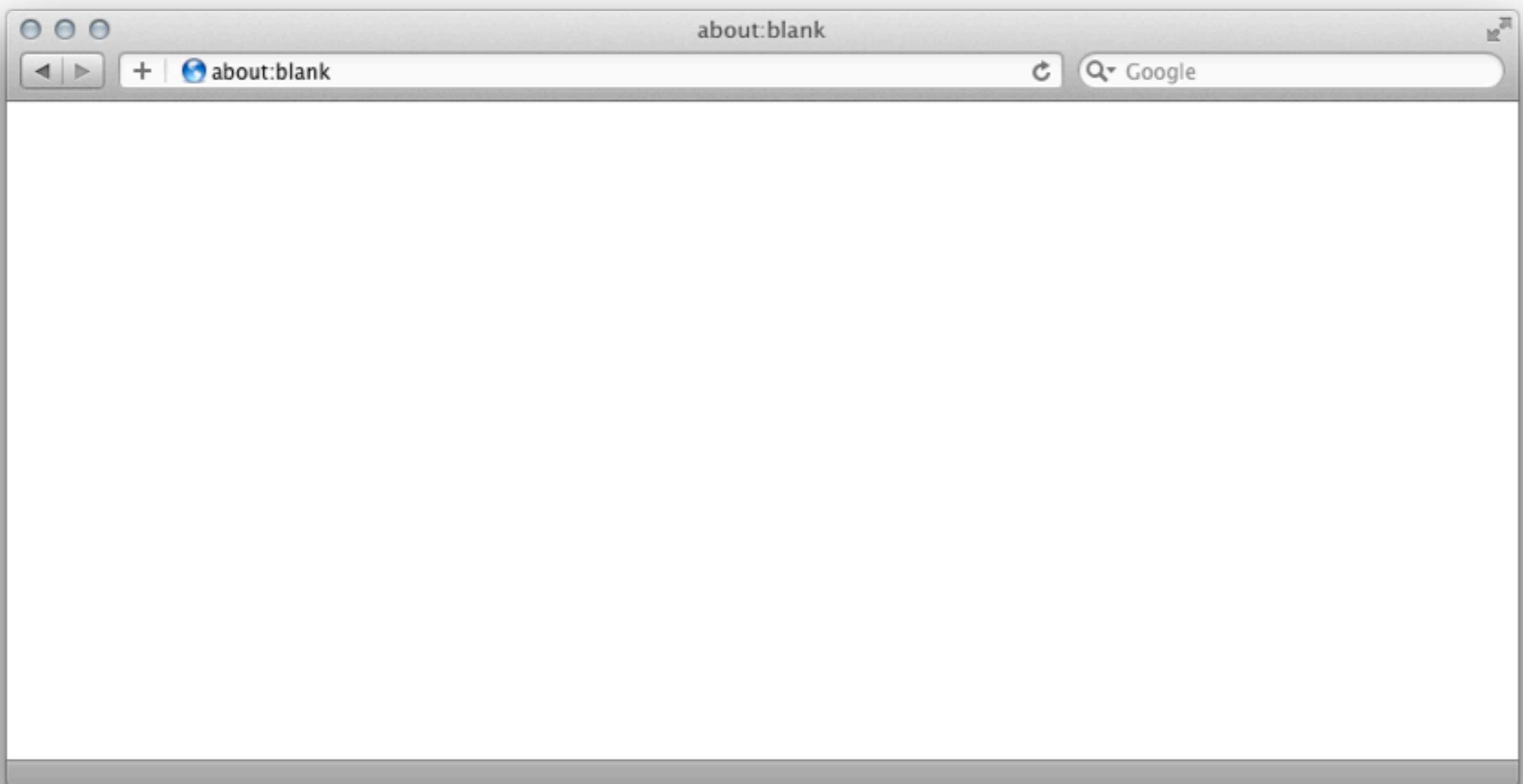


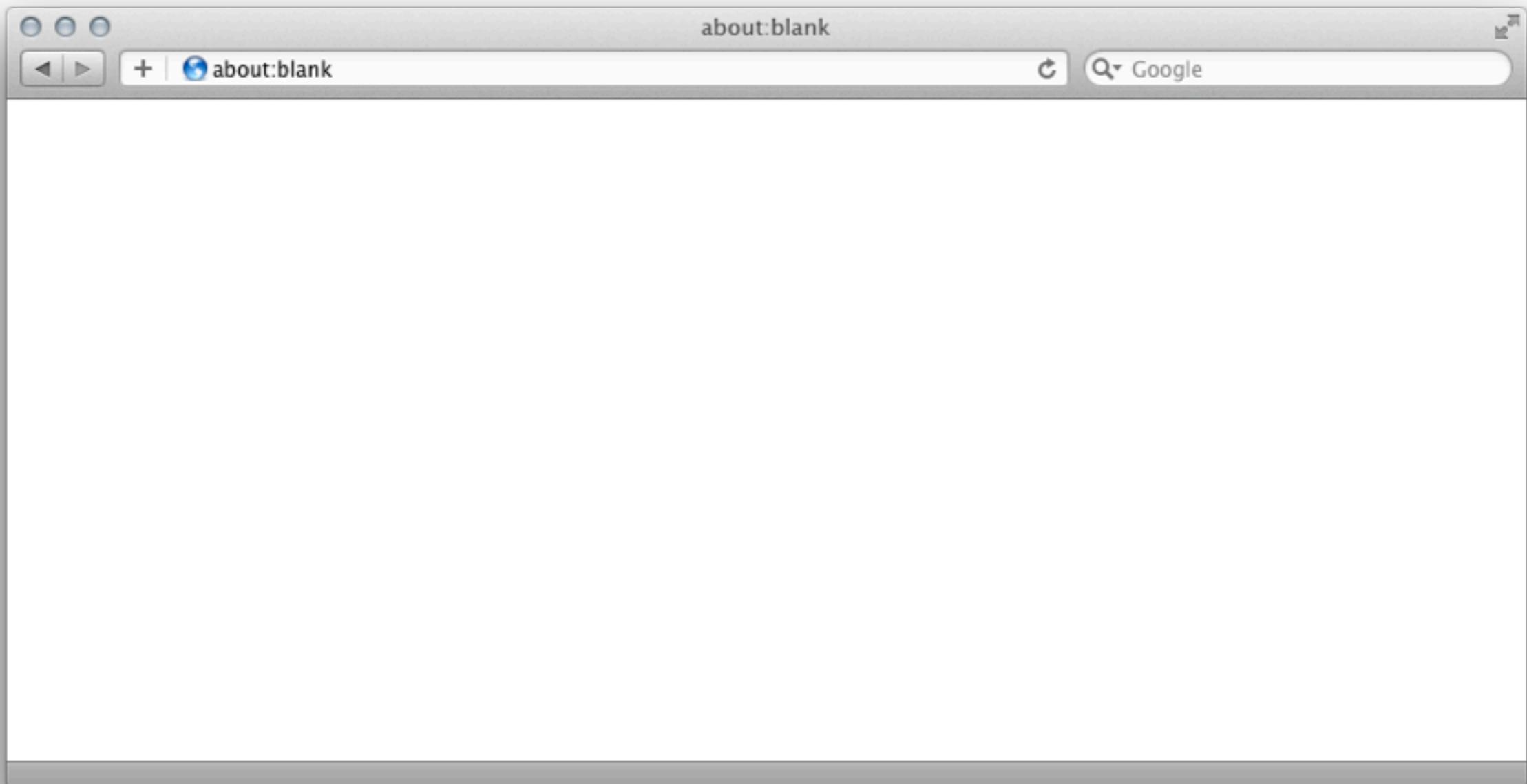
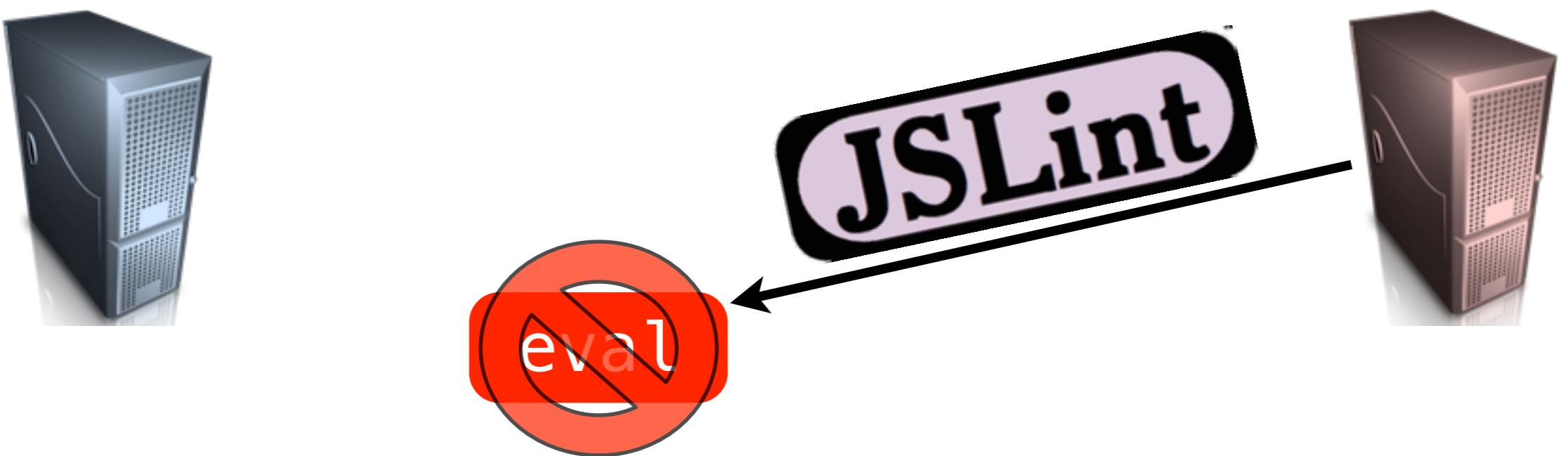


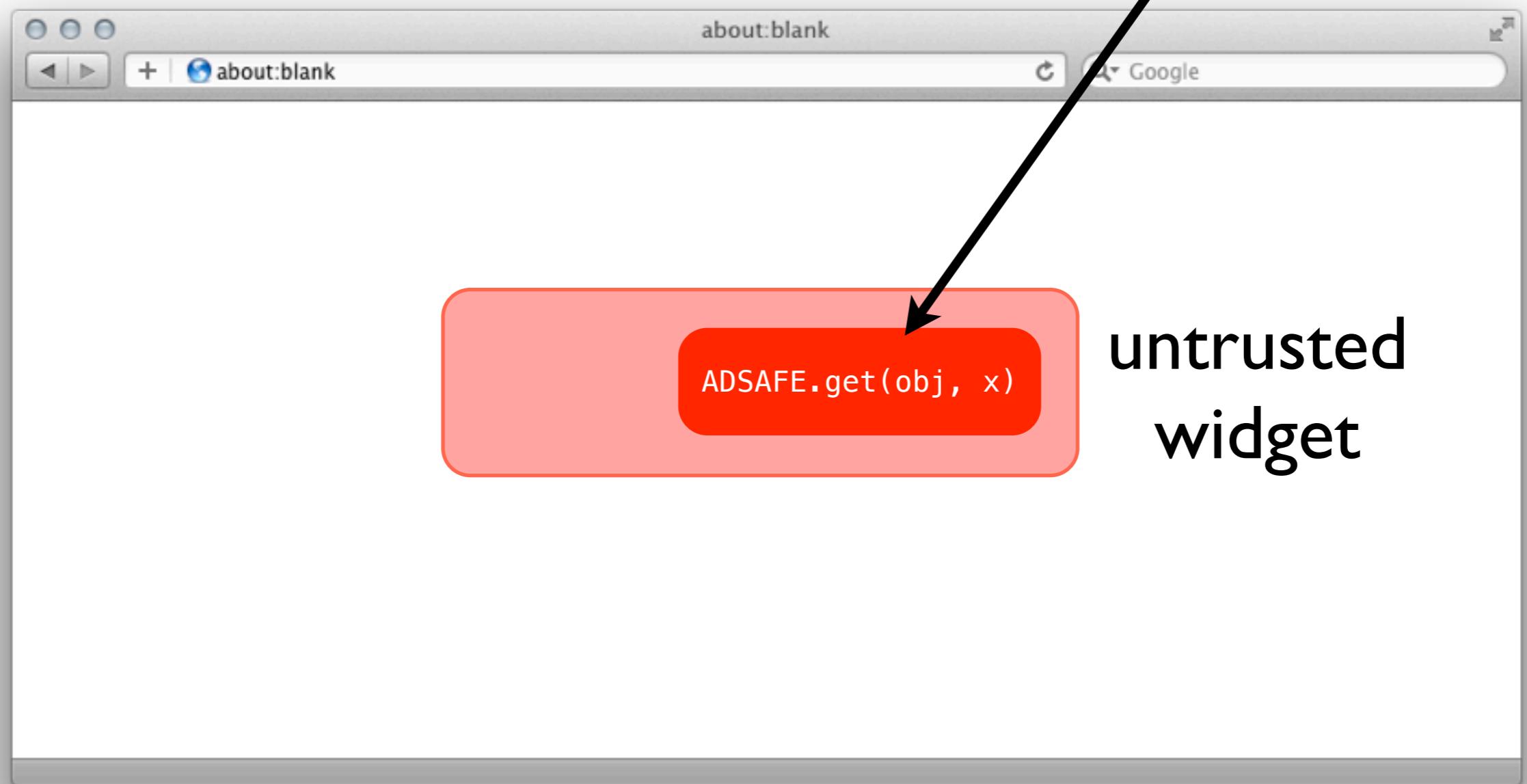
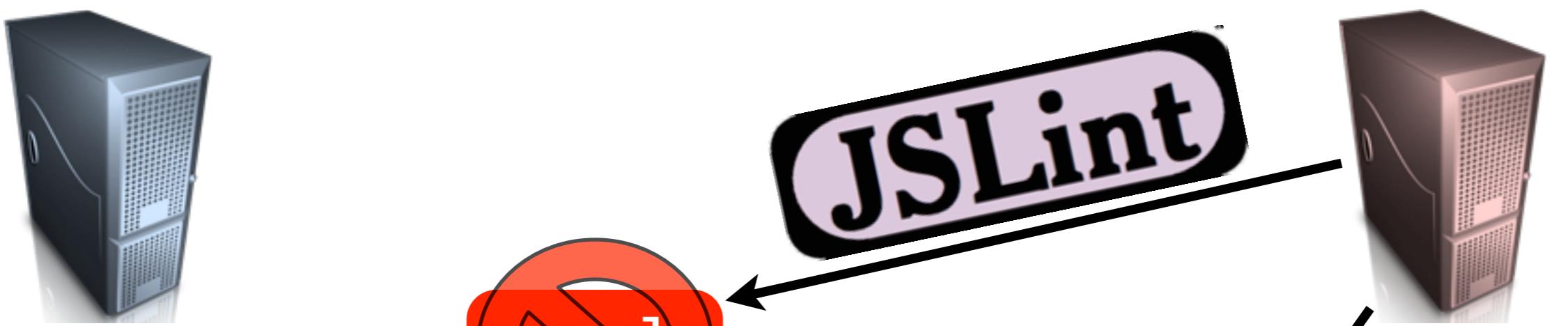


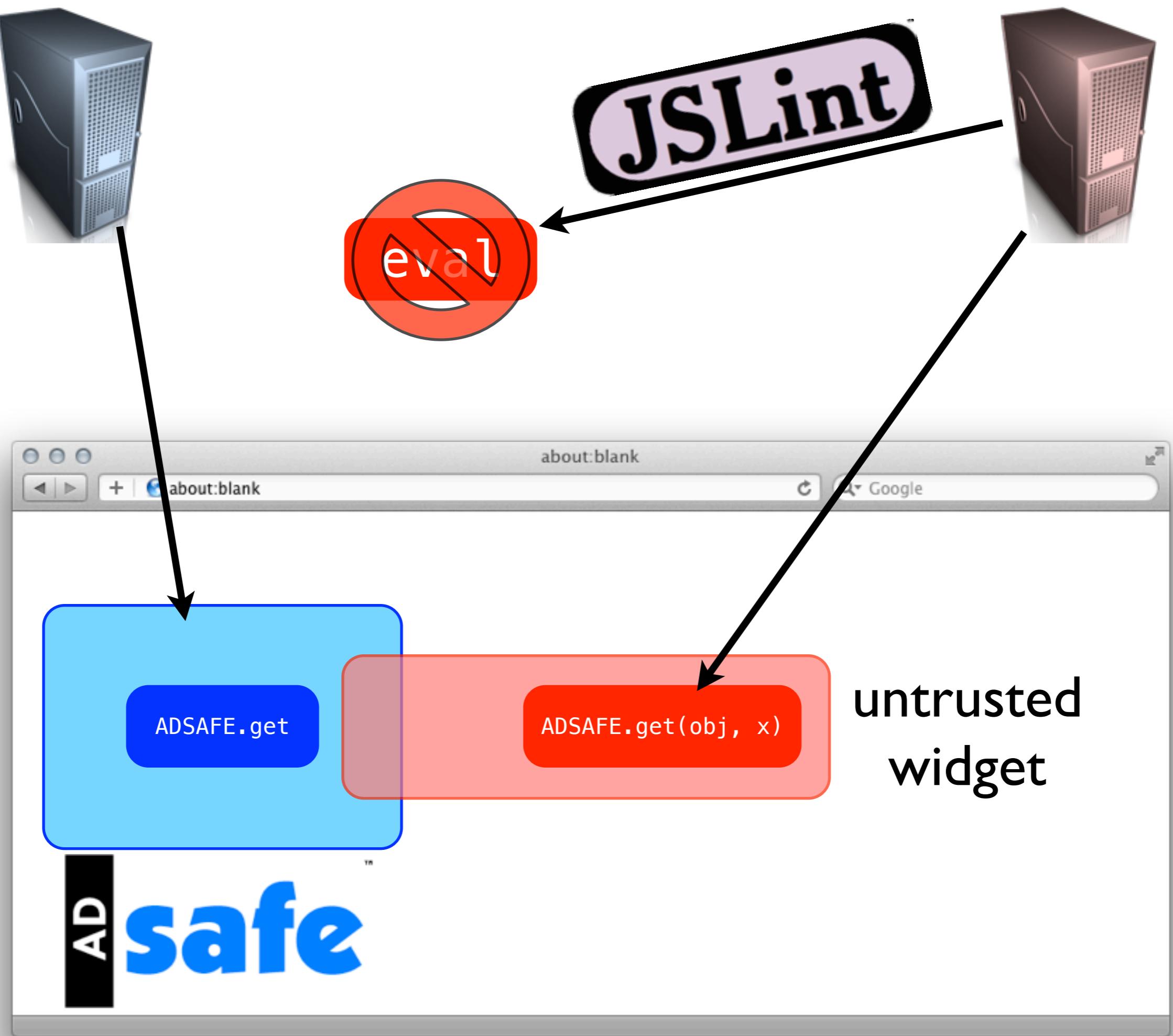














- 1,800 LOC adsafe.js library
- 50 calls to three kinds of assertions
- 40 type-tests
- 5 regular-expression based checks
- 60 privileged DOM method calls



- 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 ADsafe

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

Definition I (ADsafe): If all embedded widgets pass JSLint, then:

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

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

Definition I (ADsafe): If all embedded widgets pass JSLint, then:

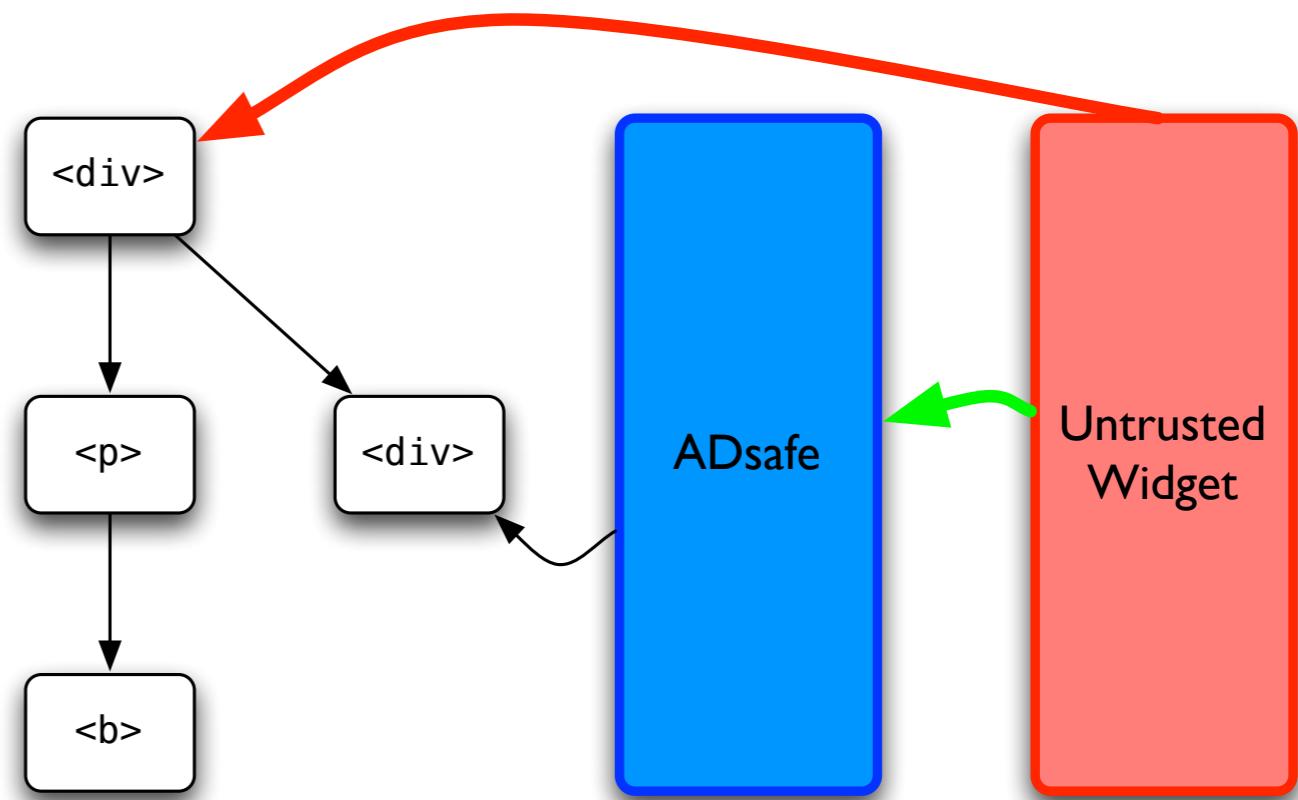


```
document.write()  
document.createElement("script")
```

