

Towards Improving the Reliability of JavaScript-based Web 2.0 Applications



Karthik Pattabiraman¹

Frolin Ocariza Jr.¹

Benjamin Zorn²

¹ University of British Columbia (UBC), ²Microsoft Research (MSR)

Web 2.0 Application: Amazon.com



Third

Web 2.0 applications allow rich UI functionality within a single web page

Web 2.0 Application: JavaScript

A screenshot of a web browser window. The address bar shows a URL from Amazon's image service. The browser's developer tools are open, displaying a large block of JavaScript code. The code is a function named 'makeVisible' on the 'S9MultiPackLayout.prototype' object. It calculates the number of visible items, their width, and their starting column. It then iterates through the item children, setting their 'display' and 'width' styles, and updating the 'src' attribute of their 'itemImages' if it's not already set. The code is syntax-highlighted with blue for keywords, red for strings, and black for other identifiers. The browser interface includes a 'Reload' button in the top right and 'Format code', 'Execute JS', and 'Error console' buttons at the bottom of the code editor.

```
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";
```

Significant amount of JavaScript code executing in the browser

Web 2.0 Application: Amazon.com



Web Apps experience errors, yet they continue to execute !

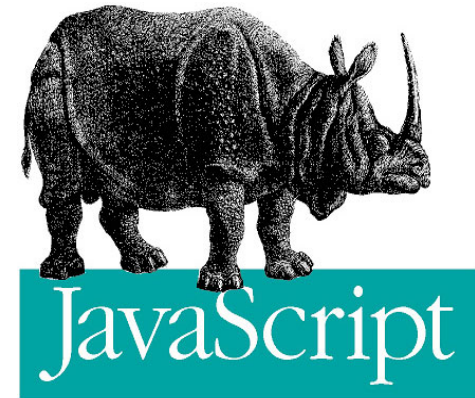
Web 2.0 Applications: Problems



Multiple Parties



Loose semantics



JavaScript



JS had to “look like Java” only less so, be Java’s dumb kid brother or boy-hostage sidekick. Plus, I had to be done **in ten days** or something worse than JS would have happened

– Brendan Eich (Inventor of JavaScript)

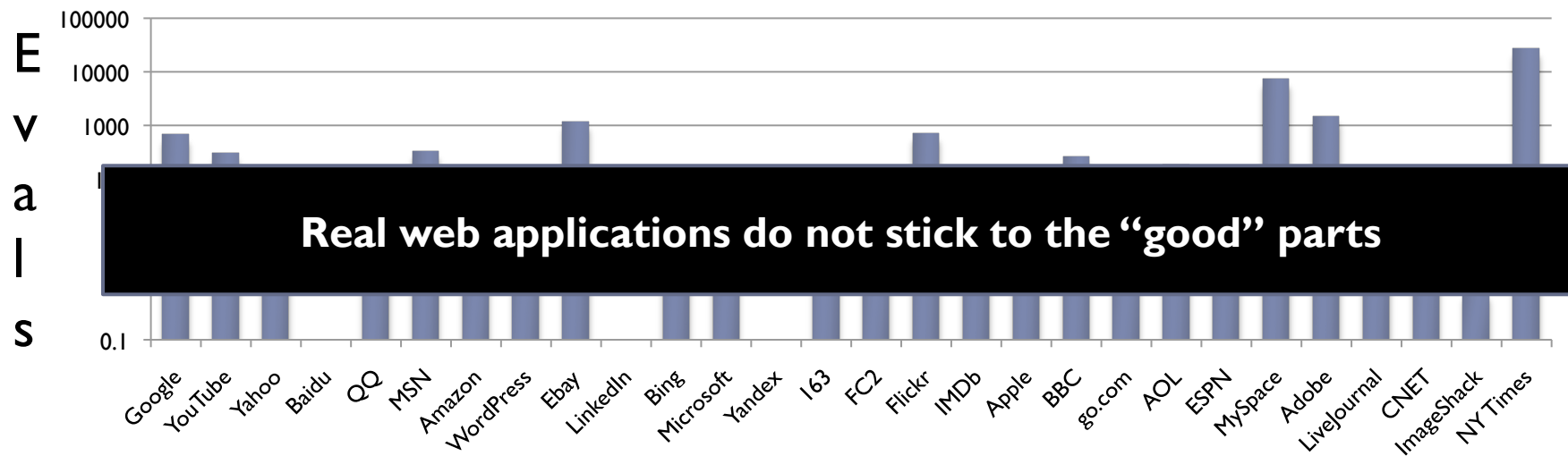
JavaScript: “Good” or “Evil” ?



Versus



Eval Calls (from Richards et al. [PLDI-2010])



Studies of JavaScript Web Applications

Performance and parallelism:

JSMeter [Ratanaworabhan-2010],
[Richards-2009], [Fortuna-2011]

Reliability

?

Security and Privacy:

[Yue-2009], Gatekeeper
[Guarnieri-2009], [Jang-2010]



Goal: Study the reliability of web applications in the “wild”

Why Reliability Matters ?

► Snapshot of iFeng.com: Leading media website in China

an error occurred when processing this directive

[an error occurred while processing this directive]

李克强宣布广州亚残运会开幕

火炬手攀登点燃主火炬|数开幕式十宗“最”
亚残运开幕解密|广州亚残运会开幕式特写

广州亚运会圆满闭幕 高清图

[组图]仁川十分钟：Rain连唱三曲|暖场演出
童谣《月光光》拉开序幕|大郅出任中国旗手

女排上演绝地逆转战胜韩国夺冠

周苏红发威女排逆转|韩国输球再斥裁判丑陋
女排逆转令洪钢哽咽|俞觉敏：我为队员骄傲

[高清]冠军球员搭讪礼仪小姐

裁判引导韩朝摔跤手赛场握手|摔跤精彩瞬间
男篮绝杀伊朗进决赛|朝鲜女足失冠背向升旗

- “铁血女将”黄蕴瑶暂列亚运英雄榜之首
- 中华台北选手罹癌参赛 携奖牌返家无遗憾
- 日本男女足亚运齐称霸 统治亚洲足坛获证
- 霍启刚温文尔雅态度和蔼 与郭晶晶差别大
- 快讯：广州亚运会发生第二起兴奋剂事件
- 阿联首绝杀韩国队 将与日本争男足金牌
- 韩朝射箭选手只关注比赛 不知两国冲突



王治郅闭幕式上挥舞国旗入场

2 of 3

Web Apps' Reliability: Challenges

- ▶ **Lack of specifications**

- ▶ Distinguish correct and incorrect executions
- ▶ No specification of correct behavior



- ▶ **Lack of tool support**

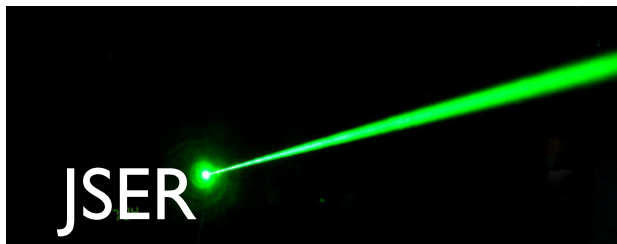
- ▶ Automatically find application errors



