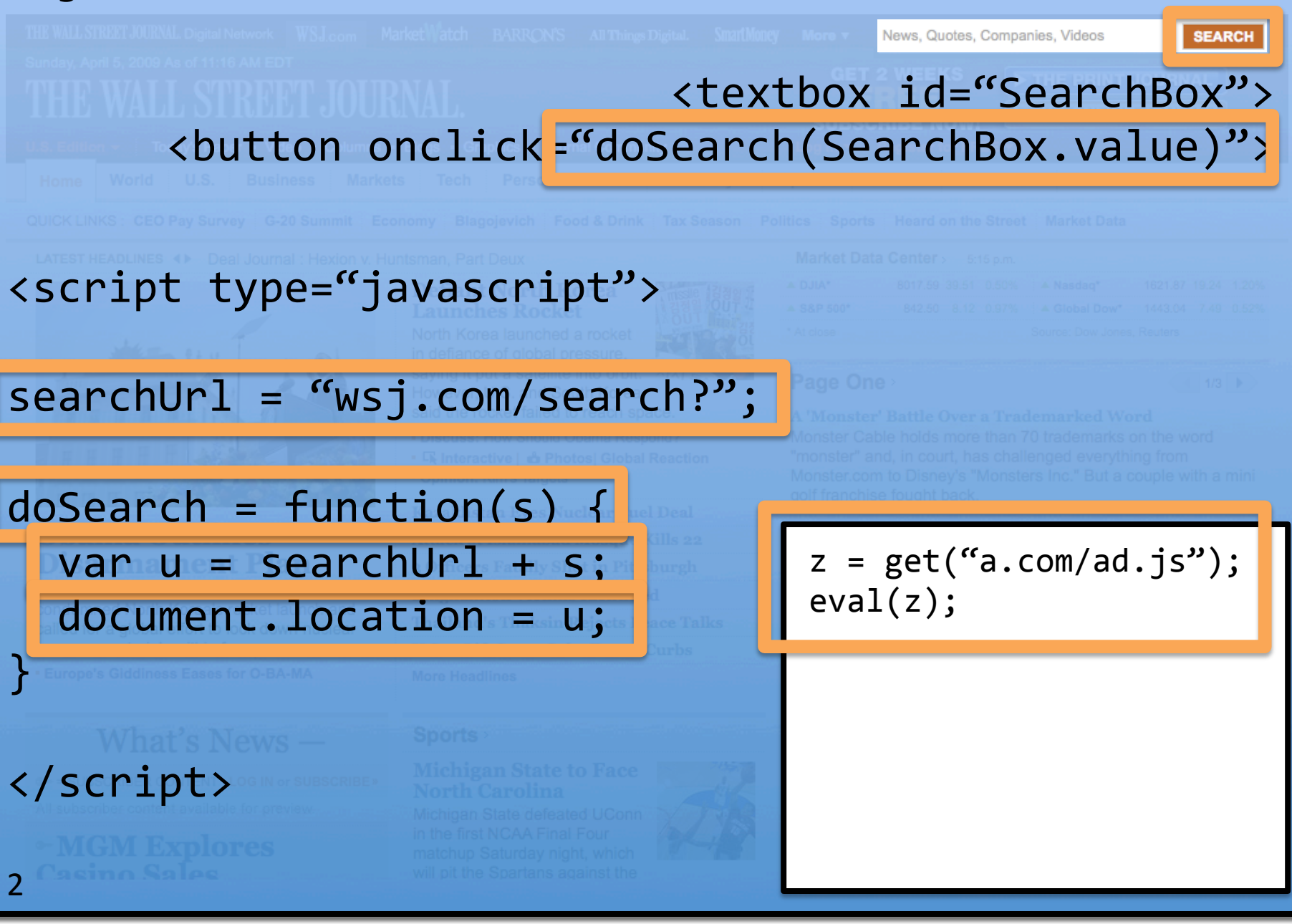
Staged Information Flow for JavaScript

Ravi Chugh, Jeff Meister, Ranjit Jhala, Sorin Lerner

UC San Diego

wsj.com




```
<textbox id="SearchBox">
<button onclick="doSearch(SearchBox.value)">

<script type="javascript">

searchUrl = "wsj.com/search?";

doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}

</script>
```

```
z = get("a.com/ad.js");
eval(z);
```

wsj.com

```
<textbox id="SearchBox">
<button onclick="doSearch(SearchBox.value)">

<script type="javascript">

searchUrl = "wsj.com/search?";

doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}

</script>
```

a.com/ad.js

```
displayAd = function() {
  ...
}
displayAd();
```

wsj.com

```
<textbox id="SearchBox">
<button onclick="doSearch(SearchBox.value)">

<script type="javascript">

searchUrl = "wsj.com/search?";

doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}

</script>
```

a.com/ad.js

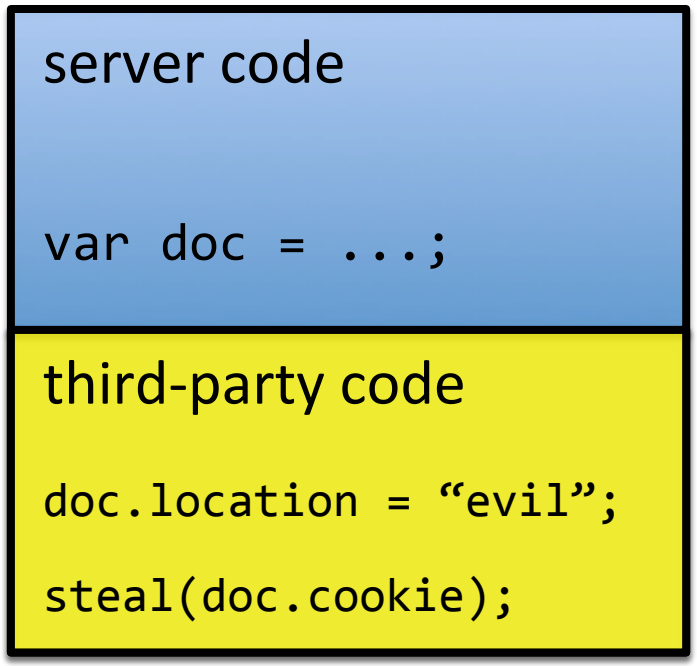
```
displayAd = function() {
  ...
}
displayAd();
searchUrl = "evil.com/";
```

evil.com

- Script navigates to malicious page
- Exploits browser vulnerability

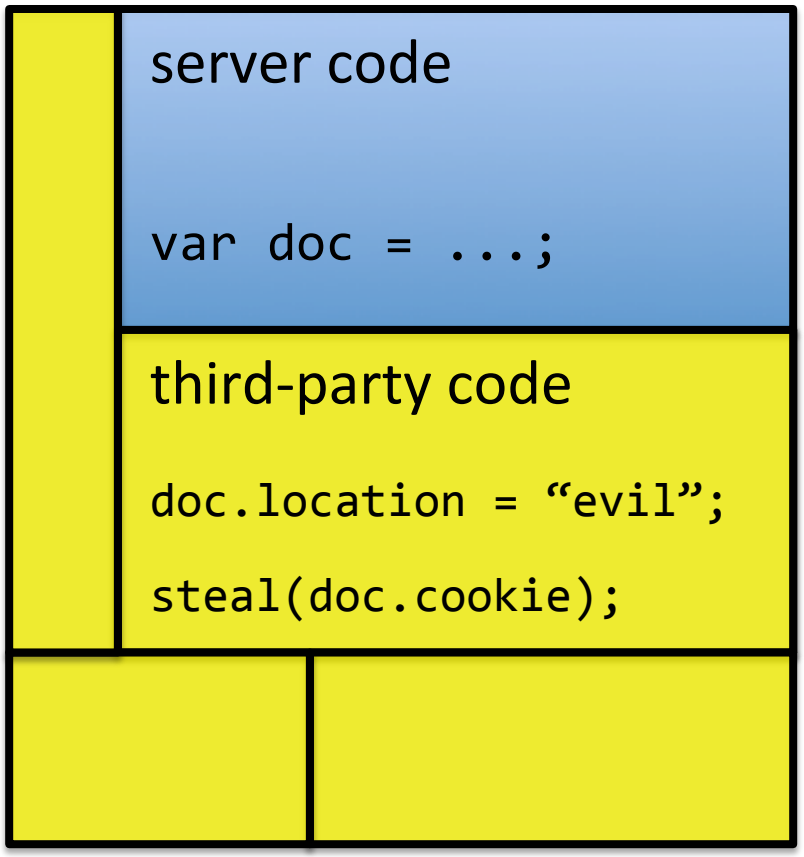
The Problem, Part 1

- Third-party code may affect sensitive data
  - e.g. writing `doc.location`
  - e.g. reading `doc.cookie`
- Information flow policies
  - e.g. integrity of `doc.location`
  - e.g. confidentiality of `doc.cookie`
- JavaScript difficulties
  - dynamic typing
  - first-class functions
  - objects, but no classes
  - prototypes