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

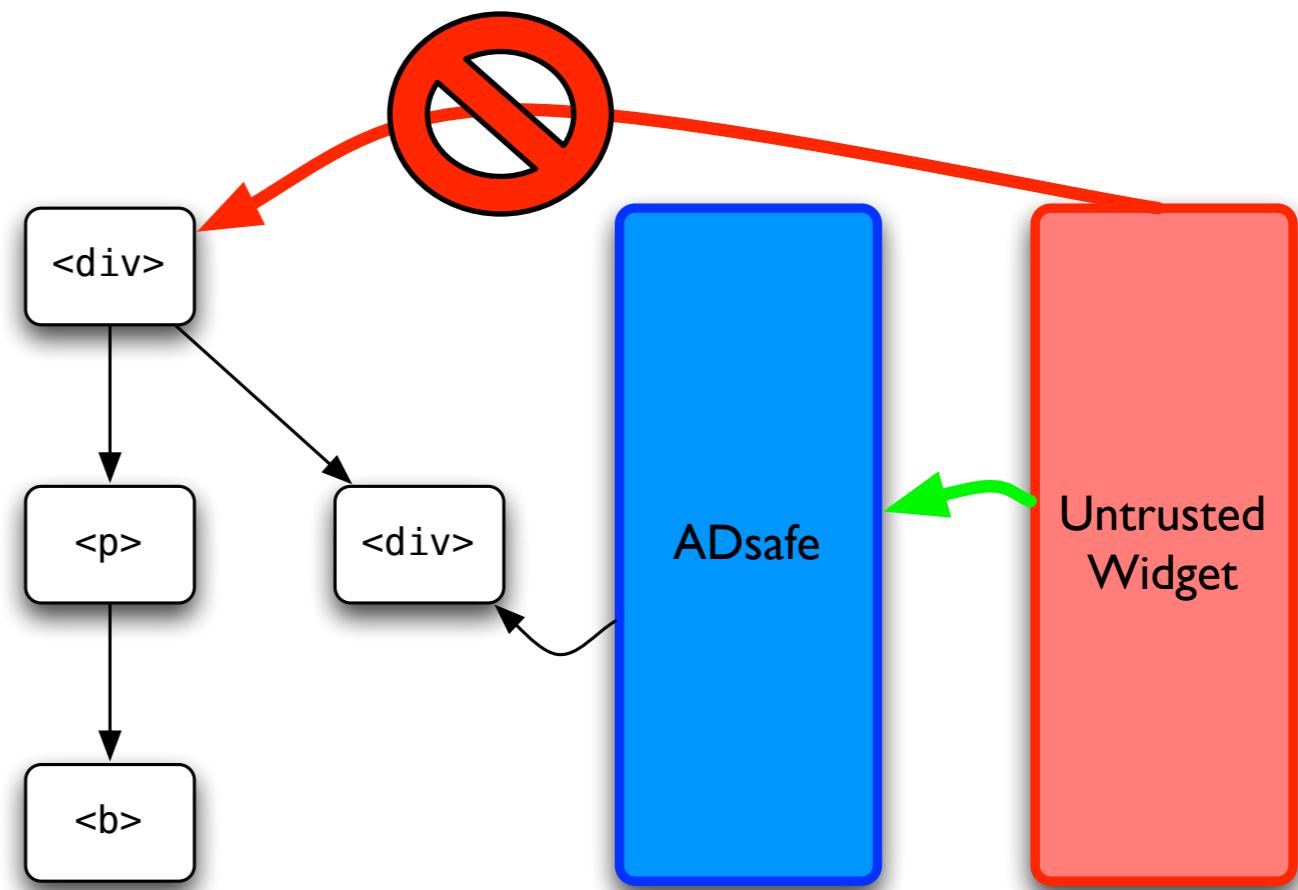
Definition I (ADsafe): If all embedded widgets pass JSLint, then:

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 I (ADsafe): 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;

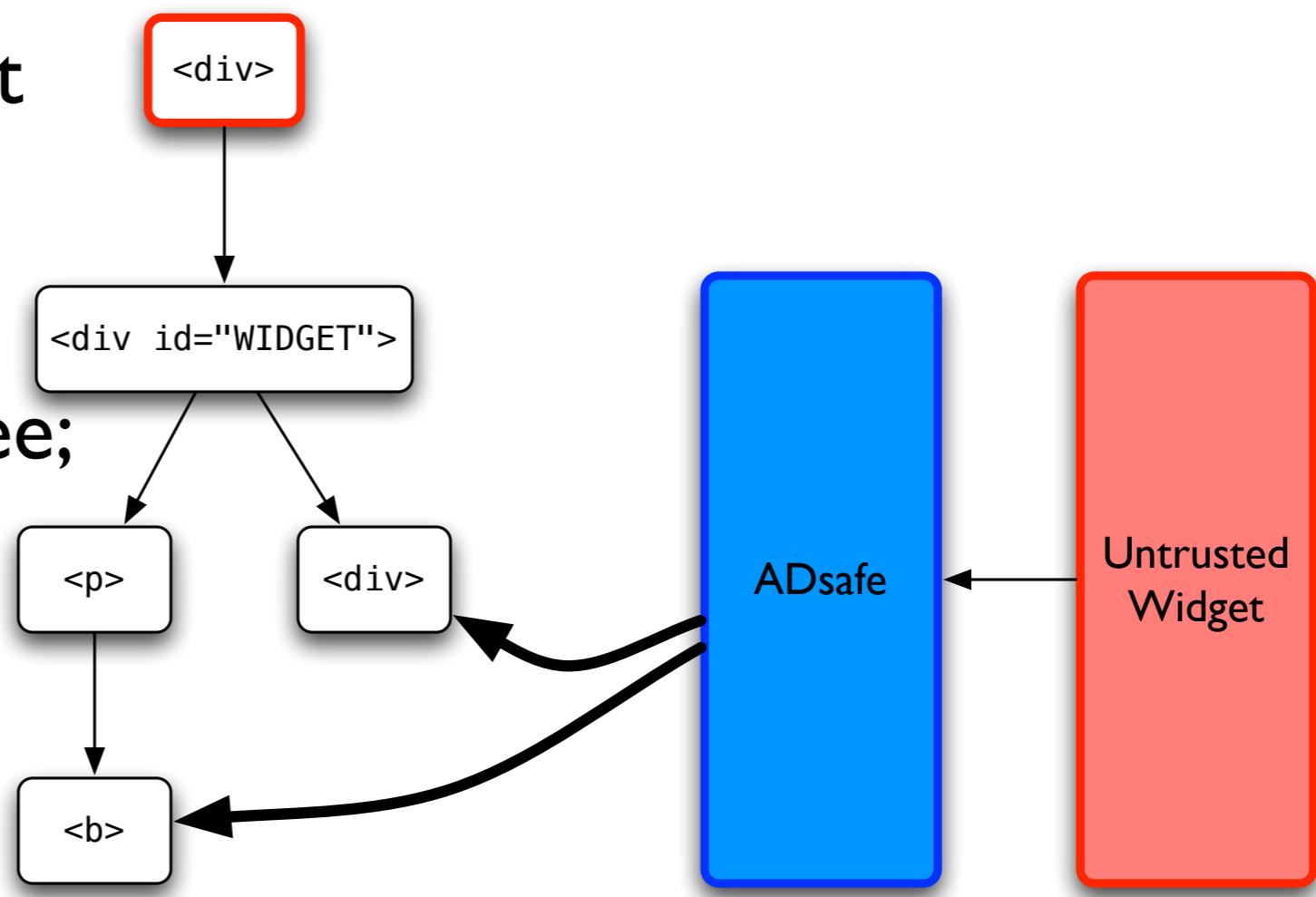


Definition I (ADsafe): If all embedded widgets pass JSLint, then:

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

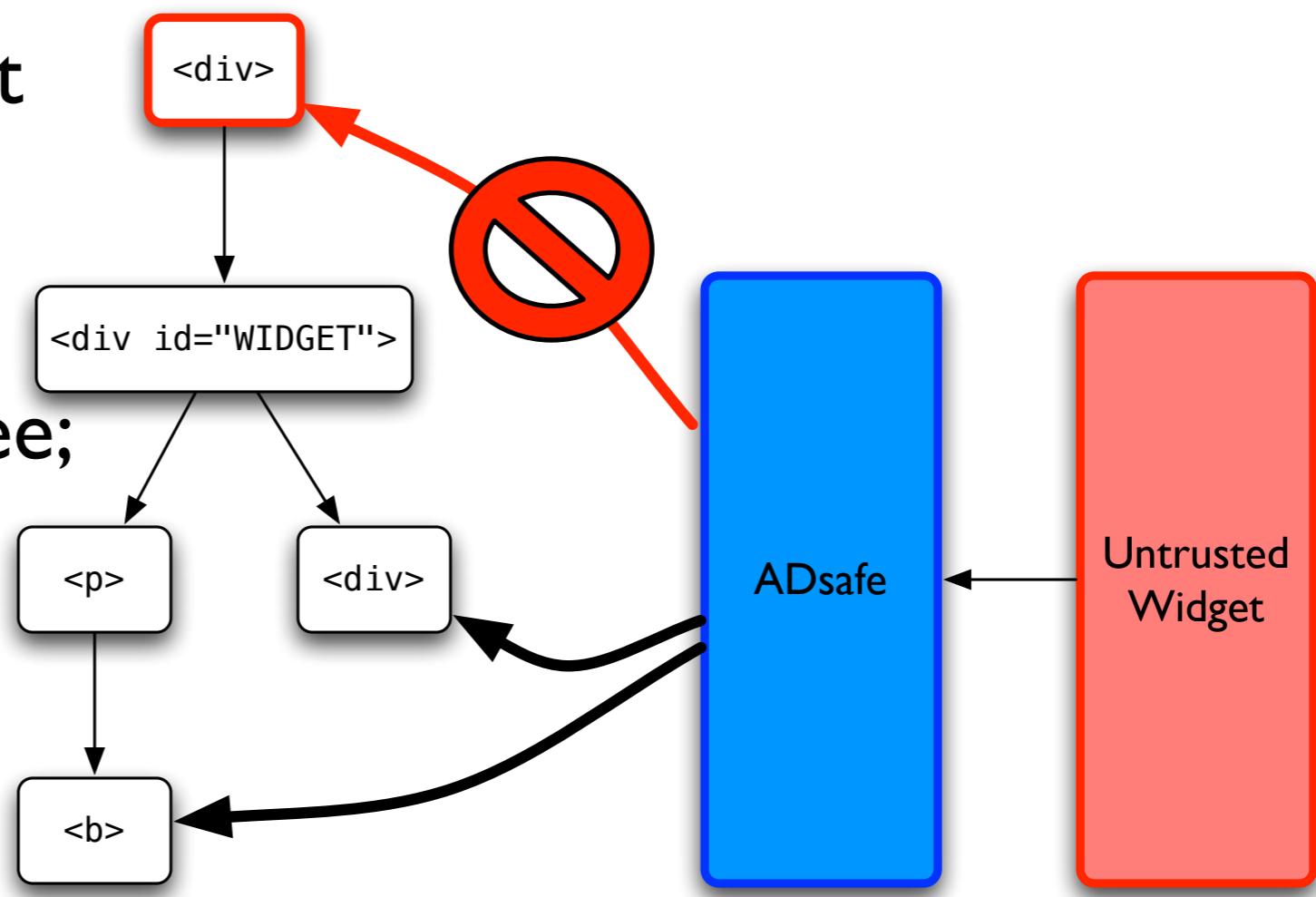


Definition I (ADsafe): If all embedded widgets pass JSLint, then:

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

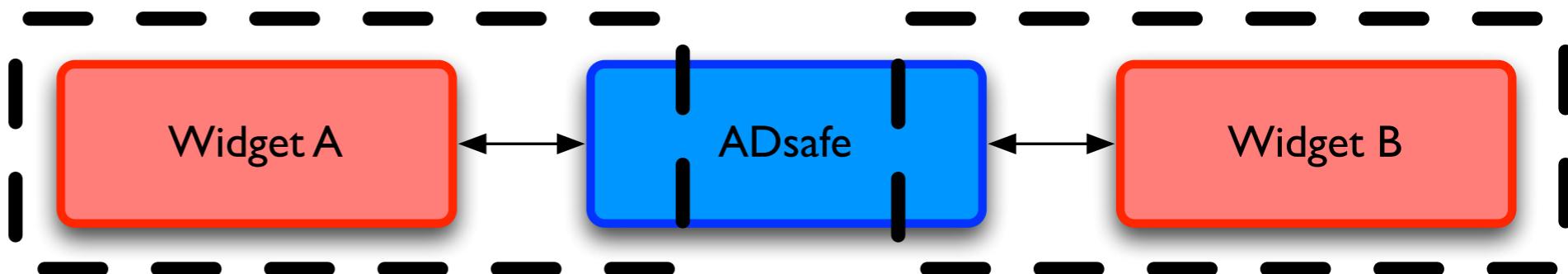
2. Widgets cannot obtain direct references to DOM nodes;

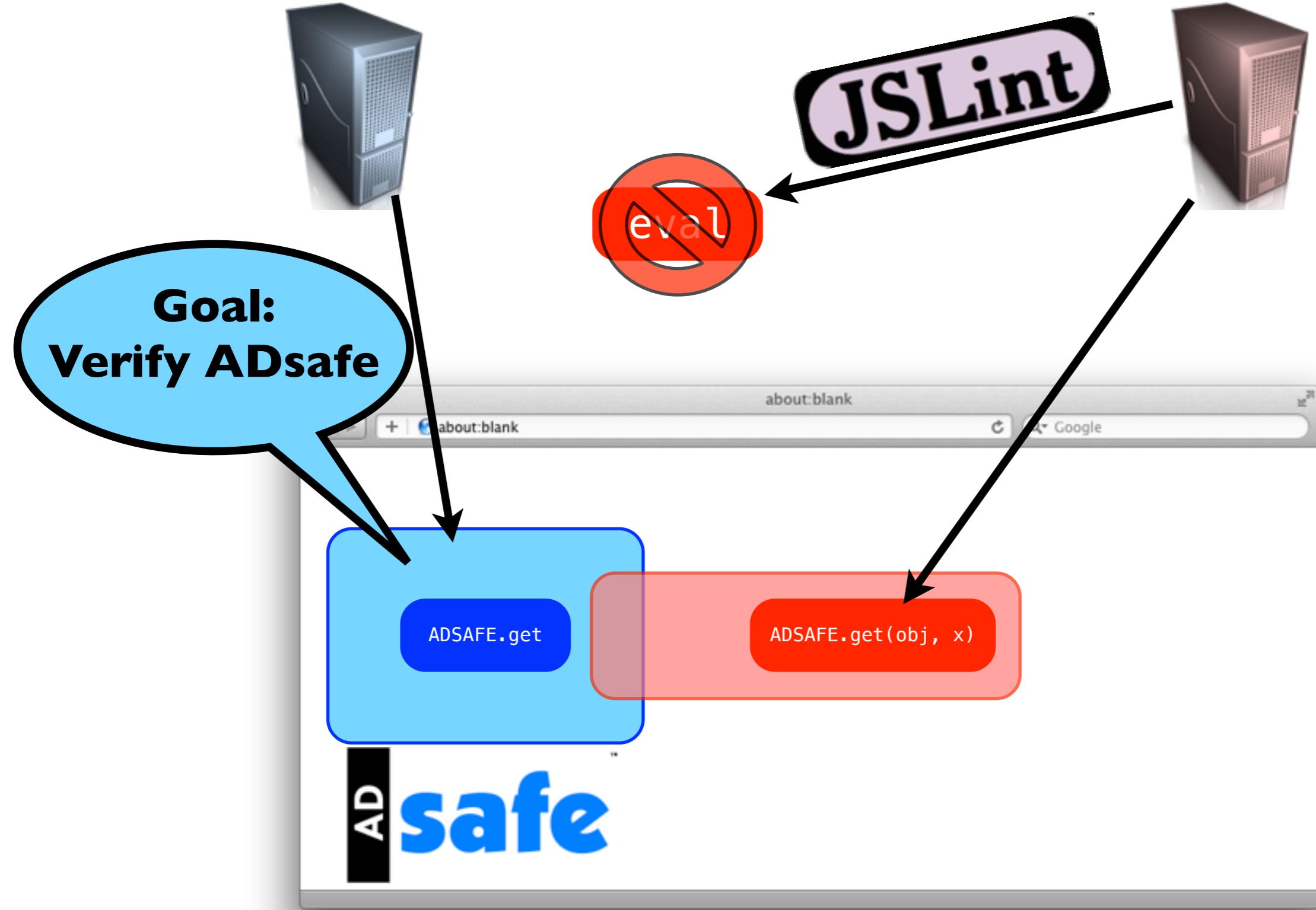
3. Widgets cannot affect the DOM outside of their subtree;
and

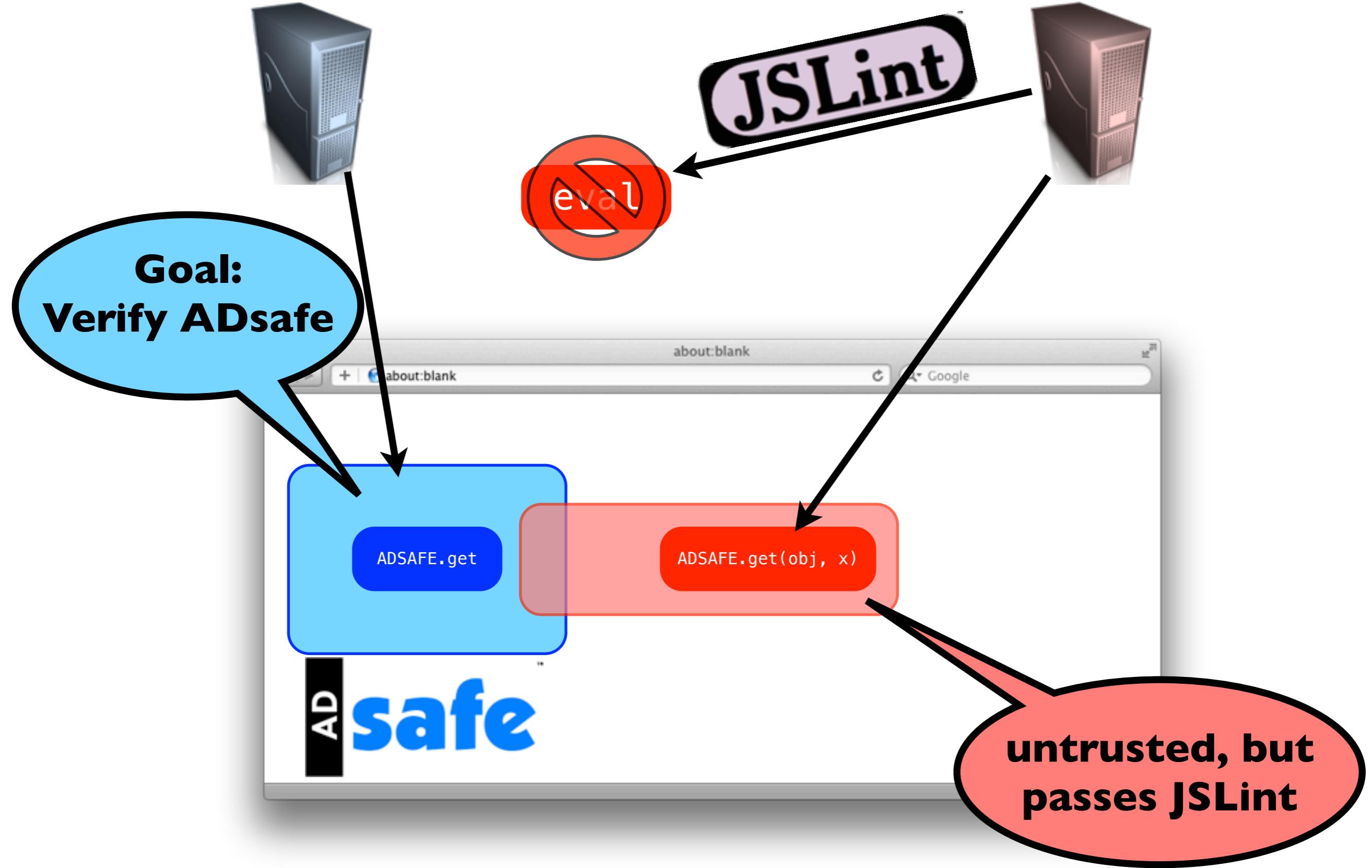


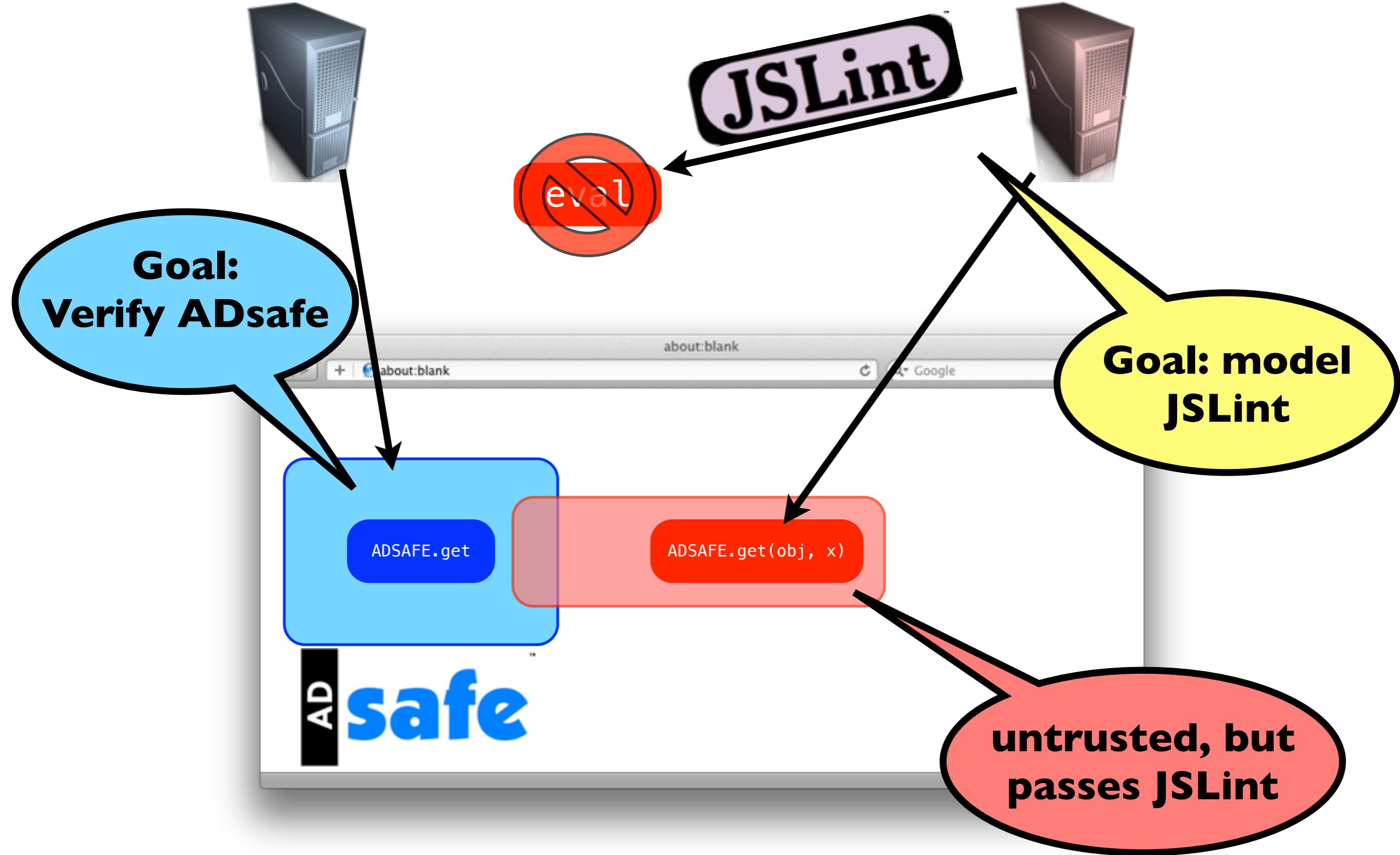
Definition I (ADsafe): If all embedded widgets pass JSLint, then:

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.





