

DoDOM: Leveraging DOM Invariants for Web 2.0 Application Robustness Testing

Karthik Pattabiraman,

University of British Columbia (UBC)

Benjamin Zorn, Microsoft Research



Web 2.0 Application: Amazon.com



Web 2.0 applications have many components, often from different domains

Web 2.0 Applications: JavaScript

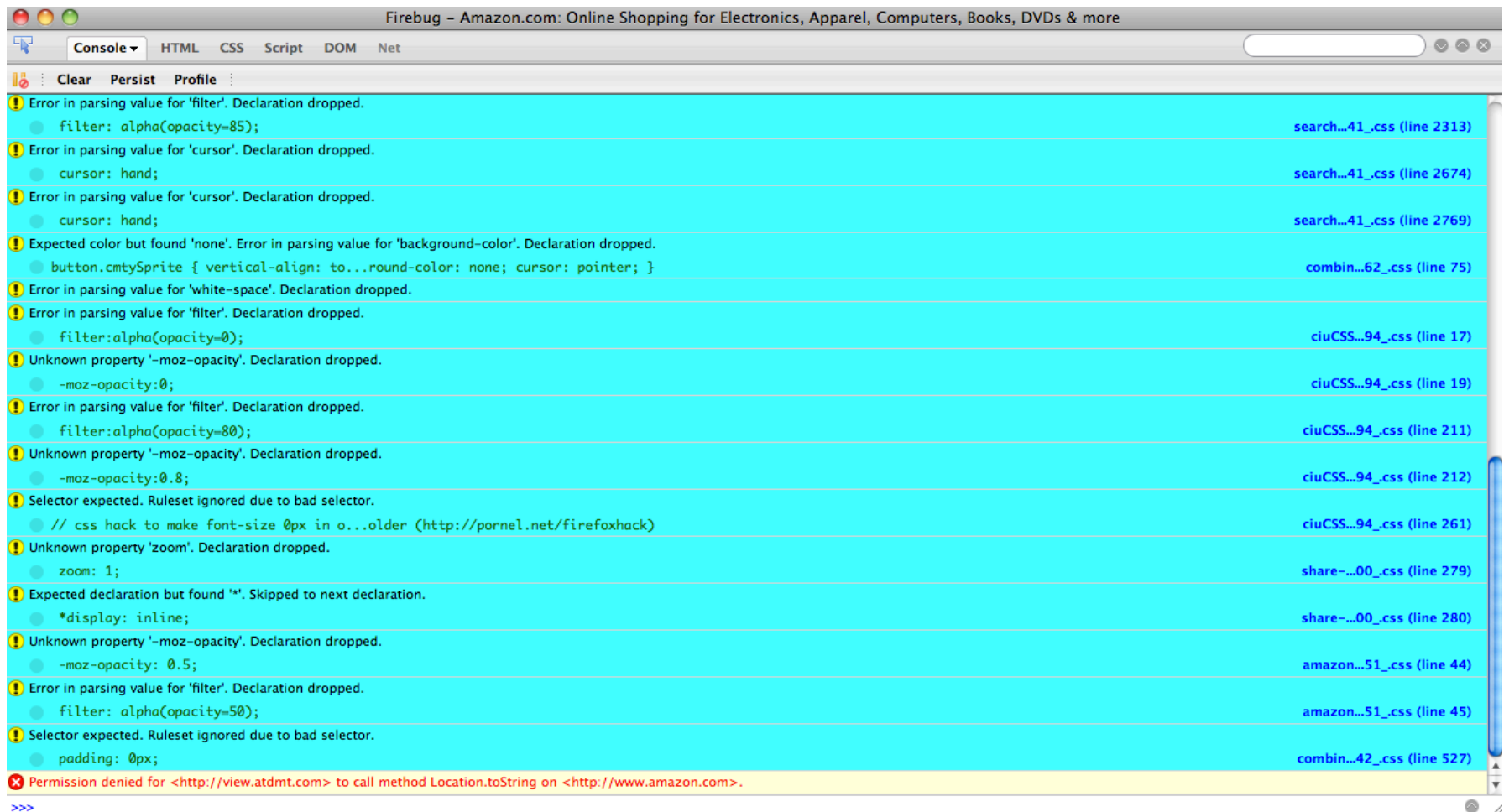


The screenshot shows a web browser window with the address bar displaying a URL from Amazon. The browser's developer tools are open, showing a large amount of JavaScript code being executed in the console. The code is for a function named `S9MultiPackLayout.prototype.makeVisible`, which appears to be a jQuery plugin for handling a multi-pack layout. The code is quite verbose, with many lines of conditional logic and DOM manipulation. The browser's status bar at the bottom shows "Format code", "Execute JS", and "Error console" buttons.

```
109.
110.
111. }
112.
113. S9MultiPackLayout.prototype.makeVisible = function() {
114.     var numProposedVisibleItems = this.numProposedVisibleItems();
115.     var lastVisibleCol = this.firstVisibleCol + numProposedVisibleItems - 1;
116.     var width = ((100 / numProposedVisibleItems)-1);
117.
118.     if (this.seedItem) {
119.         this.seedItem.style.width = width + "%";
120.         this.itemChildren[0].style.display = "";
121.         this.itemChildren[0].style.width = "100%";
122.         this.otherItems.style.width = (98 - width) + "%";
123.         var widthWithoutSeed = ((100 / (numProposedVisibleItems-1))-1);
124.         for (var i = 1; i < this.itemChildren.length; i++) {
125.             if ((i >= this.firstVisibleCol) && (i <= lastVisibleCol)) {
126.                 this.itemChildren[i].style.display = "";
127.                 this.itemChildren[i].style.width = widthWithoutSeed + "%";
128.                 if (this.itemImages[i].getAttribute("url")) {
129.                     this.itemImages[i].src = this.itemImages[i].getAttribute("url");
130.                     this.itemImages[i].setAttribute("url", "");
131.                 }
132.             } else {
133.                 this.itemChildren[i].style.display = "none";
134.             }
135.         }
136.     } else {
137.         for (var i = 0; i < this.itemChildren.length; i++) {
138.             if ((i >= this.firstVisibleCol) && (i <= lastVisibleCol)) {
139.                 this.itemChildren[i].style.display = "";
140.                 this.itemChildren[i].style.width = width + "%";
141.                 if (this.itemImages[i].getAttribute("url")) {
142.                     this.itemImages[i].src = this.itemImages[i].getAttribute("url");
143.                     this.itemImages[i].setAttribute("url", "");
144.                 }
145.             } else {
146.                 this.itemChildren[i].style.display = "none";
147.             }
148.         }
149.     }
150. }
```

Significant amount of JavaScript code executing in the browser

Web 2.0 Application: Amazon.com



Web Apps experience errors, yet they continue executing !

Problem

Web 2.0 applications error-prone because they

1. Are distributed and asynchronous
2. Integrate code and data from multiple domains
3. Have loose error-detection semantics

Need to test web app robustness using fault-injections

Challenging because web applications' behavior is non-deterministic from one execution to another