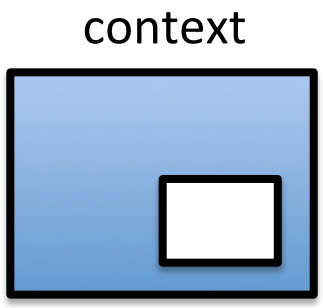


The Problem, Part 2

- Entire code not available until runtime
- Arrives in stages



Our Staged Approach: Server

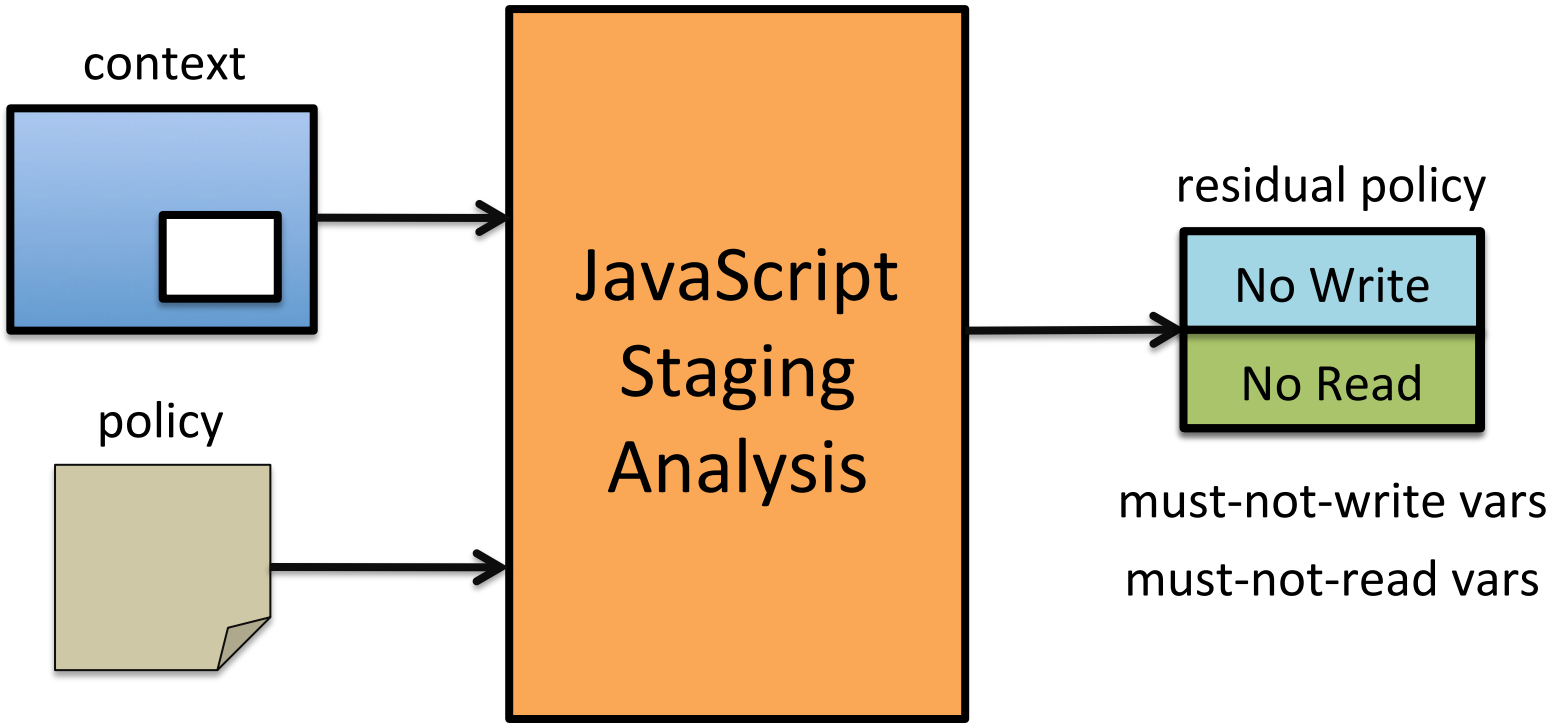


Information Flow Policies

Confidentiality policy:
x should not be read

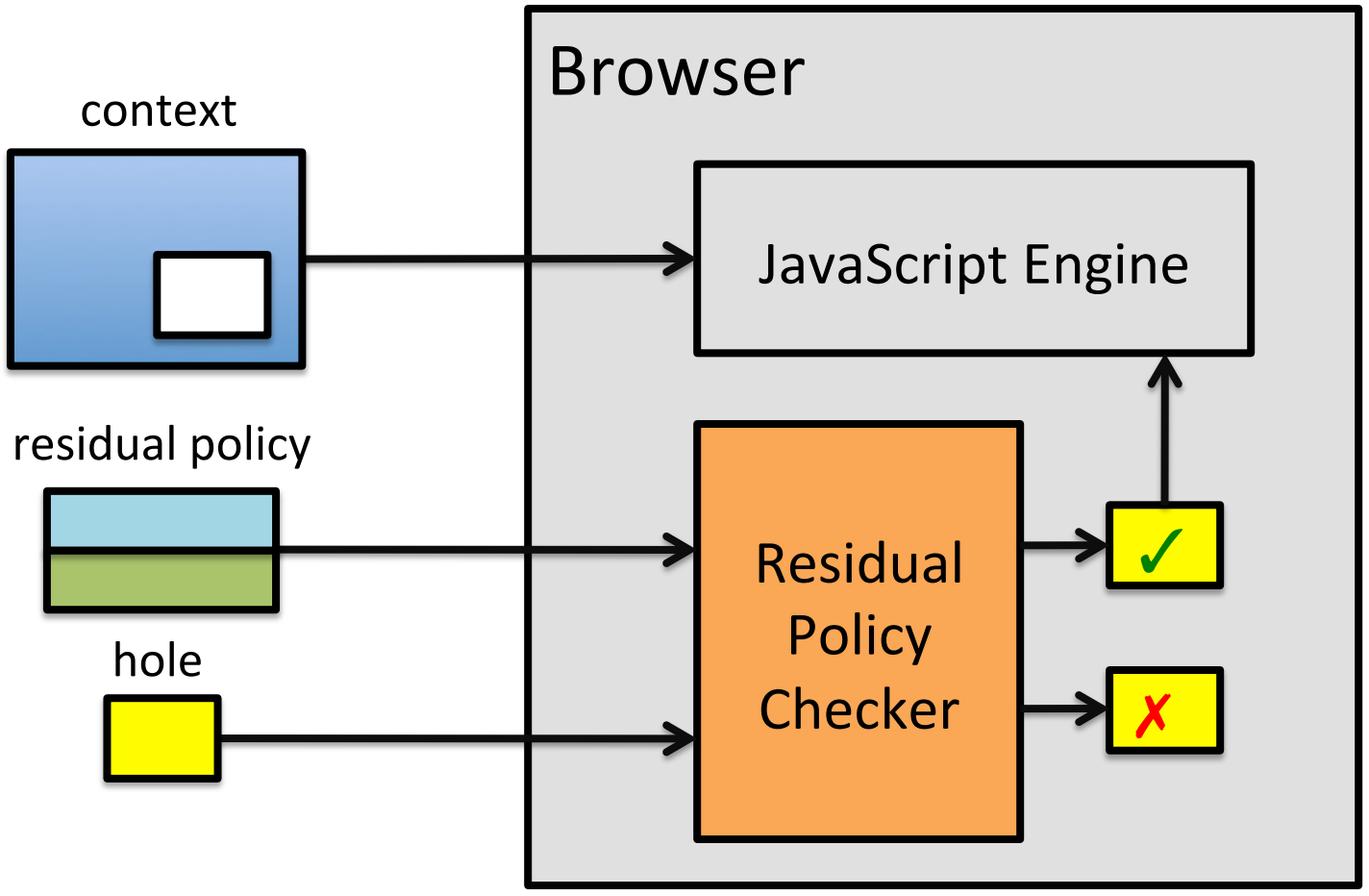
Integrity policy:
x should not be written

Our Staged Approach: Server



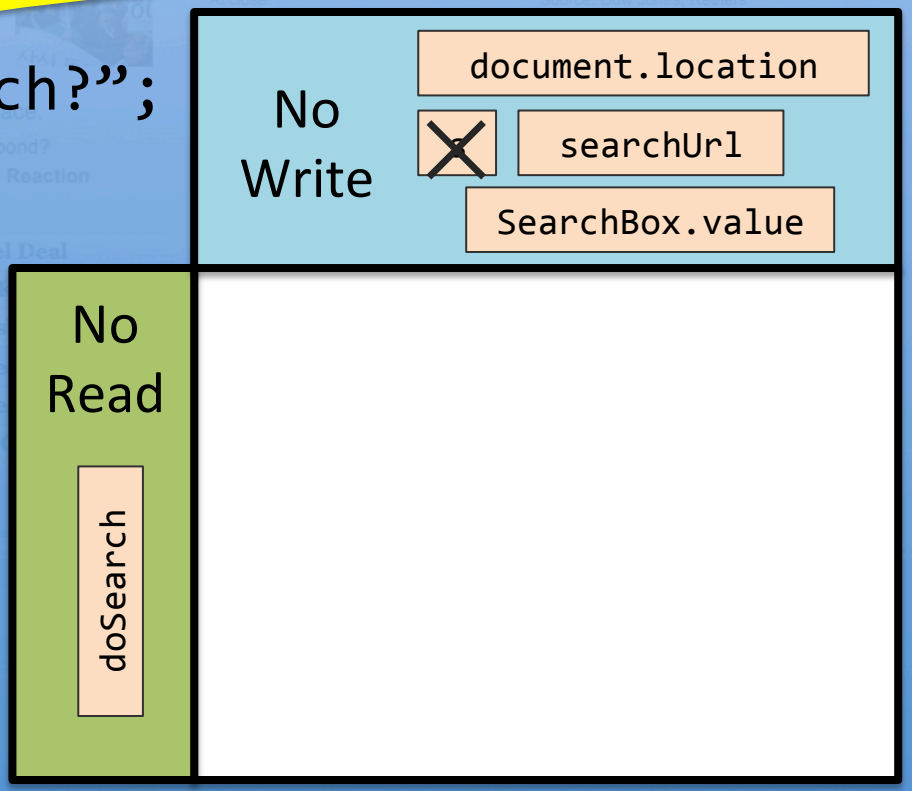
- Summarizes how loaded code must behave
- Syntactically enforceable for speed

Our Staged Approach: Client



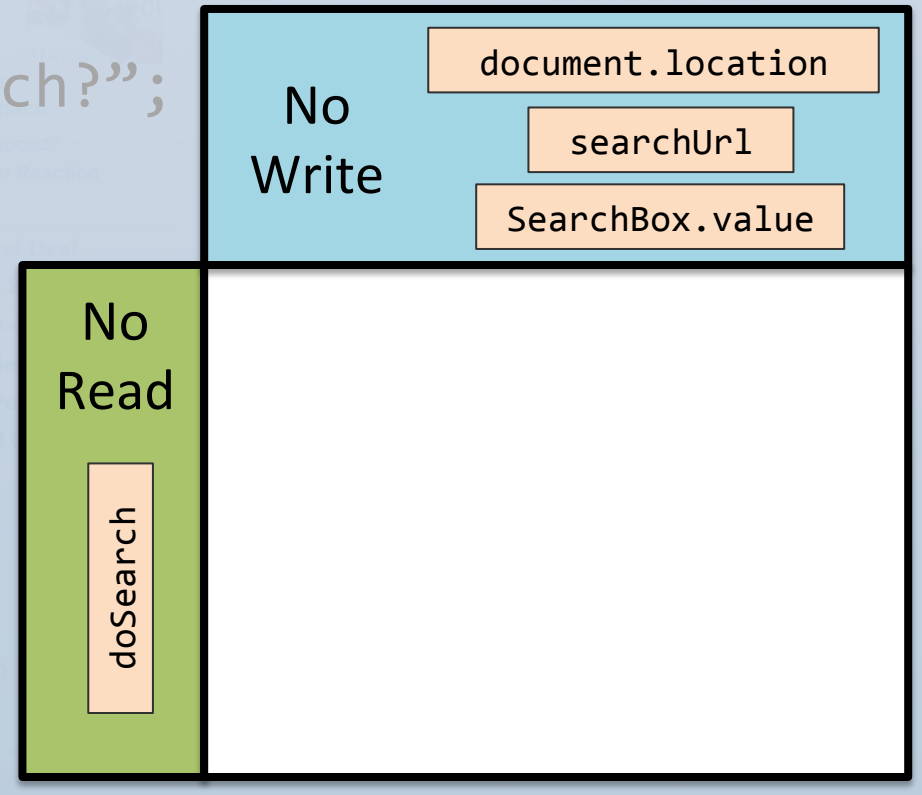
wsj.com

```
<textbox id="SearchBox">
<button onclick="doSearch(SearchBox.value)">

<script type="javascript">

searchUrl = "wsj.com/search?";

doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}

</script>
```



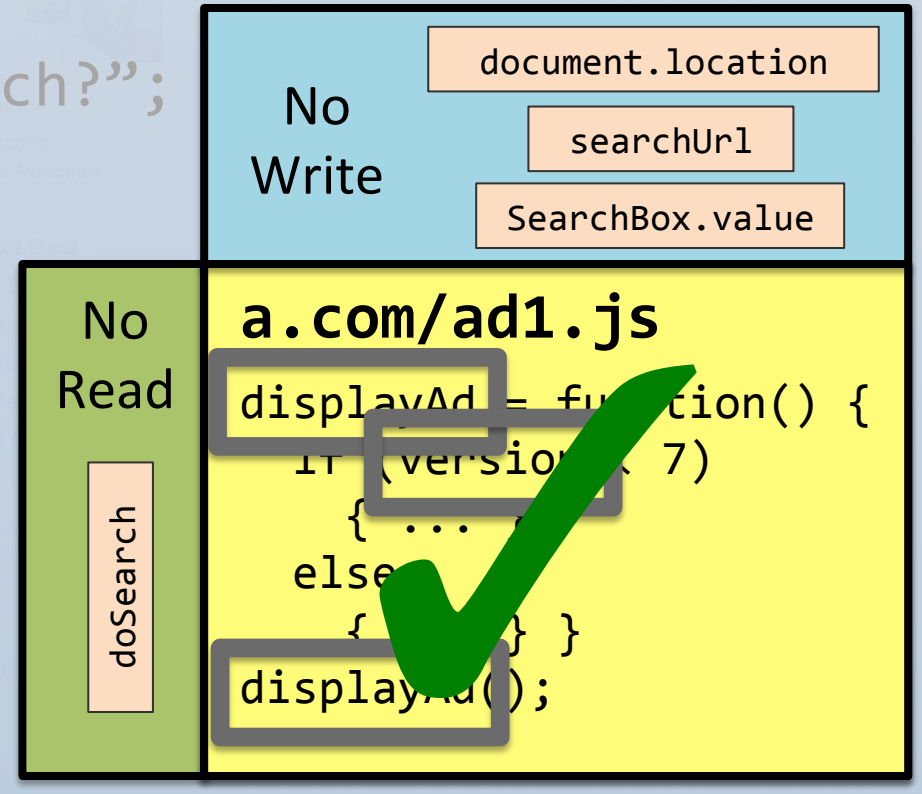
wsj.com

```
<textbox id="SearchBox">
<button onclick="doSearch(SearchBox.value)">

<script type="javascript">

searchUrl = "wsj.com/search?";

doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}

</script>
```



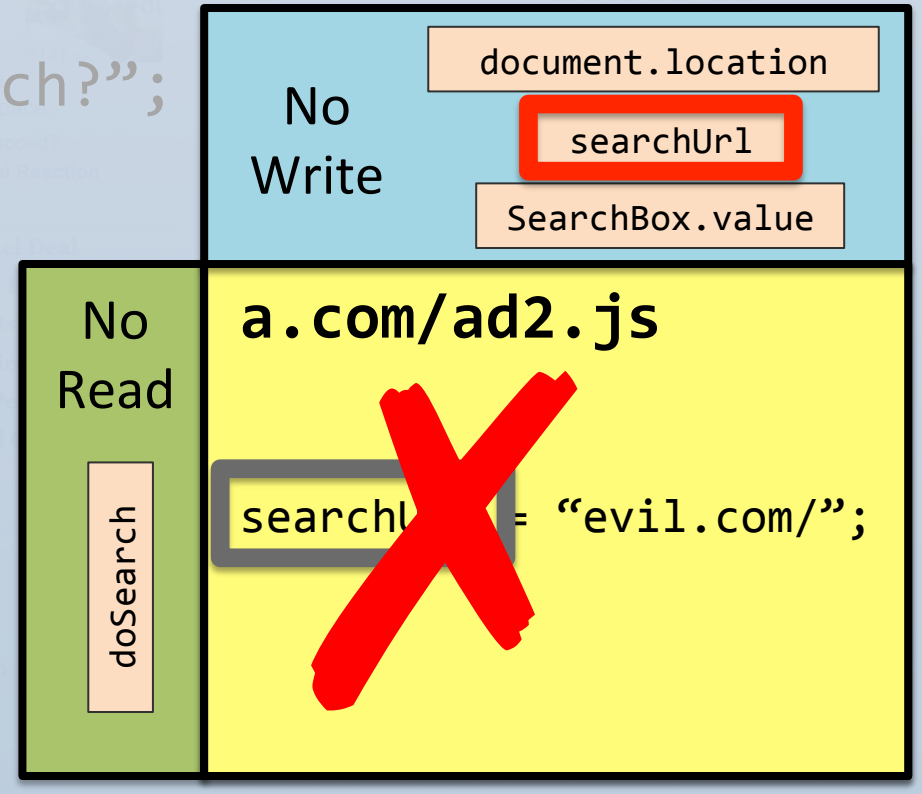
wsj.com

```
<textbox id="SearchBox">
<button onclick="doSearch(SearchBox.value)">

<script type="javascript">

searchUrl = "wsj.com/search?";

doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}

</script>
```