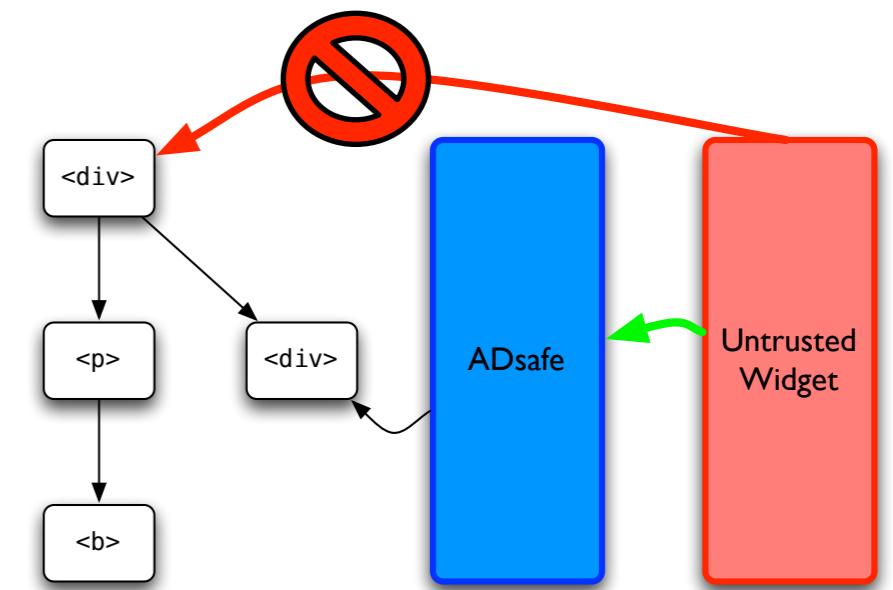




JSLint ensures:

no DOM
references

node



“Widgets cannot obtain direct references to DOM nodes.”

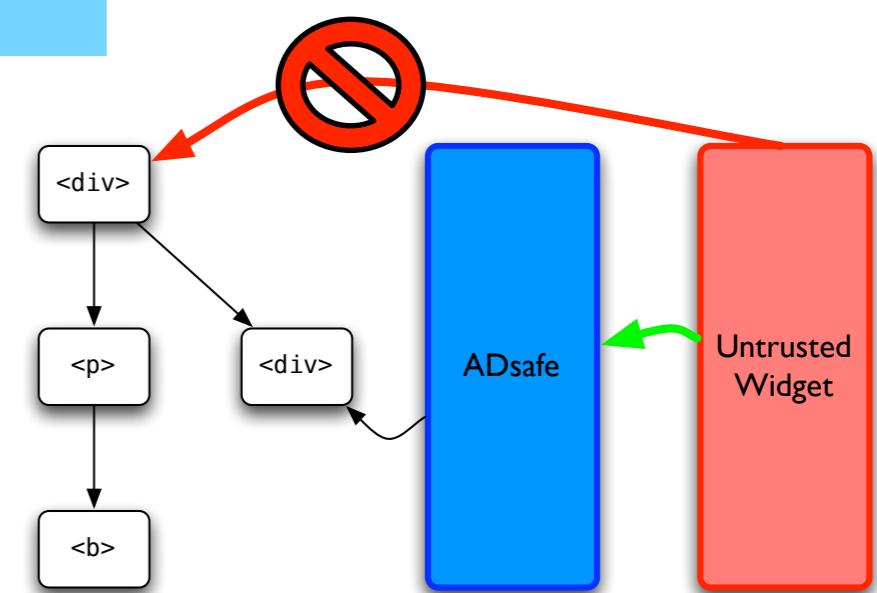
JSLint ensures:

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

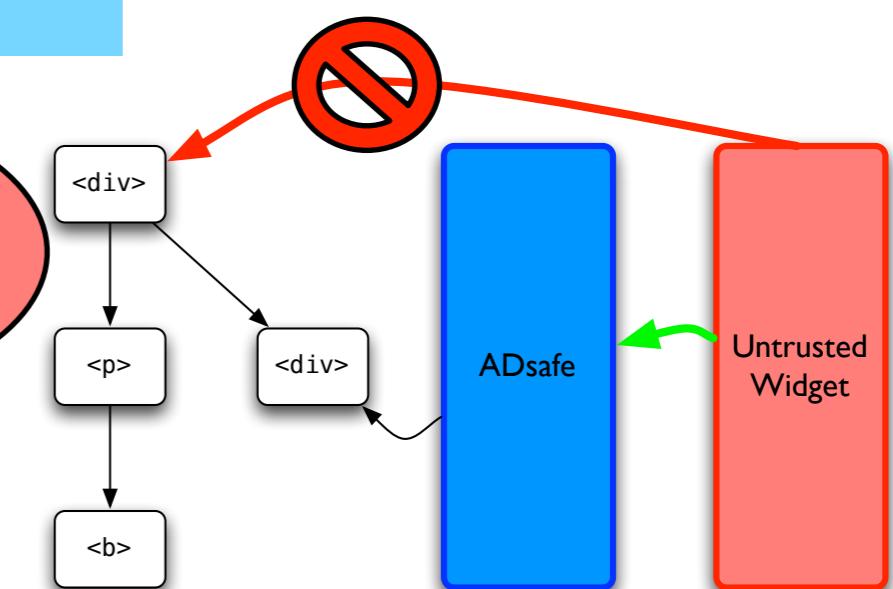
node

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.”*

JSInt ensures:

no DOM
references

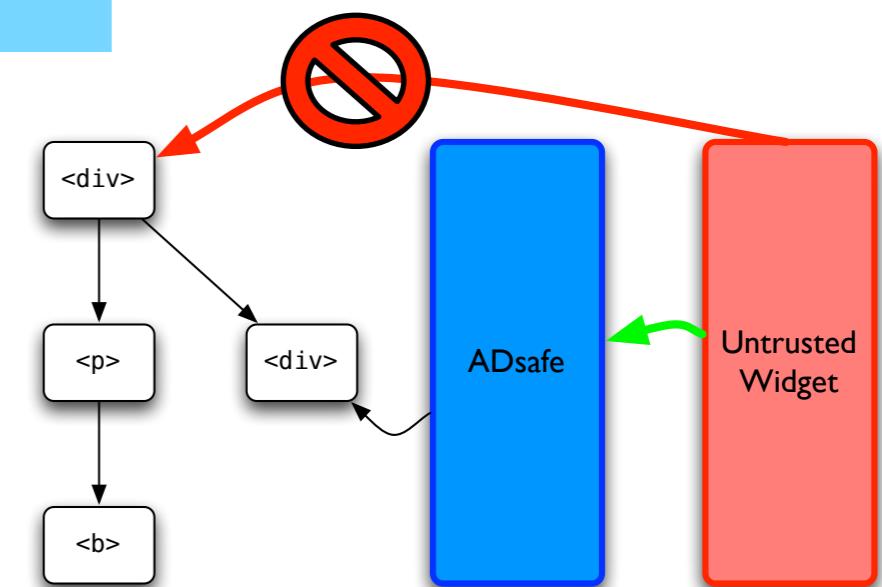
node

ADsafe ensures:
only “safe”
methods on
bunches

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

JSInt ensures:
__nodes__ is
“private”

bunch.__nodes__



“Widgets cannot obtain direct references to DOM nodes.”

JSInt ensures:

no DOM
references

node

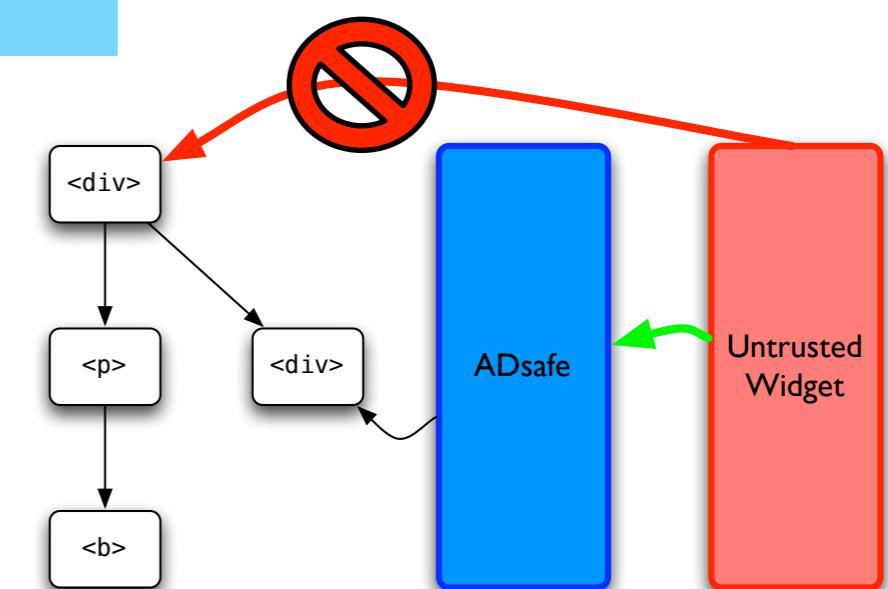
ADsafe ensures:
only “safe”
methods on
bunches

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

JSInt ensures:
__nodes__ is
“private”

bunch.__nodes__

bunch.append(...)



“Widgets cannot obtain direct
references to DOM nodes.”

Exploit append to
return nodes?

JSInt ensures:

no DOM
references

node

ADsafe ensures:

only “safe”
methods on
bunches

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

JSInt ensures:

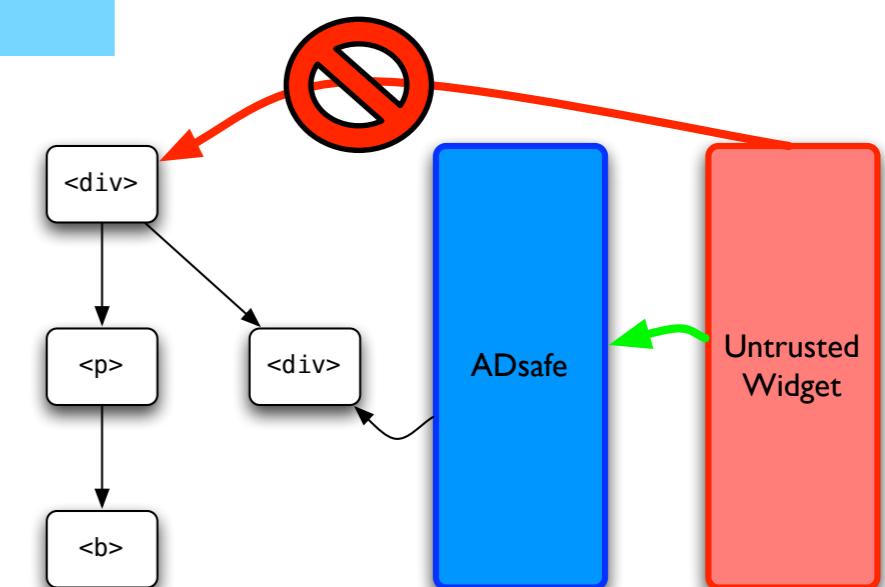
__nodes__ is
“private”

bunch.__nodes__

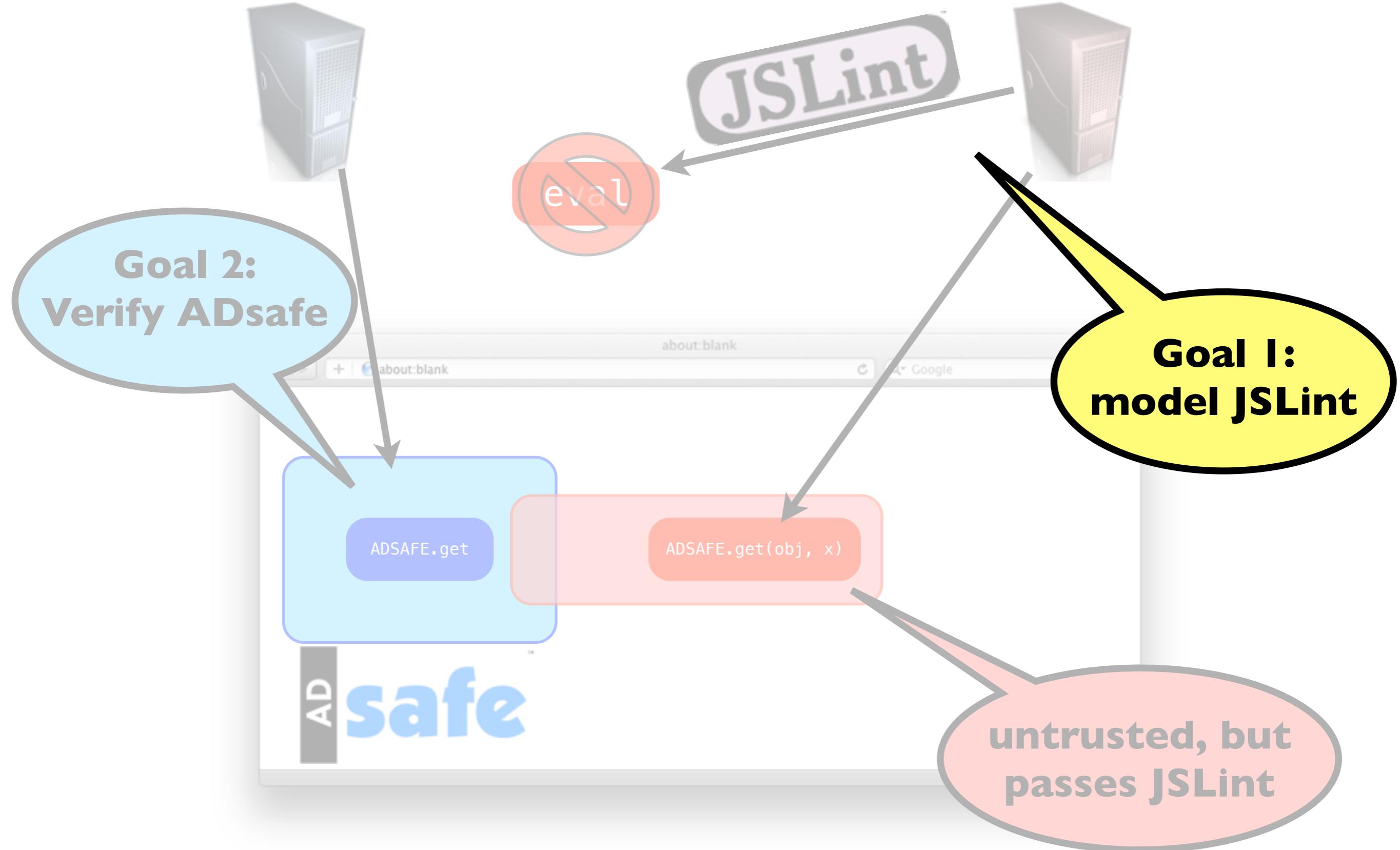
ADsafe ensures:

DOM nodes are
not returned

bunch.append(...)



“Widgets cannot obtain direct references to DOM nodes.”



- ✓ 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 +

{

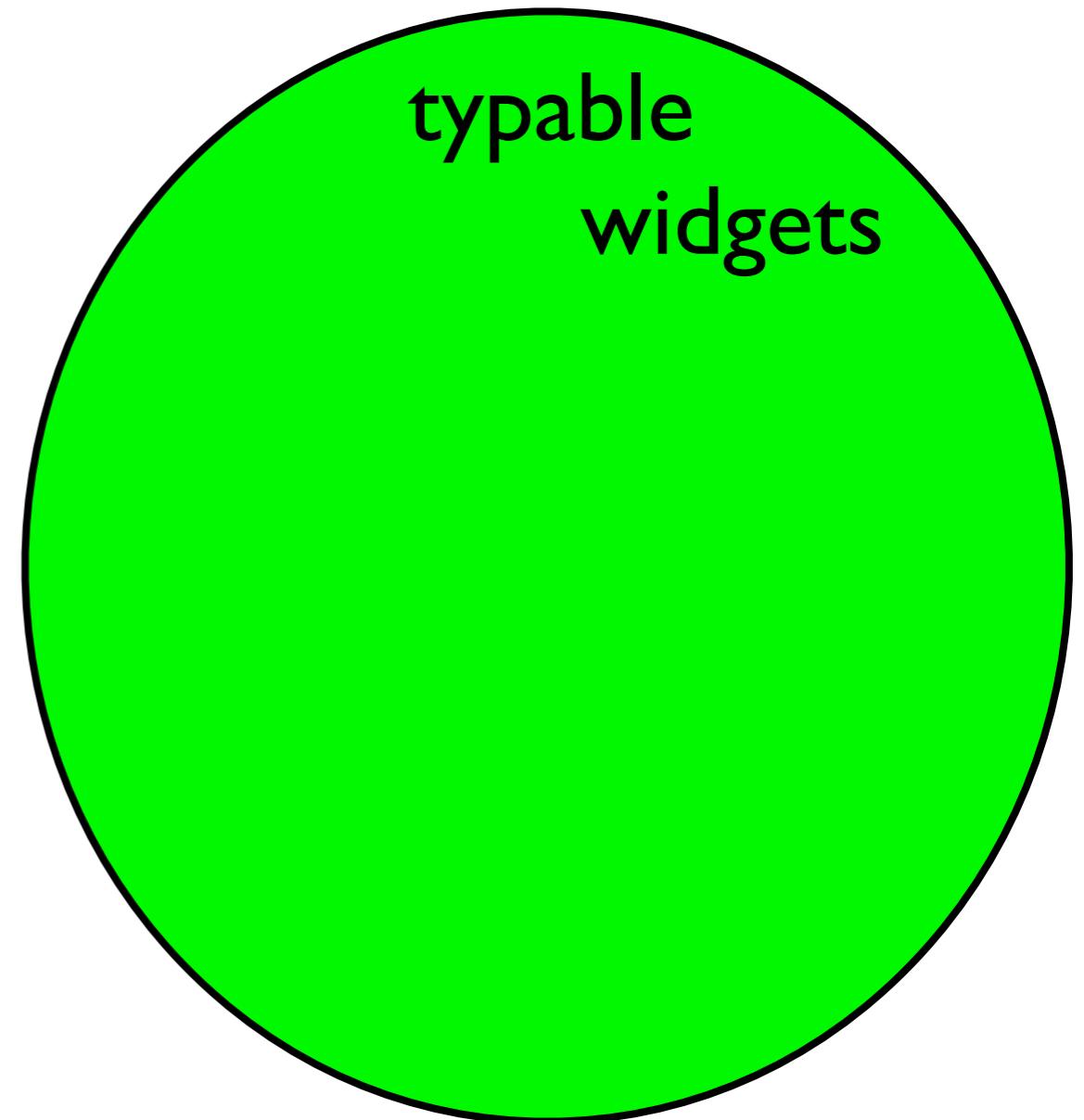
- ★: Widget
- __nodes__: Array<Node>
- caller: ☠
- prototype: ☠
- ...
- code : Widget × ... → Widget
- __proto__: Object + Function + Array + ...