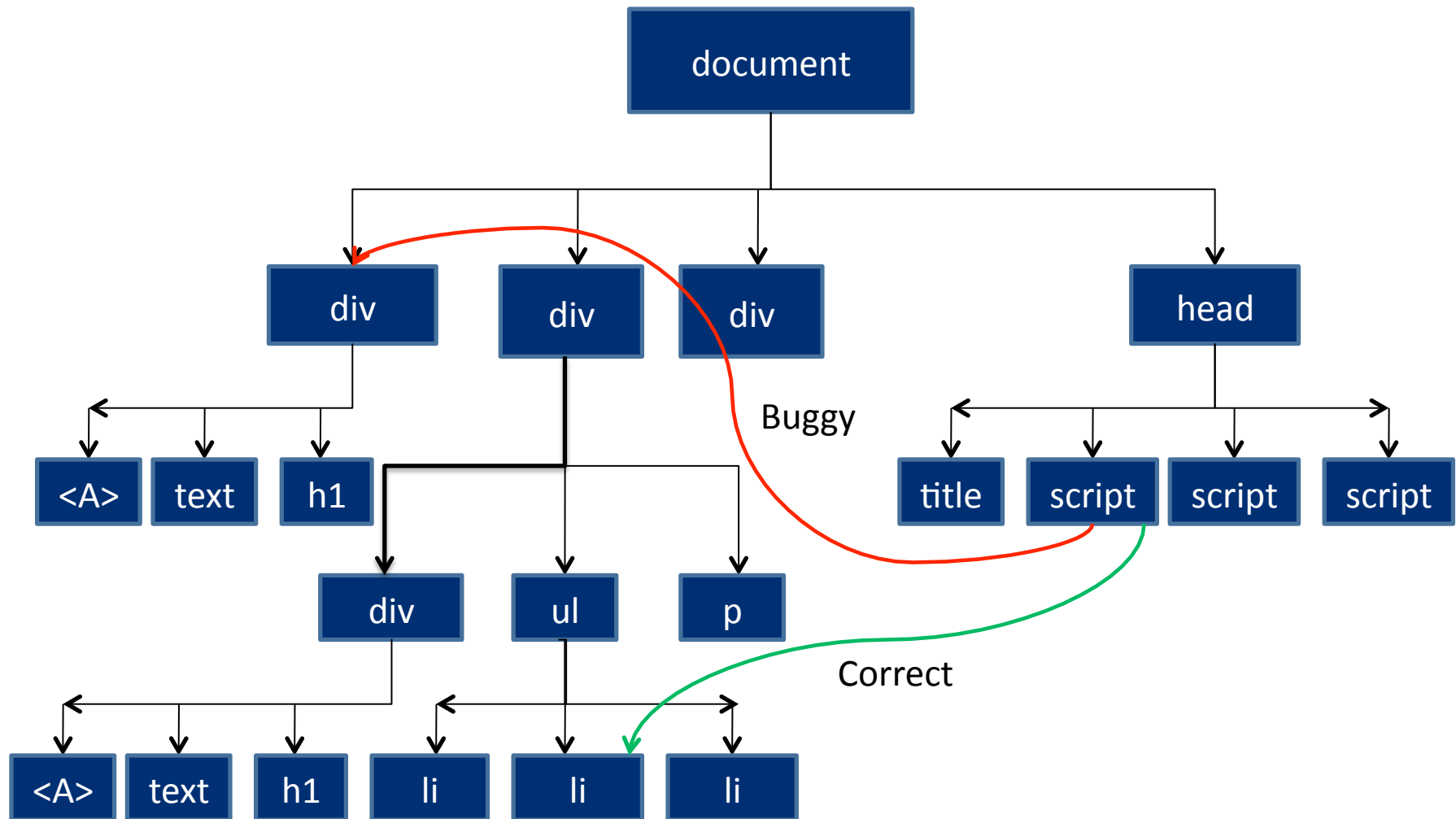
This Paper

- **Builds a characterization of web applications' behavior for robustness testing (fault-injections)**
- **Goals**
 - Automated, i.e., no manual intervention
 - Language and platform neutral
 - Capable of handling real web applications
- **Approach: Dynamic extraction of DOM-based invariants in web applications - DoDOM**

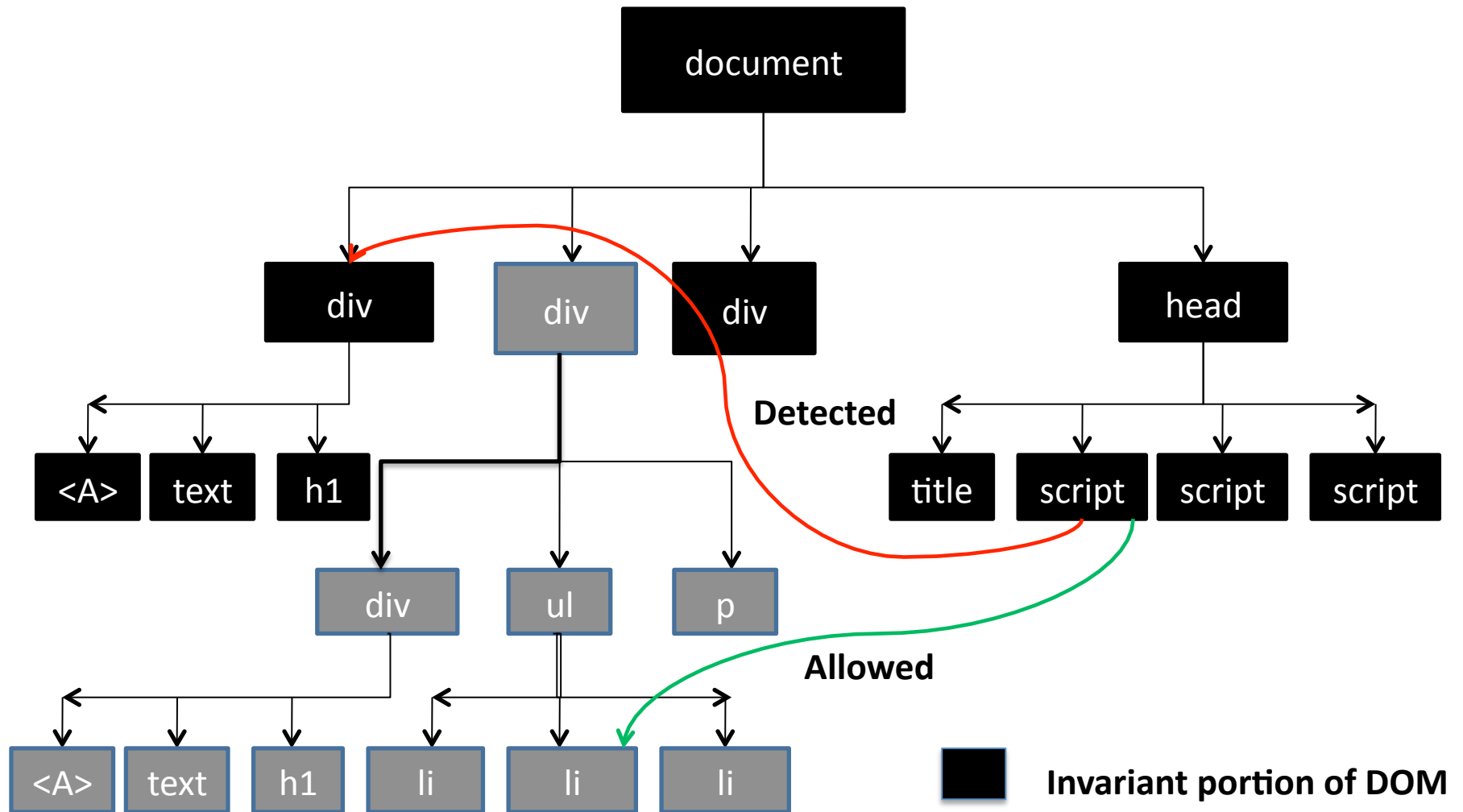
Example: HTML

```
<html>
<head>
  <title> .... </title>
  <script> .... </script>
  <script> ... </script>
</head>
<body>
  <div> <A> .... </A> <text> ... </text> <h1> ... </h1> </div>
  <div>
    <div>.... </div>
    <ul> <li>...</li> <li>...</li><li> ... </li> </ul>
    <p> ... </p>
  </div>
  <div> ... </div>
</body>
</html>
```

Example: DOM



Example: Invariant DOM



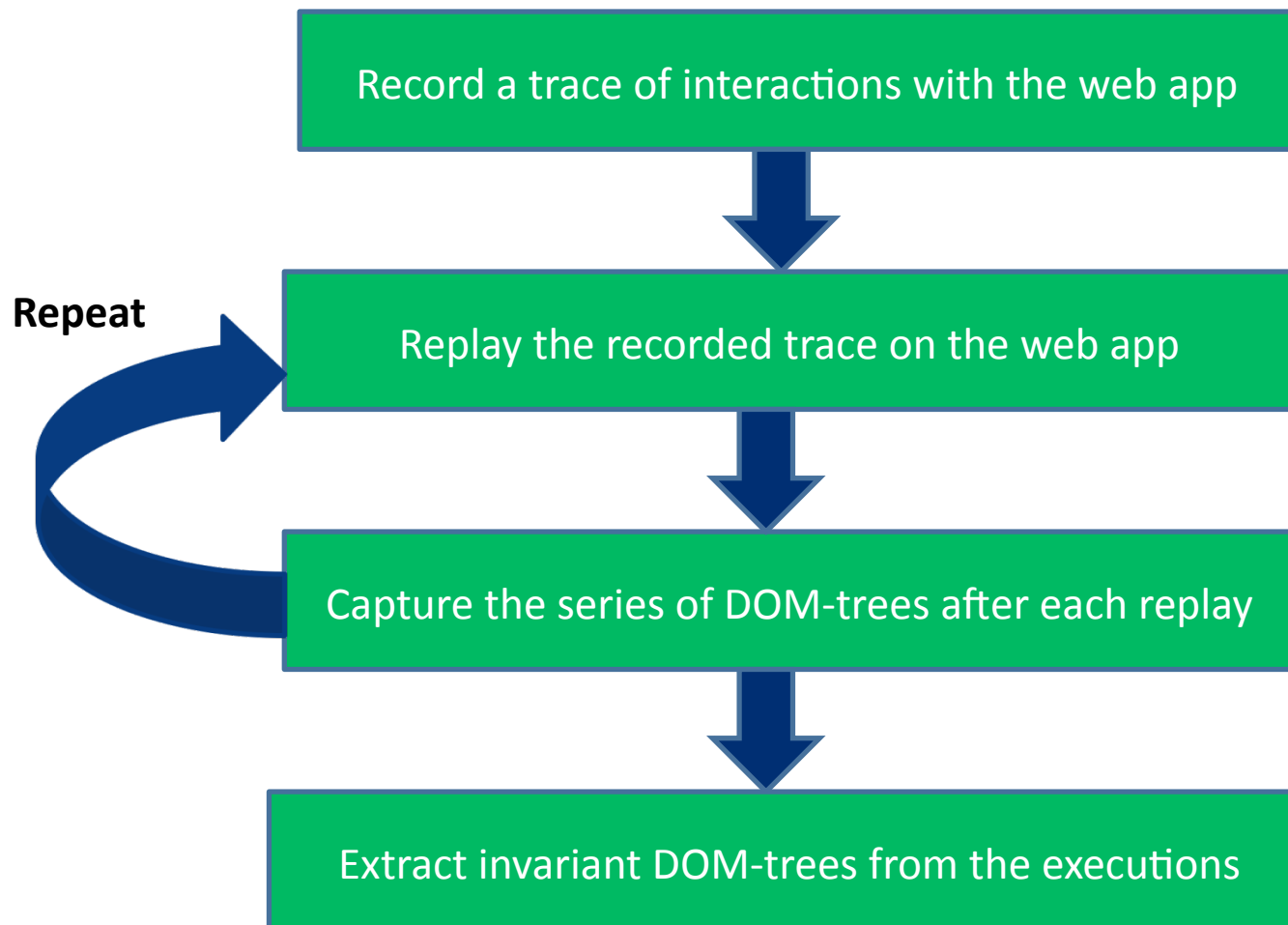
DoDOM: Contributions

- **DOM invariants can be learned dynamically**
 - Automated tool called DoDOM
- **Invariants exist in many Web 2.0 applications**
 - Stabilize within a few executions
 - Incur very few false-positives
- **Invariants are useful for error detection**
 - Both for event errors and domain failures