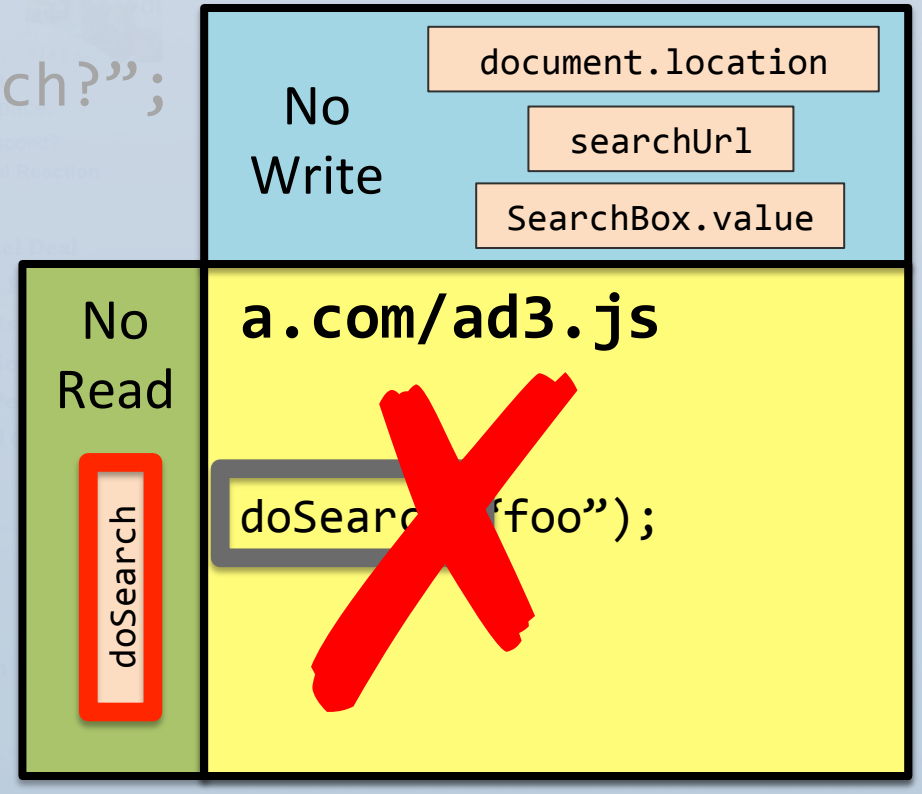
wsj.com

```
<textbox id="SearchBox">
<button onclick="doSearch(SearchBox.value)">

<script type="javascript">

searchUrl = "wsj.com/search?";

doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}

</script>
```



wsj.com

```
<textbox id="SearchBox">
<button onclick="doSearch(SearchBox.value)">

<script type="javascript">

searchUrl = "wsj.com/search?";

doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}

</script>
```



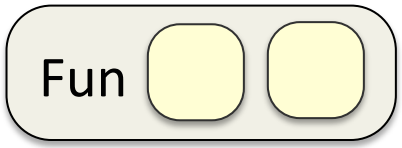
Outline

- Overview
- JavaScript Static Analysis
- Computing Residual Policies
- Additional Challenges
- Evaluation

Information Flow Graph

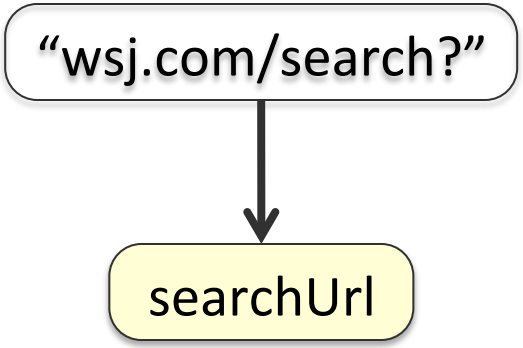
- Analysis tracks information flow in program
- Flow-insensitive, set constraint-based
- Graph representation:
  - program constants, variables, edges

  - special nodes for function declarations and calls

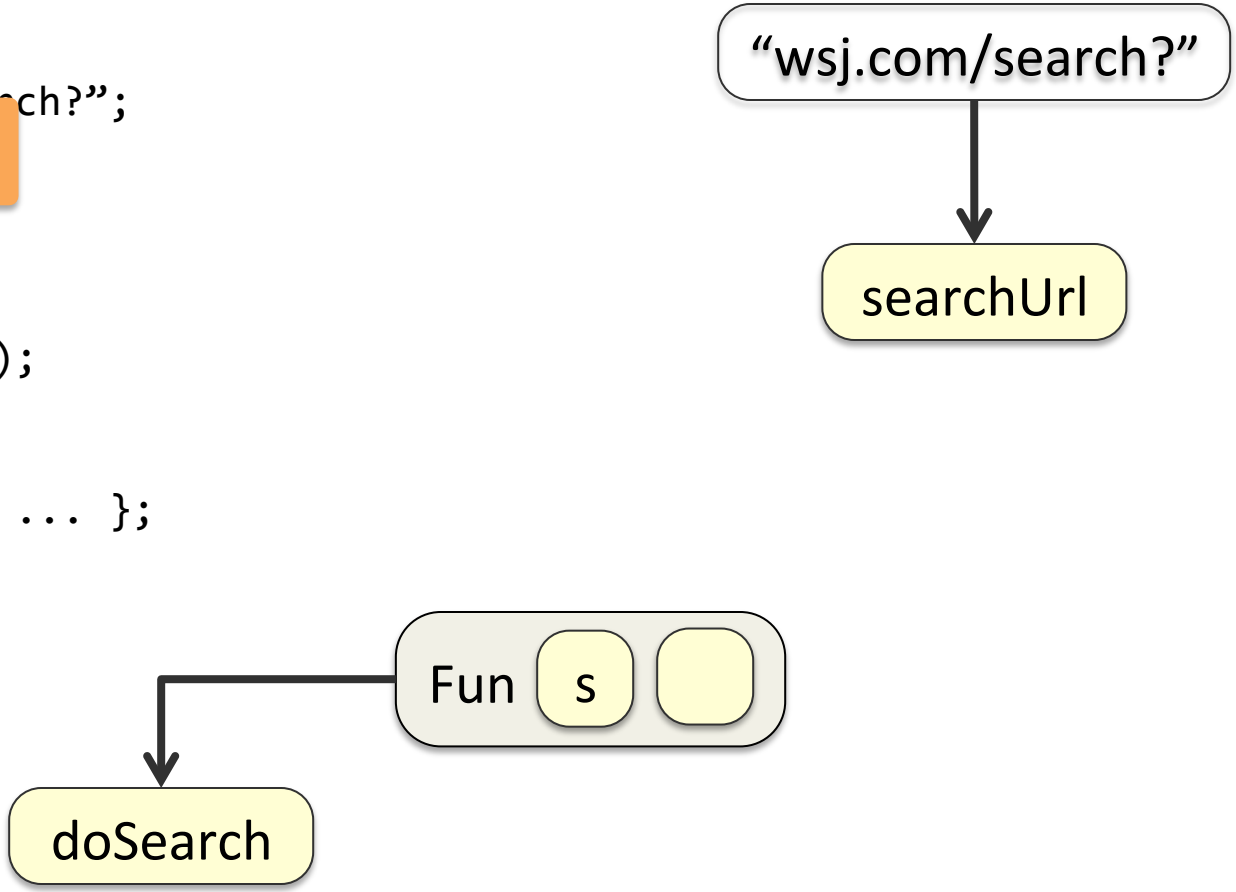


```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad1.js */
displayAd = function() { ... };
displayAd();
```

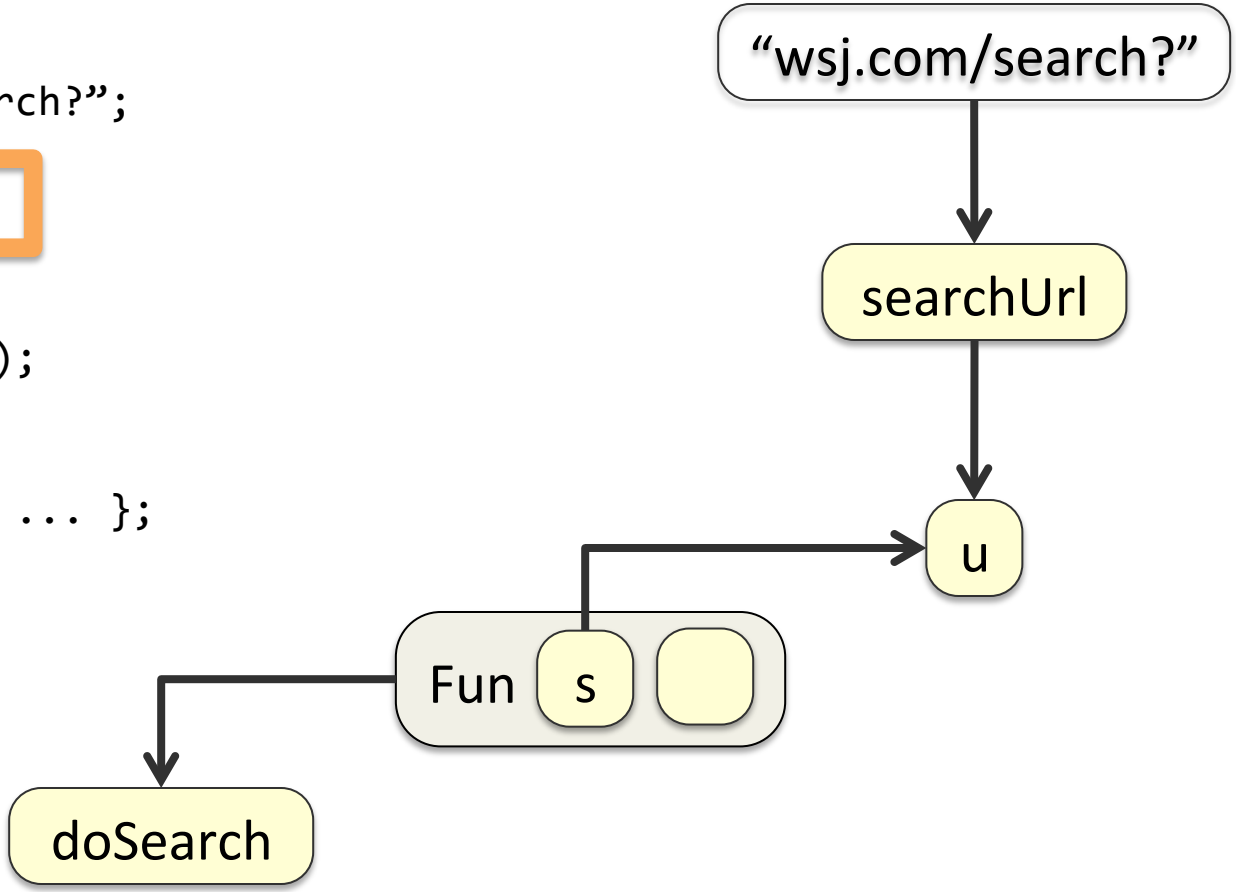
```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad1.js */
displayAd = function() { ... };
displayAd();
```