}

JSLint

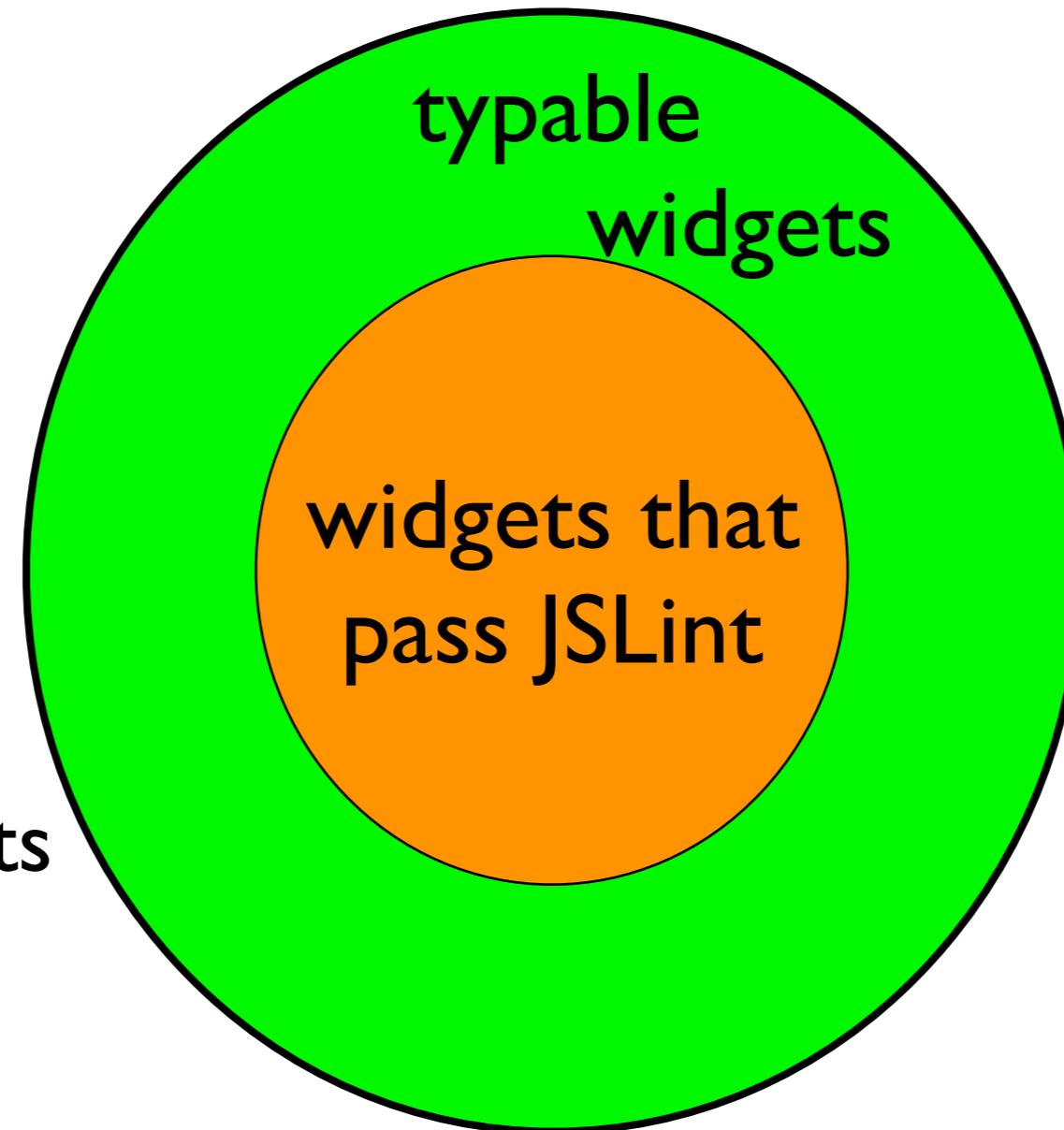


Widget type-checker



JSLint

Widget type-checker



Claim:

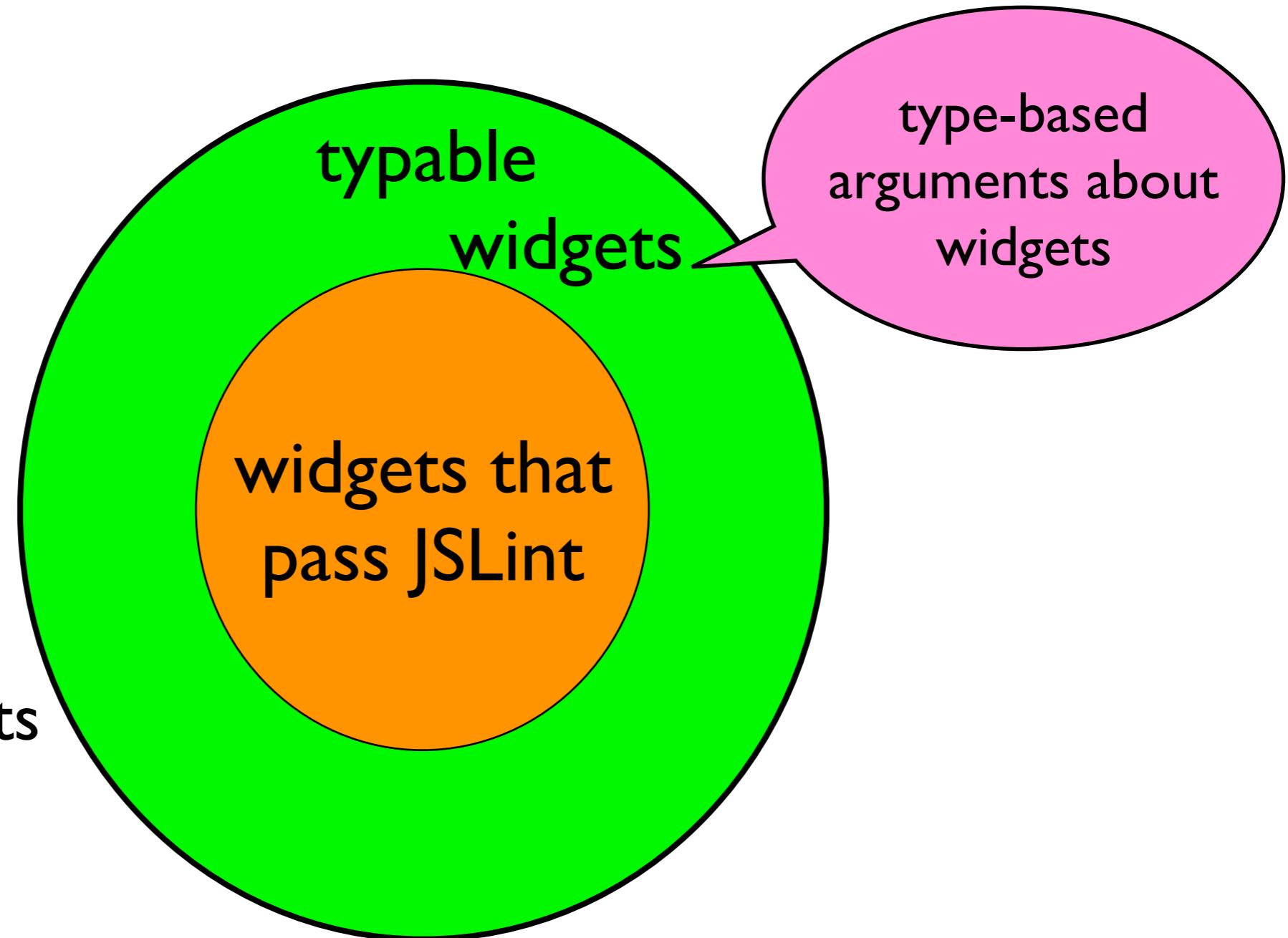
evidence:

1,100 LOC of tests

or, passing JSLint \Rightarrow Widget-typable

JSLint

Widget type-checker

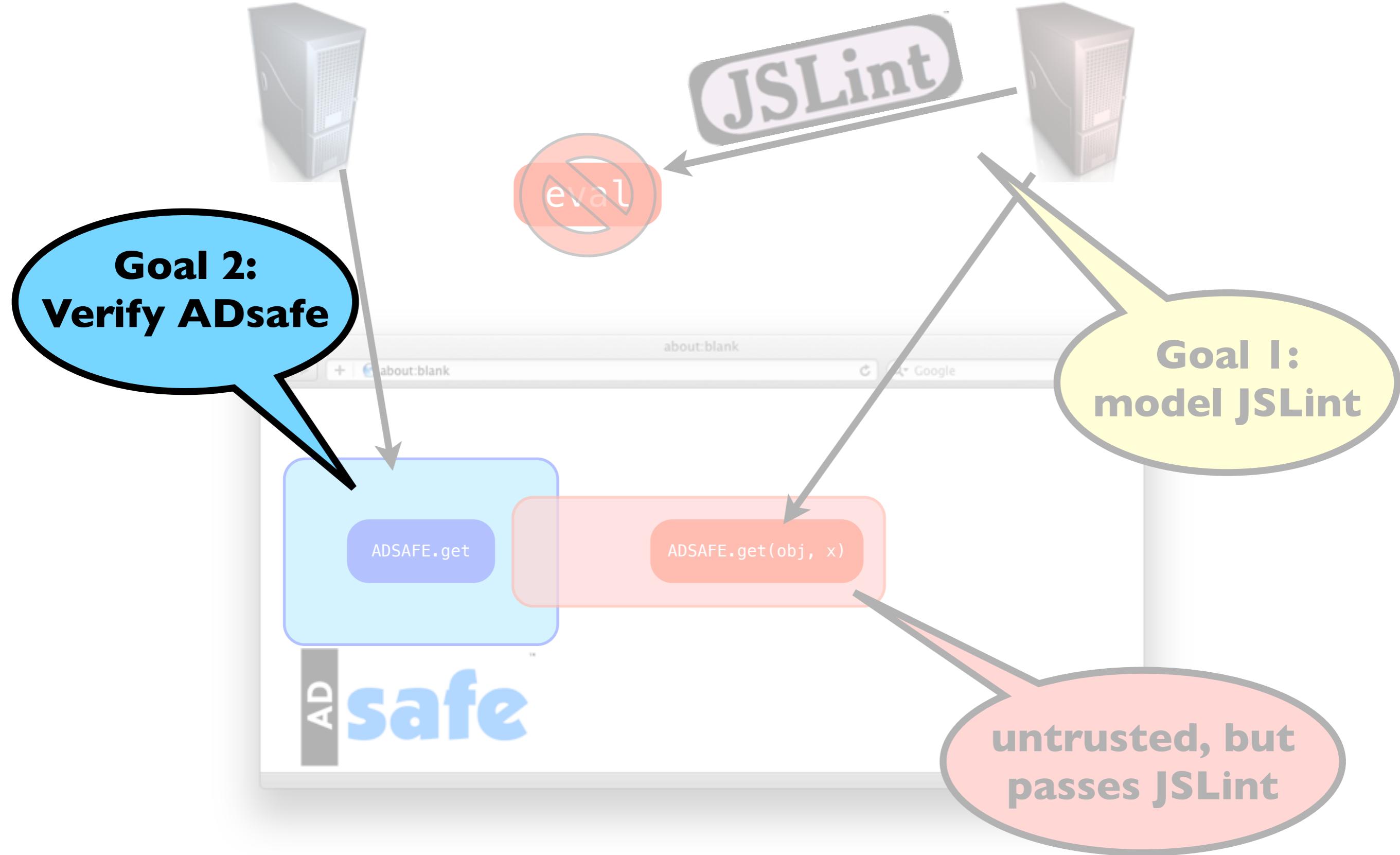


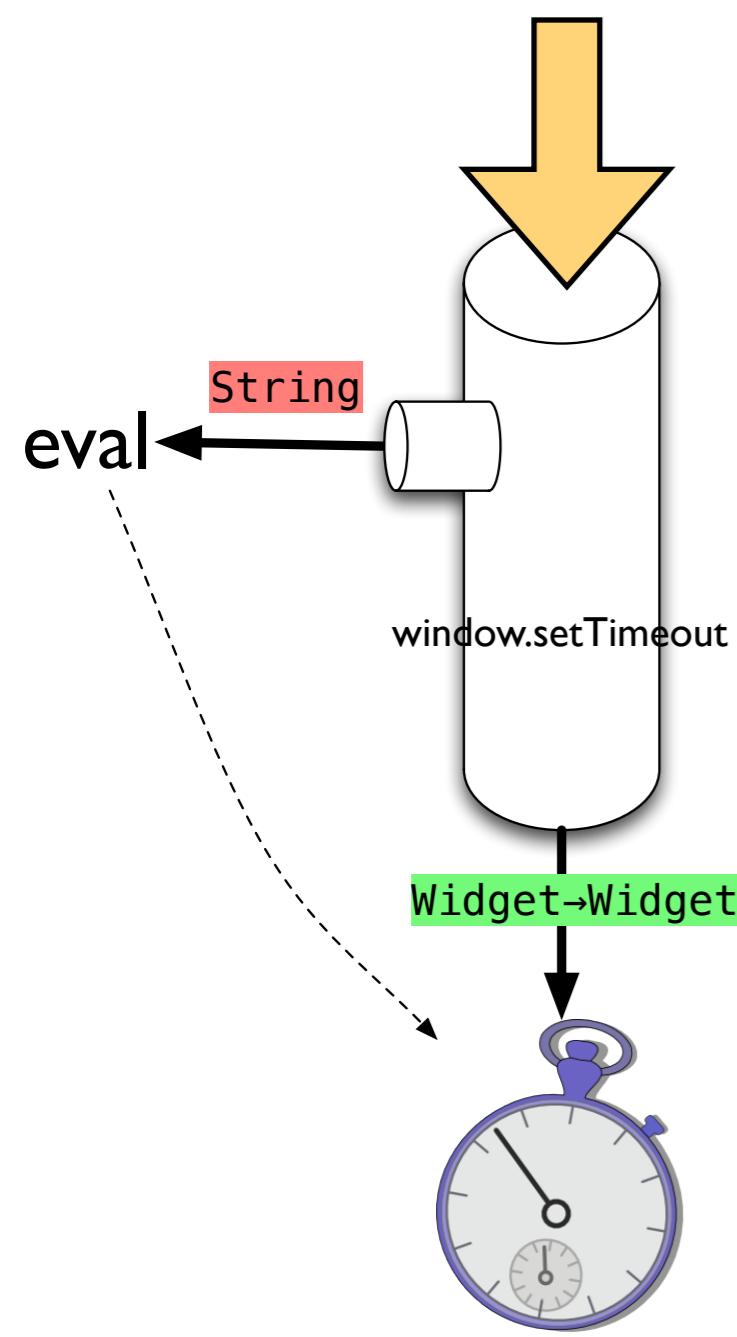
Claim:

evidence:

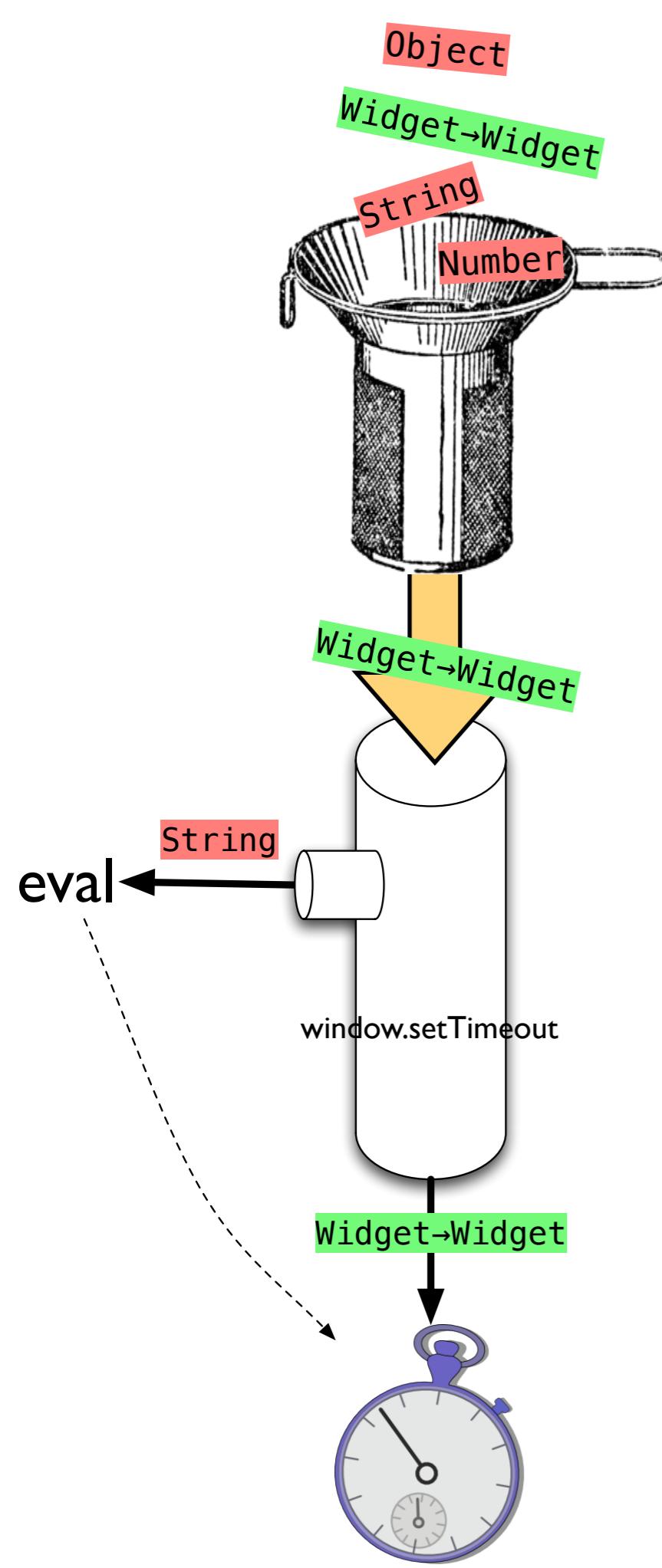
1,100 LOC of tests

or, passing JSLint \Rightarrow Widget-typable

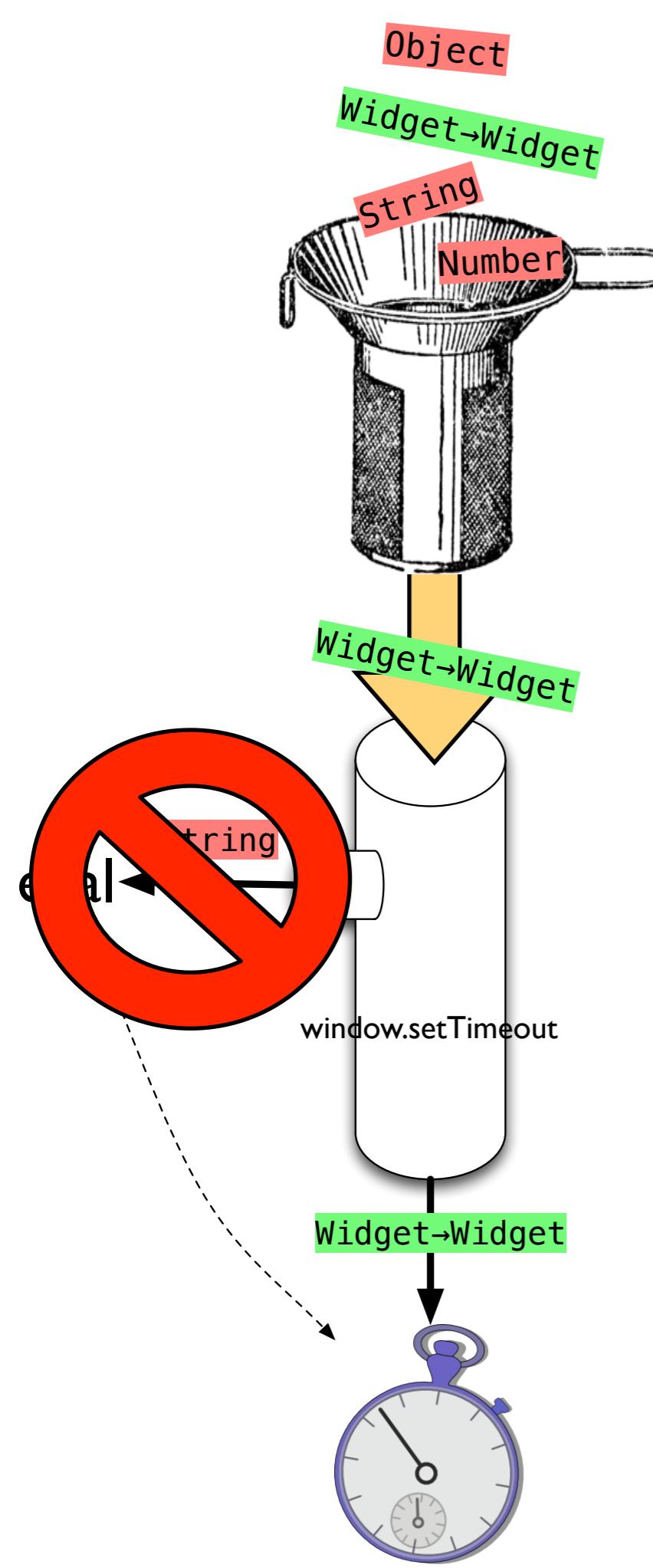




```
window.setTimeout(callback, delay);
```

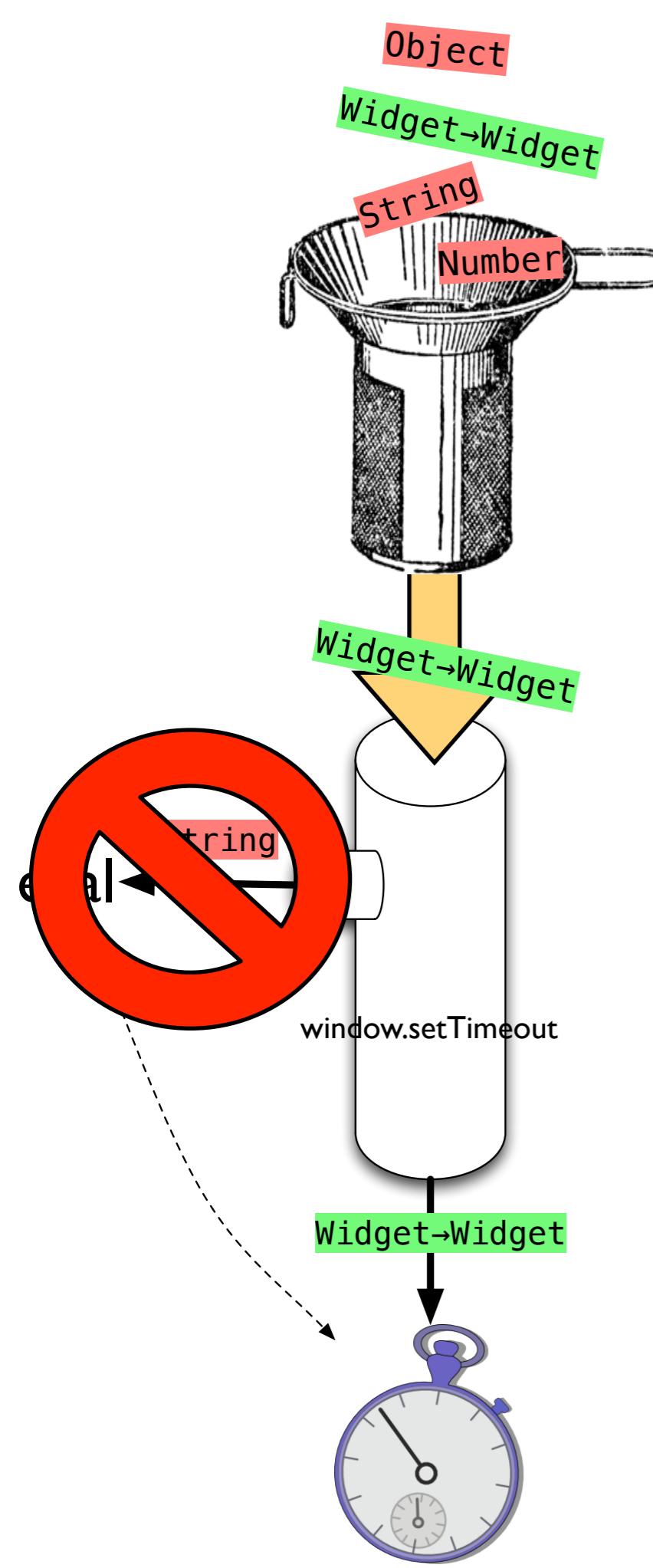


```
/*: Widget × Widget → Widget */
ADSAFE.later = function(callback, delay)
{
  if (typeof callback !== "function") {
    throw "expected function";
  }
  window.setTimeout(callback, delay);
}
```



```
Object
Widget->Widget
String
Number
Widget->Widget
Widget->Widget
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);
}
```