Overview: Evaluating JavaScript Reliability

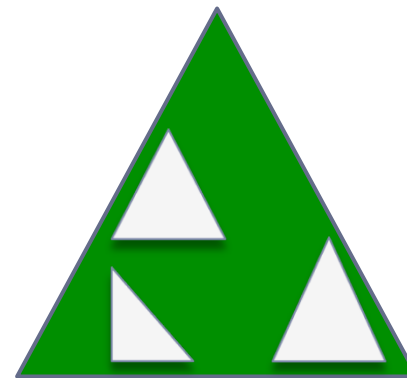
JSER: Error messages

- ▶ Studied error messages printed to the JS console



DoDOM: DOM Invariants

- ▶ Extracted invariant DOMs over multiple executions



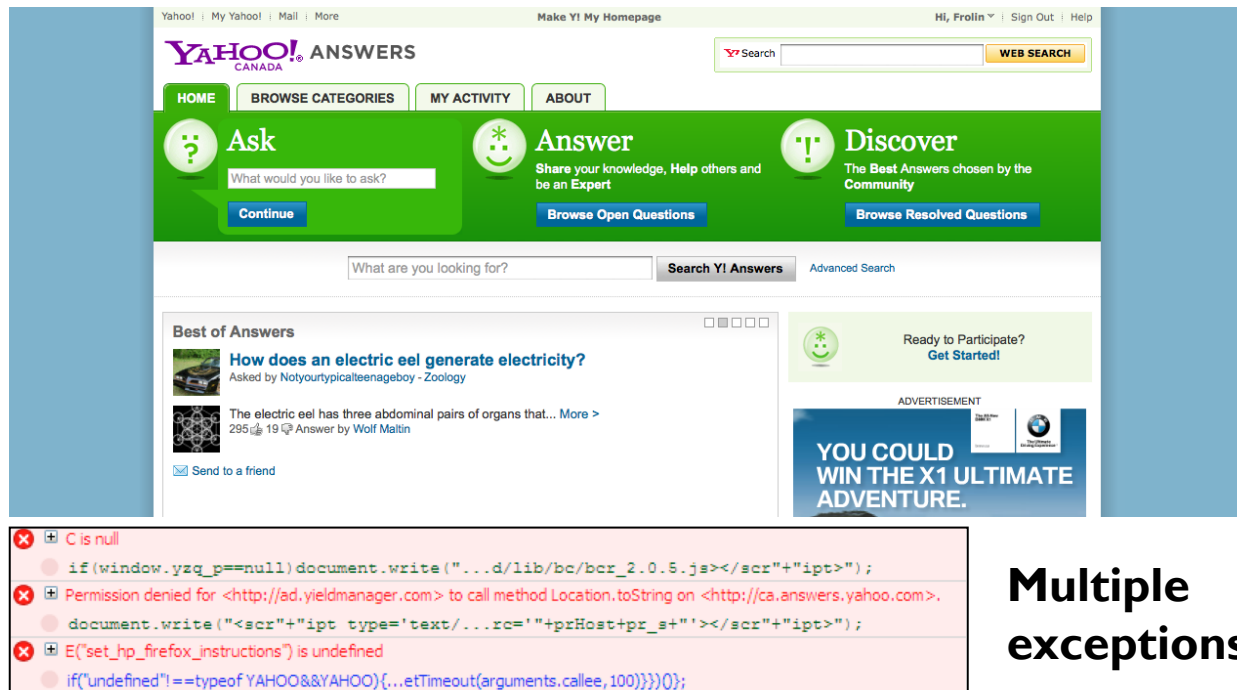
DoDOM

This Talk

- ▶ Motivation and Approach
- ▶ Two approaches for evaluating JS Reliability
 - ▶ Error messages – JSER [Under submission]
 - ▶ DOM Invariants – DoDOM [ISSRE 2010]
- ▶ Future Directions and Conclusions

JSER: JavaScript Error Messages

- ▶ Any exception thrown by JS code is logged to JS console



The screenshot shows the Yahoo! Answers homepage. The page has a green header with navigation links: HOME, BROWSE CATEGORIES, MY ACTIVITY, and ABOUT. Below the header, there are three main sections: 'Ask' (with a question input field and a 'Continue' button), 'Answer' (with a 'Share your knowledge' prompt and a 'Browse Open Questions' button), and 'Discover' (with a 'The Best Answers' prompt and a 'Browse Resolved Questions' button). A search bar is located below these sections. The main content area displays 'Best of Answers' with a featured question: 'How does an electric eel generate electricity?'. To the right, there is an advertisement for 'YOU COULD WIN THE X1 ULTIMATE ADVENTURE'.

Overlaid on the bottom left of the screenshot is a JavaScript error console window showing the following messages:

- C is null**
- if(window.yzq_p==null) document.write("...d/lib/bc/bcr_2.0.5.js"</scr+"ipt>");**
- Permission denied for <http://ad.yieldmanager.com> to call method Location.toString on <http://ca.answers.yahoo.com>.**
- document.write("<scr"+"ipt type='text/...rc='"+prHost+pr_s+"'"</scr"+"ipt>");**
- E("set_hp_firefox_instructions") is undefined**
- if("undefined"!==typeof YAHOO&&YAHOO){...etTimeout(arguments.callee,100)}}0);**

Multiple exceptions

JSER: Why Error Messages ?

- ▶ No false positives unlike static analysis
- ▶ Challenging to analyze JavaScript statically
- ▶ Capture interactions with the DOM



Vs.



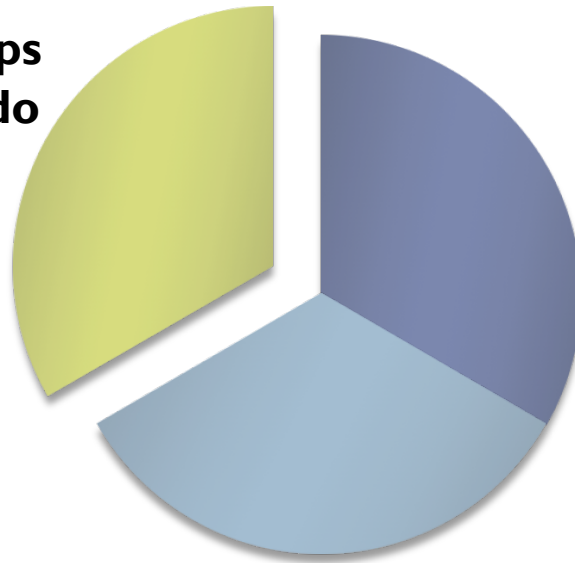
JSER: Tools

- ▶ Chose 50 web applications from the Alexa top 100
- ▶ Created test suites for normal interactions in Selenium
- ▶ Capture JavaScript Errors printed to Firebug console



JSER: Research Questions

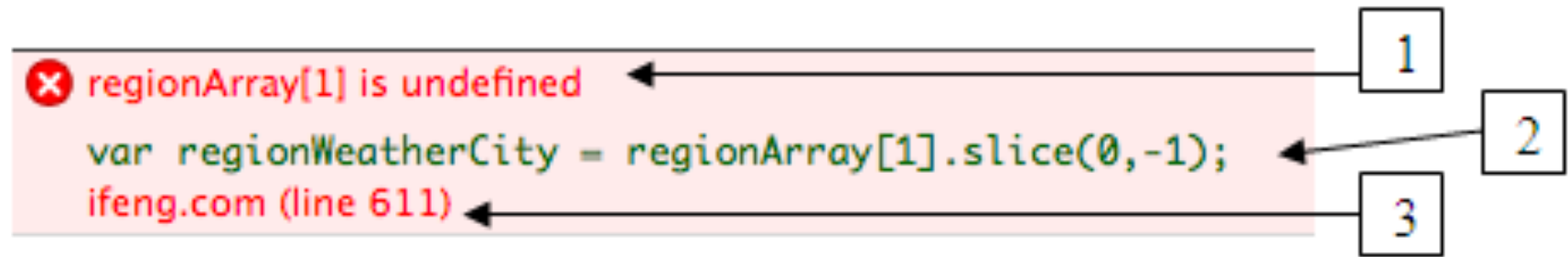
Do errors occur in web apps and if so, what categories do they fall in ?



How do errors correlate with static and dynamic characteristics of the app?

How do errors vary by speed of testing ? Are they all deterministic ?

Errors and their classification: Method



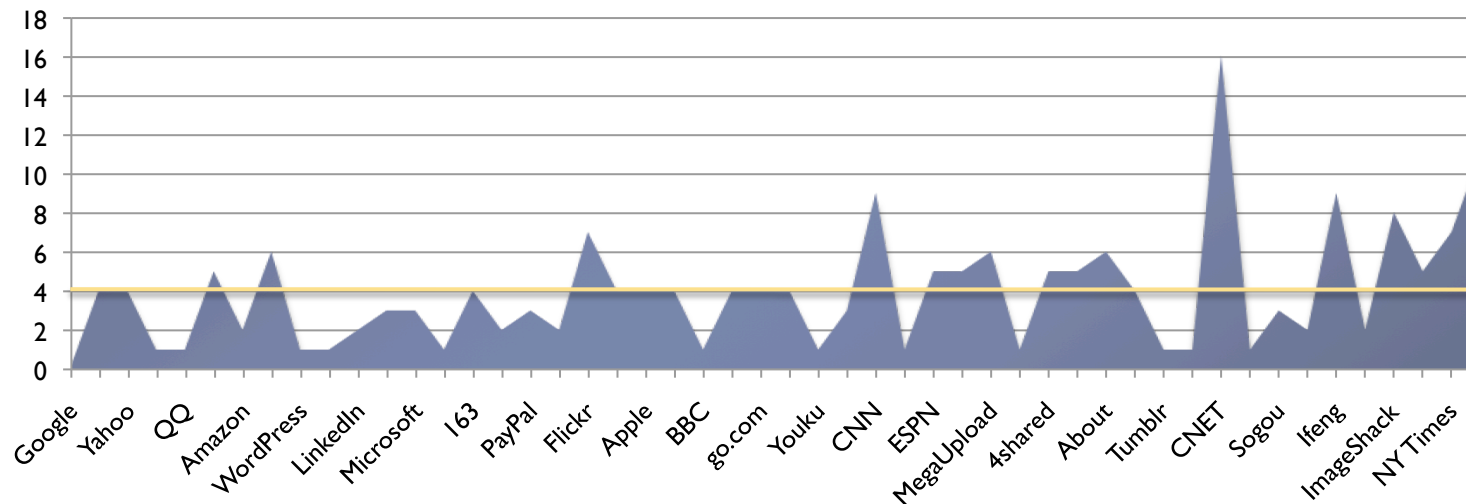
1. Description of error message
2. Line of code corresponding to error
3. Domain number and line number

Two errors are different if any attribute is different

Errors and their classification: Results

- ▶ **Average of 4 distinct error messages for each app**
 - ▶ **Standard dev: 3**
 - ▶ **Max: 16 (Cnet)**
 - ▶ **Min: 0 (Google)**