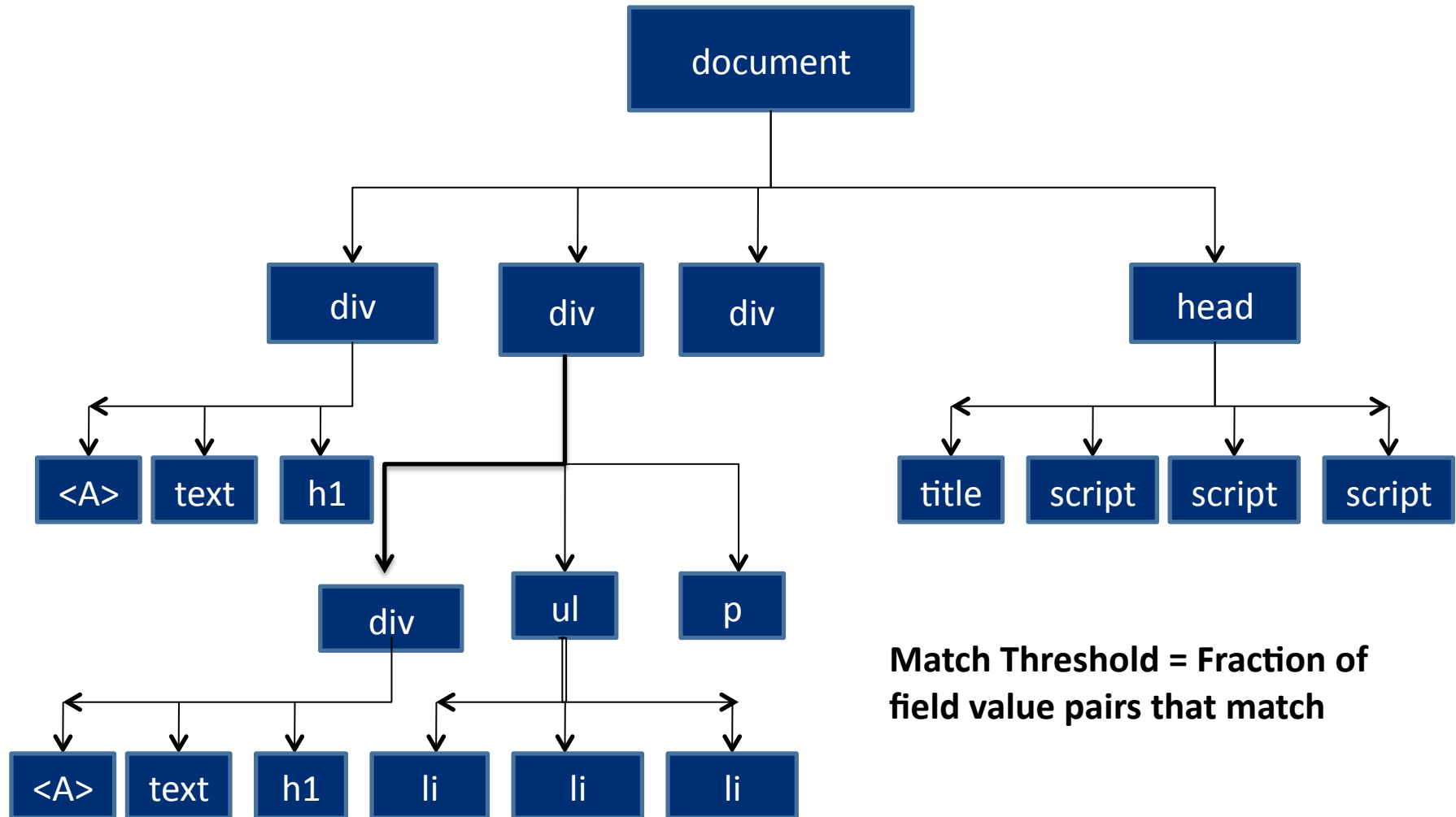
DoDOM: Contributions

- **DOM invariants can be learned dynamically**
 - Automated tool called DoDOM
- **Invariants exist in many Web 2.0 applications**
 - Stabilize within a few executions
 - Incur very few false-positives
- **Invariants are useful for error detection**
 - Both for event errors and domain failures