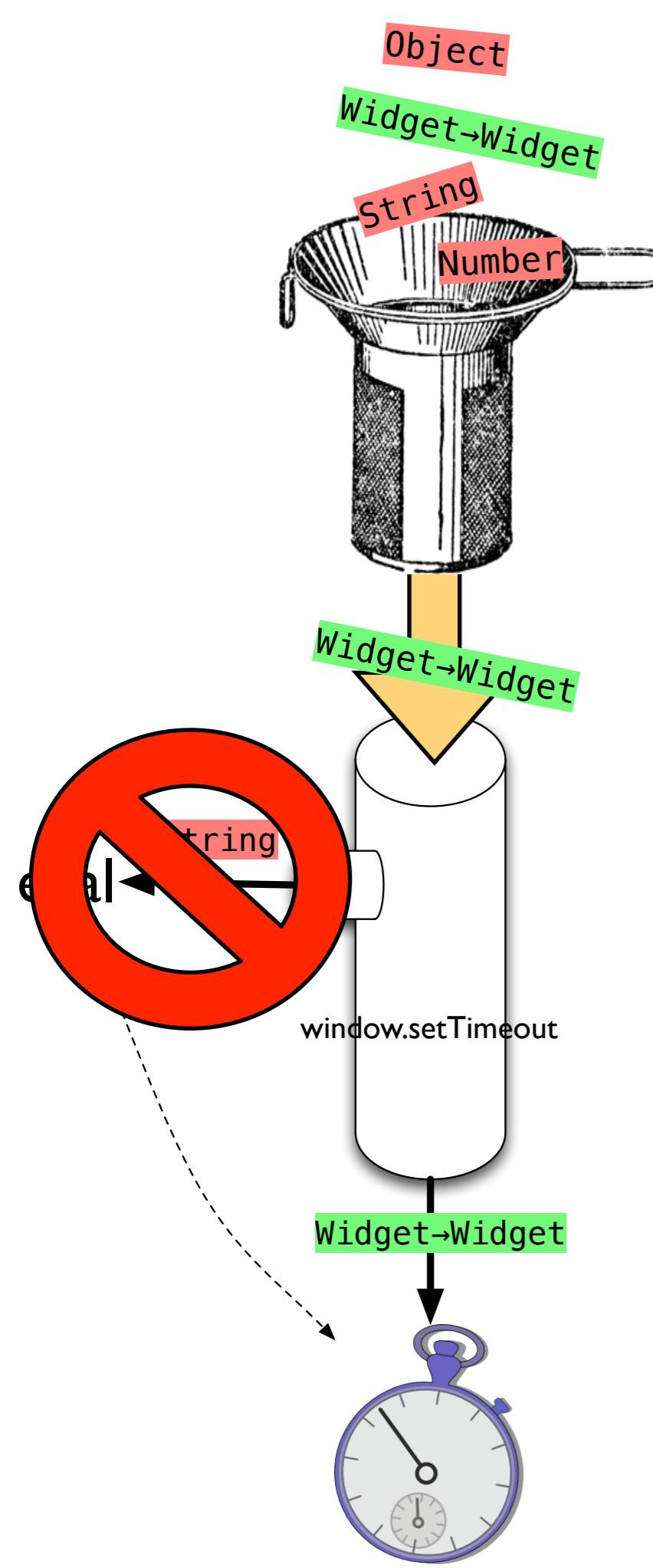


```
window : {  
  eval: 💀,  
  setTimeout : (Widget x ... → Widget) x Widget → Undefined,  
  ...  
}
```



Widget

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

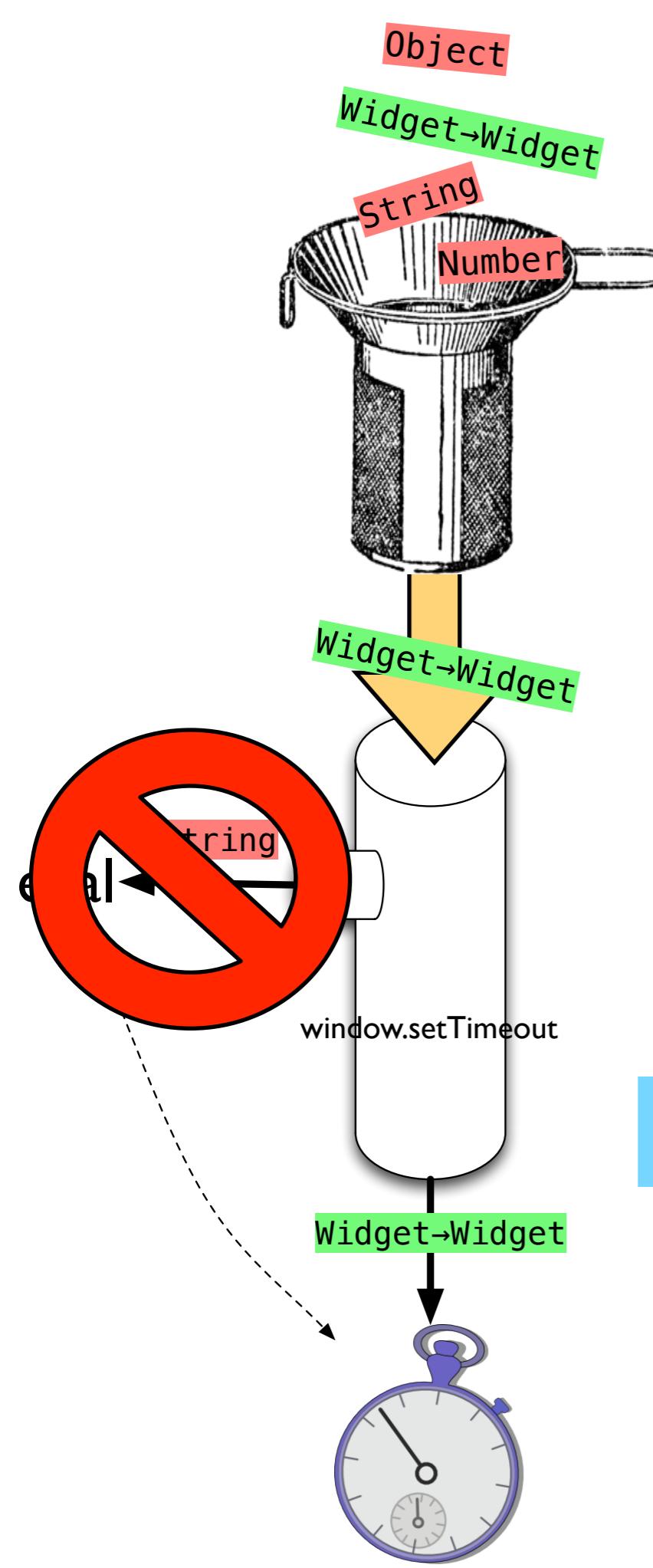


```
window : {  
  eval: 💀,  
  setTimeout : (Widget x ... → Widget) x Widget → Undefined,  
  ...  
}
```



Widget

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



```
window : {  
  eval: 💀,  
  setTimeout : (Widget x ... → Widget) x Widget → Undefined,  
  ...  
}
```



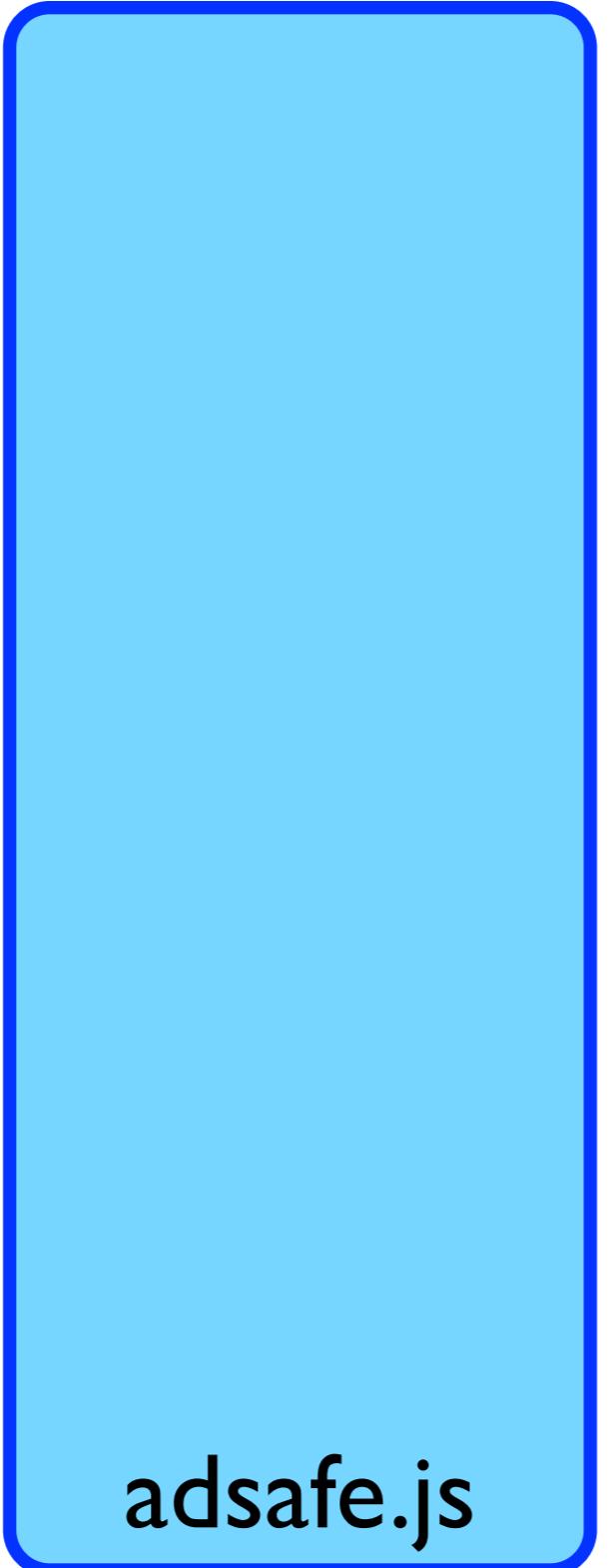
Widget

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

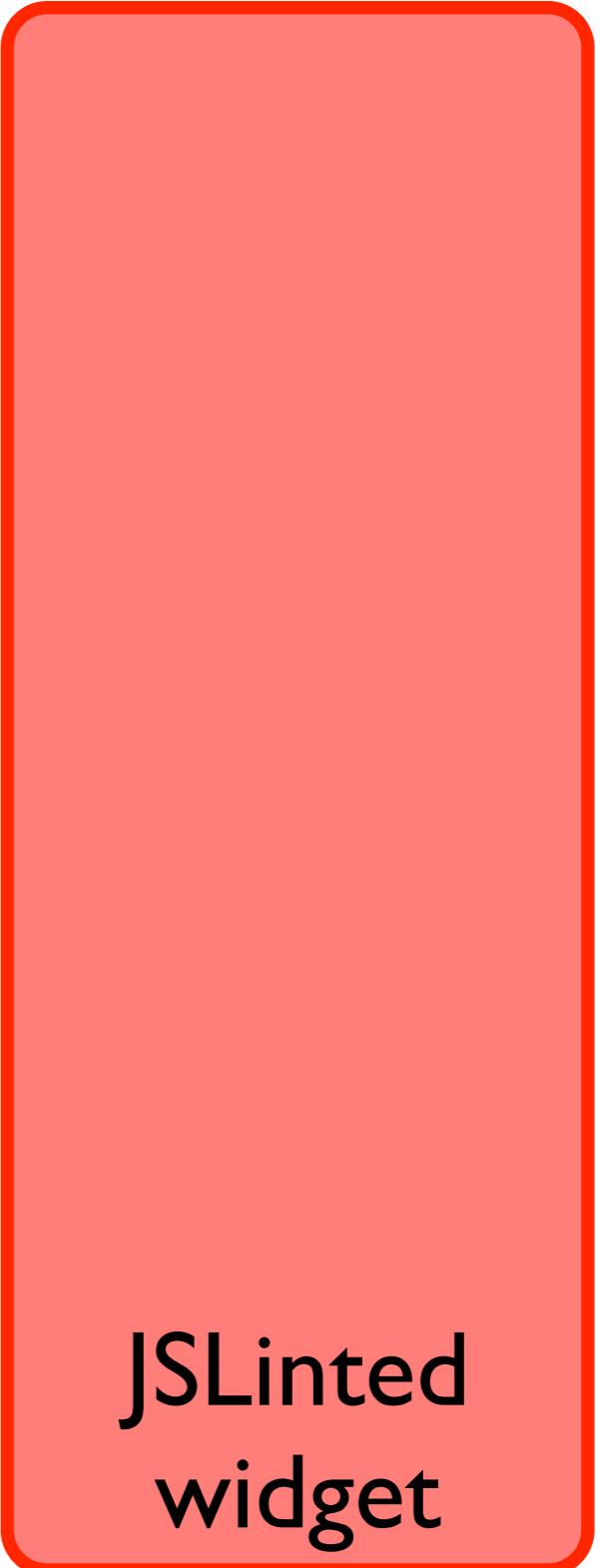
`Widget × ... → Widget`

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

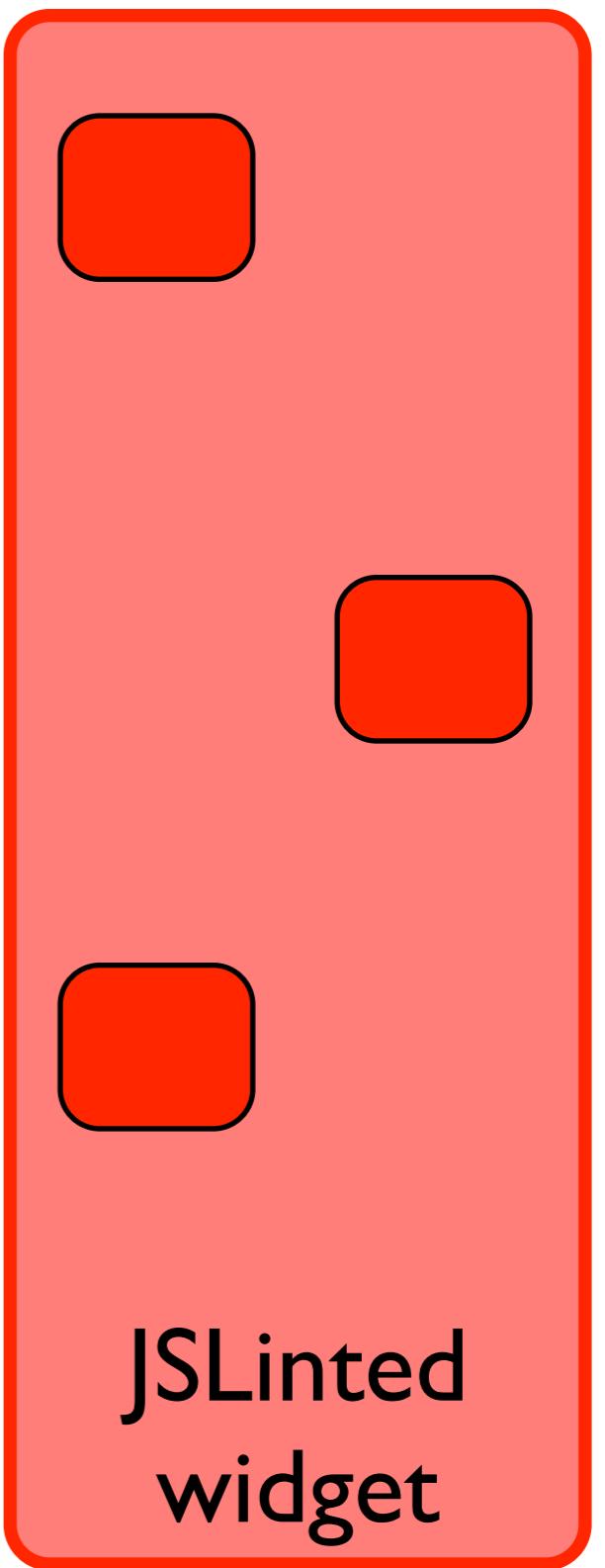
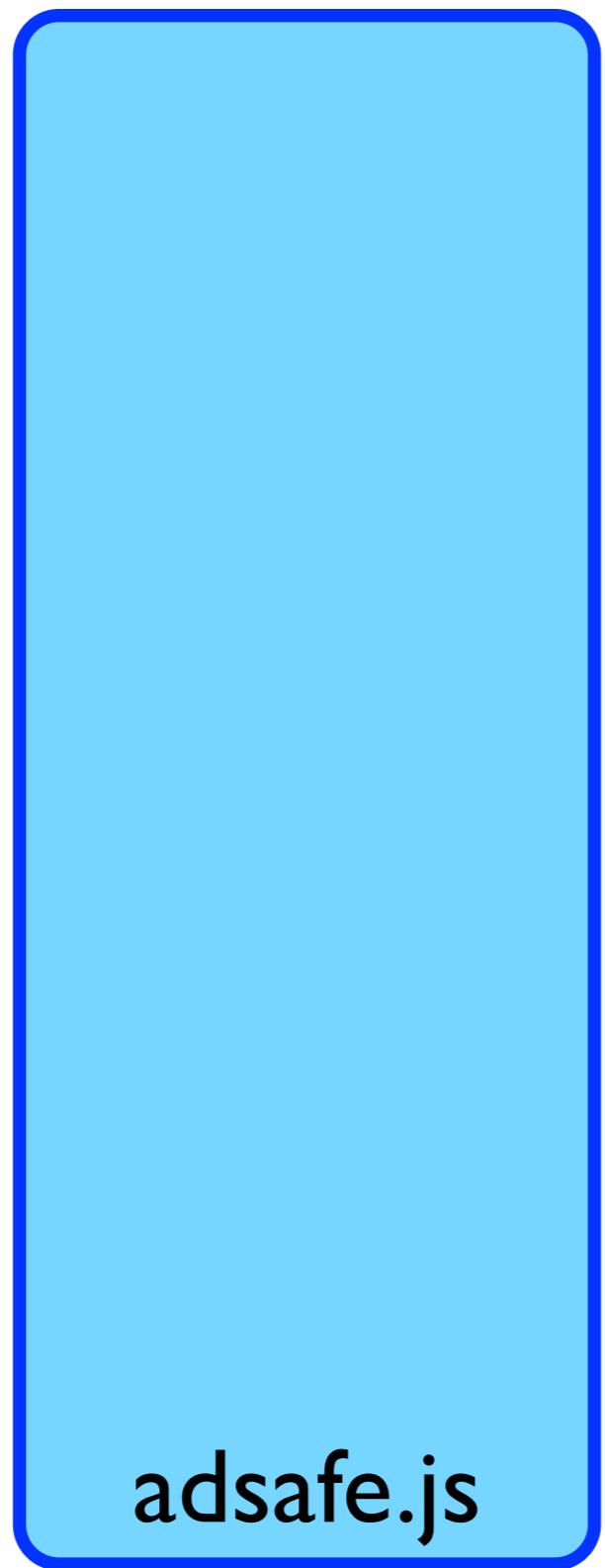
—Politz et al. USENIX Security 2011 and Guha, Saftoiu, Krishnamurthi. ESOP 2011.

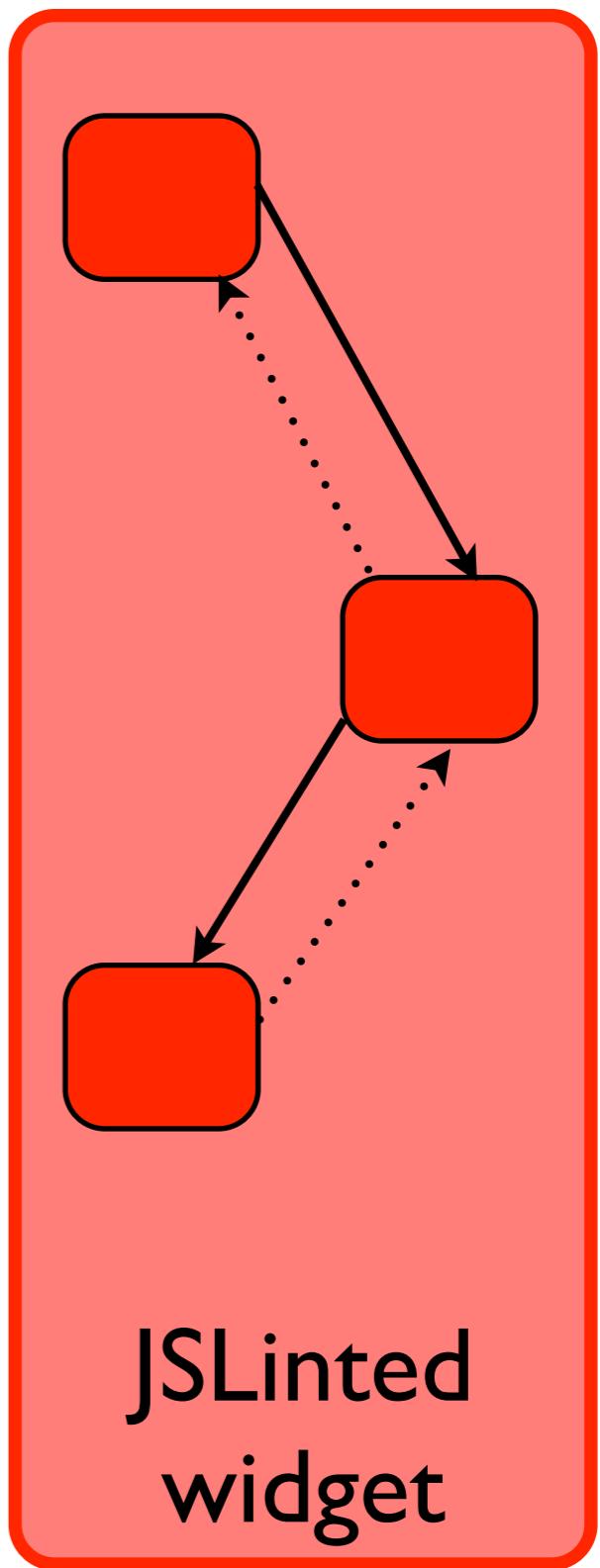
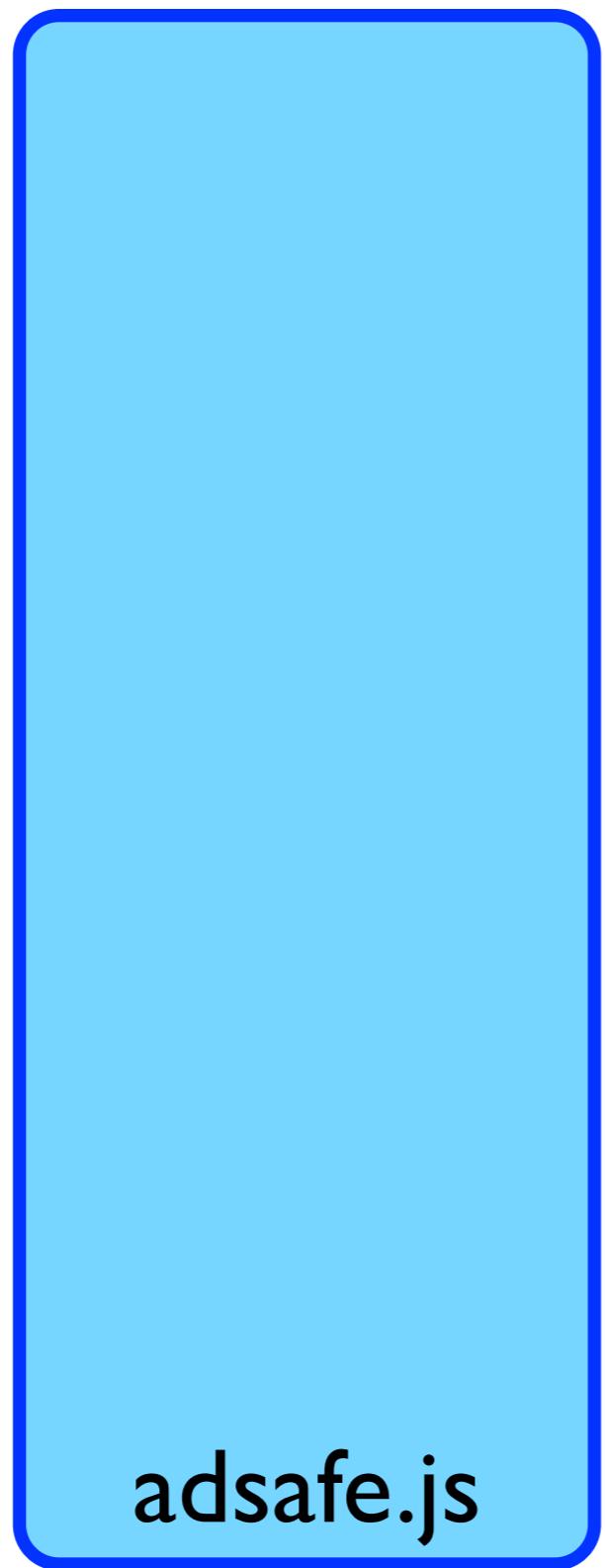