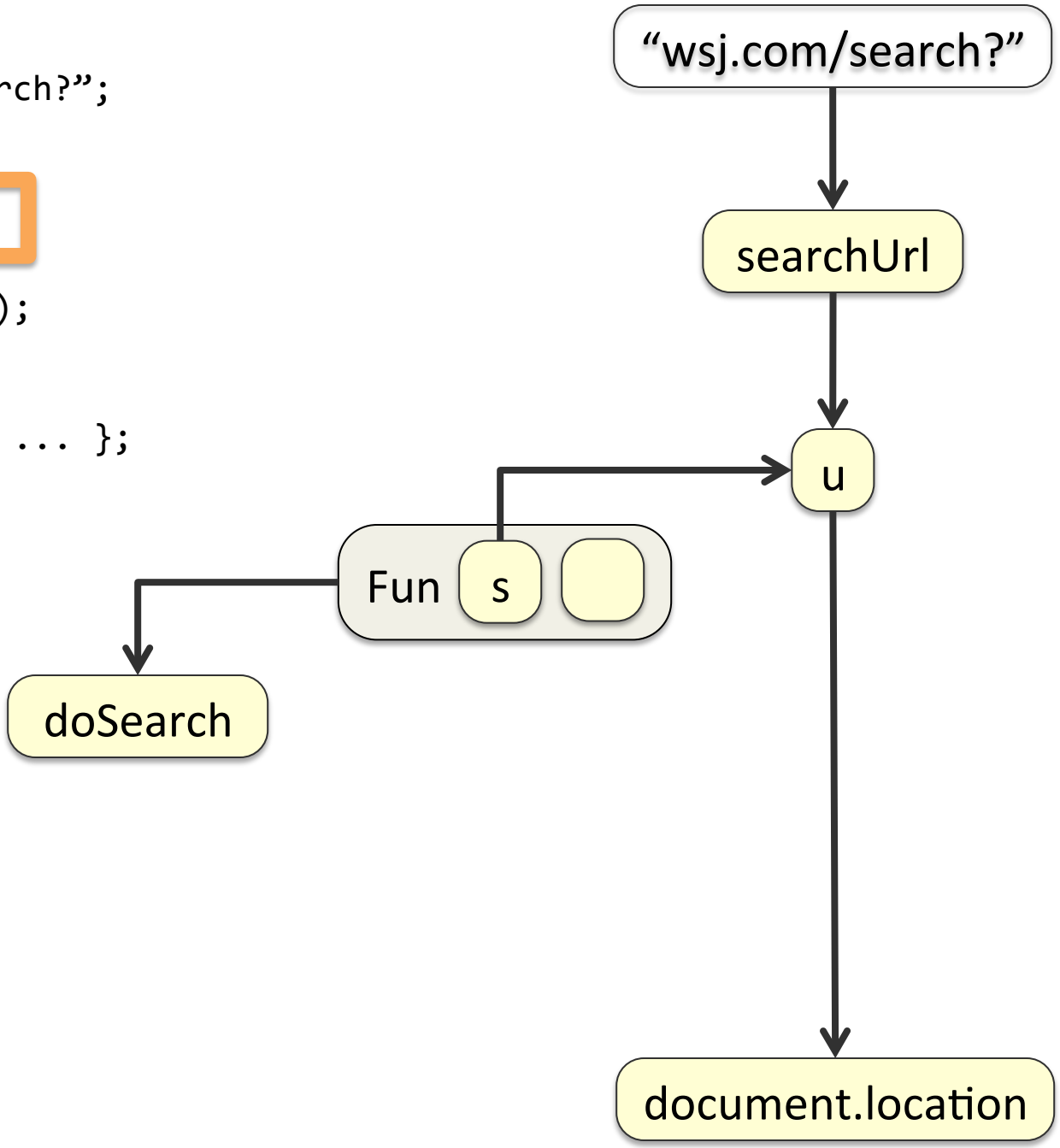
```
searchUrl = "wsj.com/search?";
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad1.js */
displayAd = function() { ... };
displayAd();
```



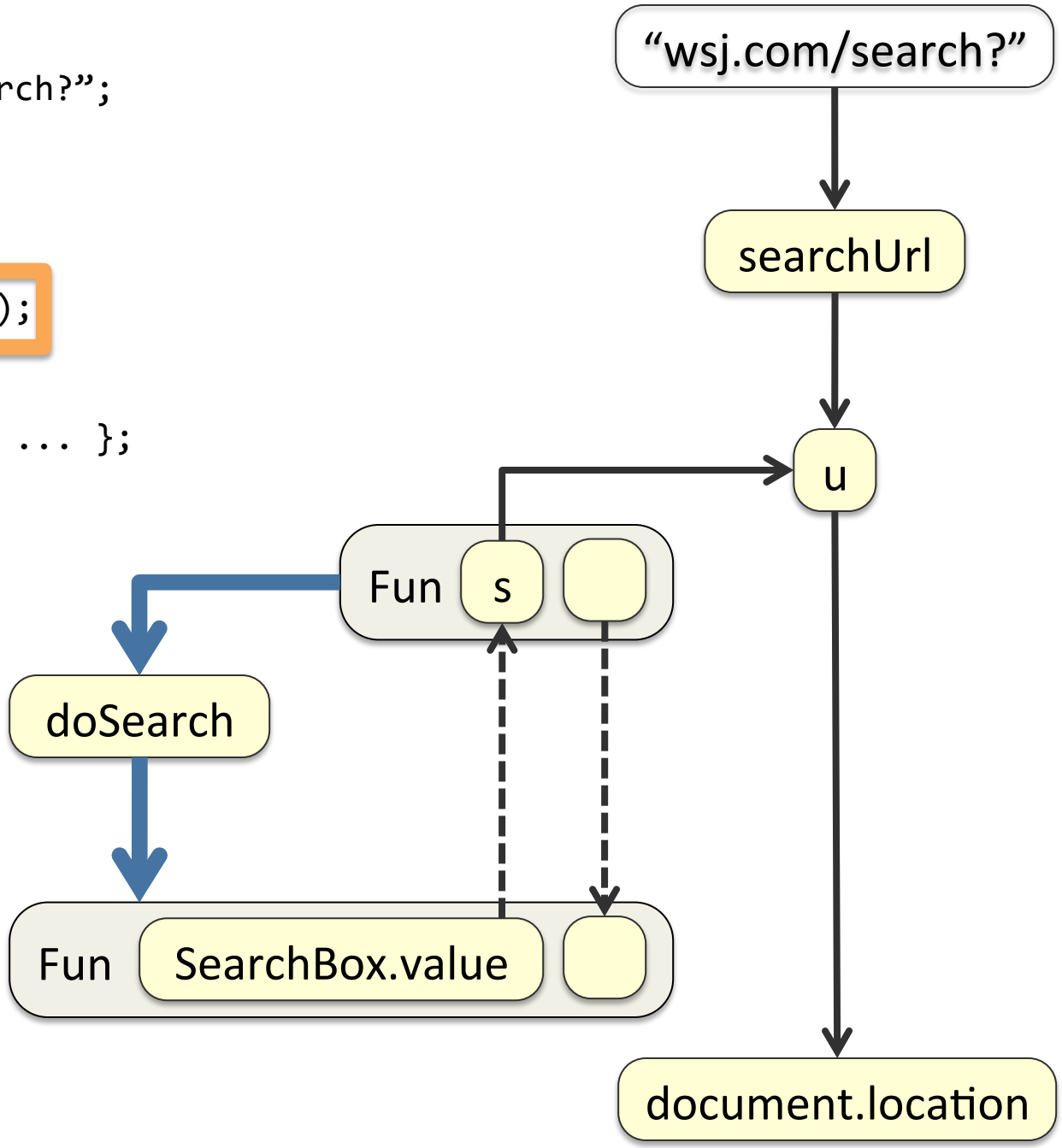
```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad1.js */
displayAd = function() { ... };
displayAd();
```



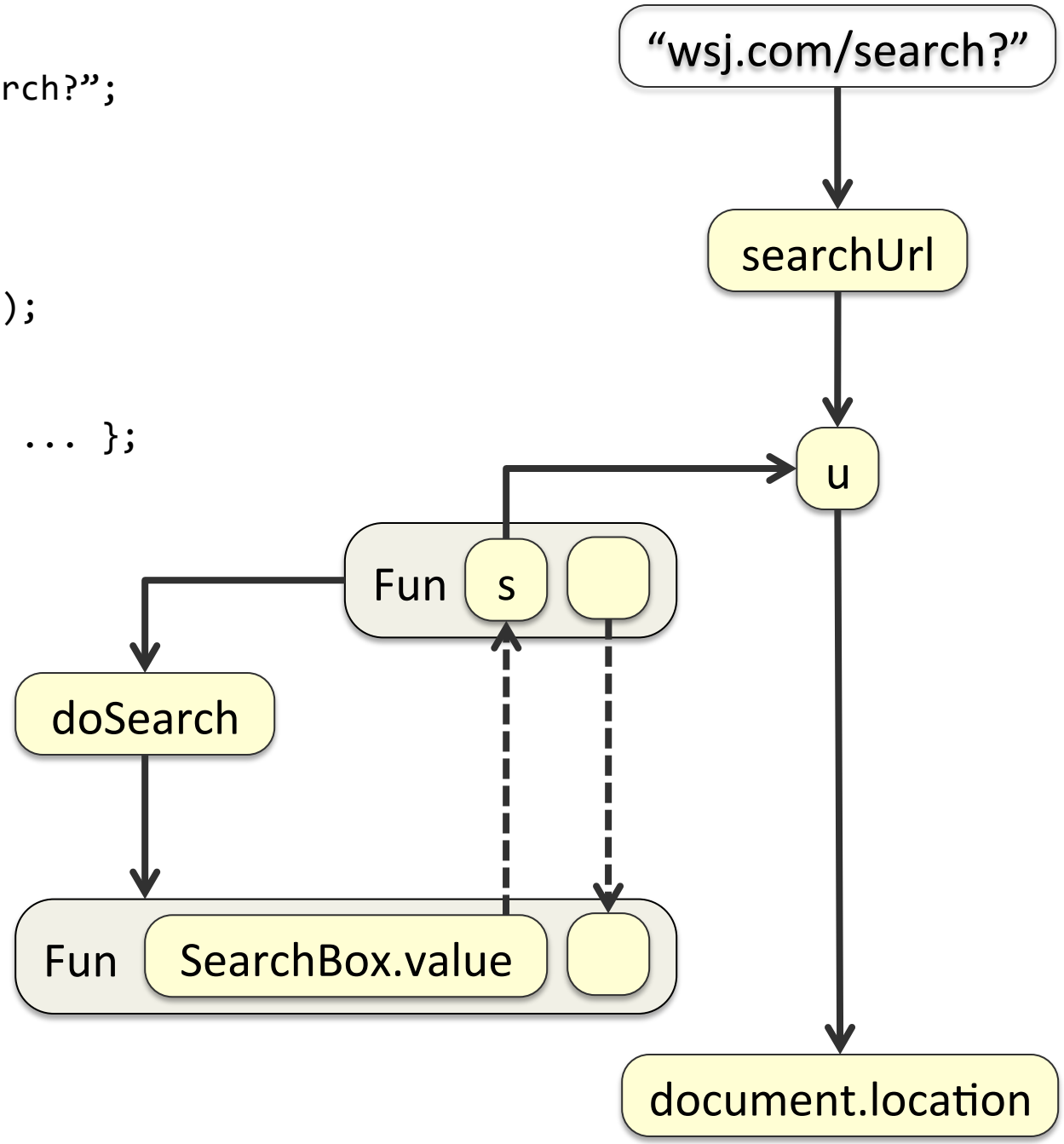
```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad1.js */
displayAd = function() { ... };
displayAd();
```



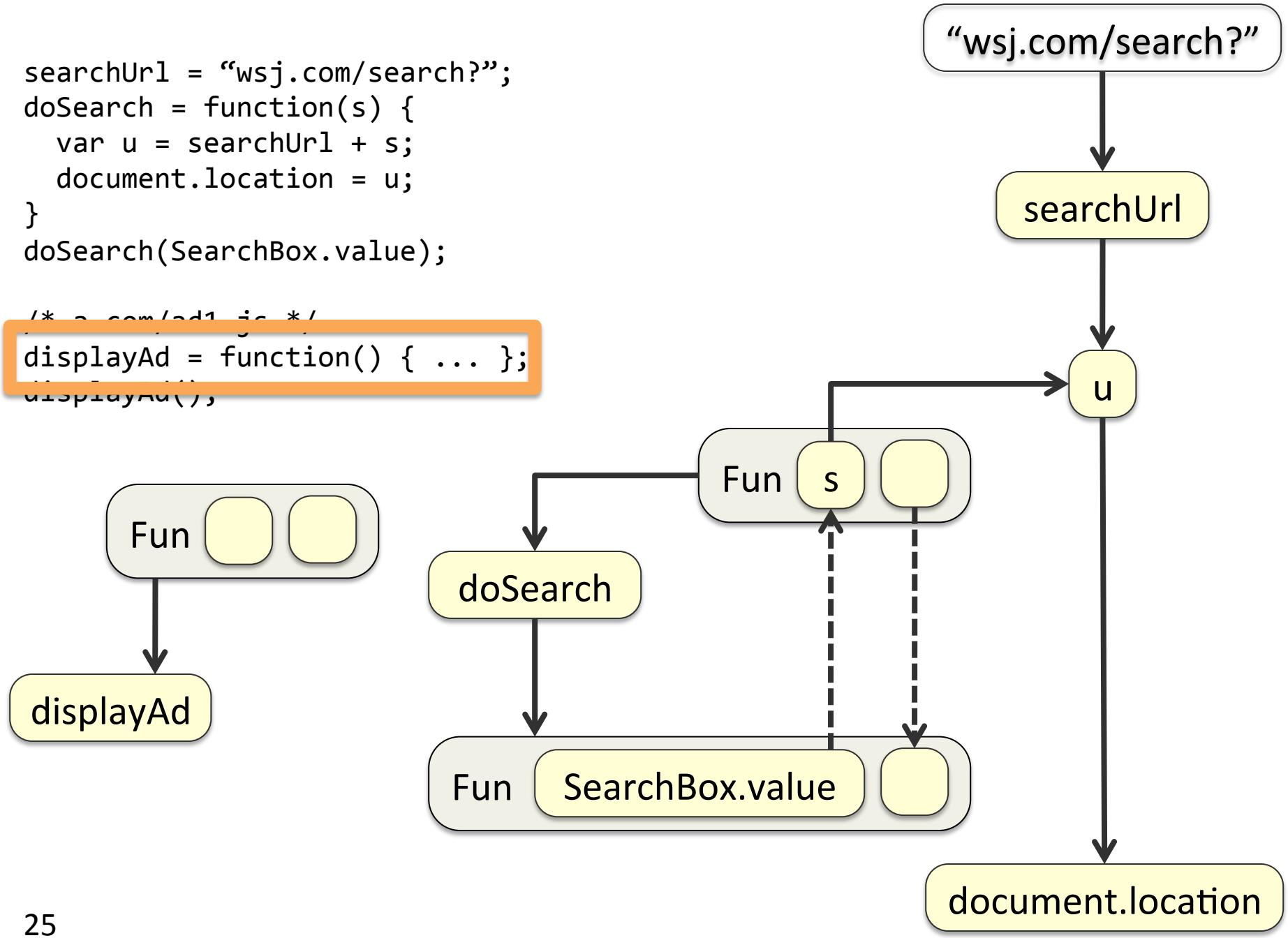
```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad1.js */
displayAd = function() { ... };
displayAd();
```