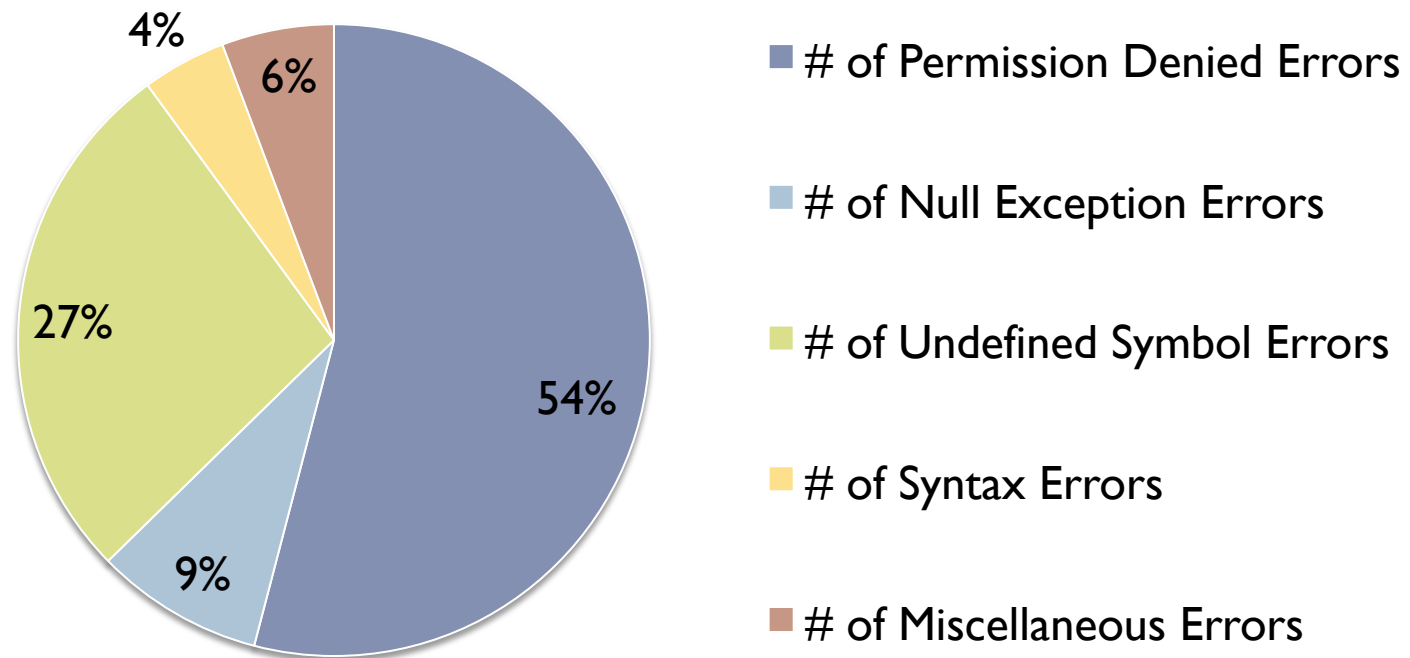
Total Distinct Errors



Errors and their classification: Results

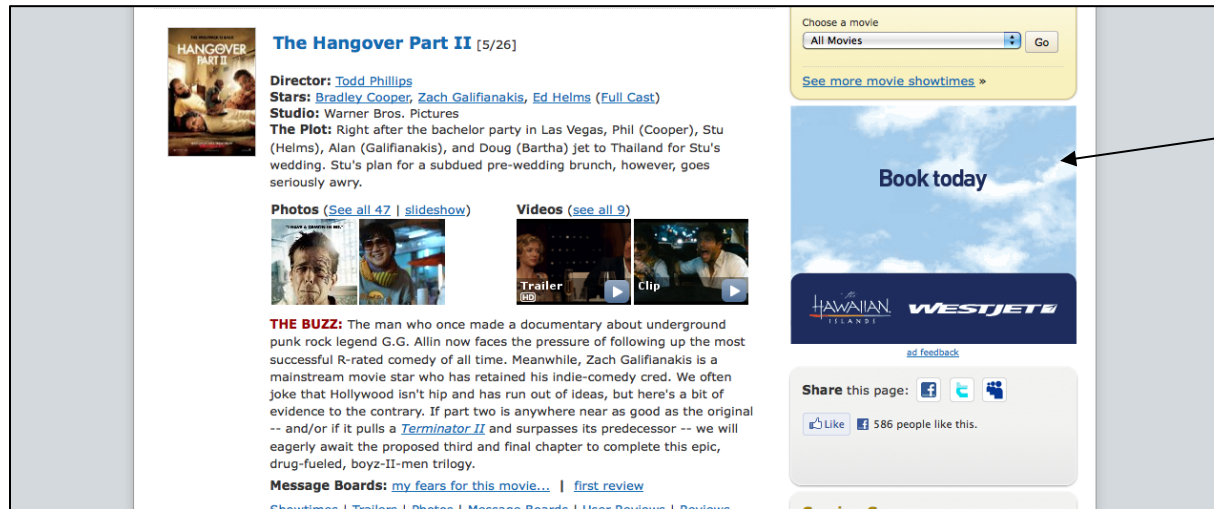
- ▶ 94 % of errors fall into four predominant categories

Distribution of Error Messages



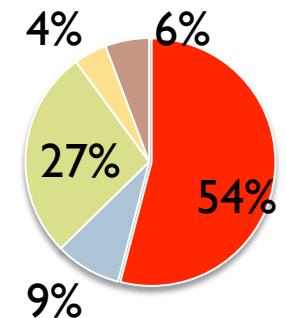
Errors and their classification:

Permission Denied Example



Taken from
imdb.com

advertisement



- Error Message:** Permission denied for <http://view.atdmt.com> to call method `Location.toString` on <http://www.imdb.com>

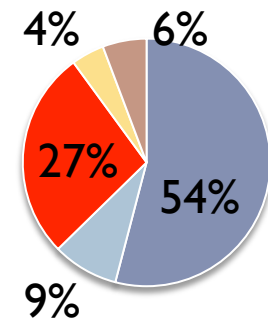
- Explanation:** Triggered by appearance of advertisement. Leads to SOP violation.

Bottom Line: JS errors may appear as a result of code written by others

Errors and their classification: Undefined Symbol Example



Taken from
cnn.com



- Error Message:** `cnn_onMemFBInit()` is undefined

// this probably isn't needed anymore

```
if (CNN_ISMemInit && CNN_IsFBInit) cnn_onMemFBInit();
```

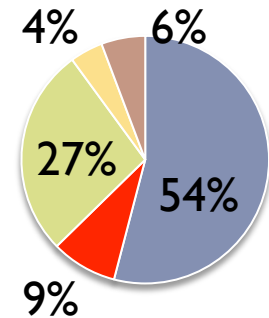
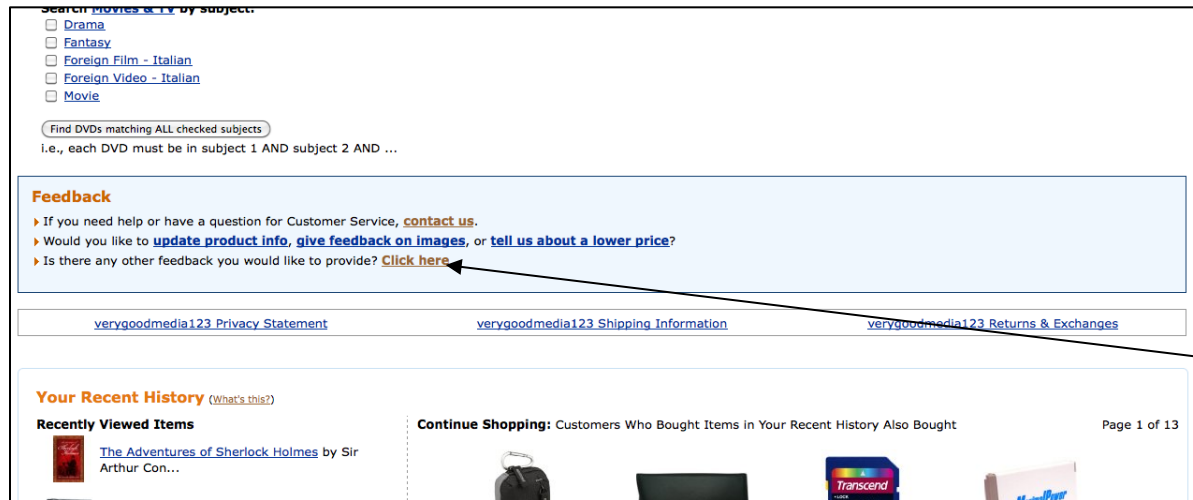
- Explanation:** Both `CNN_IsMemInit` and `CNN_IsFBInit` set to true

- Bottom Line:** JS code is difficult to maintain

Errors and their classification:

Null Exception Example

Taken from
amazon.com



Causes error
on click

- Error Message:** `document.getElementById("inappDiv")` is null

```
document.getElementById("inappDiv").style.display = 'none';
```