adsafe.js

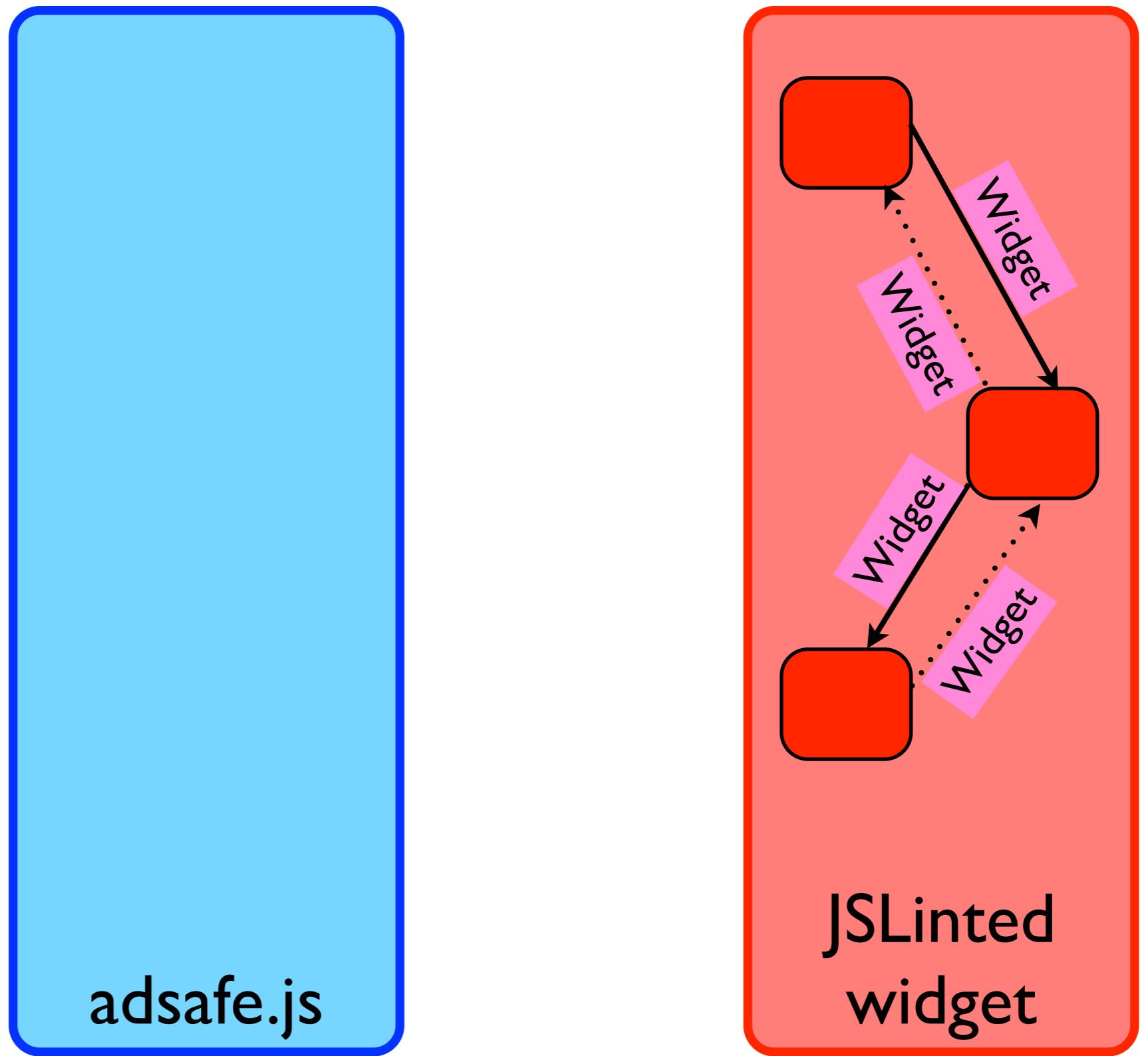


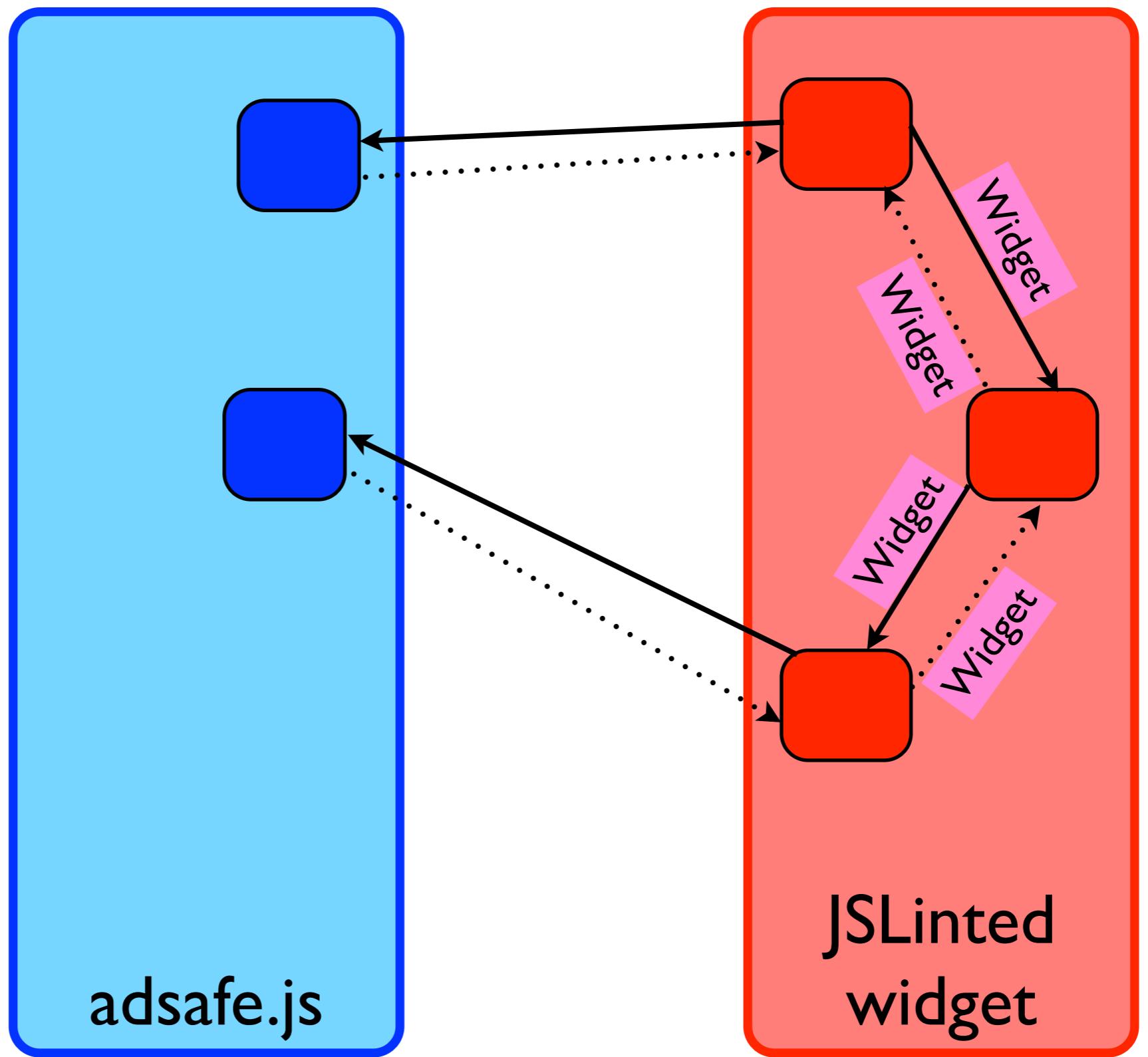
**JS Linted
widget**

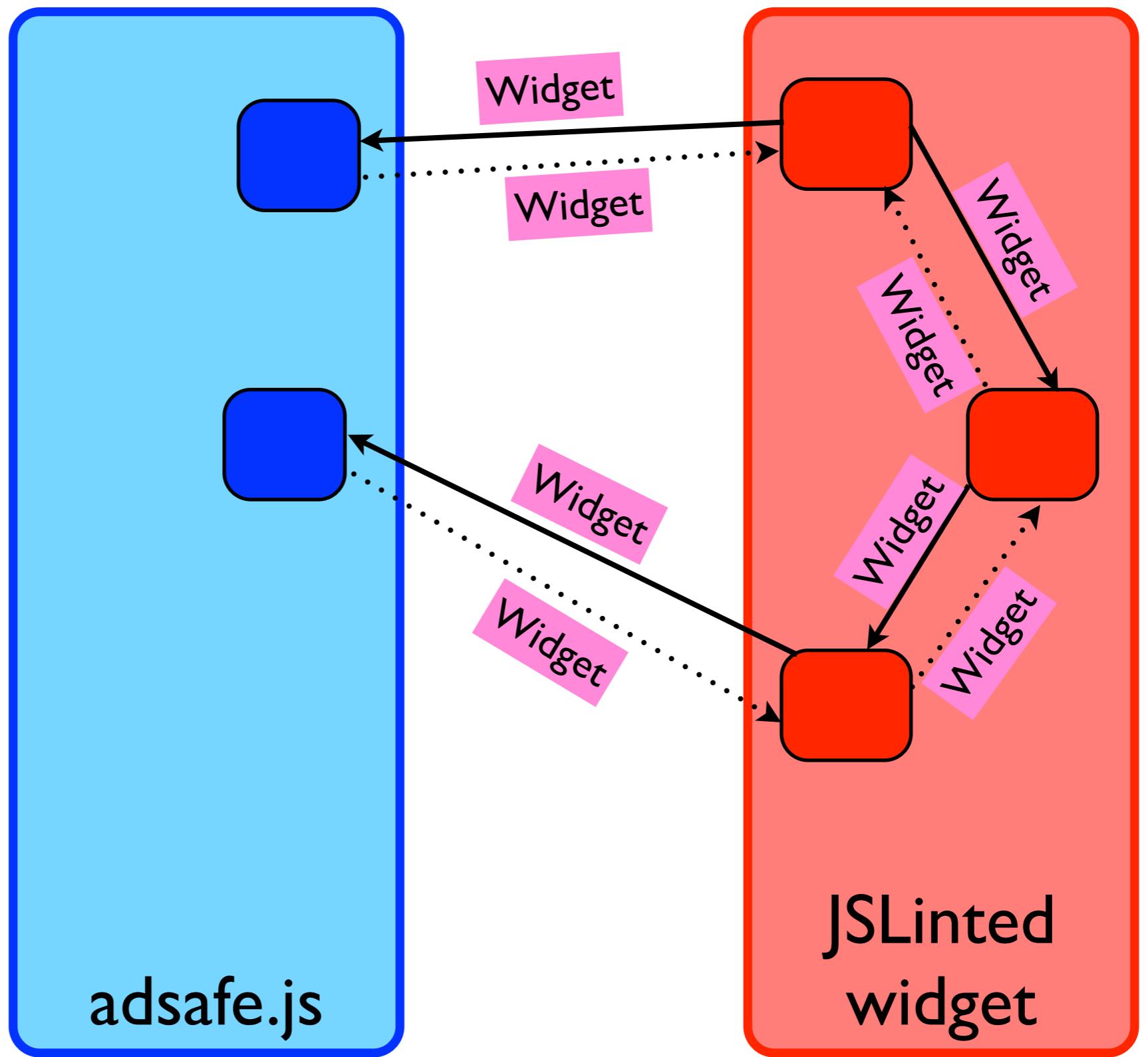


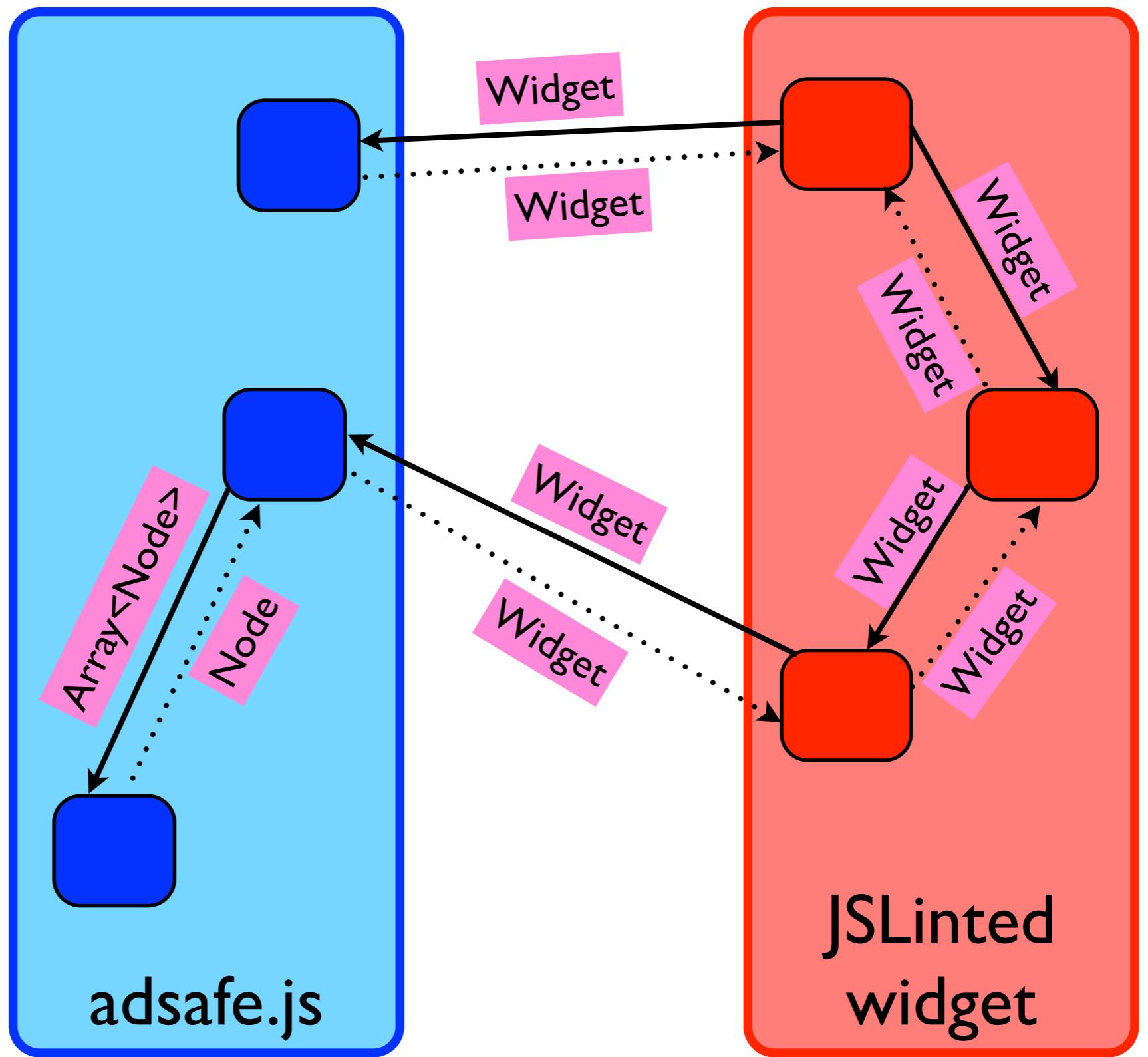


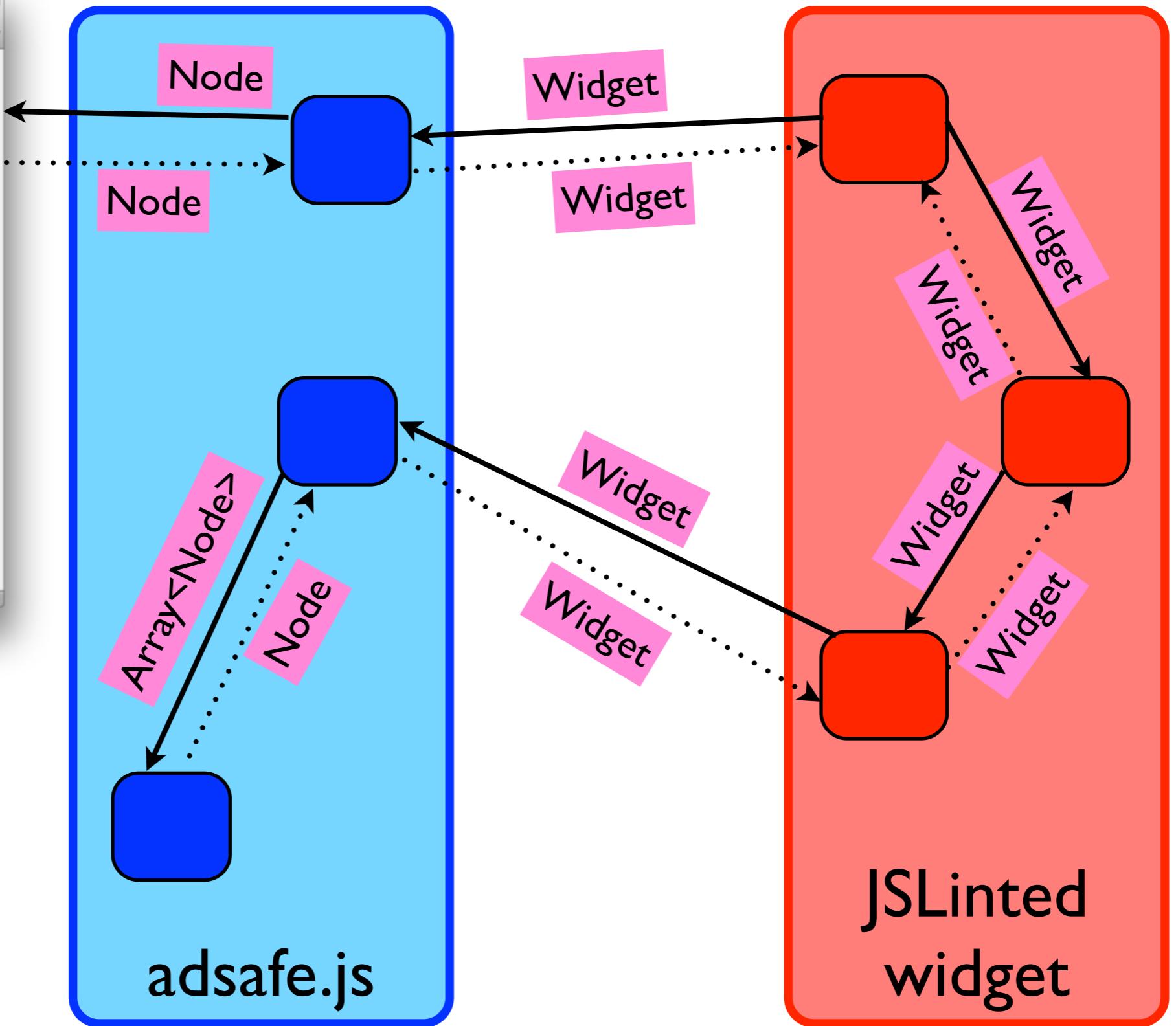
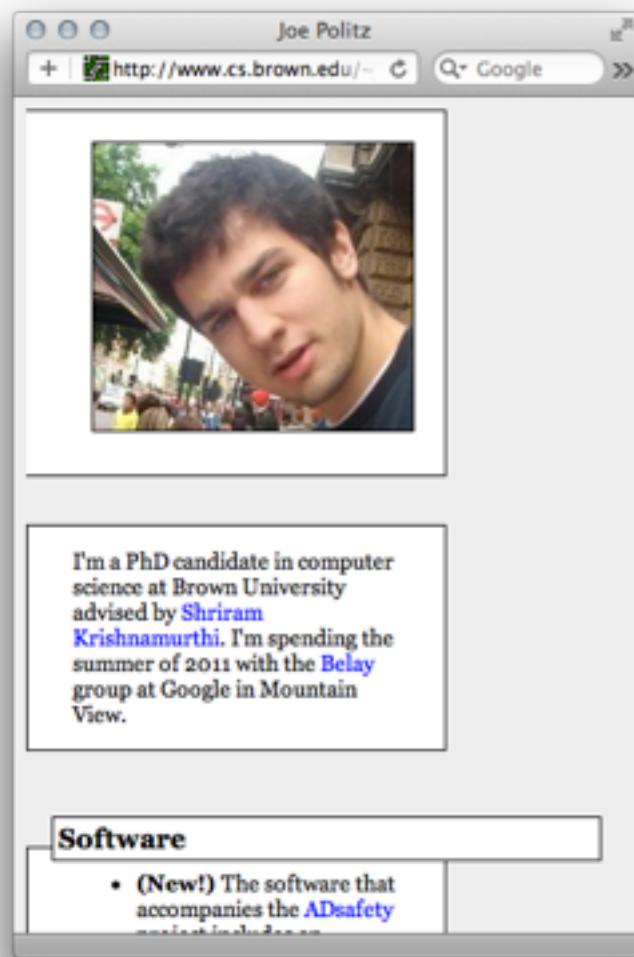
**JS Linted
widget**