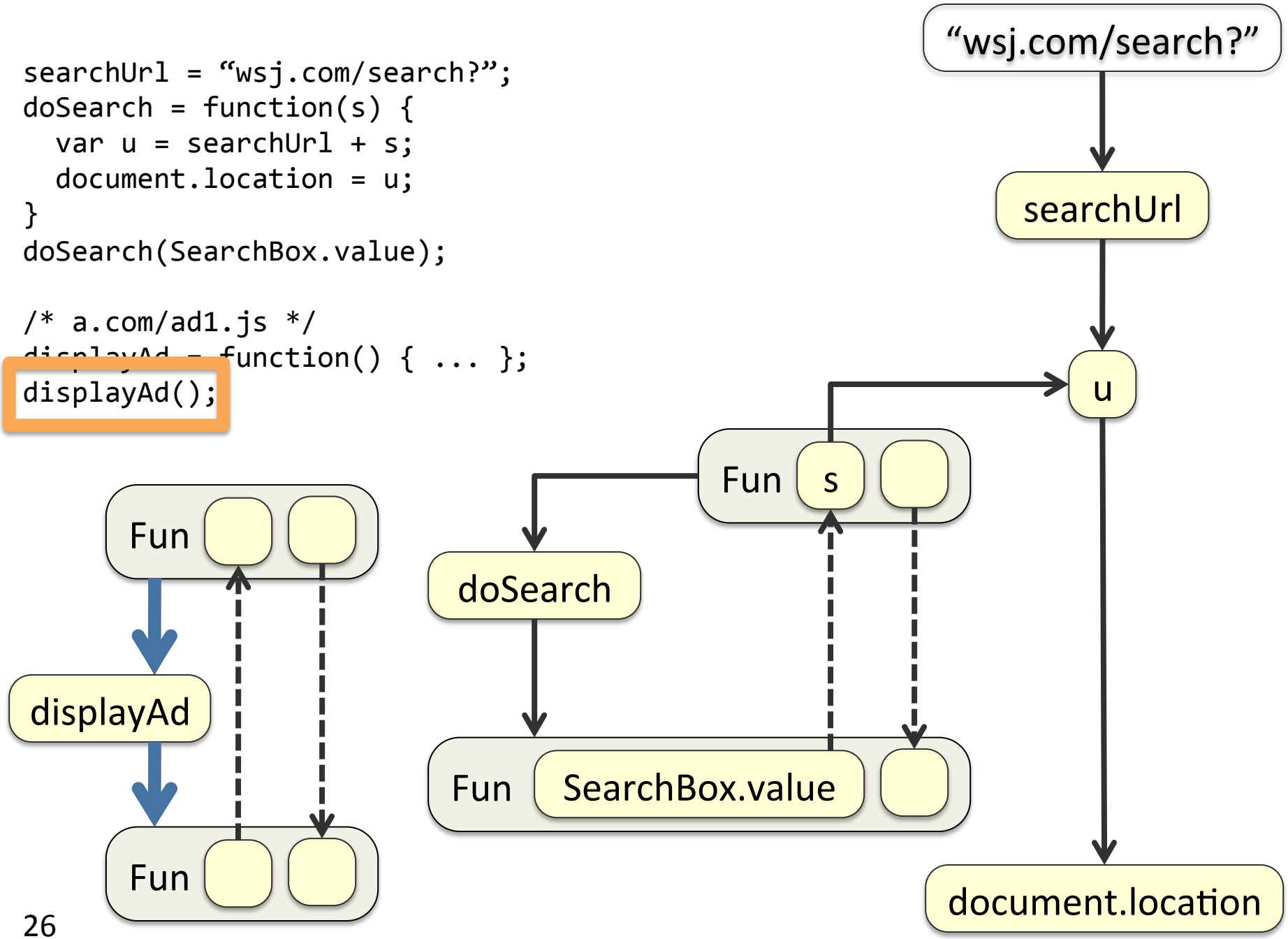
```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad1.js */
displayAd = function() { ... };
displayAd();
```



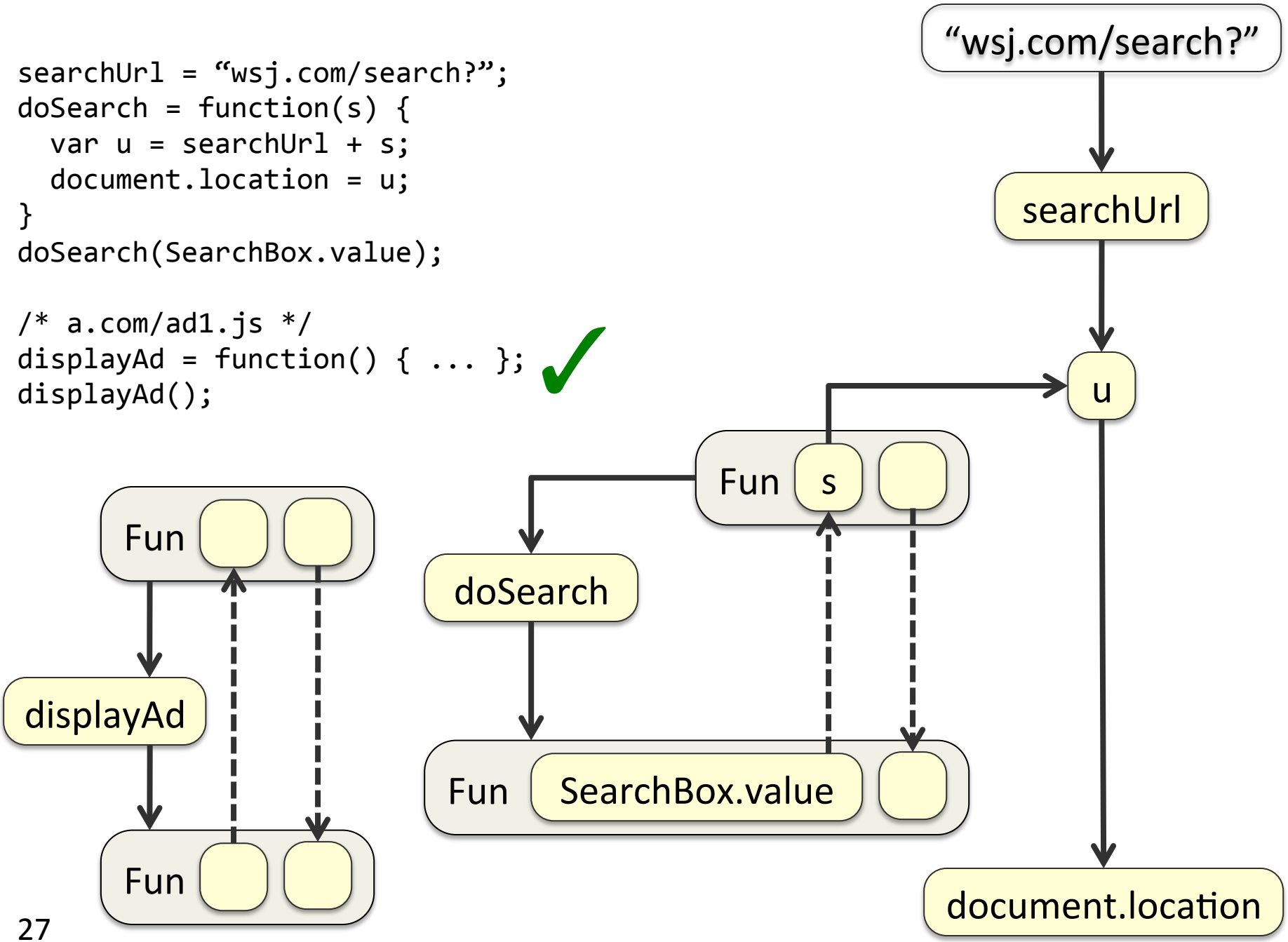
```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad1.js */
displayAd = function() { ... };
displayAd();
```



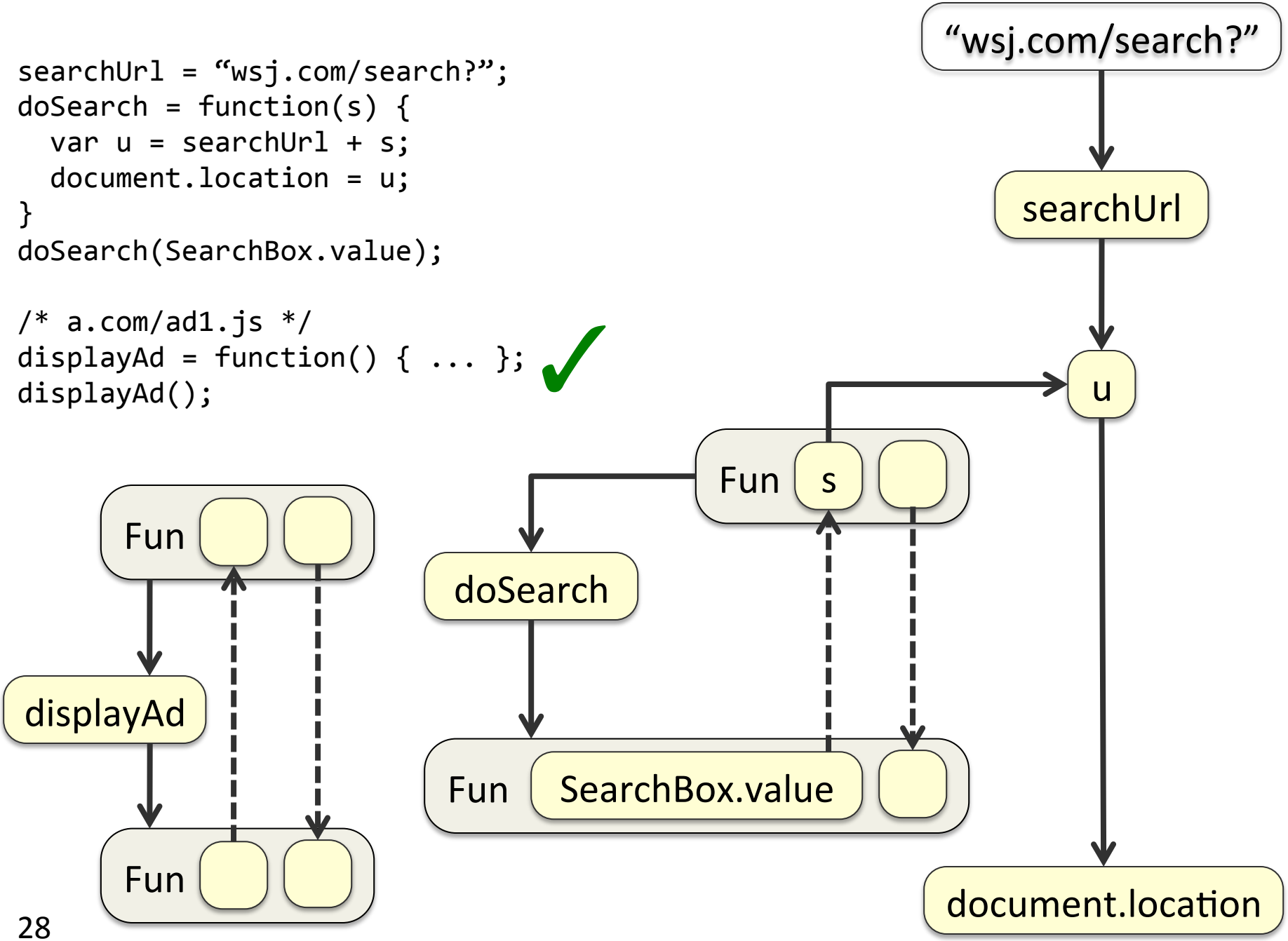
```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad1.js */
displayAd = function() { ... };
displayAd();
```



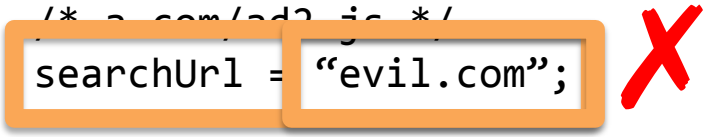
```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad1.js */
displayAd = function() { ... };
displayAd();
```