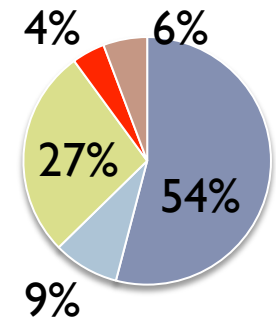
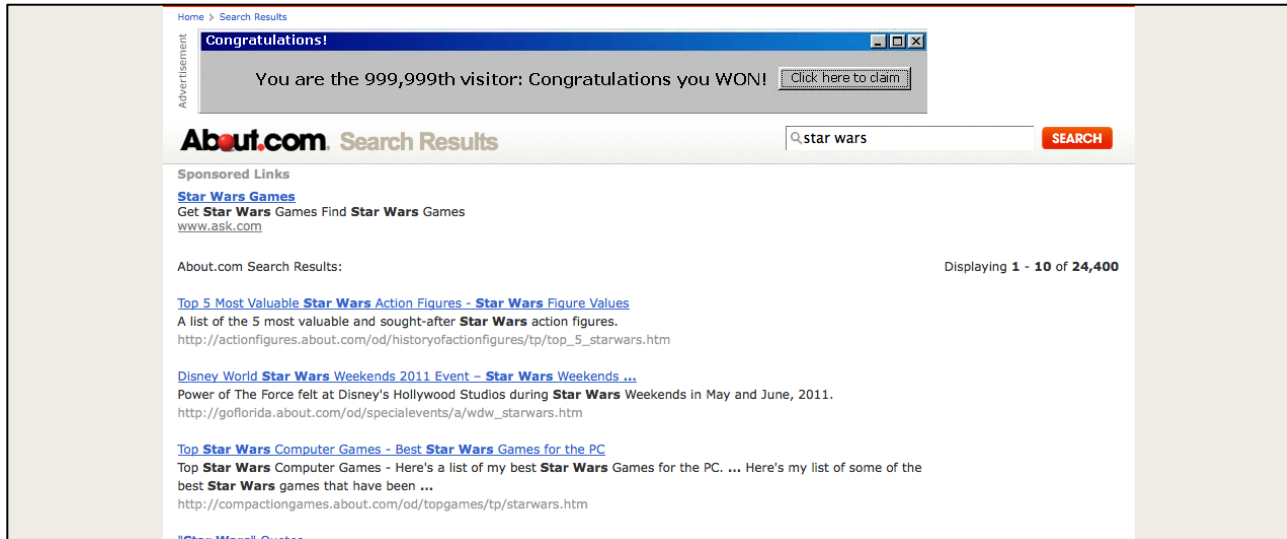
- Explanation:** `inappDiv` was only defined for users who are logged in

- Bottom Line:** JS code may depend on the DOM

Errors and their classification:

Syntax Error Example

Taken from
about.com



- **Error Message:** unterminated string literal

```
zGPU = 'http://movies.about.com/od/onlinemovies  
Movies_Available_on_the_Internet.html' "
```

- **Bottom Line:** JS code is sometimes not well-tested



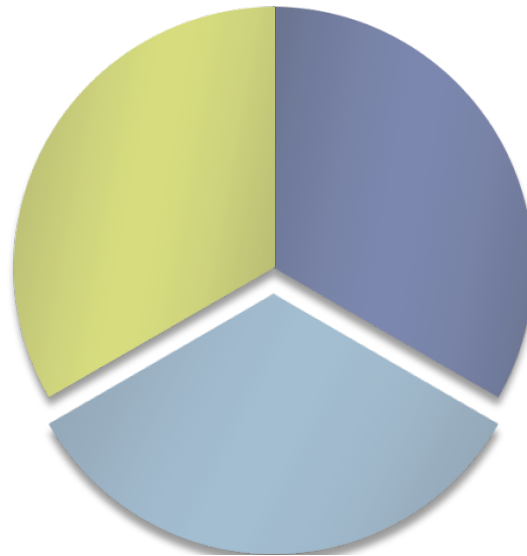
Errors and their classification: Insights

- ▶ **Errors can occur due to third-party code**
 - ▶ Permission denied errors
- ▶ **Errors can occur due to the DOM**
 - ▶ Null exception errors
- ▶ **JavaScript code can be difficult to maintain**
 - ▶ Undefined symbol errors
- ▶ **JavaScript code is sometimes not well-tested**
 - ▶ Syntax errors



Research Questions

Do errors occur in web apps
and if so, what categories do
they fall in ?

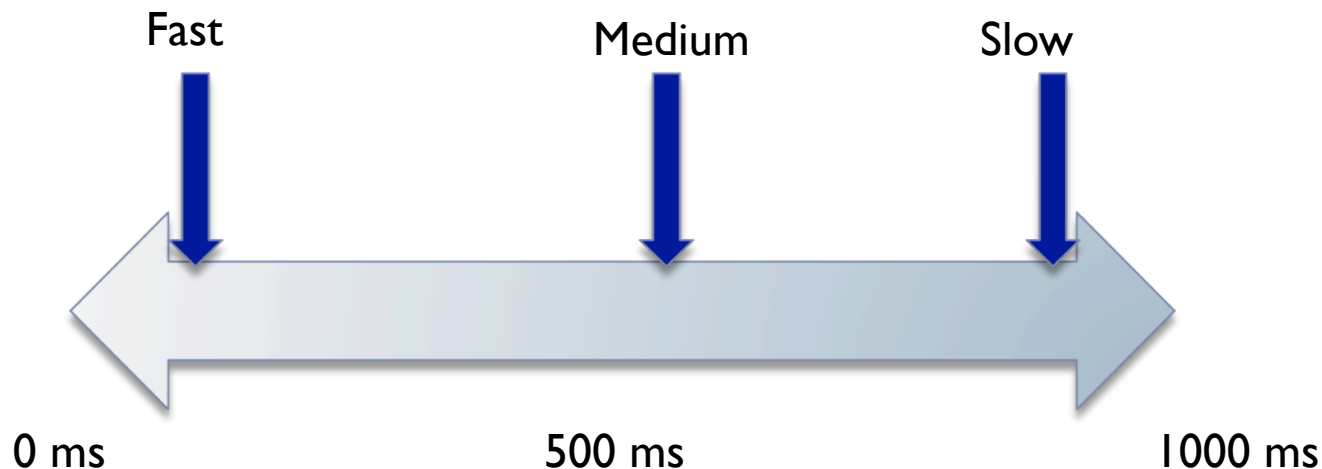


How do errors correlate
with static and dynamic
characteristics of the app?

**How do errors vary
by speed of testing ?
Are they all
deterministic ?**

Effect of Testing Speed: Method

- ▶ Varied testing speed for replaying events in Selenium
- ▶ Performed three executions in each testing speed

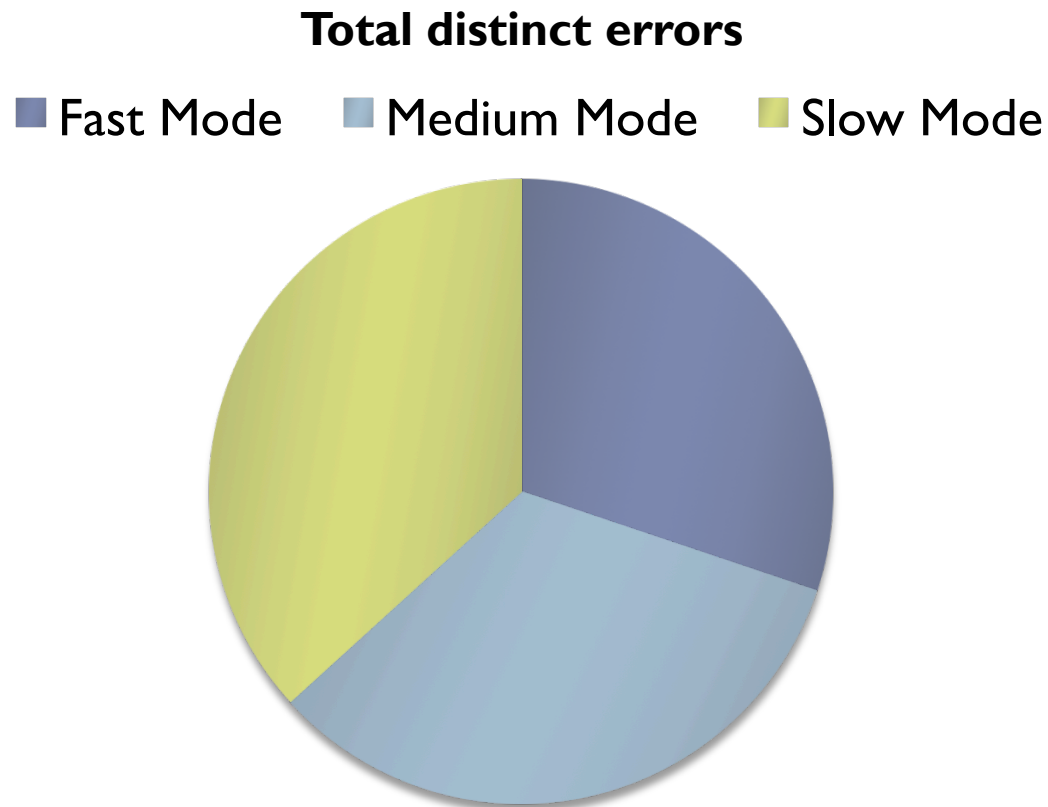


Effect of Testing Speed: Example

Error Message (shortened)	F 1	F 2	F 3	M 1	M 2	M 3	S 1	S 2	S 3
Permission Denied for view.atdmt.com to call <fname> on marquee.blogs.cnn.com	4	4	4	1	3	3	2	2	3
Permission Denied for view.atdmt.com to call <fname> on www.cnn.com	20	17	20	22	22	16	25	20	16
Permission Denied for ad.doubleclick.net to call <fname> on www.cnn.com	8	16	13	3	6	4	7	12	11
targetWindow.cnnad showAd is not a function	0	2	5	0	0	0	0	0	0
window.parent.CSIManager is un- defined	0	0	0	0	0	0	1	1	0

Effect of Testing Speed: Frequencies

- ▶ All three testing modes expose similar error frequencies
 - ▶ Slow mode exposes the most number of errors !