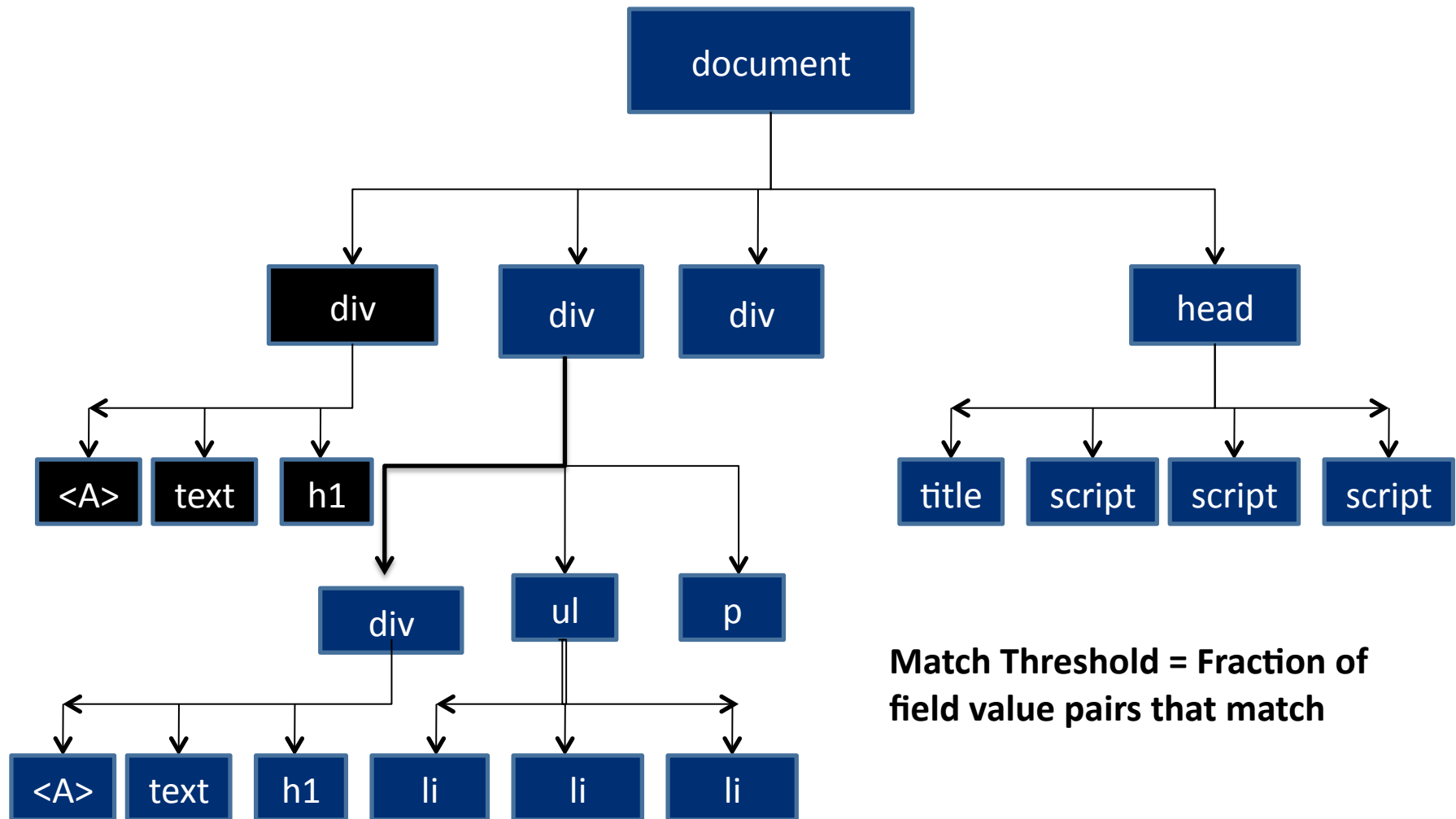
DoDOM: Approach



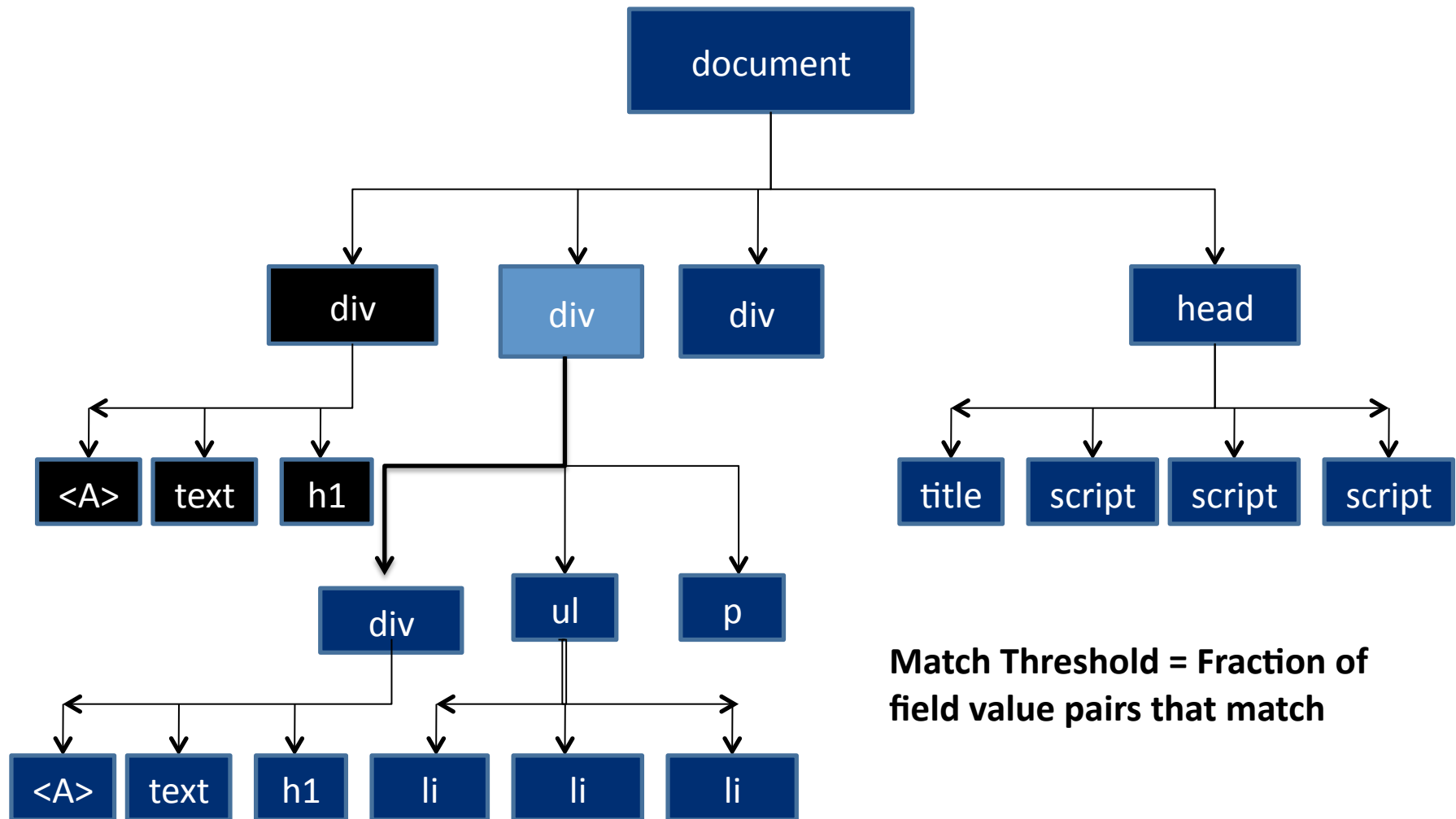
DoDOM: Invariant Extraction



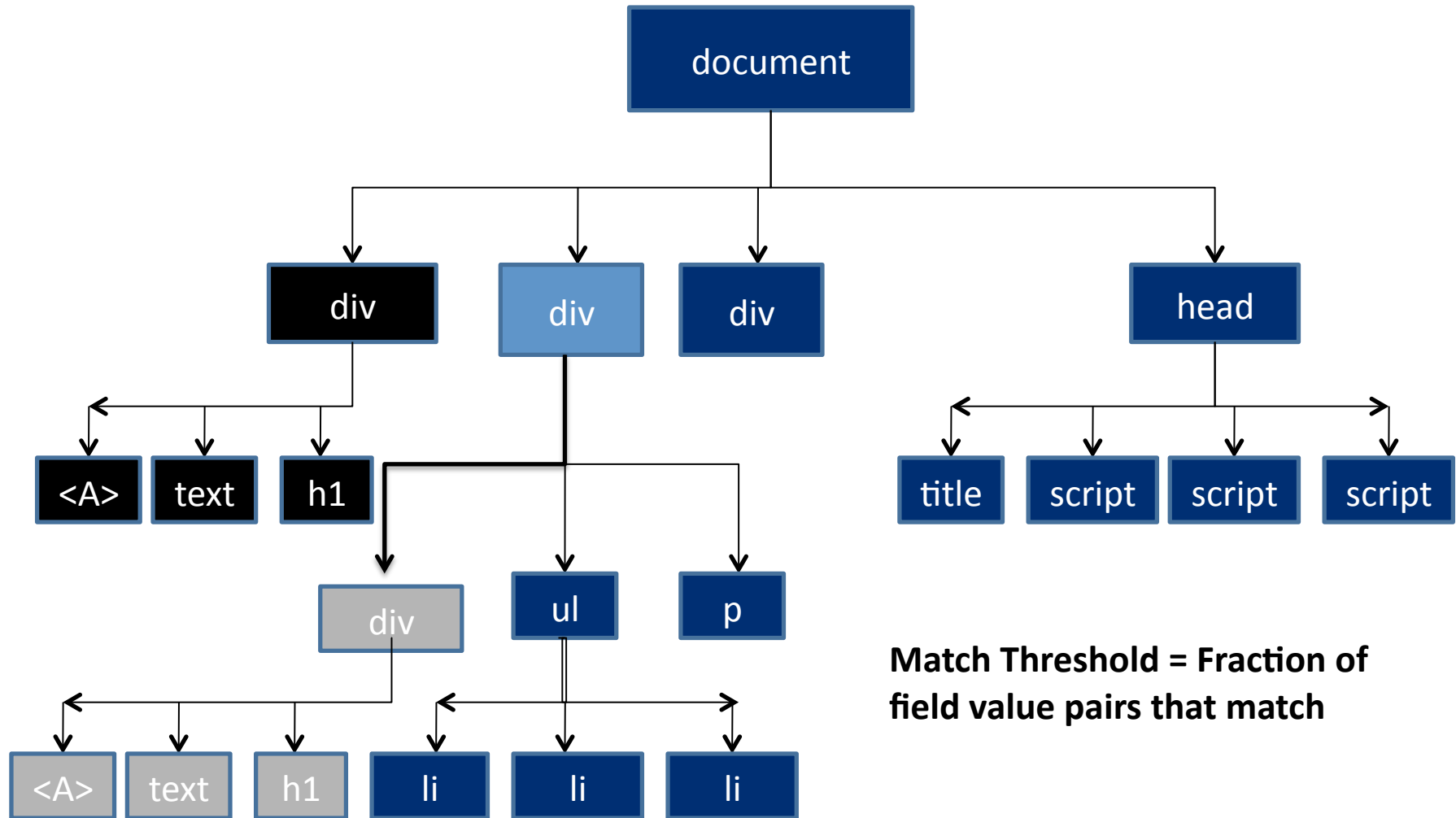
DoDOM: Invariant Extraction



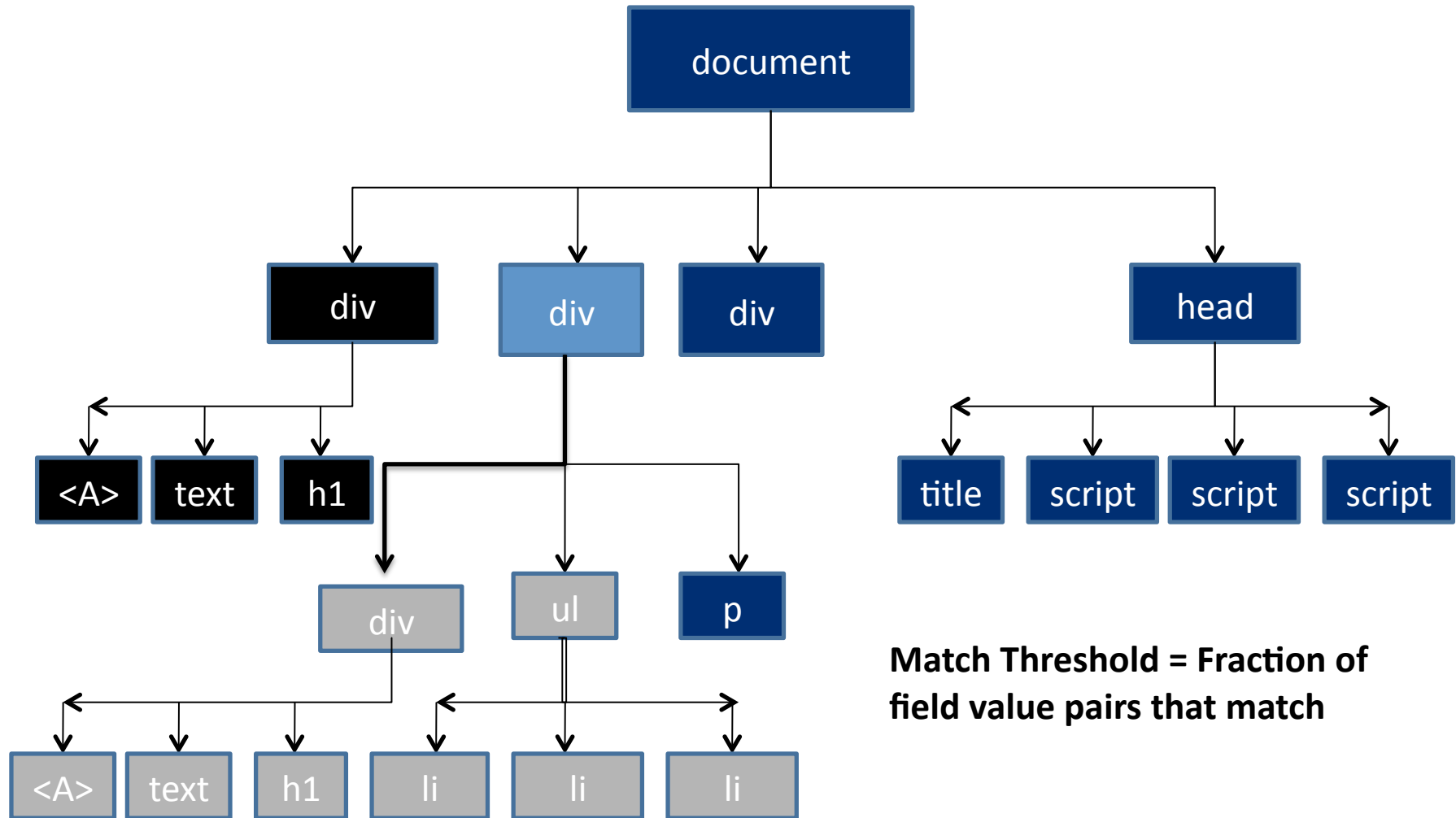
DoDOM: Invariant Extraction