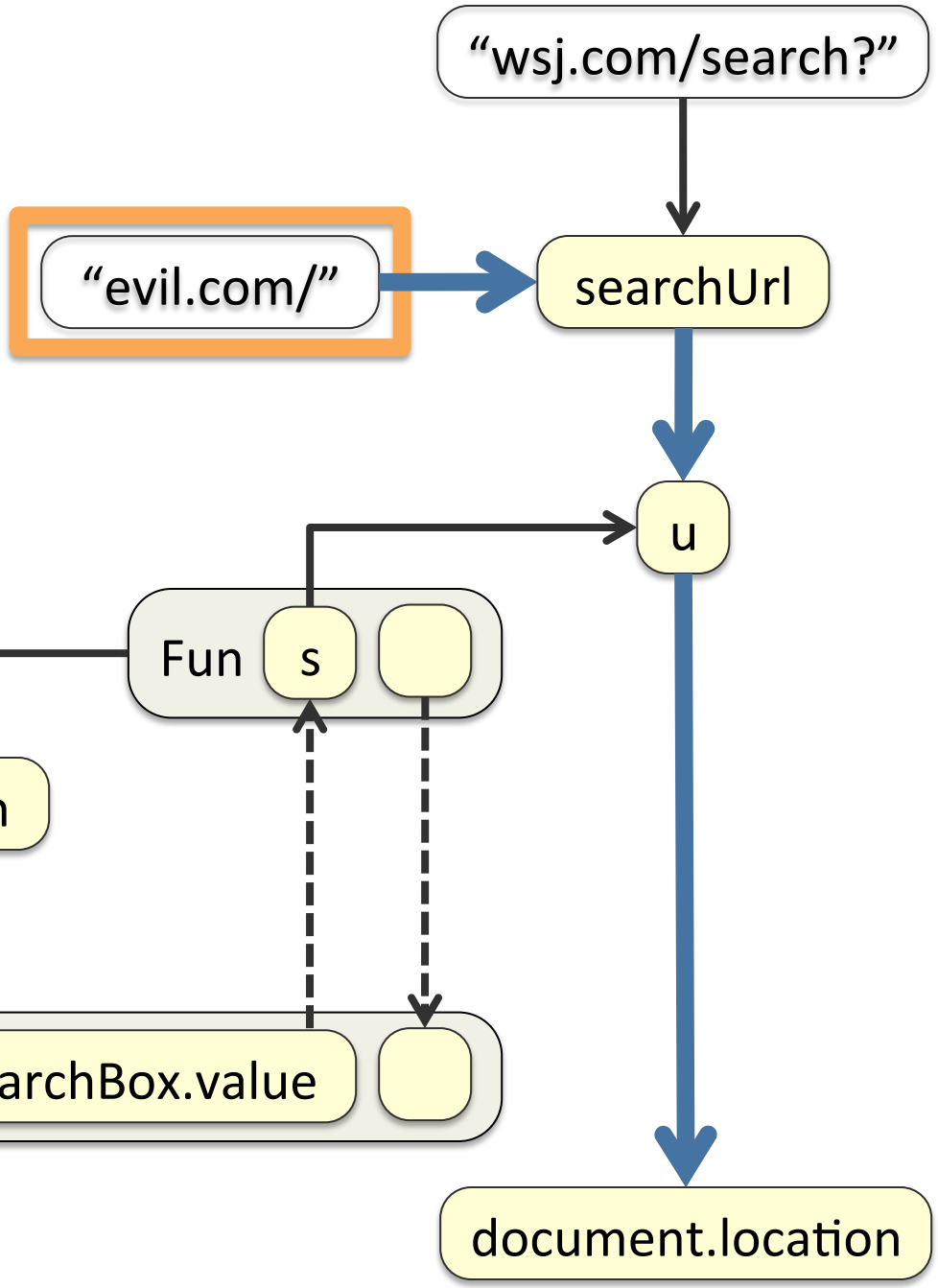


```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad1.js */
displayAd = function() { ... };
displayAd();
```

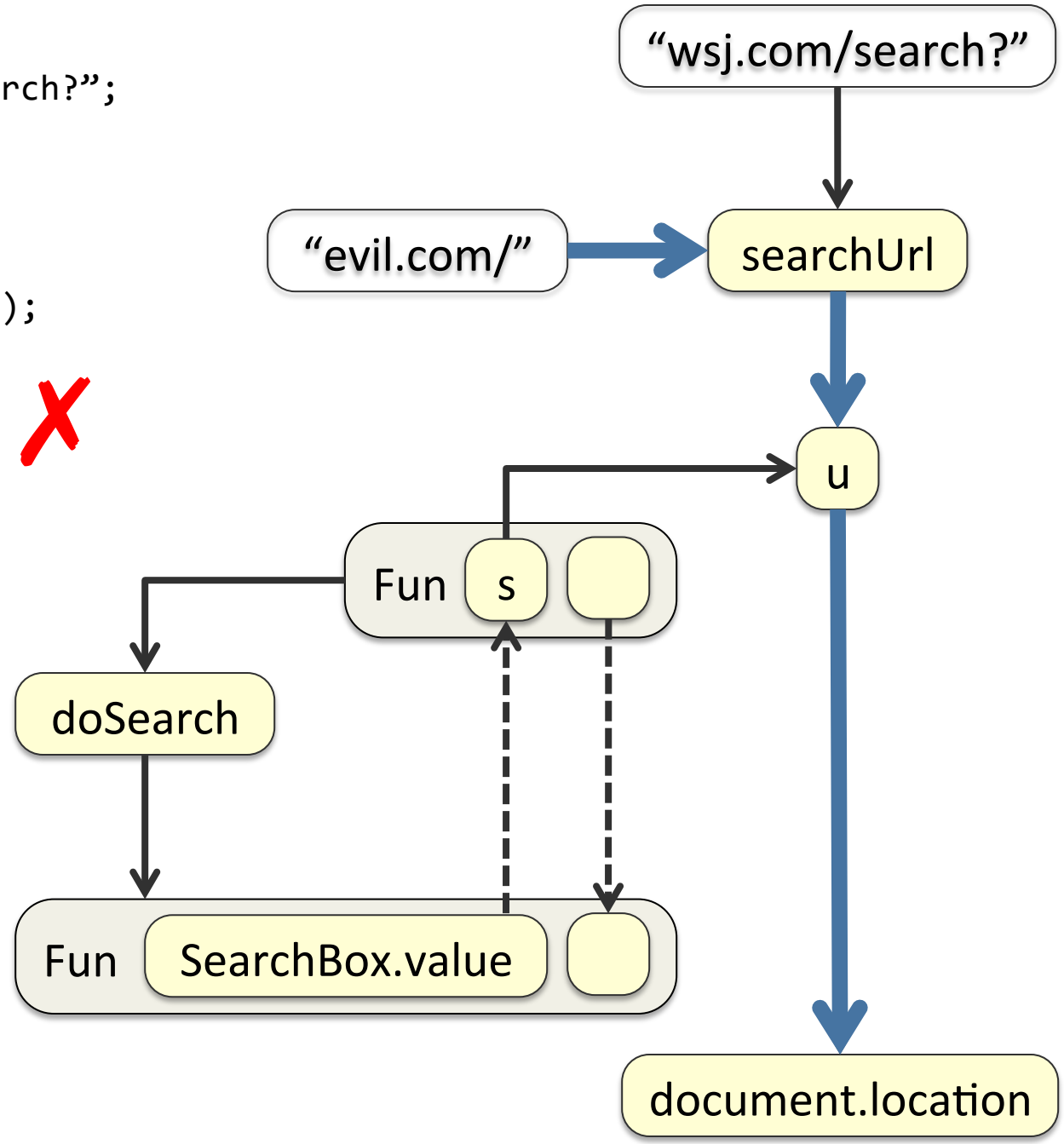


```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad2.js */
searchUrl = “evil.com”;
```

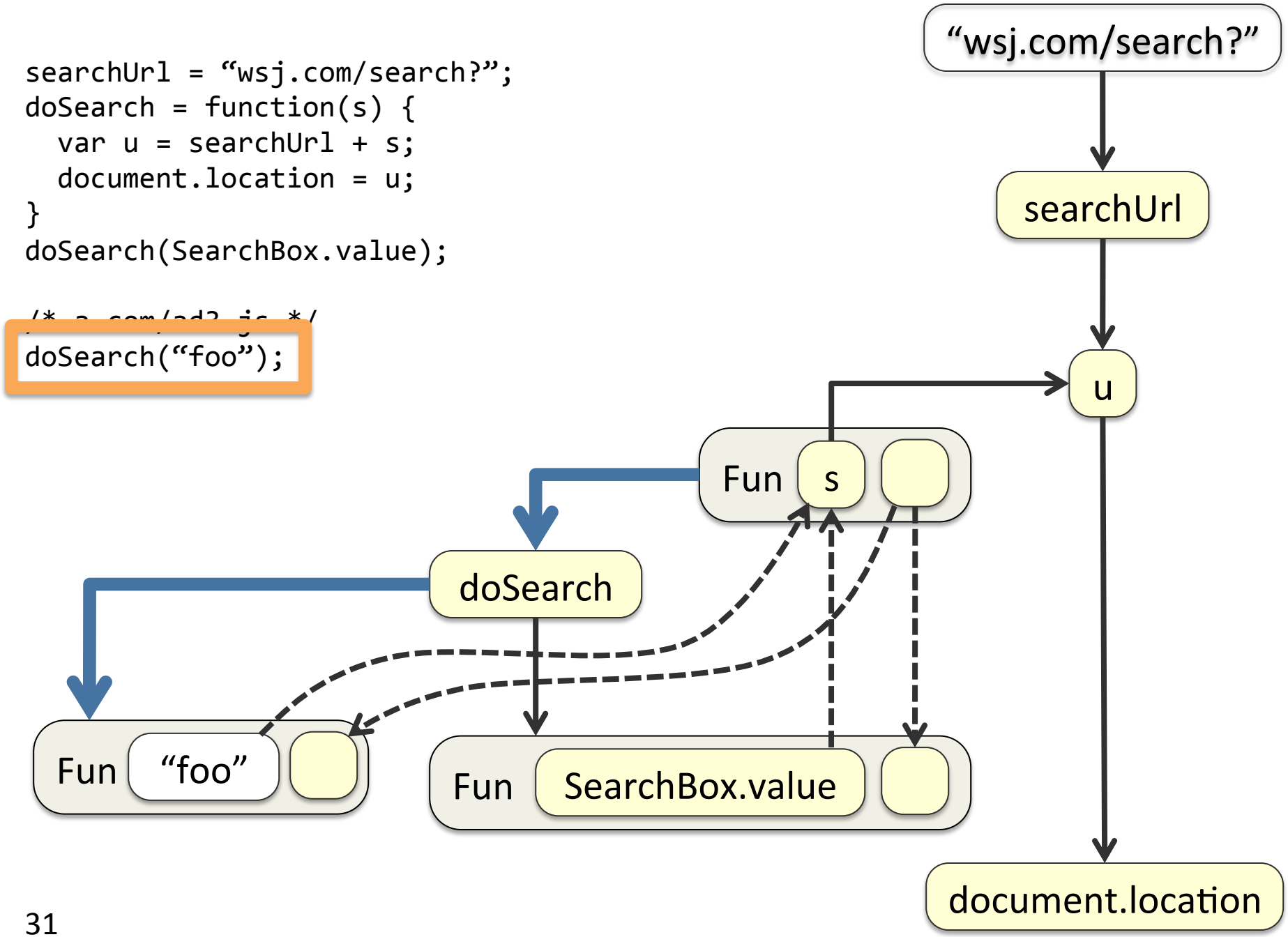




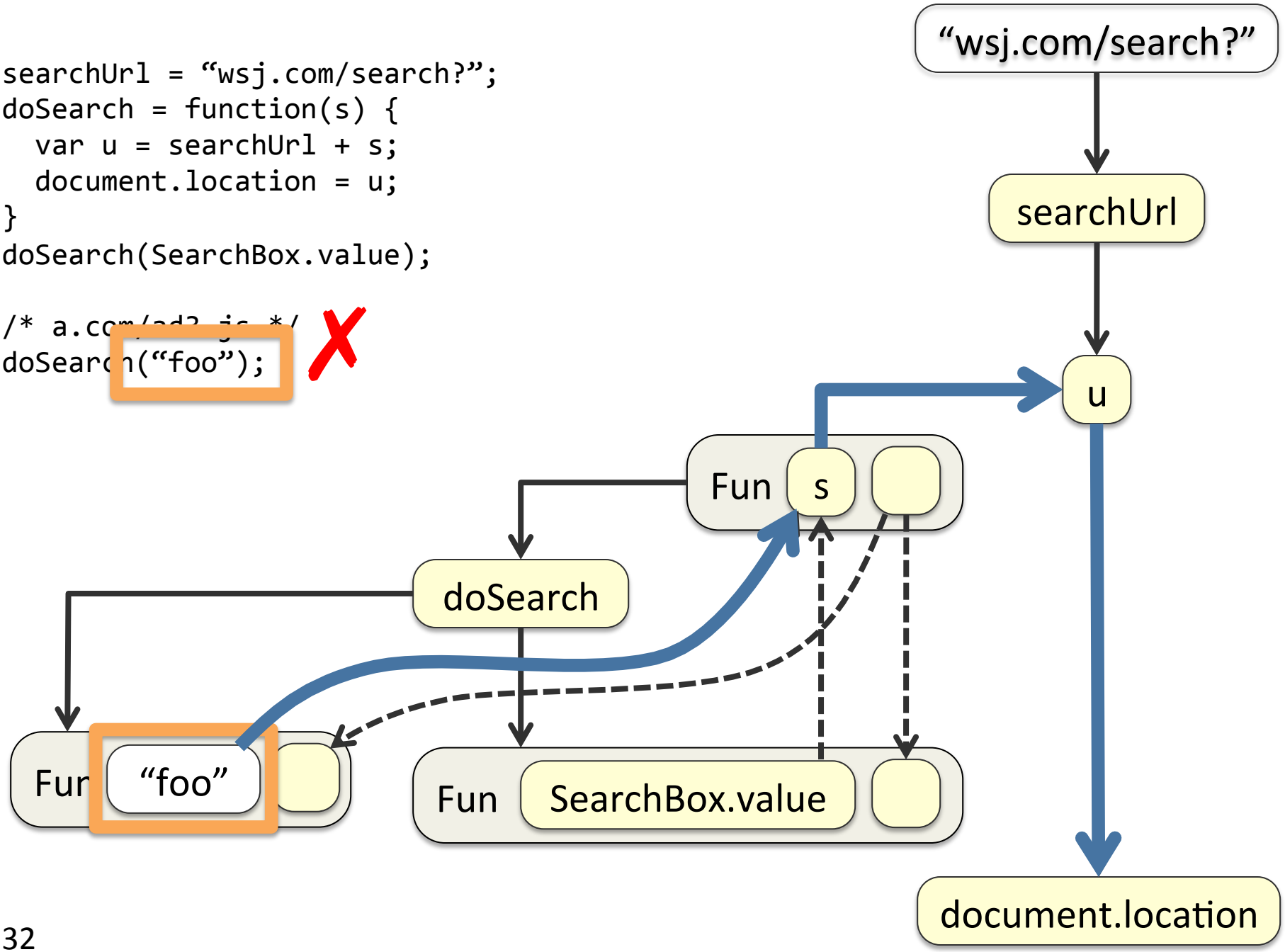
```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad2.js */
searchUrl = “evil.com”;
```



```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad2.js */
doSearch(“foo”);
```



```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);

/* a.com/ad2.js */
doSearch(“foo”);
```