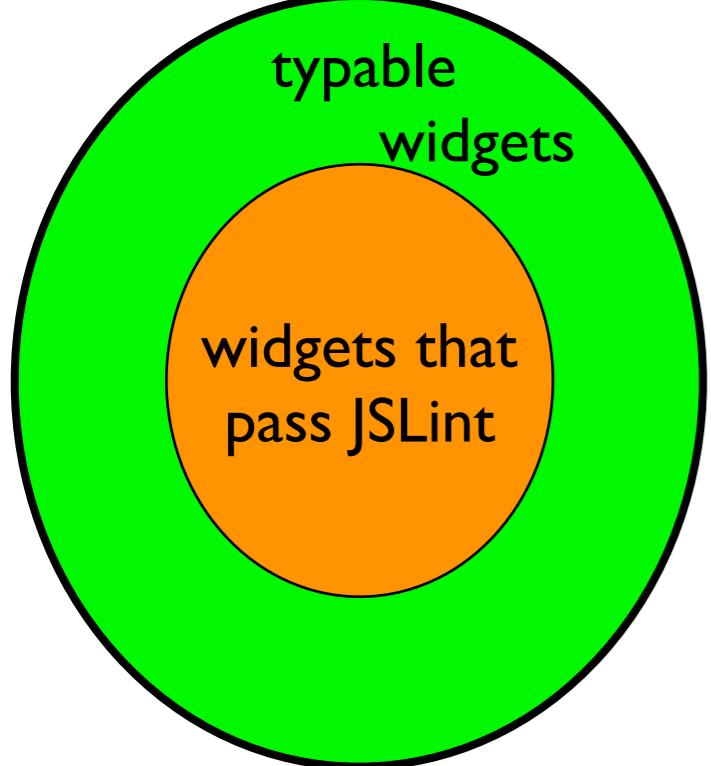




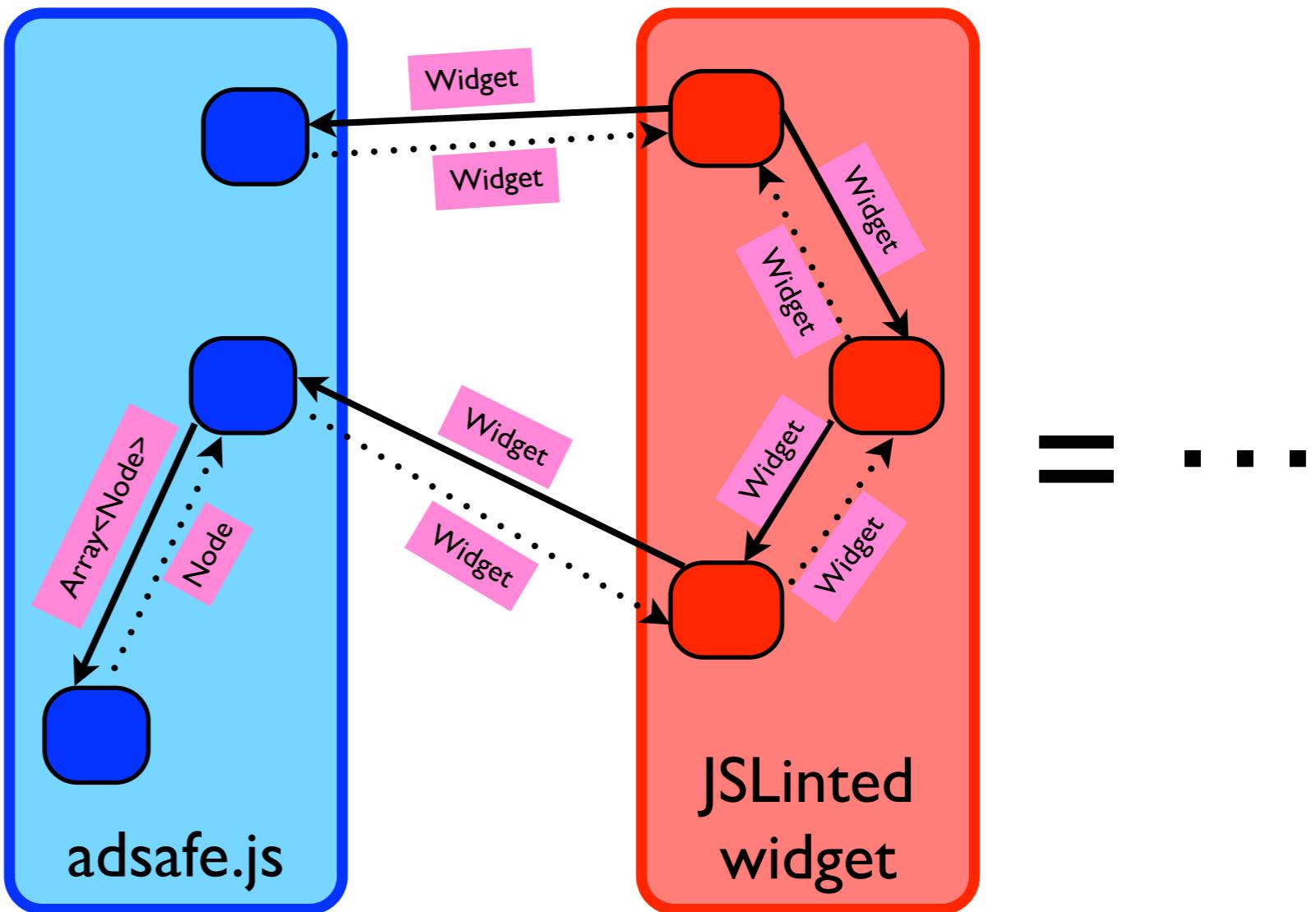






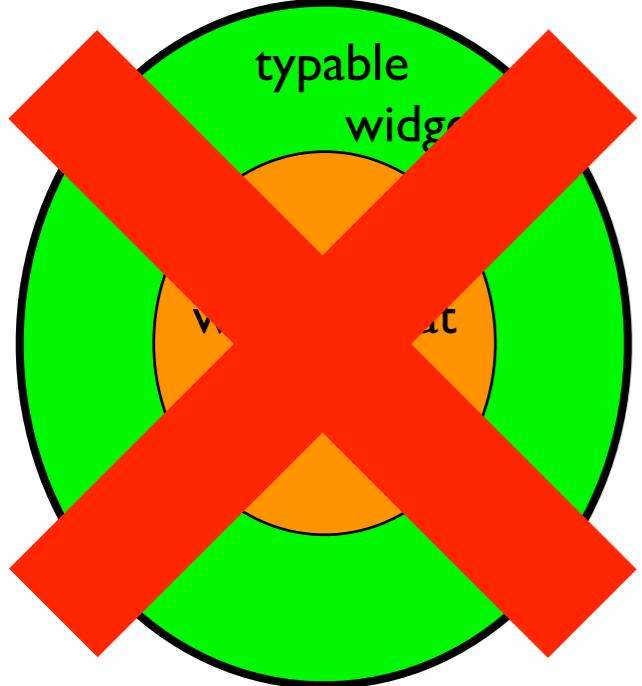


+

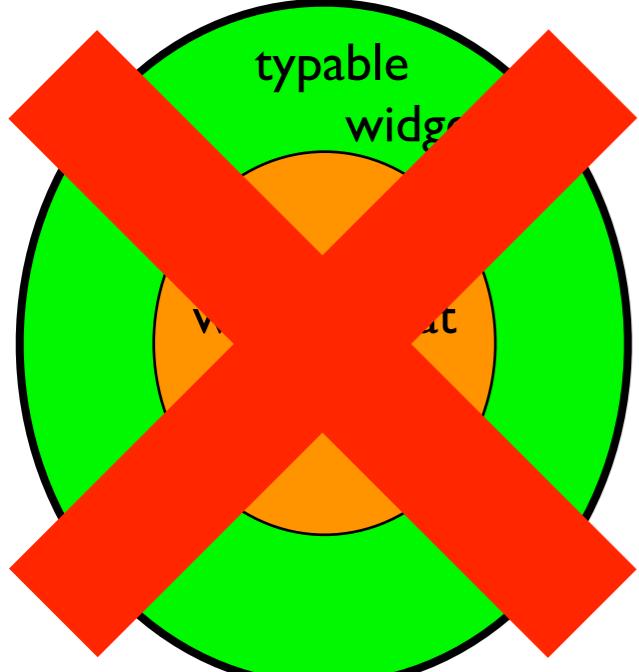


JSLint model

Type-checked ADsafe

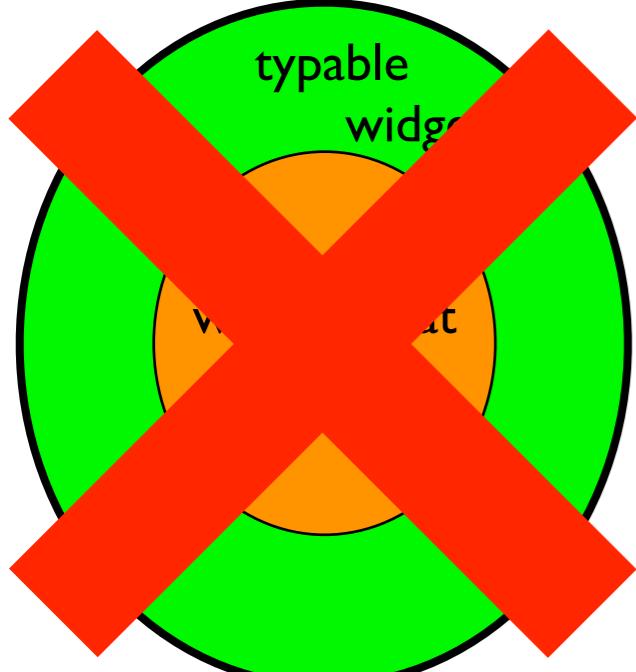


```
var fakeNode = {  
  tagName: "div",  
  appendChild: function(elt) {  
    var win = elt.ownerDocument.defaultView;  
    win.eval("alert('hacked')");  
  }  
};
```



Rejected by JSLint

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



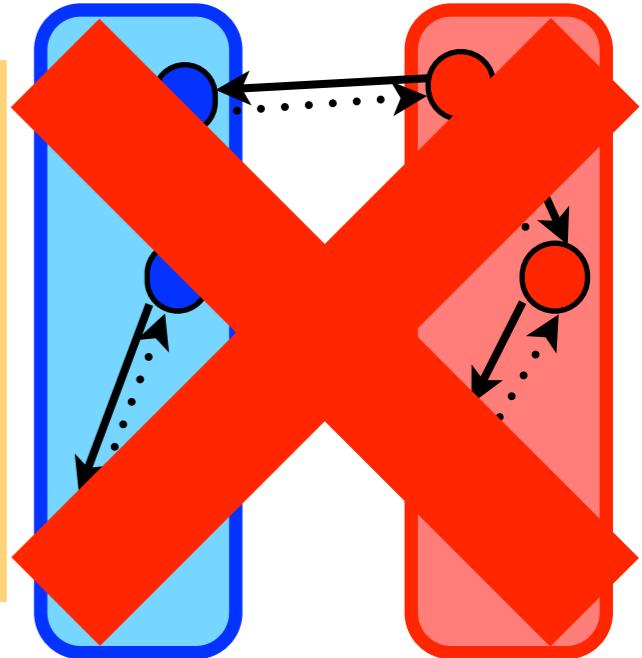
Rejected by JSLint

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

Accepted by JSLint

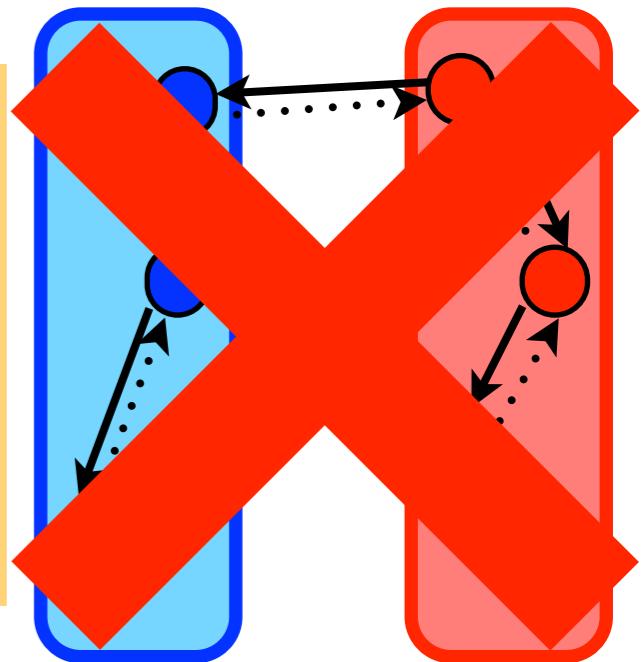
type error: expected Array<HTML>,
received Array<Widget>

```
/*: 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 ...
}
```



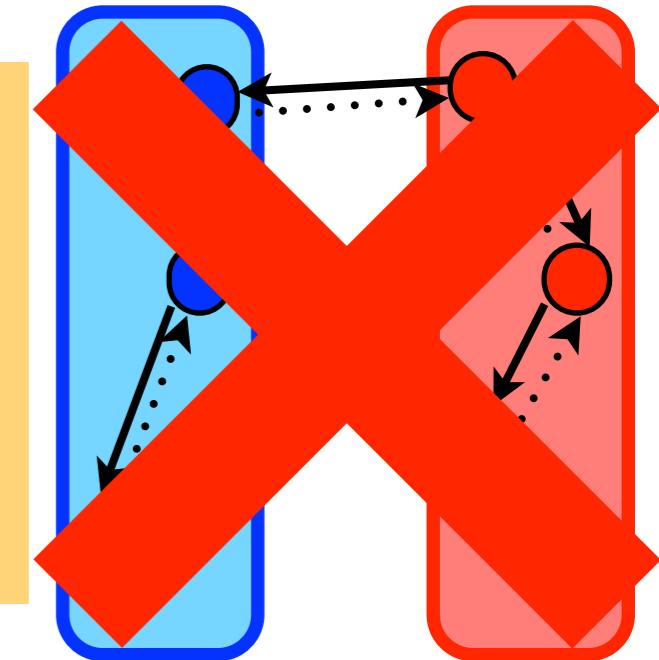
```
/*: Widget × Widget → Widget */
WrappedElt.prototype.style = function(name, val) {
    var regexp = new RegExp("url");
    if (regexp.test(val)) {
        return error();
    }
    ... this.__node__.style[name] = val ...
}
```

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

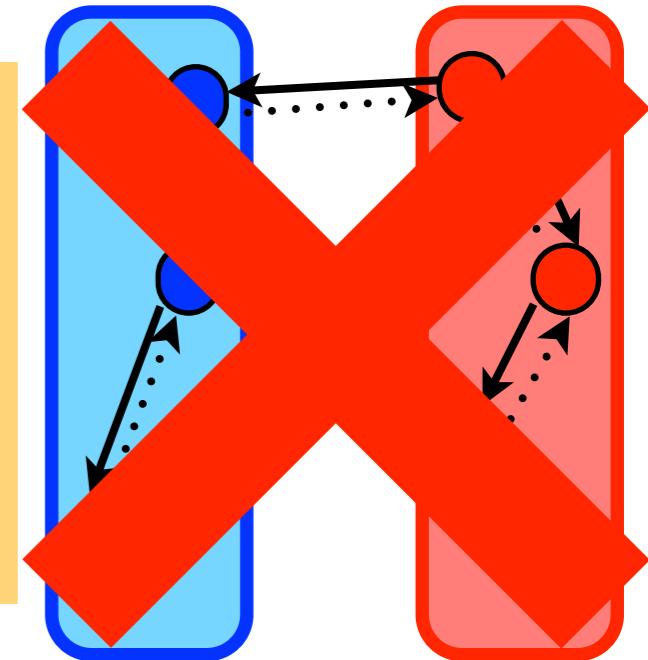
```

/*: 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')";
        }
    }
};

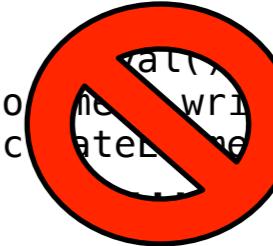

```

passes safety check

returns bad value

Definition I (ADsafe): If all embedded widgets pass JSLint, then:

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

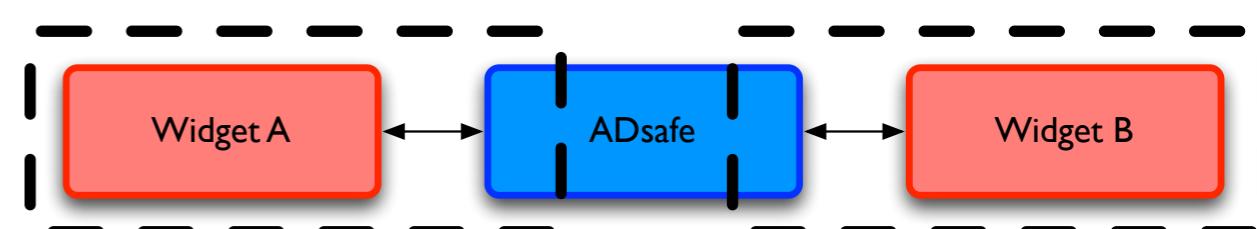
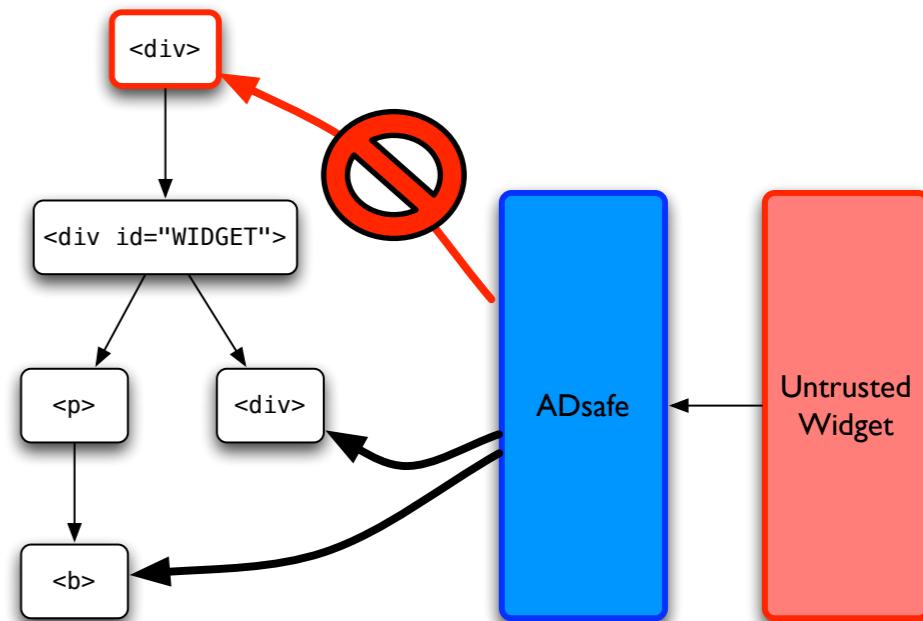
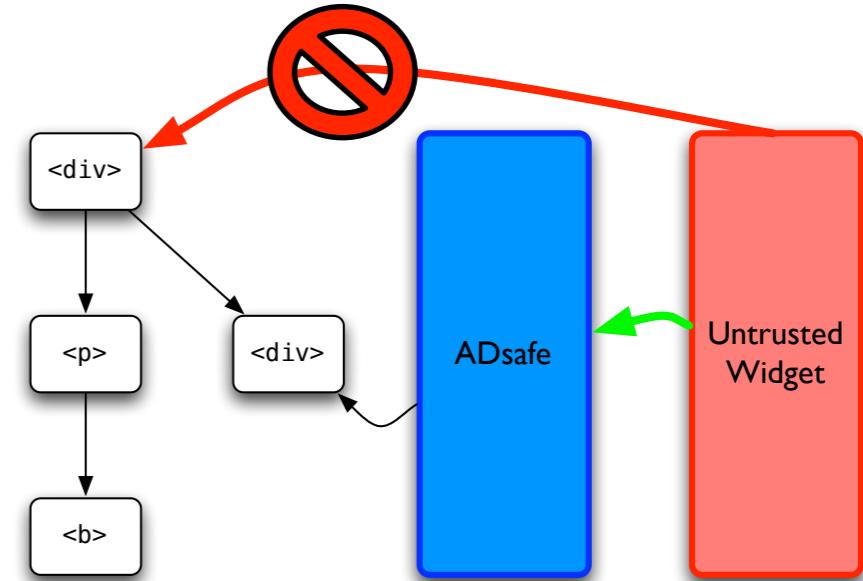


1. Widgets cannot load new code at runtime, or cause ADsafe to load new code on their behalf;

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.