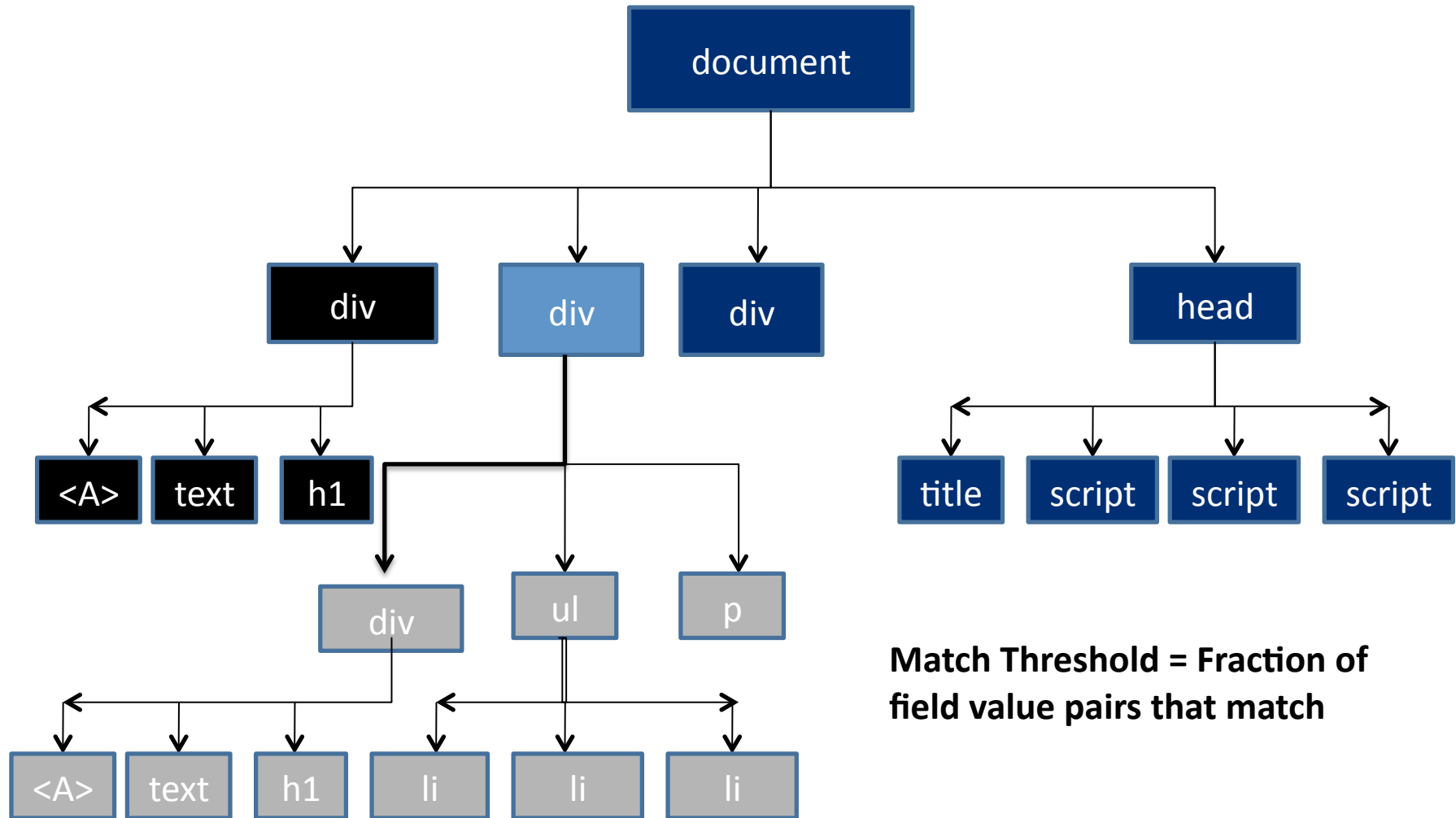
DoDOM: Invariant Extraction



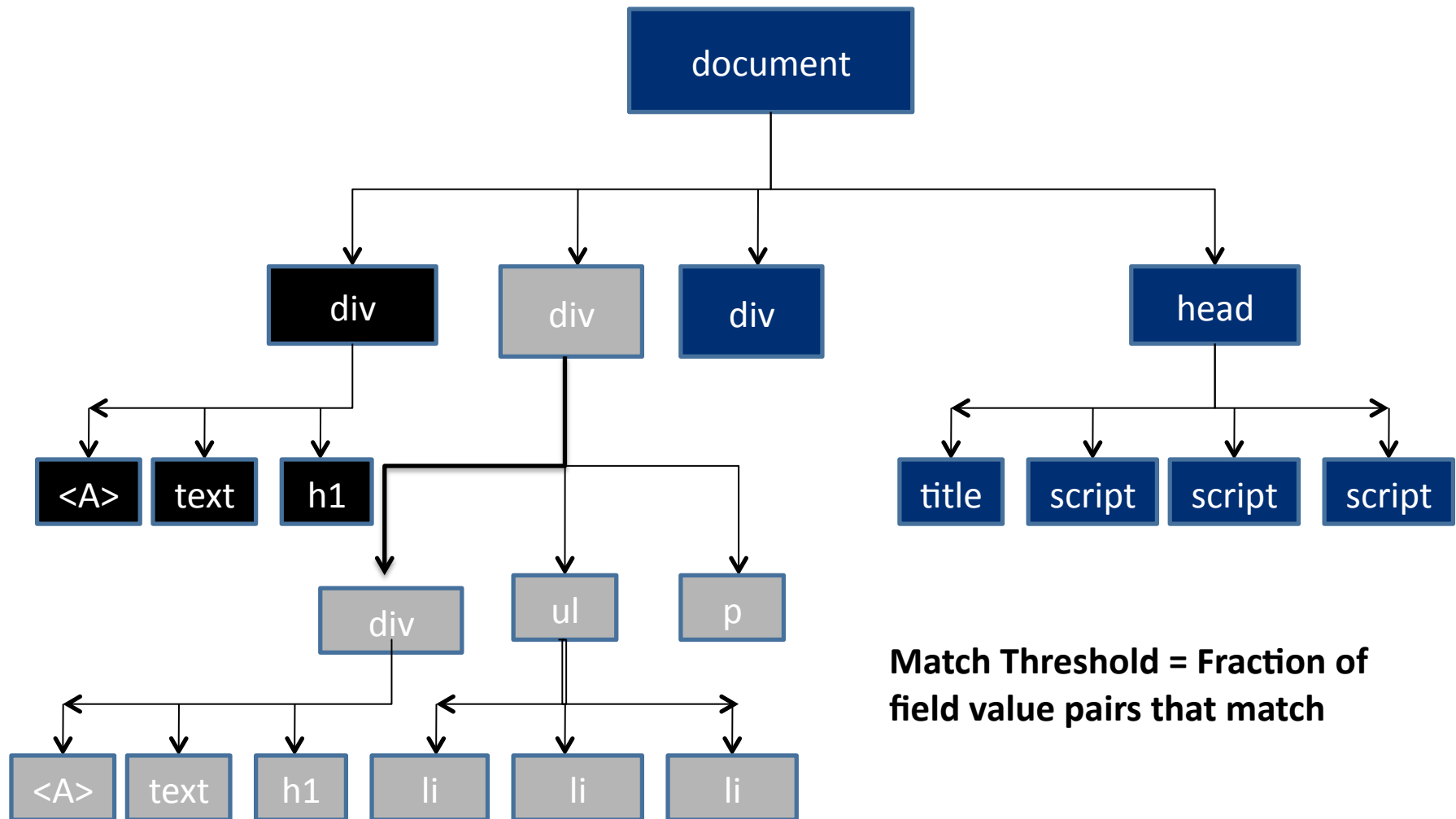
DoDOM: Invariant Extraction



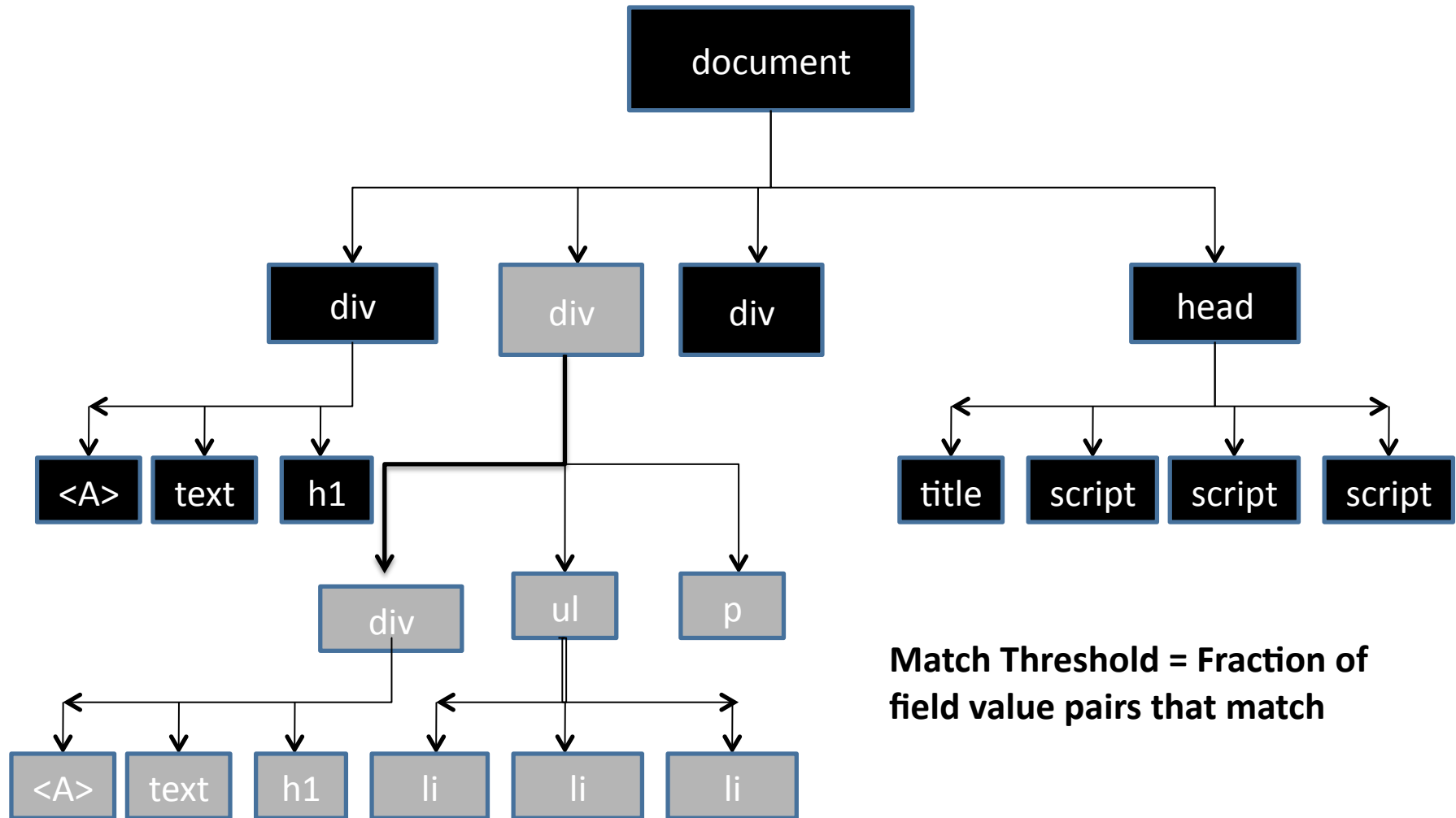
DoDOM: Invariant Extraction



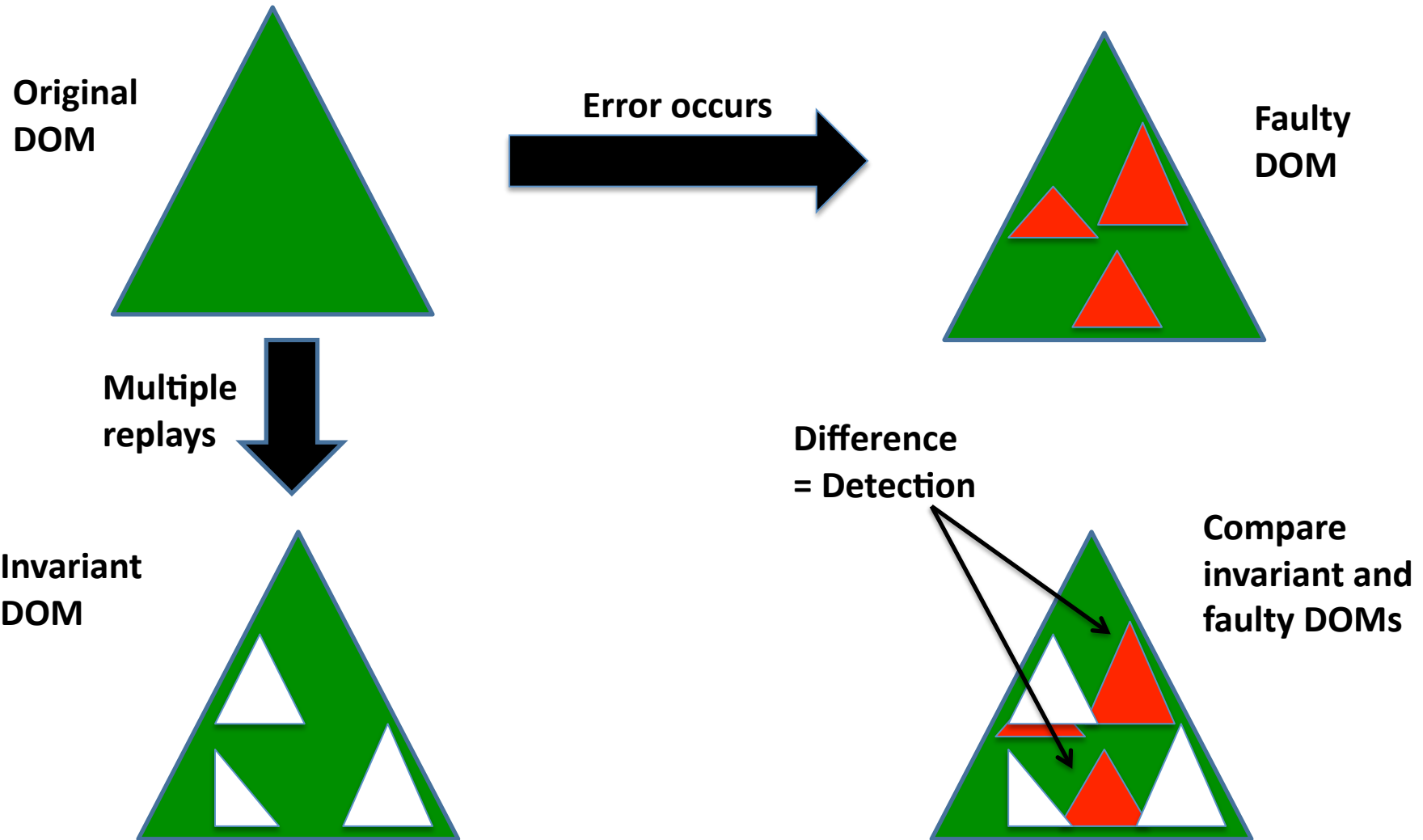
DoDOM: Invariant Extraction



DoDOM: Invariant Extraction