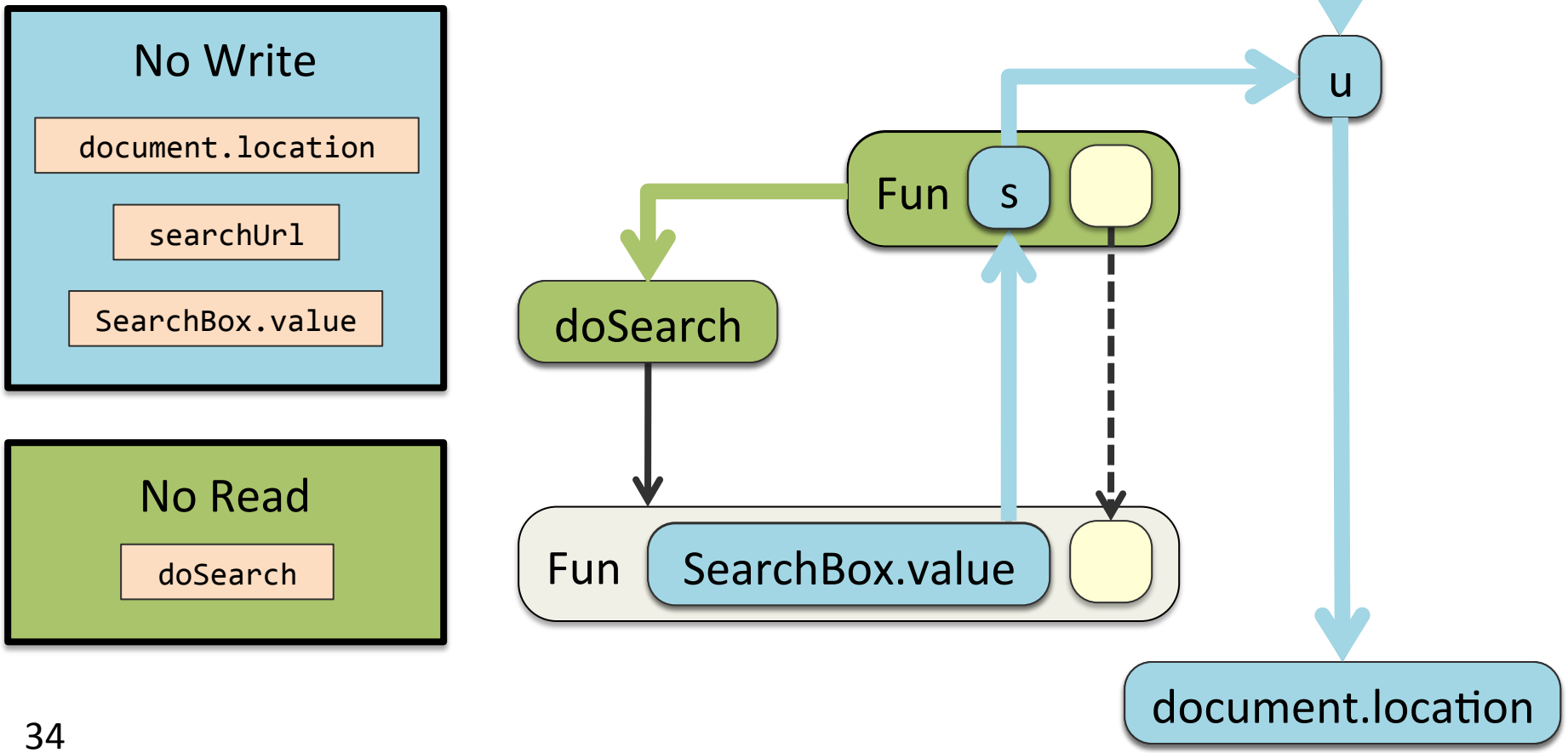


Outline

- Overview
- JavaScript Static Analysis
- Computing Residual Policies
- Additional Challenges
- Evaluation

```
searchUrl = “wsj.com/search?”;
doSearch = function(s) {
  var u = searchUrl + s;
  document.location = u;
}
doSearch(SearchBox.value);
```

Add taint to sensitive data and propagate

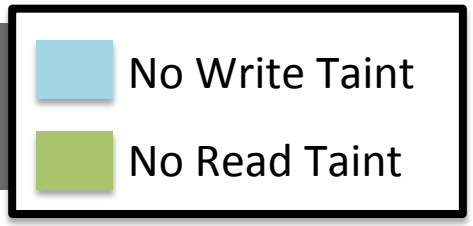


Residual Policies

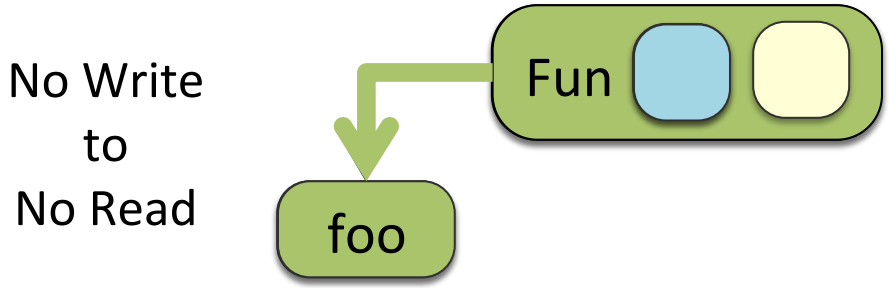
- Difficulties:
  - Aliasing
  - First-class functions
  - Don't want flow analysis in browser

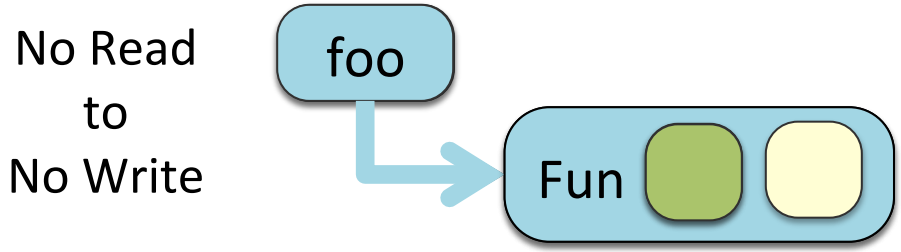
- Solution:
  - Conservatively taint functions
  - Conservatively taint fields

Tainted Functions



- Transfer taints from parameters to functions

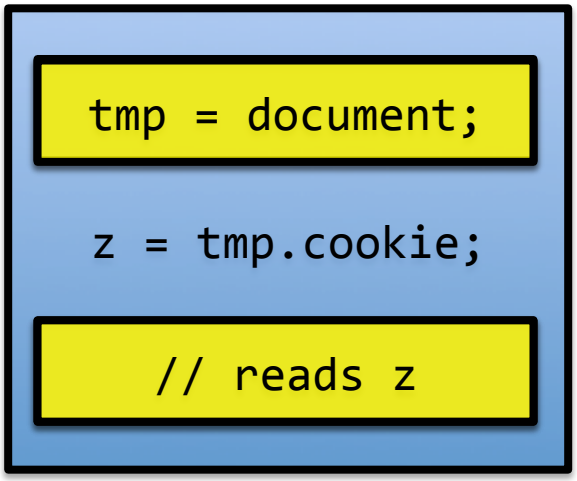


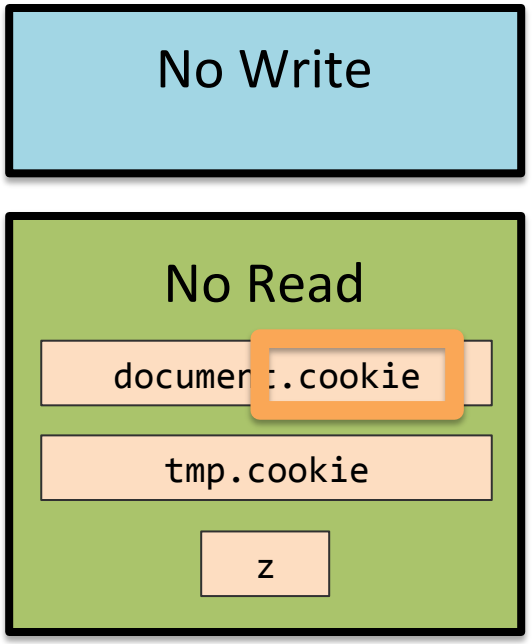



```
// hole redefines foo
foo = function(t) {
  // reads t, hence cookie
}

foo(document.cookie);
```

- Transfer taints from return values to functions

Aliasing and Tainted Fields





- Residual policy misses future aliasing
- Conservative approach:

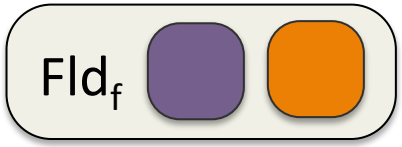
  if field f is tainted for some object, f tainted for all

Outline

- Overview
- JavaScript Static Analysis
- Computing Residual Policies
- Additional Challenges
- Evaluation

Objects

- Used pervasively in JavaScript
- Hence, analysis must be field-sensitive
- Encode "setter" and "getter" for field f using



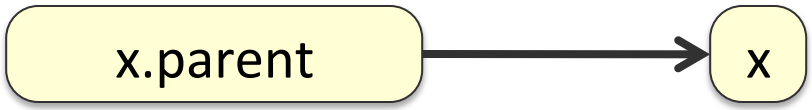
- Fields can be dynamically added

```
x = { f:1 };
x.g = 2;
```

- Initially assume no fields
- Iteratively add constraints until fixpoint

Prototypes

- JavaScript uses prototype-based inheritance
- Intuitively, each object x
  - has a link to its parent
  - inherits parent's fields



- Ensures each object has fields of its ancestors

Indirect Flows

```
if (document.cookie == “foo”) {
   y = 1;
}
```

- Propagate taints along indirect flow edges
- But not program values

Outline

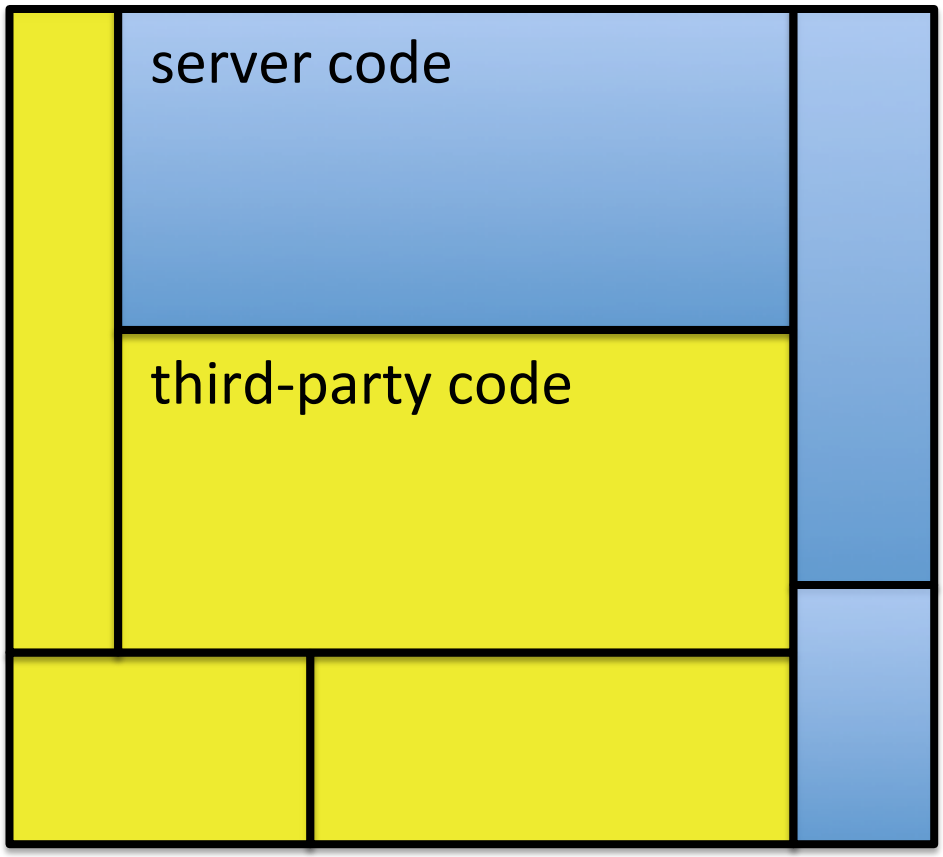
- Overview
- JavaScript Static Analysis
- Computing Residual Policies
- Additional Challenges
- Evaluation