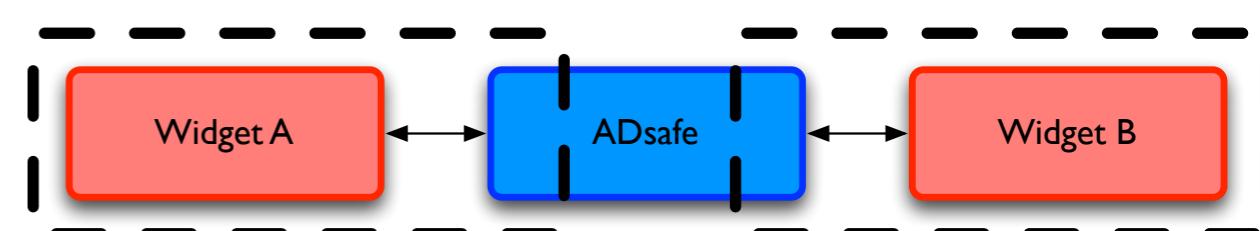
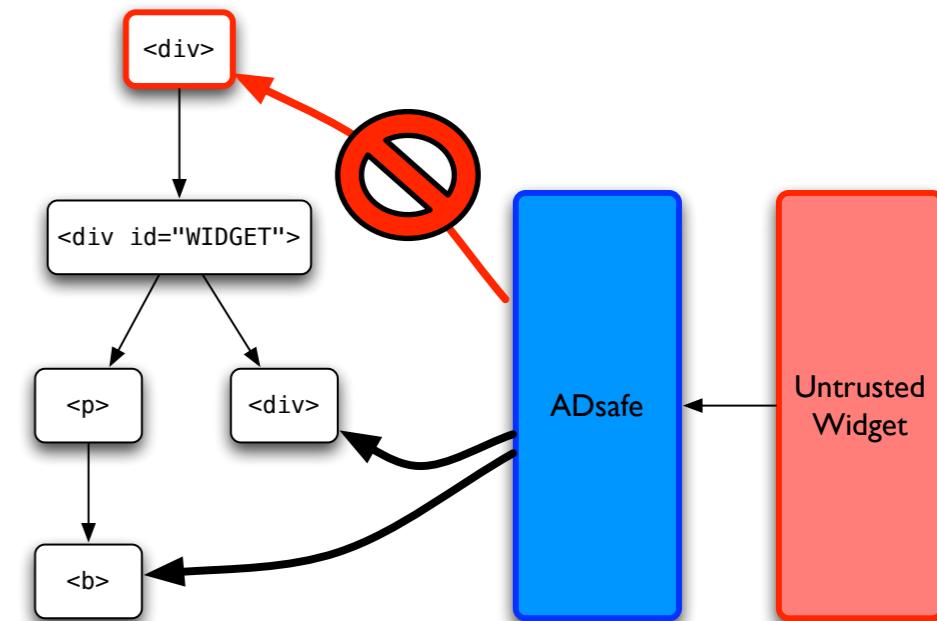
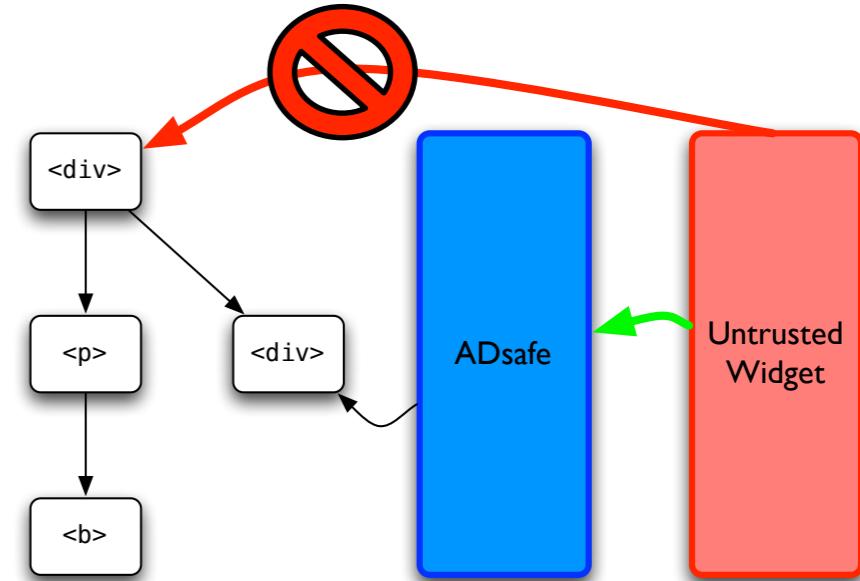
Definition I (ADsafe): If all embedded widgets pass JSLint, then:

✓ 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;

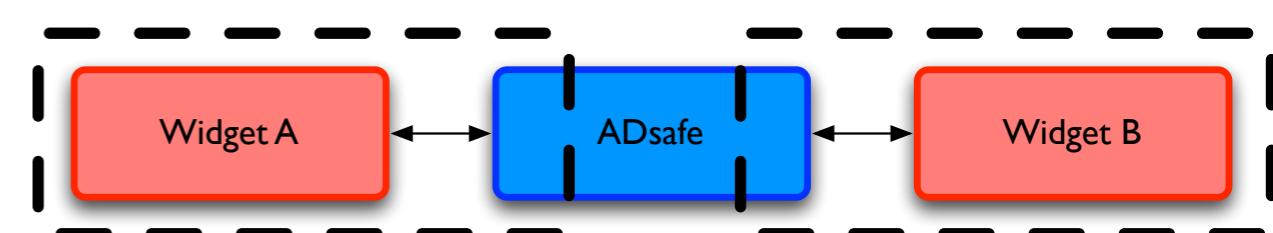
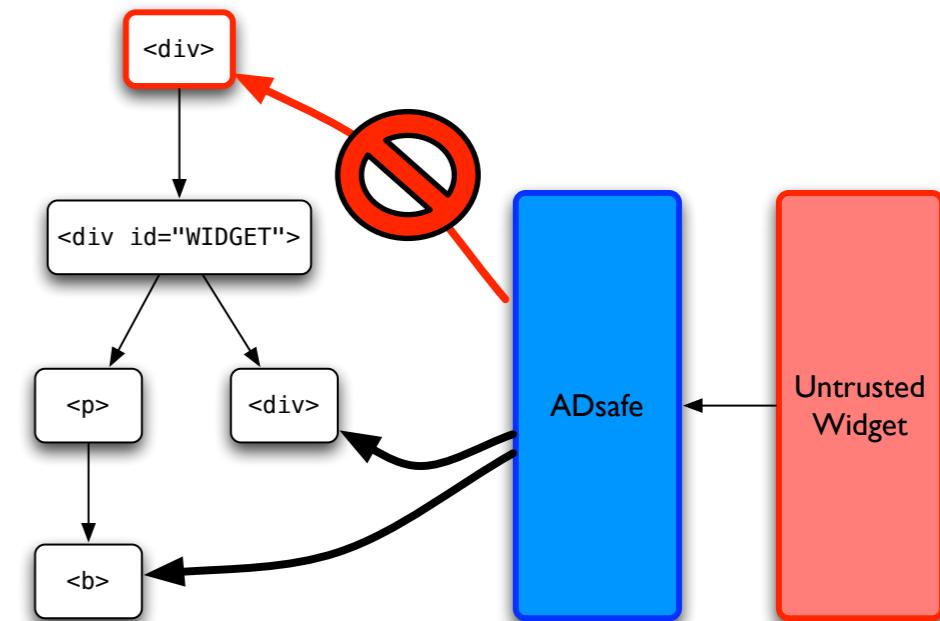
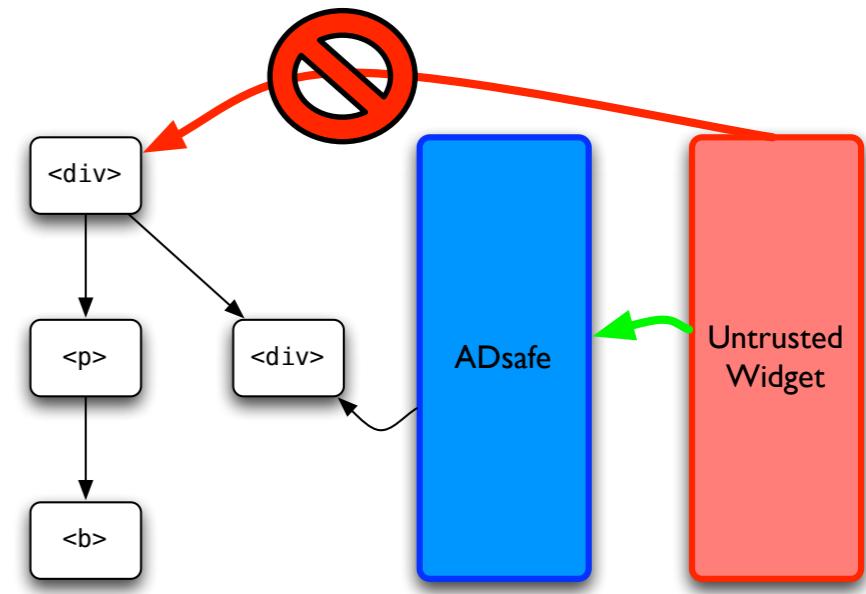
3. Widgets cannot affect the DOM outside of their subtree; and

4. Multiple widgets on the same page cannot communicate.



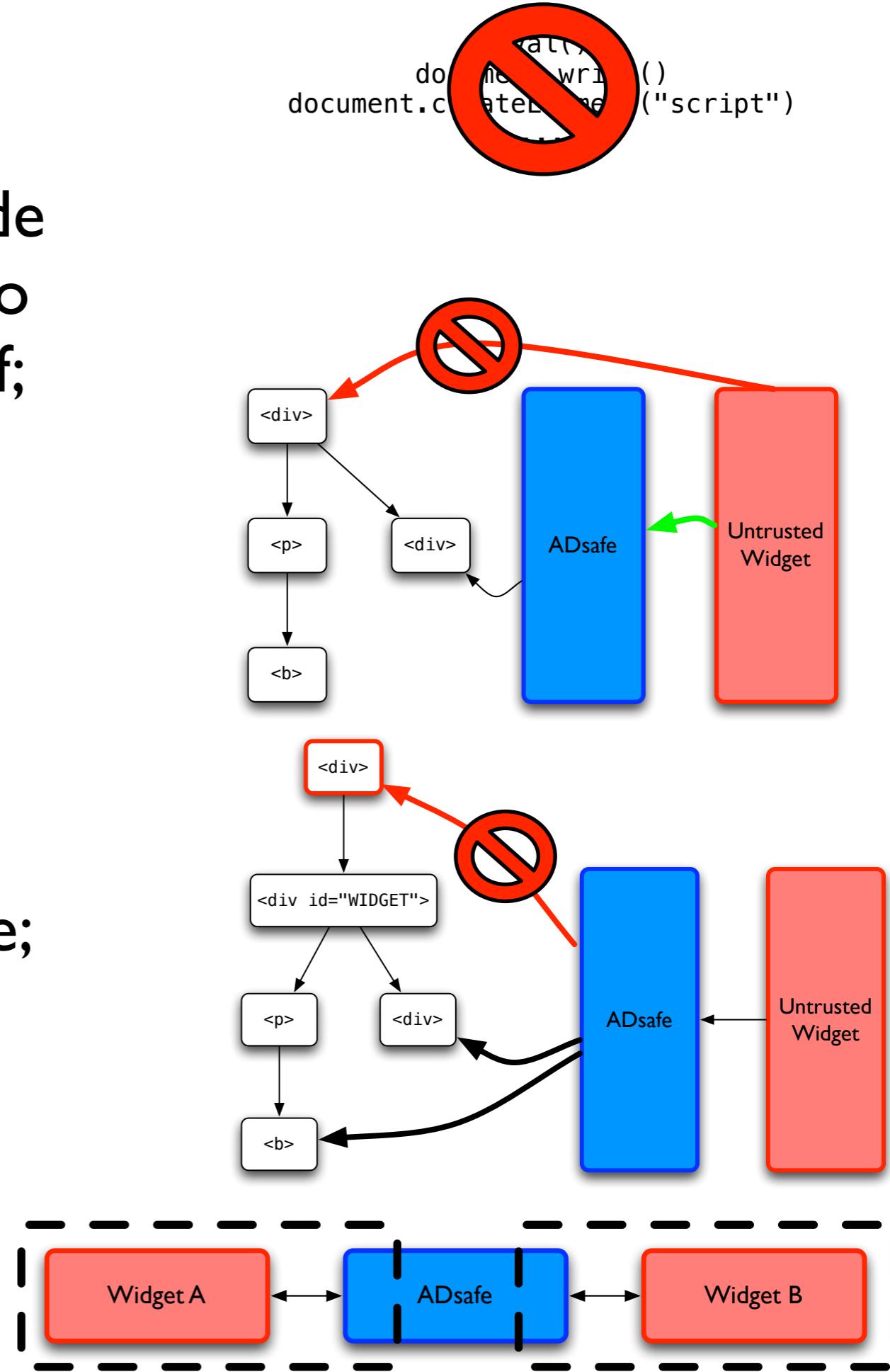
Definition I (ADsafe): If all embedded widgets pass JSLint, then:

- ✓ 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.



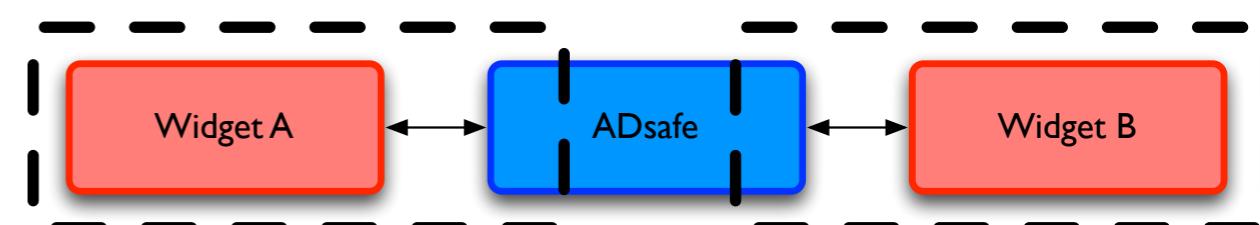
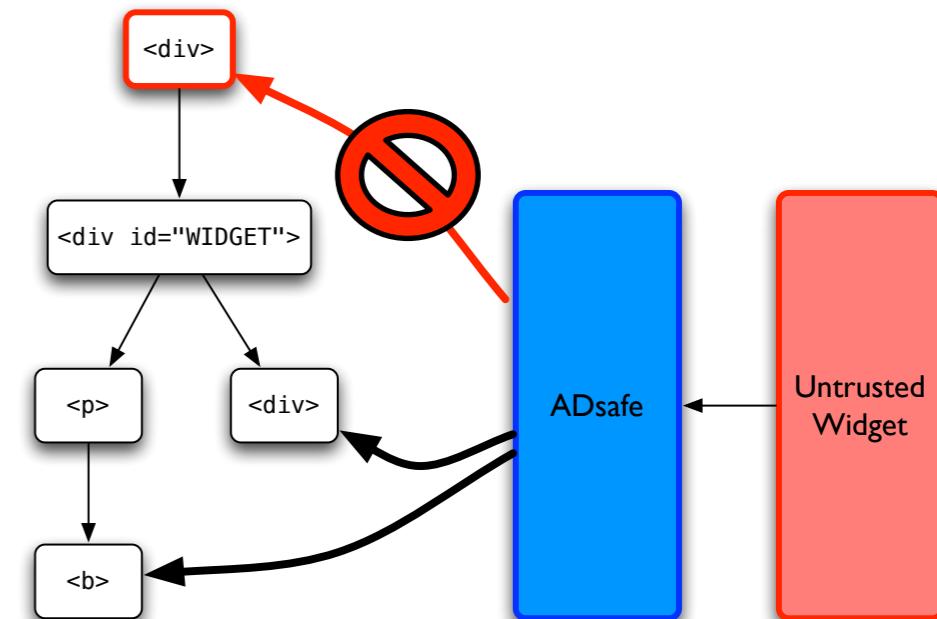
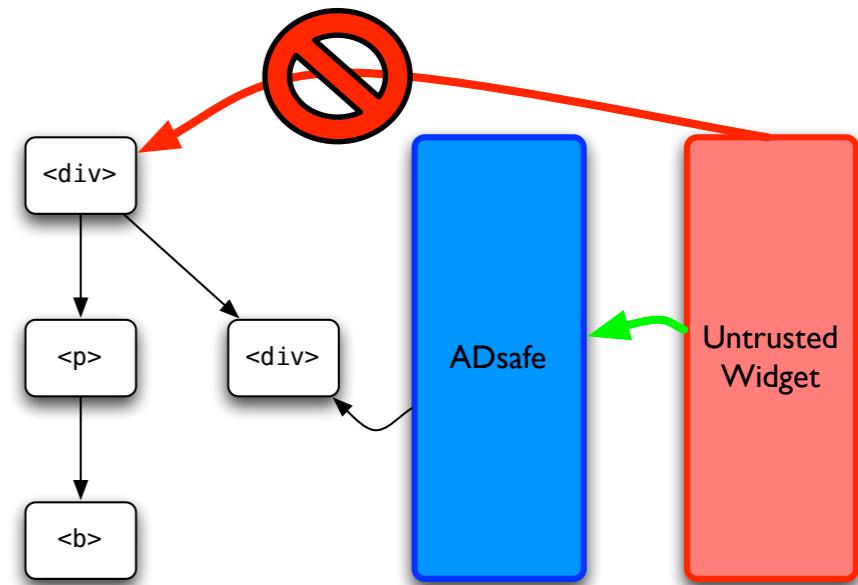
Definition I (ADsafe): If all embedded widgets pass JSLint, then:

- ✓ 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 I (ADsafe): 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;
- cloud 3. Widgets cannot affect the DOM outside of their subtree; and
- Retracted** 4. multiple widgets on the same page cannot communicate.

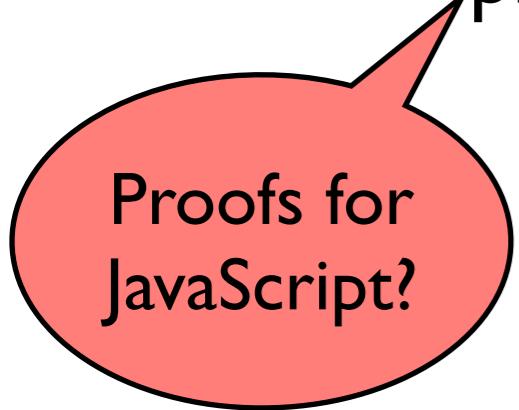




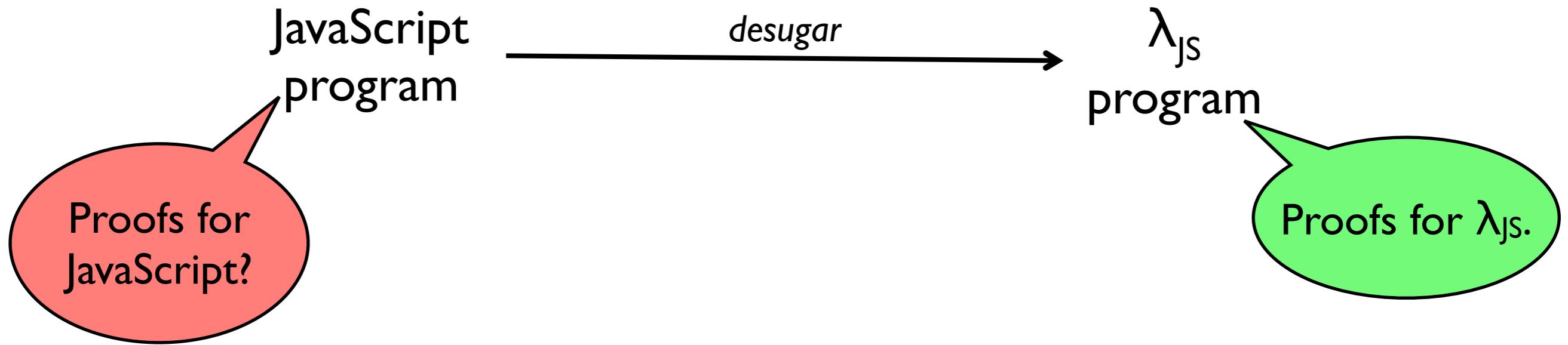
Caveats:

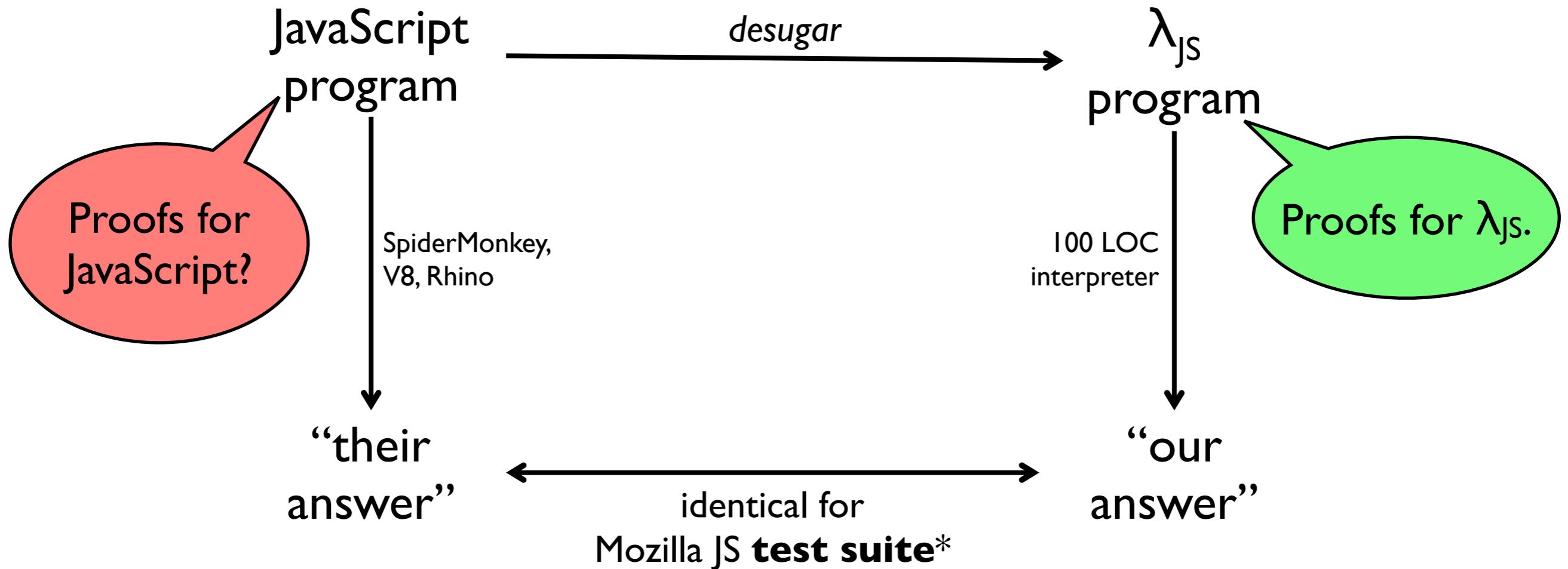
- II LOC unverified
- subtree property unverified

JavaScript
program



Proofs for
JavaScript?





— 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 +

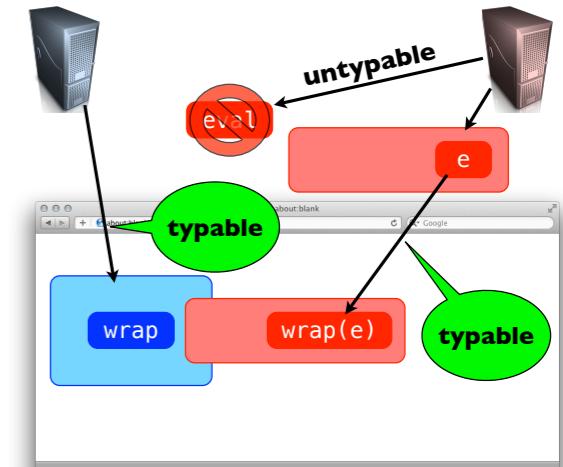
{ ★: Widget
 __nodes__: Array<Node>
 caller: 💀
 prototype: 💀
 ...
 code : Widget × ... → Widget
 __proto__: Object + Function + Array + ... }

Conclusion

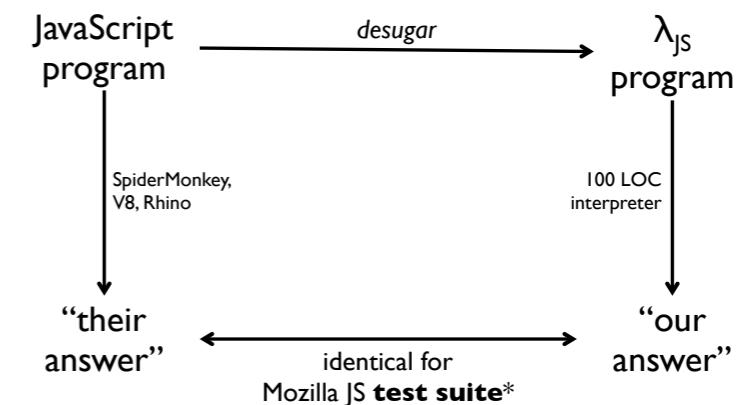
I. Model sandbox as a type system

2. Object types for JavaScript (★ and ☠)

3. Proofs over tractable JavaScript semantics



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





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

Widget = $\mu\alpha.$

Str \cup Num \cup Null \cup Bool \cup Undef \cup
Object \cup Function
proto : UBunch \cup Array \cup RegExp
 \cup String \cup Number \cup Boolean,
 $\star : \alpha,$
code : [Global \cup α] $\alpha \cdots \rightarrow \alpha,$
"__nodes__" : Array<HTML> \cup Undef,
"__star__" : Bool \cup Undef,
"caller" : , "callee" : ,
"eval" : , "prototype" : ,
"watch" : , "constructor" : ,
"__proto__" : , "unwatch" : ,
"arguments" : , "valueOf" : Absent,
"toString" : Absent

Ref {