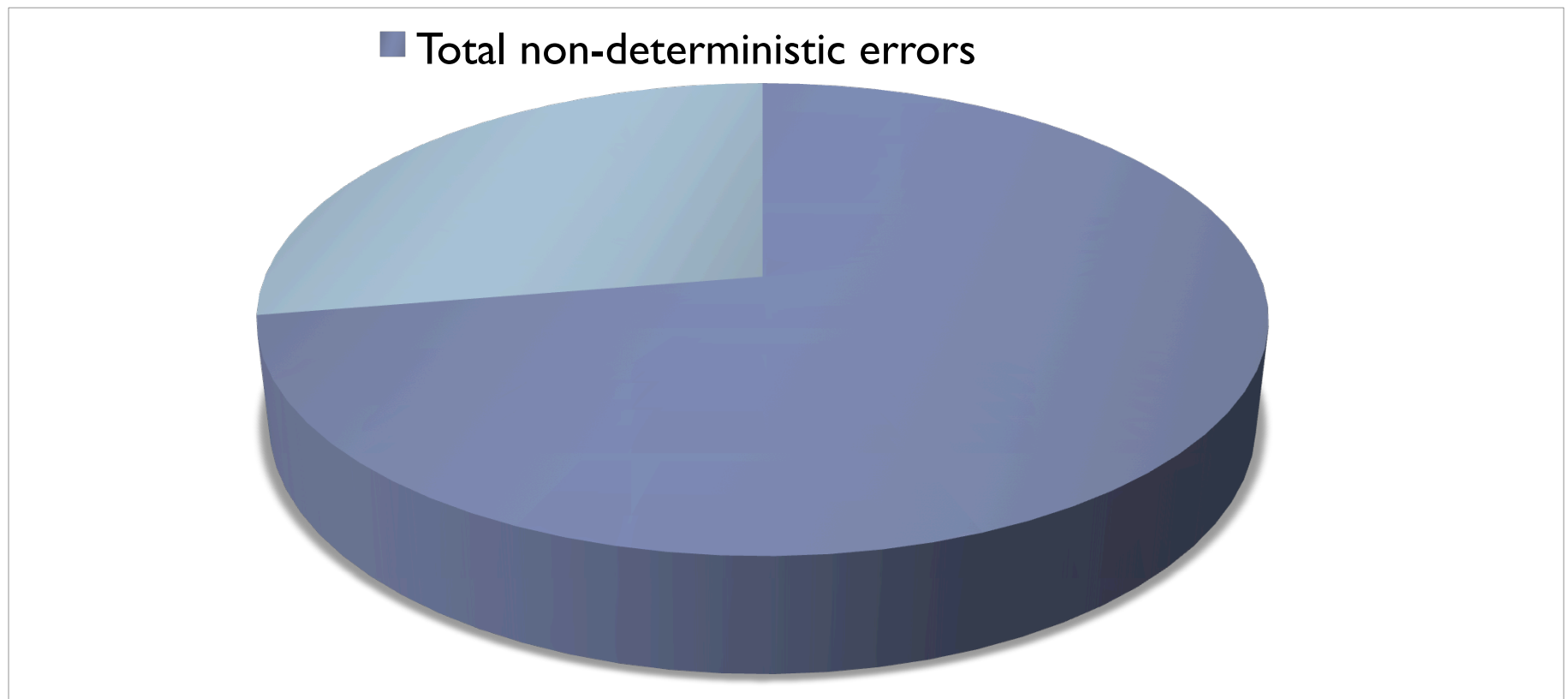


Effect of Testing Speed: Example

Error Message (shortened)	F 1	F 2	F 3	M 1	M 2	M 3	S 1	S 2	S 3
Permission Denied for view.atdmt.com to call <fname> on marquee.blogs.cnn.com	4	4	4	1	3	3	2	2	3
Permission Denied for view.atdmt.com to call <fname> on www.cnn.com	20	17	20	22	22	16	25	20	16
Permission Denied for ad.doubleclick.net to call <fname> on www.cnn.com	8	16	13	3	6	4	7	12	11
targetWindow.cnnad showAd is not a function	0	2	5	0	0	0	0	0	0
window.parent.CSIManager is un- defined	0	0	0	0	0	0	1	1	0

Effect of Testing Speed: Non-Determinism

- ▶ More than 70% of distinct errors are non-deterministic



Effect of Testing Speed: Insights

- ▶ **Different modes expose different no. of errors**
- ▶ **Different reasons for errors**
 - ▶ **Fast/Medium Modes:** Due to rapid page transitions
 - ▶ **Slow Mode:** Due to errors in advertisements
- ▶ **More than 70% of errors are non-deterministic**



Research Questions

Do errors occur in web apps and if so, what categories do they fall in ?



How do errors correlate with static and dynamic characteristics of the app?

How do errors vary by speed of testing ? Are they all deterministic ?

Static/Dynamic Correlations

Static Characteristics

Measured using Phoenix & Firebug plugins

- Alexa Rank
- Bytes of JavaScript code
- Number of domains
- Domains containing JS

Dynamic Characteristics

From Richards et al. [PLDI – 2010]

- ▶ Number of called functions
- ▶ Number of eval calls
- ▶ Properties deleted
- ▶ Object inheritance overridings

Static/Dynamic Correlations: Summary

Static Characteristics

Measured using Phoenix & Firebug plugins

- **Alexa Rank**
- Bytes of JavaScript code
- **Number of domains**
- **Domains containing JS**

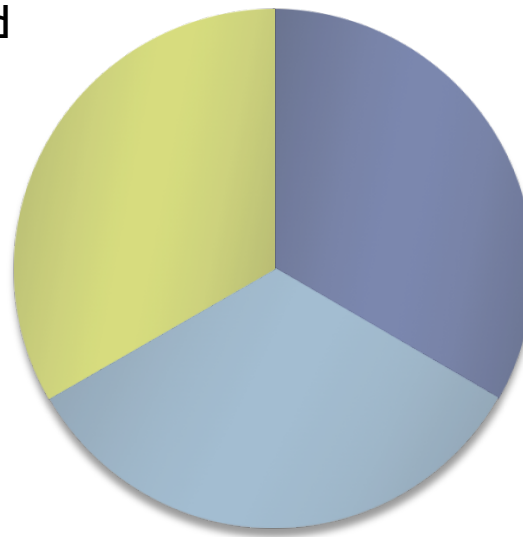
Dynamic Characteristics

From Richards et al. [PLDI – 2010]

- ▶ Number of called functions
- ▶ Number of eval calls
- ▶ Properties deleted
- ▶ Object inheritance overridings

Research Questions: Answers

Average of four errors in each app. Errors fall into four well-defined categories



Closest related with one of the domains, in one of domains with JS, Alexa rank of the app?

Errors vary by speed of testing. Majority of errors are non-deterministic?

Implications of the Results

▶ **Programmers**

- ▶ Need to make code robust against other code/scripts
- ▶ Make sure interactions with DOM are checked

▶ **Testers**

- ▶ Perform integration testing to see effects of ads
- ▶ Need to test at multiple testing speeds, multiple times

▶ **Static analysis tool developers**

- ▶ Target most common classes of errors
- ▶ Need to model the DOM in the analysis

This Talk

- ▶ Motivation and Approach
- ▶ Two approaches for evaluating JS Reliability
 - ▶ Error messages – JSER [Under submission]
 - ▶ DOM invariants – DoDOM [ISSRE 2010]
- ▶ Future Directions and Conclusions