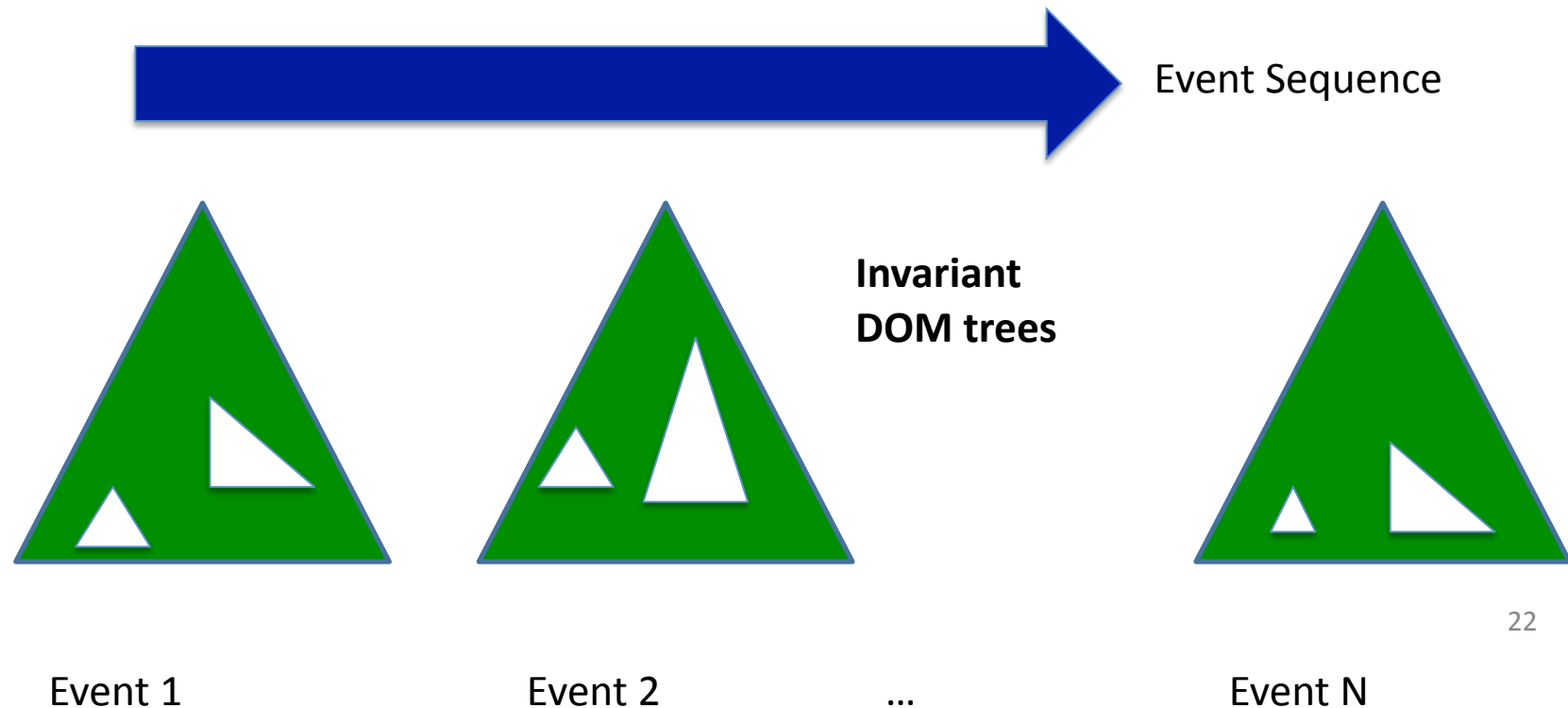
DoDOM: Error Detection



Invariant DOM Sequences

Event Sequence: Sequence of events recorded in a trace

DOM Tree Sequence: Sequence of DOM trees after each event



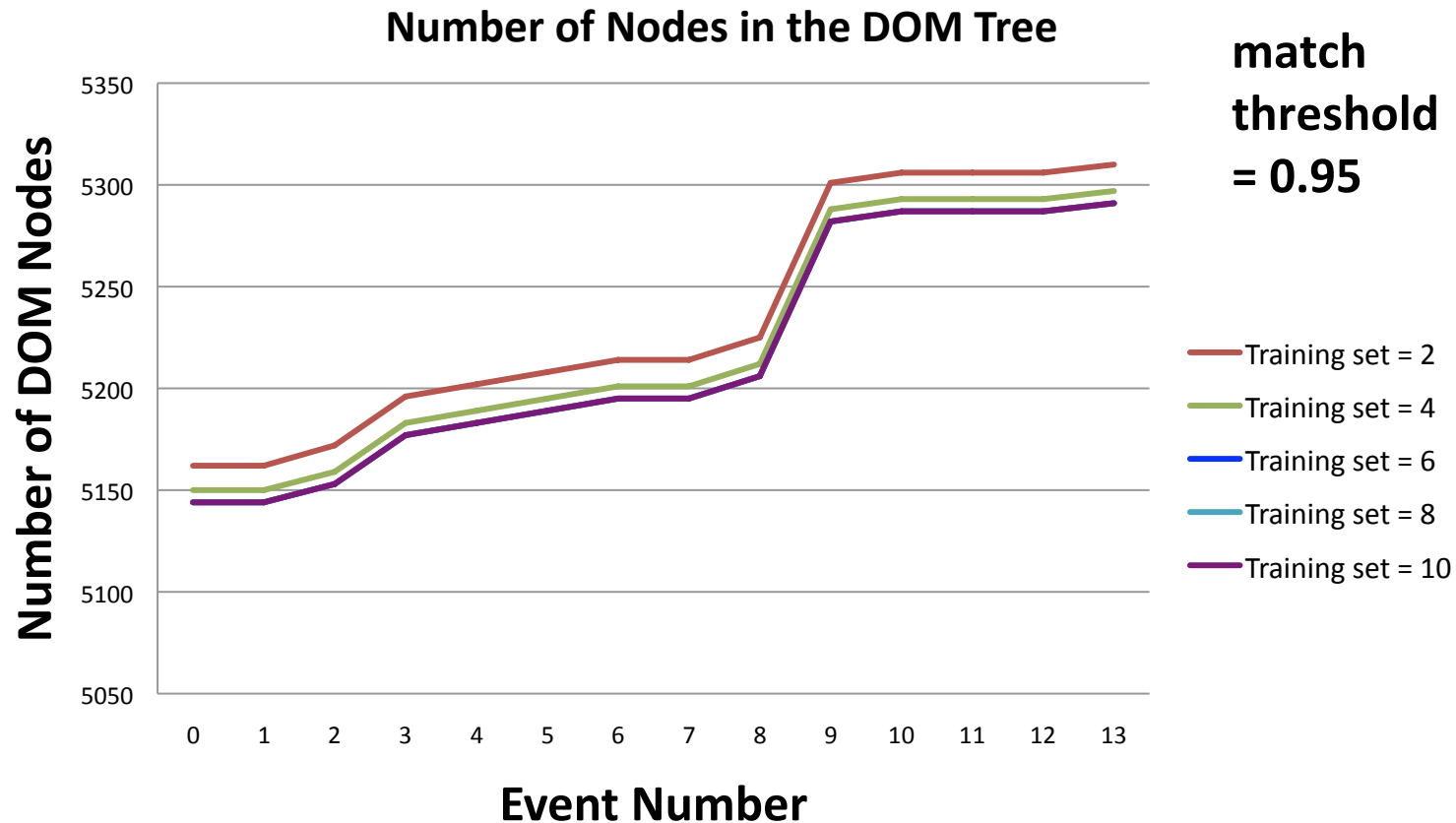
DoDOM: Contributions

- **DOM invariants can be learned dynamically**
 - Automated tool called DoDOM
- **Invariants exist in many Web 2.0 applications**
 - Stabilize within a few executions
 - Incur very few false-positives
- **Invariants are useful for error detection**
 - Both for event errors and domain failures