Implementation

- Flow analysis and residual policy generator
  - parse JavaScript (JSure)
  - generate set constraints (6,000 lines of OCaml)
  - solve constraints (Banshee + 400 lines of C)
- Stand-alone residual policy checker
  - not yet incorporated into browser
- JavaScript collector
  - Firefox extension (500 lines of JavaScript)

Experimental Setup

- Collect JavaScript for Alexa top 100 web sites



Context:
all server code

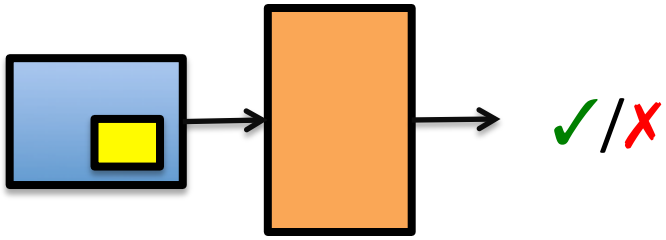
Hole:
all third-party code

97/100 have JavaScript

63/97 have holes

Experimental Setup

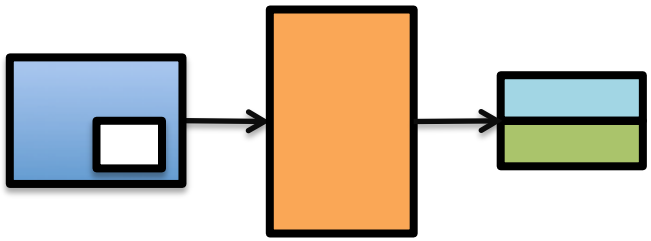
- Information flow analysis on context + hole

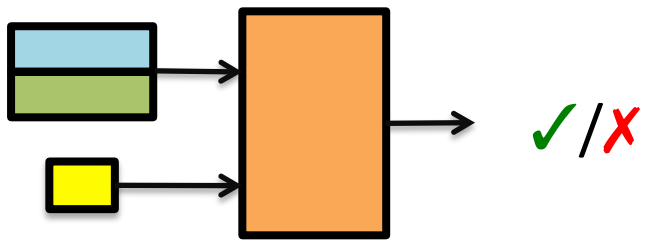


`cookie` confidentiality

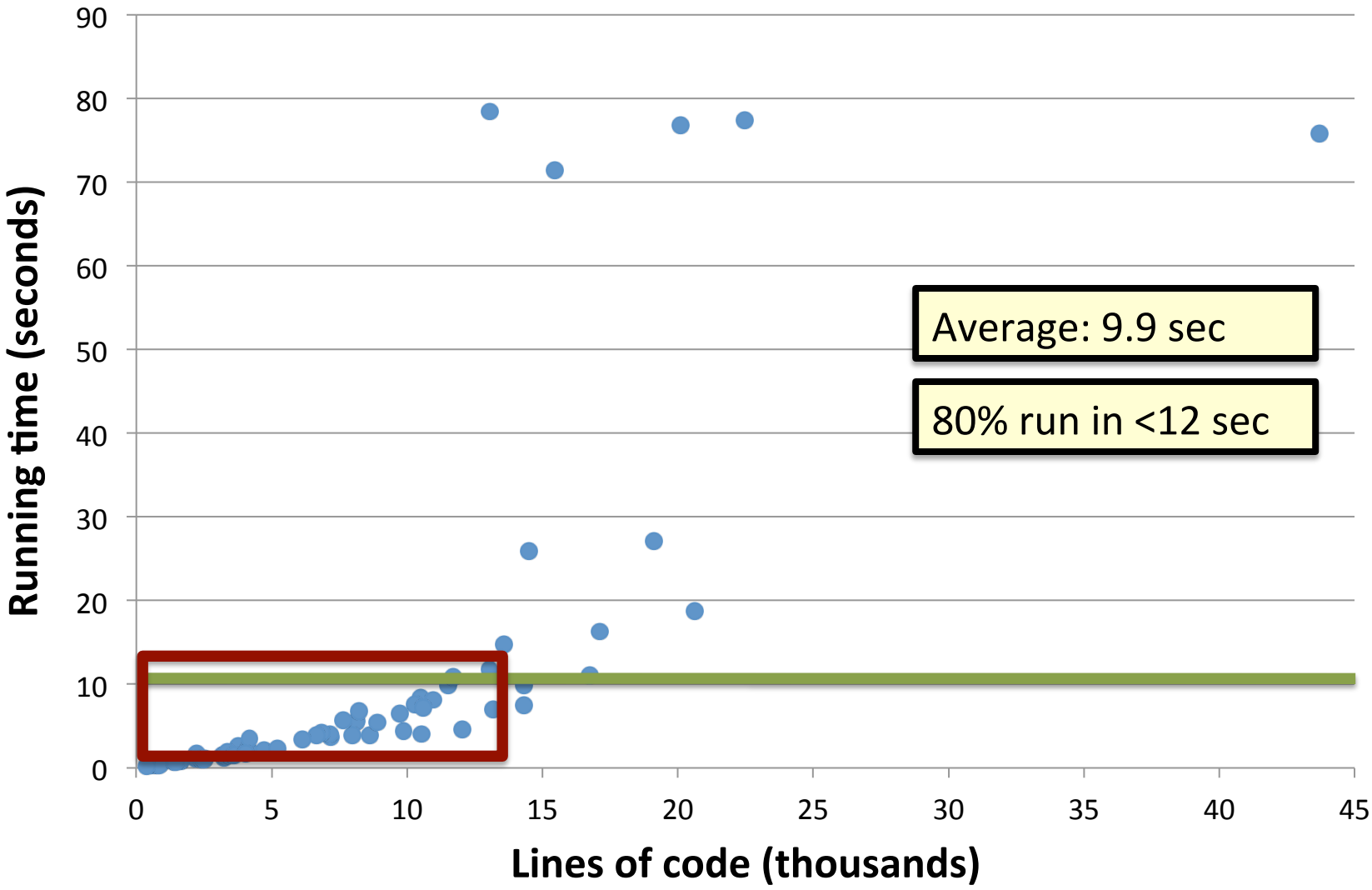
`location` integrity

- Compute residual policy, check it on hole



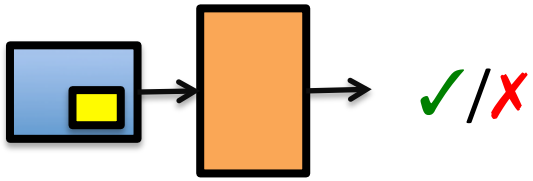


Scalability of Full Analysis



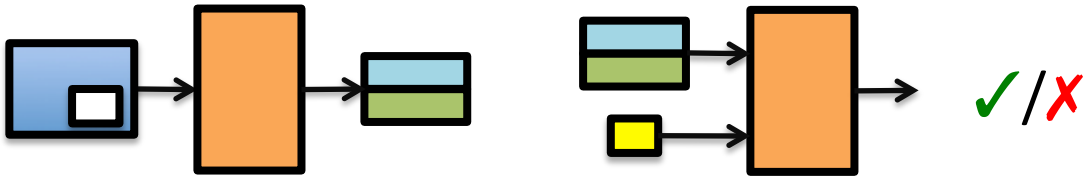
Average Running Times

Full Analysis



9.9 sec

Staged Analysis

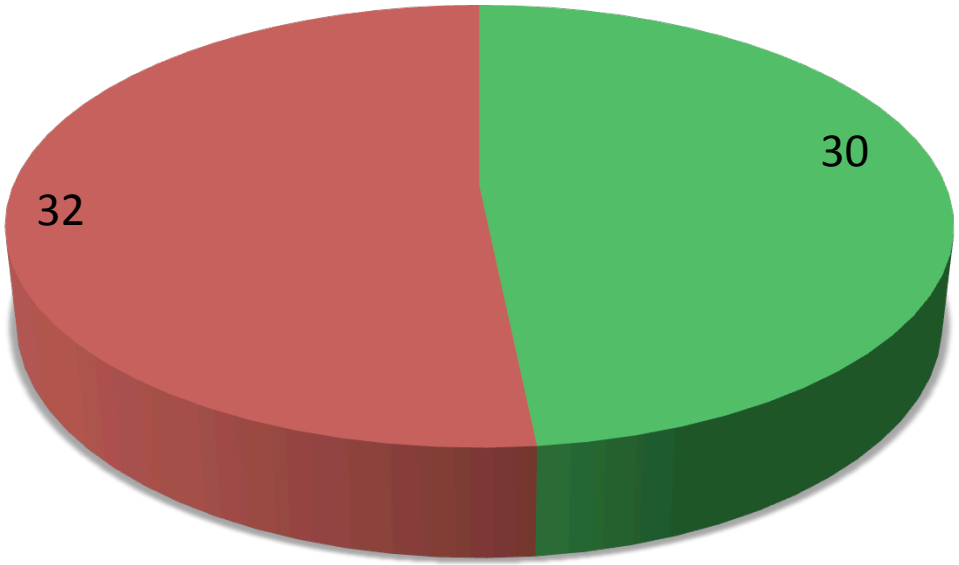


14.0 sec

0.13 sec

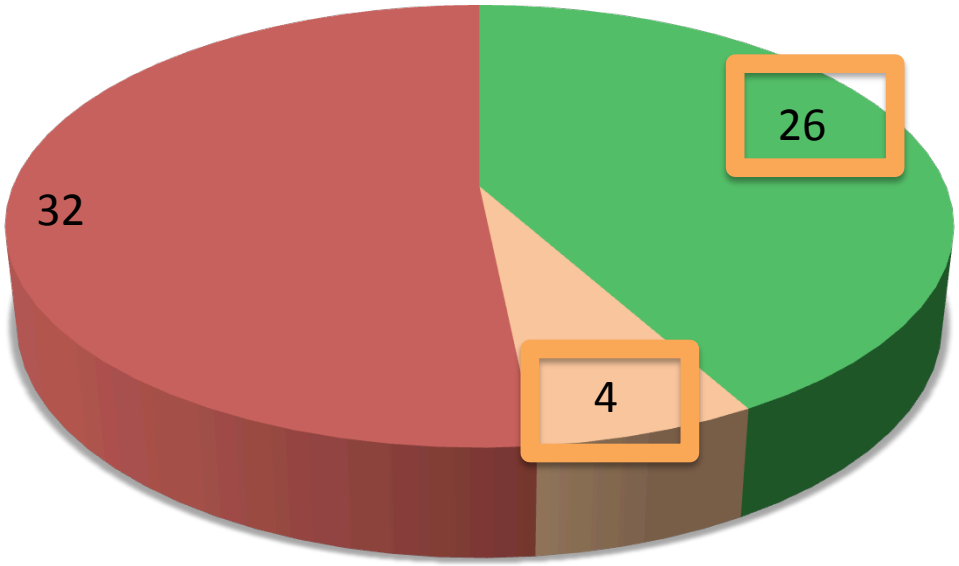
Results of Analysis: Full

- Hole satisfies cookie policy? ✓ 30 ✗ 32



Results of Analysis: Staged

- Hole satisfies cookie policy? ✓ 30 ✗ 32



Residual checker:

- 26/30 safe
- Imprecision:
  4 false positives

Future Work

- Context-sensitivity
- Dynamically-constructed field names
- Test more complicated policies
- Embed residual policy checker in browser

Related Work

- Information flow
  - type systems
  - dynamic instrumentation
- JavaScript analysis
  - types [Thiemann 05, Anderson et al. 05]
  - dynamic policies [Chander et al. 07]
  - static analysis [Guarnieri/Livshits 09]
- Browser security
  - finer-grained interaction between scripts [Howell et al. 07]

Summary

- JavaScript static analysis is scalable
- Residual checks are fast enough for client
- Residual policies precisely capture information flow

Thanks!

Extra Slides

Information Flow Policies

Confidentiality of x:
x should not affect hole variables

directly or indirectly

```
holeVar = x;
```

```
if (x) { holeVar = 1 };
```

Integrity of x:
hole variables should not affect x

directly or indirectly

```
x = holeVar;
```

```
if (holeVar) { x = 1 };
```

Fields

Running Times

	cookie policy	location policy
Flow analysis on context + hole	9.9	10.7
Computing residual policy	14.0	28.4
Checking residual policy	0.13	0.12

- Full analysis too slow to run on client
- Quick to compute residual policy on server
- Small run-time overhead to check
  - running time includes parsing time
  - parser is not optimized for speed

Results of Staged Analysis

Full: Policy Satisfied?	Staged: Policy Satisfied?	cookie policy	location policy
✓	✓	26	49
	✗ (imprecision)	4	8
✗	✗	32	5
	✓ (soundness)	0	0

- Residual policy usually agrees with full information flow analysis
- Imprecision from tainted functions/fields
- No false negatives