Motivation

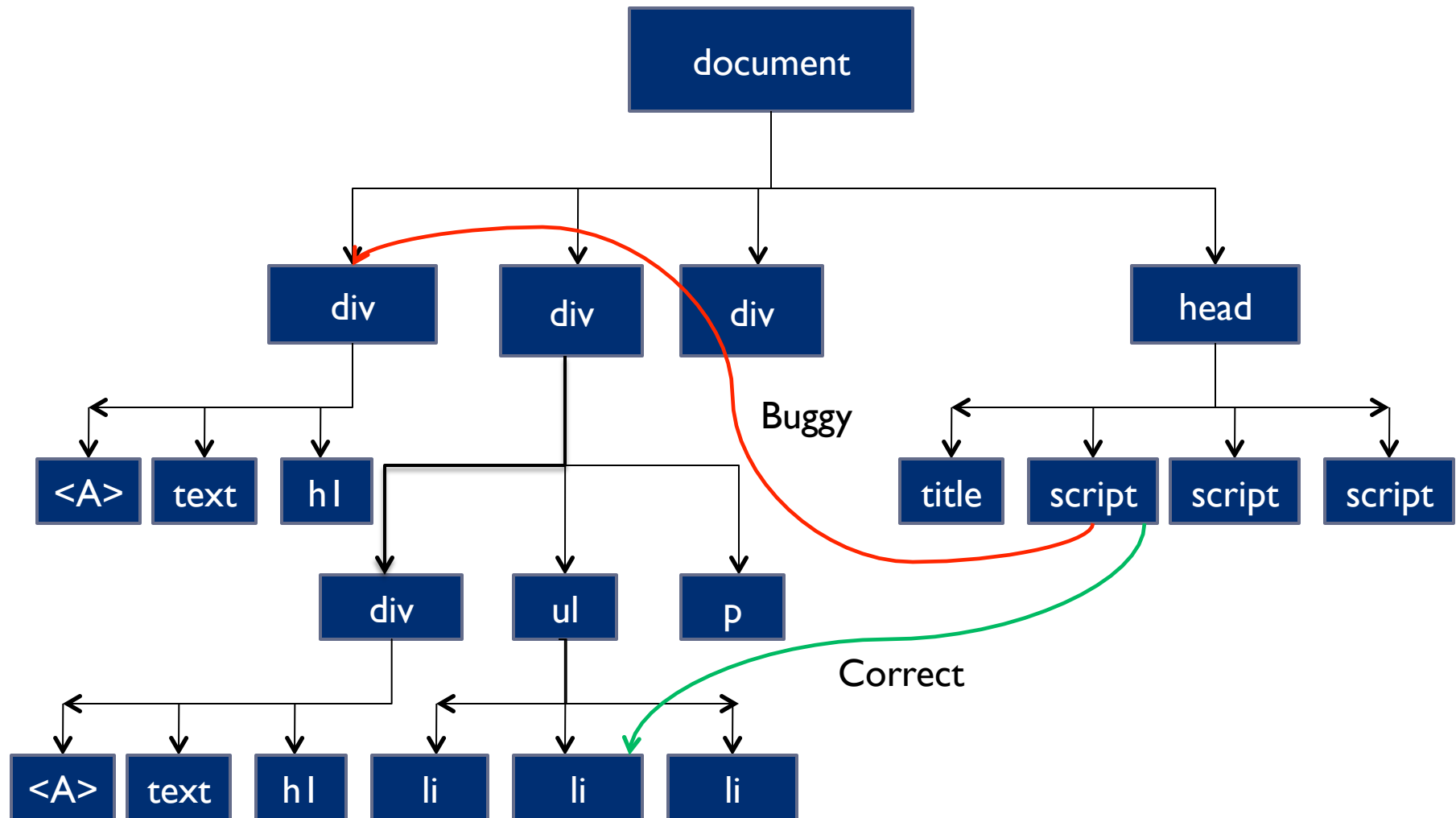
- ▶ **Error Messages are limited in representativeness**
 - ▶ Not all errors result in error messages
- ▶ **Goal: Detect errors that do not result in messages**
 - ▶ Do not require programmer intervention
- ▶ **Challenges**
 - ▶ Non-determinism in application