Experimental Setup

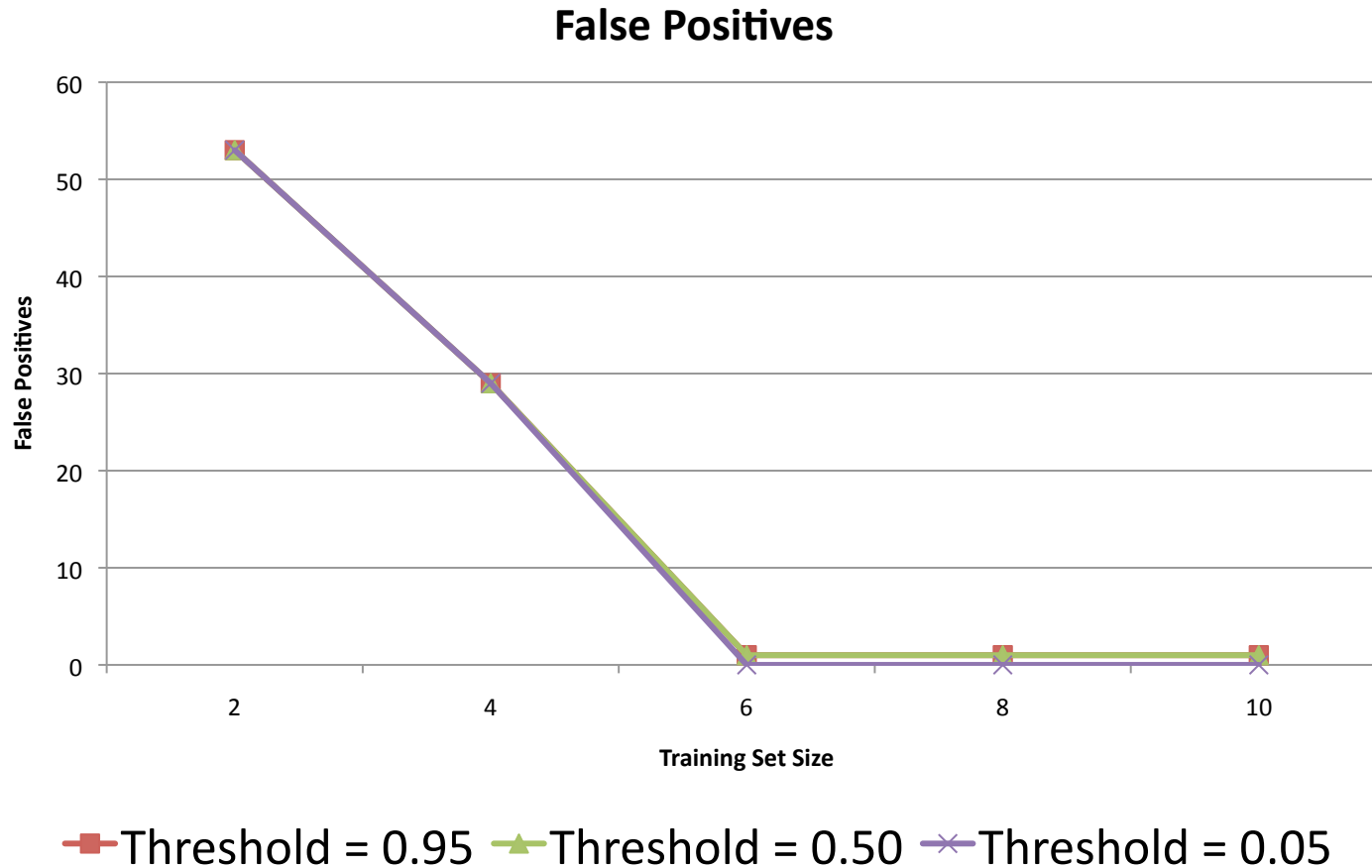
- **Focused mainly on results from Slashdot appln.**
 - Nearly 10000 lines of JS code in the application
 - User interaction by reading and expanding comments
 - Gathered a trace of 13 events with DoDOM
- **Did a replay of the trace 58 times (with DoDOM)**
 - Analyzed the DOM trees and extracted invariant DOM
 - Varied training set size from 1 to 10 executions
 - Chose match thresholds of 0.05, 0.50 and 0.95

Results-1 (Invariants)



**Convergence of DOM-trees with a training set size of 6 (~10%)
Invariant DOM tree contains about 99% of original DOM tree nodes**

Results - 2 (False-positives)



False positives drop sharply after 6 executions (10% of total)

DoDOM: Contributions

- **DOM invariants can be learned dynamically**
 - Automated tool called DoDOM
- **Invariants exist in many Web 2.0 applications**
 - Stabilize within a few executions
 - Incur very few false-positives
- **Invariants are useful for error detection**
 - Both for event errors and domain failures

Fault Injection Experiments

Event Errors

- Emulated by dropping events from the trace one at a time, during replays

Trace



Fault-injected Trace



Domain Failures

- Blocked domains one at a time during replays
 - All scripts from domain blocked during replays

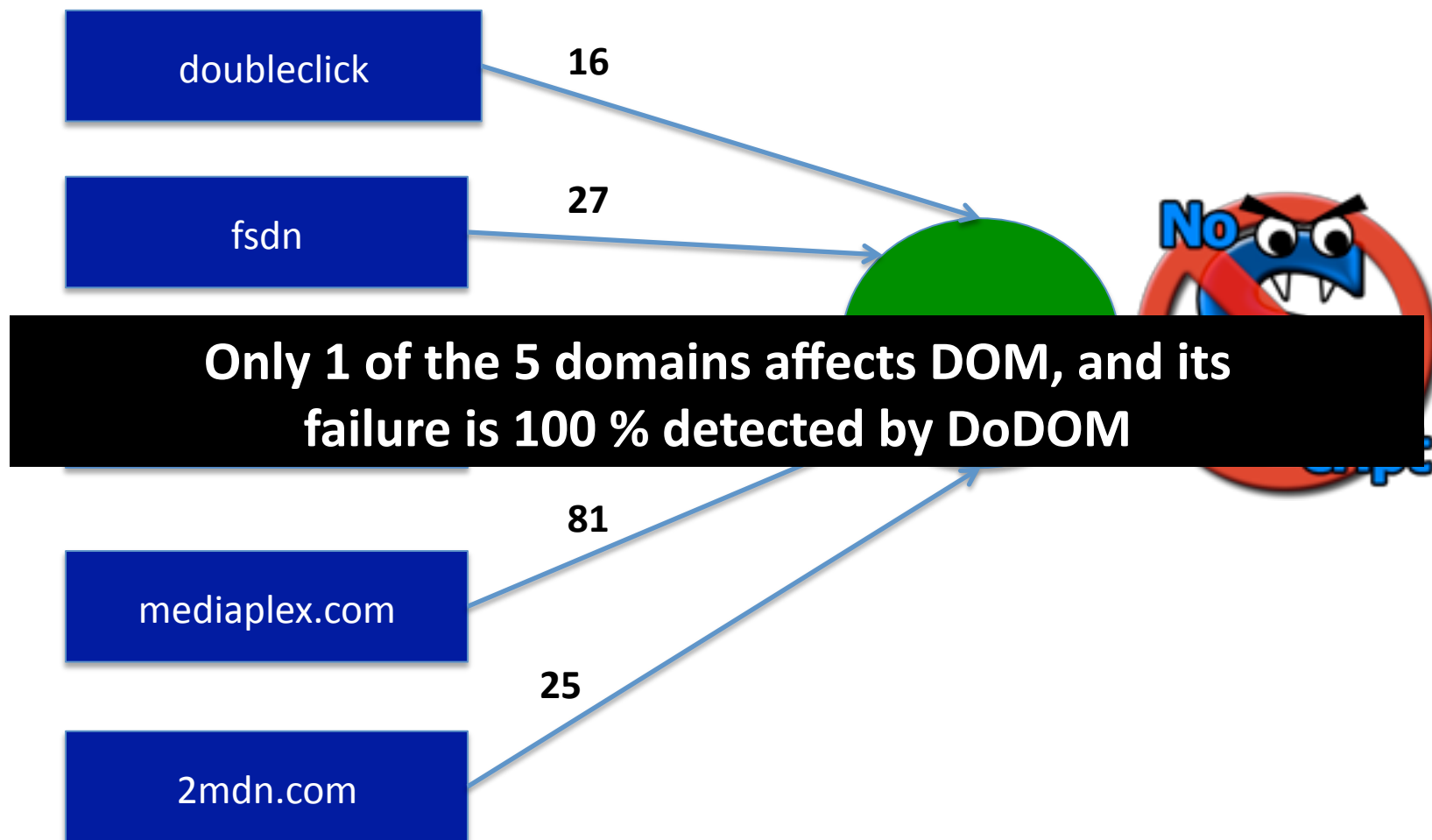


Results - 3 (Event Errors)

Event No	Affects DOM ?	Injected	Detected
1	No	5	0
2	Yes	5	5
3	No	5	0
4	No	5	0
5	No	5	0
6	Yes	5	5
7	No	5	0
8	Yes	5	5
9	No	5	0
10	Yes	5	5
11	No	5	0
12	No	5	0
13	Yes	5	5

100 % detection for event failures whose handlers affect the DOM

Results – 4 (Domain Failures)



DoDOM: Contributions

- **DOM invariants can be learned dynamically**
 - Automated tool called DoDOM
- **Invariants exist in many Web 2.0 applications**
 - Stabilize within a few executions
 - Incur very few false-positives
- **Invariants are useful for error detection**
 - Both for event errors and domain failures

Other Websites

Training Set Size	JavaPetStore	CNN.com
2	397	2413
4	397	2408
6	387	2407
8	387	2407
10	387	2407

Training set size of 6 is sufficient for deriving invariants

Related Work

- **Regression testing of Web 1.0 Applications**
 - [Sprenkle'05], [Dobolyi'09]
- **Automated testing of Web 2.0 Applications**
 - [Marchetto'08], ATUSA [Mesbah'09]
- **Dynamic invariant detection in programs**
 - DAIKON [Ernst'01], HeapMD[Chilimbi'06]

Summary

- **Web 2.0 applications are error prone**
 - Multiple scripts interact in ad-hoc ways
- **DOM invariants for robustness testing**
 - **DoDOM** tool to dynamically learn the invariants
 - Real web applications exhibit invariants
 - High coverage for errors that impact DOM
- **Future Work**
 - Extension to security attacks
 - Extension to richer fault models

DoDOM: Implementation