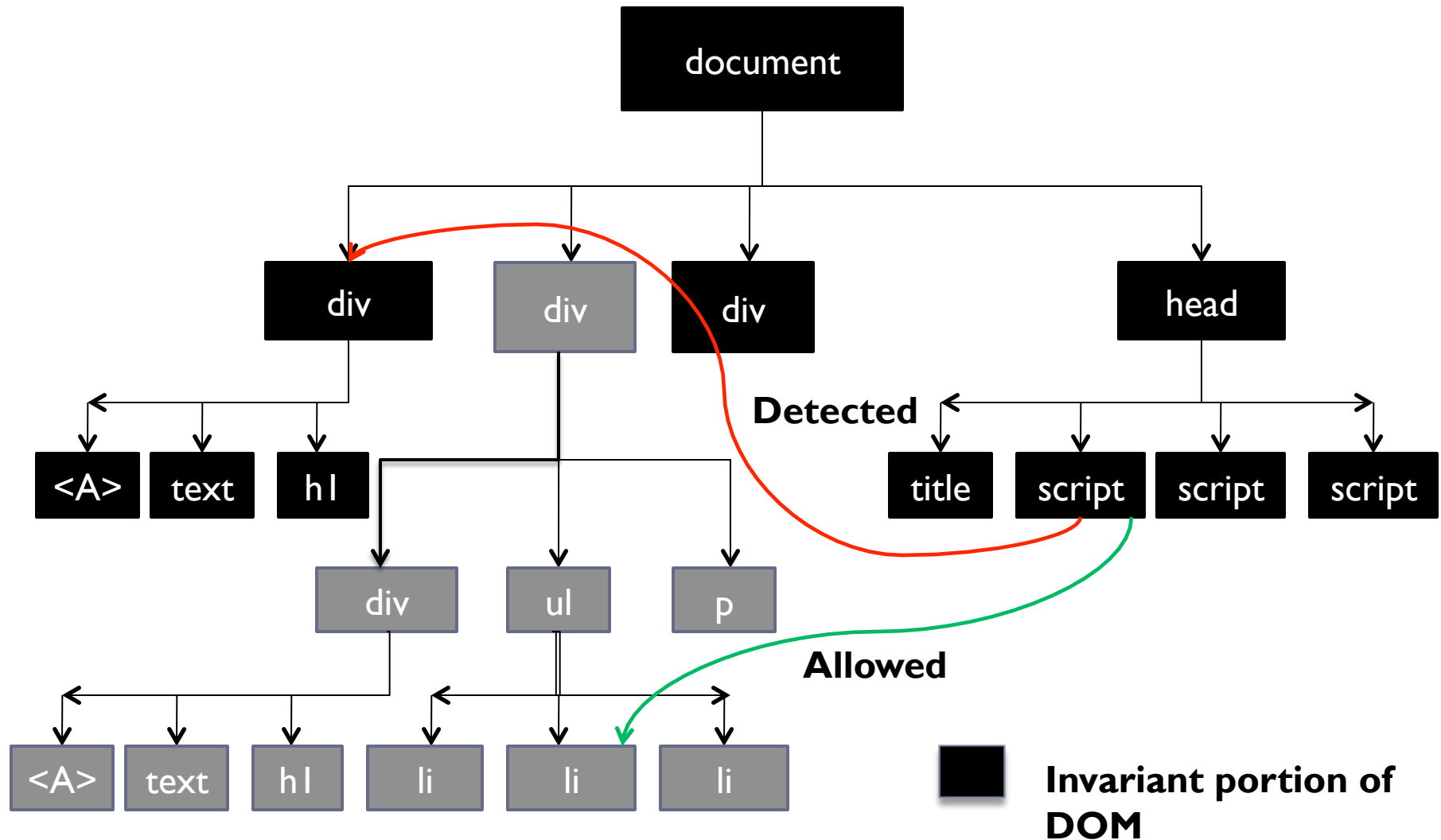
Example: HTML

```
<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



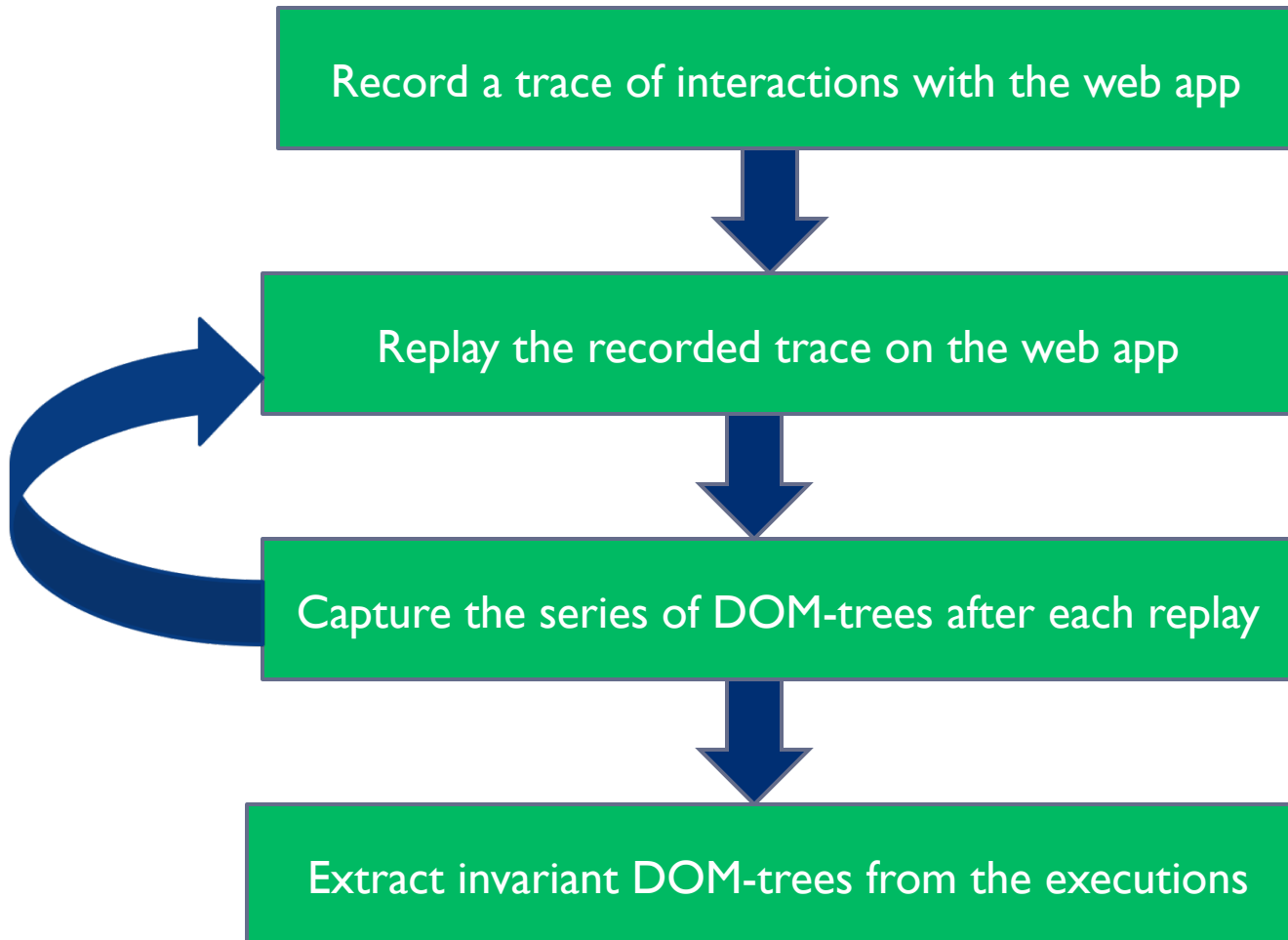
Example: Invariant DOM



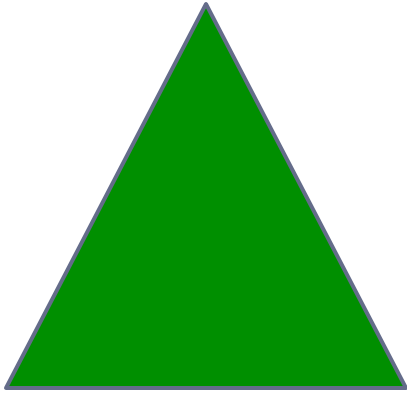
DoDOM: Contributions

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

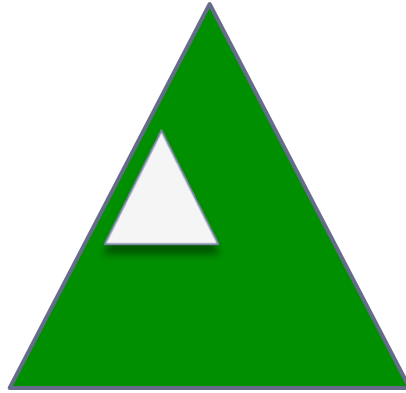
DoDOM: Approach



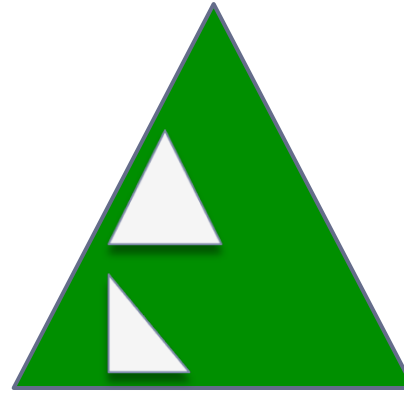
DoDOM: Invariant Extraction



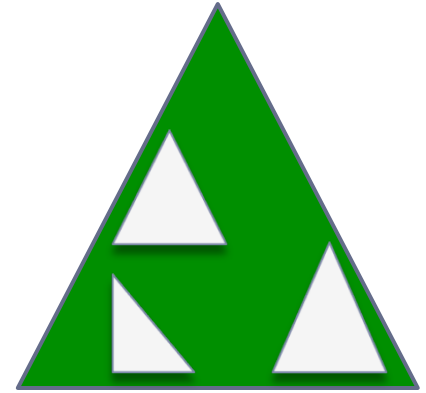
Execution 1



Execution 2



Execution 3

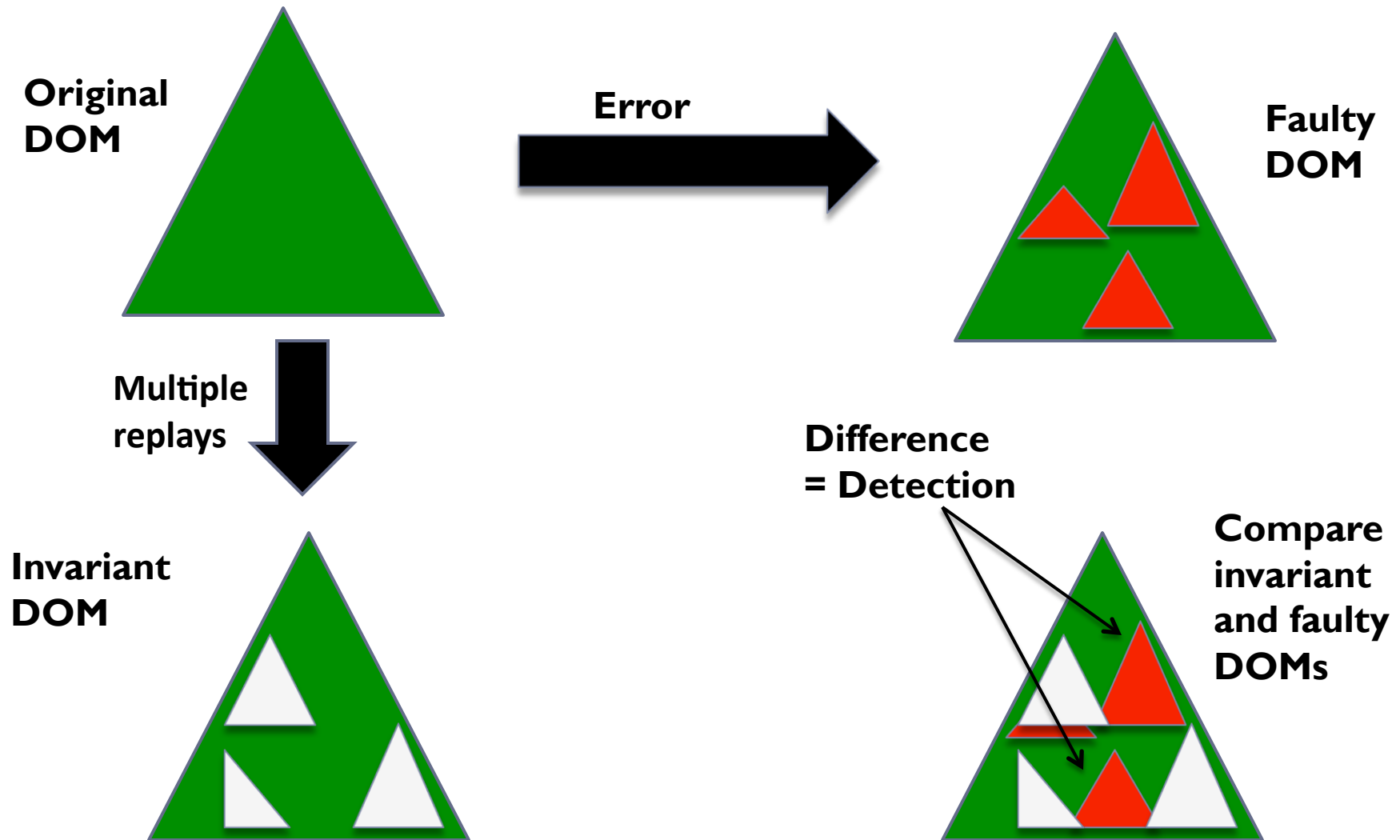


Execution 4



Invariant DOM

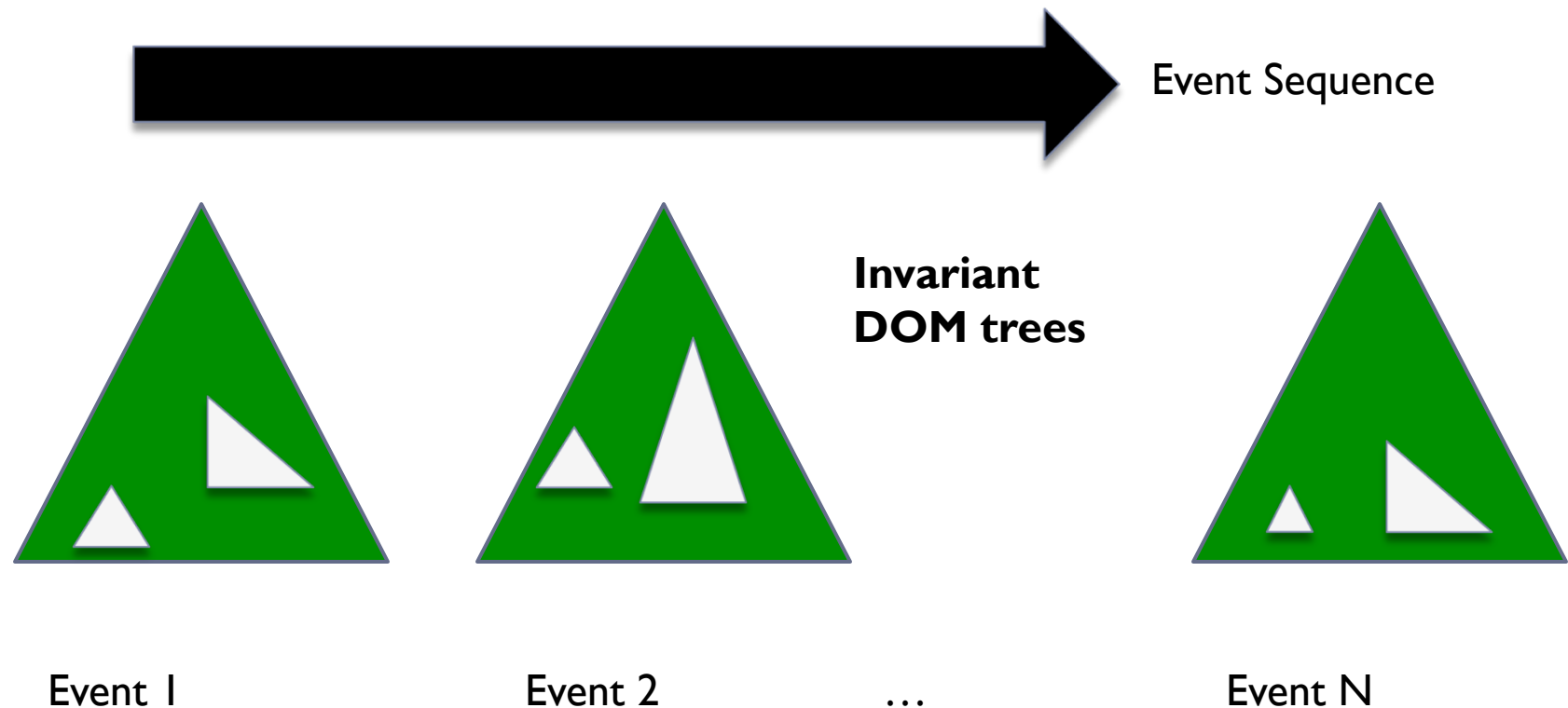
DoDOM: Error Detection



DoDOM: Invariant DOM Sequences

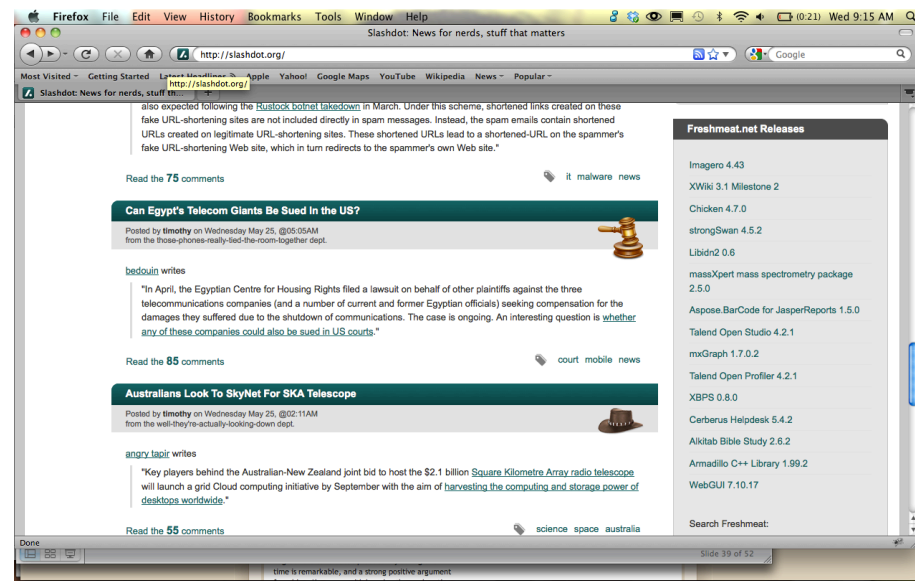
Event Sequence: Sequence of events recorded in a trace

DOM Tree Sequence: Sequence of DOM trees after each event

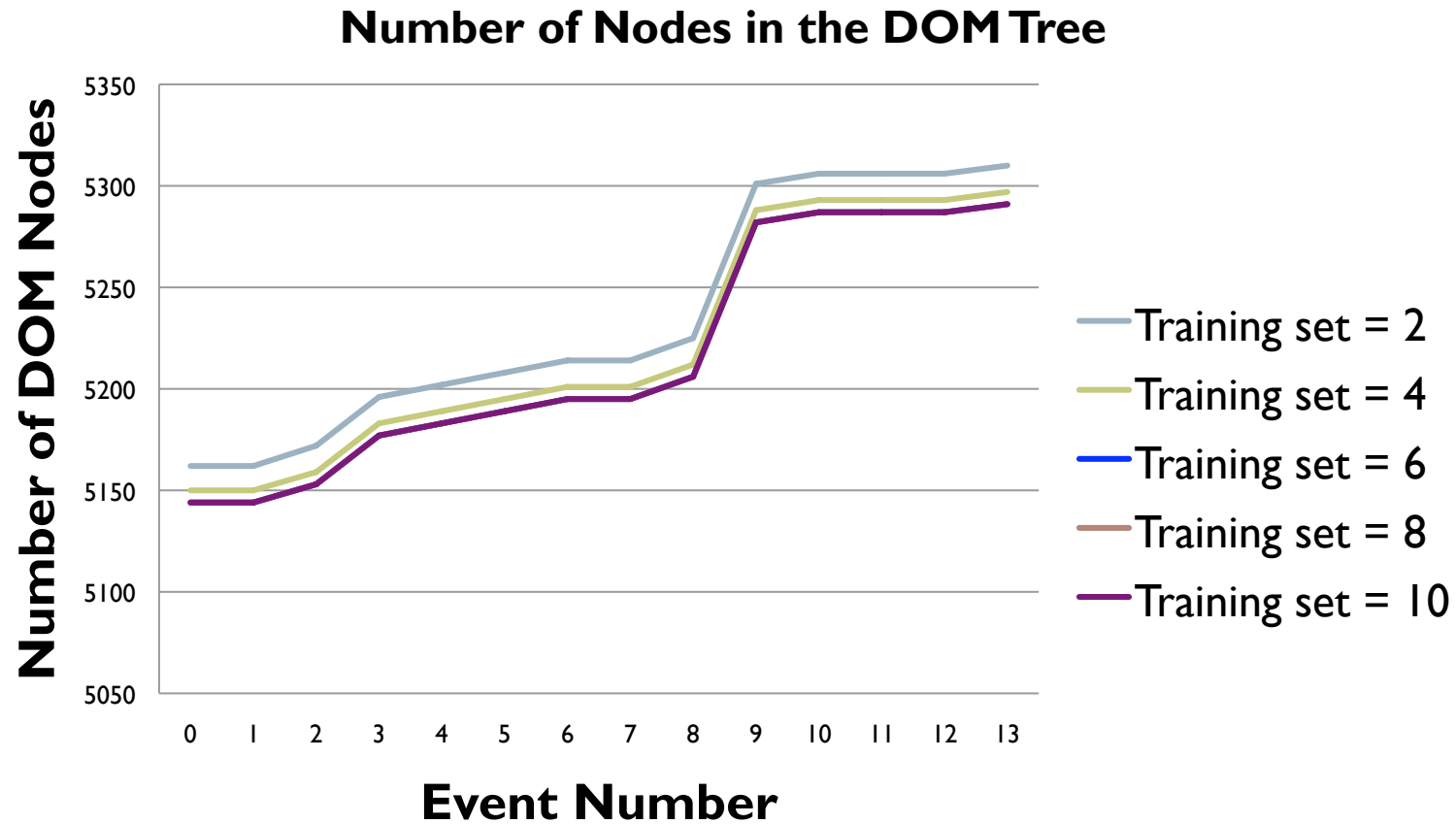


Experimental Setup

- ▶ **Focused mainly on results from Slashdot web app**
 - ▶ 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)**



Results: Invariant DOM



Convergence of DOM-trees with a training set size of 6 (10%)

Results: Fault Injection Experiments

Event Errors

- ▶ Dropped events from the trace one at a time
- ▶ 100% detection for events that impact the DOM

Trace



Fault-injected Trace



Domain Failures

- ▶ Blocked domains one at a time using NoScript
- ▶ Only one of five domains impacted the DOM



DoDOM: Summary

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

This Talk

- ▶ Motivation and Approach
- ▶ Two approaches for evaluating JS Reliability
 - ▶ Error messages - JSER[Under submission]
 - ▶ DOM invariants – DoDOM [ISSRE 2010]
- ▶ Future Directions and Conclusions

Future Directions

- ▶ **Static analysis to augment dynamic analysis**

- ▶ Seed with errors found by dynamic analysis
- ▶ Prioritize errors based on past experience



- ▶ **Analyze the impact of an error message**

- ▶ Does it impact application's functionality ?
- ▶ Does it affect other users of the application ?



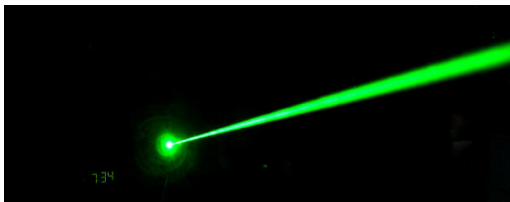
- ▶ **Fault Injection**

- ▶ Inject realistic faults in the application
- ▶ Study its robustness under faults



Conclusions

- ▶ **Web 2.0 applications' reliability is challenging**
- ▶ **Measured the reliability of web apps in the “wild” as a first step to improving their reliability**
 - ▶ **JSER**: Based on error messages
 - ▶ **DoDOM**: Based on DOM invariants



<http://www.>