Empirical Studies of JavaScript-based Web Application Reliability

Karthik Pattabiraman
Froilan Ocariza
Kartik Bajaj
Ali Mesbah
Benjamin Zorn

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

• Building reliable and secure software applications

• Compiler & runtime techniques for error resilience
  – Partitioning data for differential resilience [ASPLOS’11]
  – Error detection in different programs [DSN’12][DSN’13]
  – Fault Injection techniques and tools [DSN’14][ISPASS’14]

• This tutorial
  – Reliability of modern web applications (Part 1)
  – Tools for building robust web applications (Part 2)
Web 2.0 Applications

- Facebook
- YouTube
- Google
- Amazon
- Tumblr
Web 2.0 applications allow rich UI functionality within a single web page
Modern Web Applications: JavaScript

- JavaScript: Implementation of ECMAScript standard
  - Client-Side JavaScript: used to develop web apps
- Executes in client’s browser – send AJAX messages
- Responsible for web application’s core functionality
- Not easy to write code in – has many “evil” features
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: Prevalence

- 97 of the Alexa top 100 websites use JavaScript
- Thousands of lines of code, often > 10,000

Copyright: Karthik Pattabiraman, 2014
JavaScript: “Good” or “Evil”?

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

Real web applications do not stick to the “good” parts

Copyright: Karthik Pattabiraman, 2014
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 and improve the reliability of JavaScript web applications
Does Reliability Matter?

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

---

Copyright: Karthik Pattabiraman, 2014
This Talk

• Motivation and Approach

• Three approaches for measuring JS Reliability
  – Stack Overflow Reports [MSR 2014] – With F. Ocariza, A. Mesbah

• Conclusion and Next Steps
JSER: JavaScript Error Messages

- All exceptions thrown are logged to JS console
JSER: Error Messages Vs. Static Analysis

• No false positives unlike static analysis

• Capture interactions with third-party code (advertisements)

• Capture interactions with the DOM
JSER: Tools

• Chose 50 web applications from Alexa top 100
• Created Selenium tests for normal interactions
• Capture JavaScript Errors printed to Firebug
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?
JSER: 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
JSER: Error Frequencies Results

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

Total Distinct Errors
JSER: Error Classification Results

- 94% of errors fall into four predominant categories

Distribution of Error Messages

- 54% of Permission Denied Errors
- 27% of Undefined Symbol Errors
- 9% of Null Exception Errors
- 6% of Syntax Errors
- 4% of Miscellaneous Errors

Copyright: Karthik Pattabiraman, 2014
JSER: Research Questions

Errors occur in web applications (4 per application on average) and fall into four categories

- How do errors vary by speed of testing? Are they all deterministic?
- How do errors correlate with static and dynamic characteristics of the app?

Copyright: Karthik Pattabiraman, 2014
JSER: Effect of Testing Speed

- Varied testing speed for replaying events in Selenium
- Performed three executions in each testing speed
### JSER: Testing Speed Results (CNN)

<table>
<thead>
<tr>
<th>Error Message (shortened)</th>
<th>F 1</th>
<th>F 2</th>
<th>F 3</th>
<th>M 1</th>
<th>M 2</th>
<th>M 3</th>
<th>S 1</th>
<th>S 2</th>
<th>S 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permission Denied for view.atdmt.com to call &lt;fname&gt; on marquee.blogs.cnn.com</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>targetWindow.cnnad showAd is not a function</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>window.parent.CSIManager is undefined</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
JSER: Effect of Testing Speed

• All three testing modes expose different errors
JSER: Non-Determinism

- More than 70% of errors: non-deterministic
Errors occur in web applications (4 per application on average) and fall into four categories.

- Error occurrences vary with speed of testing.
- About 70% of errors are non-deterministic.

How do errors correlate with static and dynamic characteristics of the app?
JSER: Static/Dynamic Characteristics

**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

Copyright: Karthik Pattabiraman, 2014
# JSER: Correlations Summary

<table>
<thead>
<tr>
<th>Static Characteristics</th>
<th>Dynamic Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Alexa Rank</td>
<td>• Number of called functions</td>
</tr>
<tr>
<td>• Bytes of JavaScript code</td>
<td>• Number of eval calls</td>
</tr>
<tr>
<td>• Number of domains</td>
<td>• Properties deleted</td>
</tr>
<tr>
<td>• Domains containing JS</td>
<td>• Object inheritance overridings</td>
</tr>
</tbody>
</table>

Copyright: Karthik Pattabiraman, 2014
Errors occur in web applications (4 per application on average) and fall into four categories.

Error occurrences vary with speed of testing. About 70% of errors are non-deterministic.

Errors correlate with Alexa rank, no of domains but not with loc or eval calls.

Copyright: Karthik Pattabiraman, 2014
JSER: 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

Copyright: Karthik Pattabiraman, 2014
This Talk

• Motivation and Approach

• Three approaches for measuring JS Reliability
  – Stack Overflow Reports [MSR 2014] – With F. Ocariza, A. Mesbah

• Conclusion and Next Steps

Copyright: Karthik Pattabiraman, 2014
Bug Report Study: Goals

• What errors/mistakes cause JavaScript faults?

• What impact do JavaScript faults have?

Bug Report Study of twelve popular, Open Source JavaScript Applications
Bug Report Study: Objects

Eight JavaScript Web Applications

- moodle
- Joomla!
- WordPress
- Drupal
- RoundCube
- Wikimedia
- TYPO3
- taskfreak!

Four JavaScript Libraries

- jQuery
- prototype
- mootools
- ember
Bug Report Study: Method

- Collect bug reports from bug repositories
  - Focus on bugs that are marked fixed to avoid spurious bugs
  - Organized into a uniform format (XML file)

- Search for all bug reports that have the word “JavaScript”
- Filter out reports that are not marked “fixed” OR the fault does not involve JS
- Pick the first 30 reports and analyze them manually to determine cause/impact

Copyright: Karthik Pattabiraman, 2014
Bug Report Study: Questions

• **RQ1**: What types of JavaScript *faults* occur in web apps?

• **RQ2**: What is the impact of JavaScript faults?

• **RQ3**: How long does it take to fix a JavaScript fault?

• **RQ4**: Are JavaScript faults browser-specific?
Bug Report Study: Categories

Incorrect Method Parameter Fault: Unexpected or invalid value passed to JS method or assigned to JS property

DOM-Related Fault: The method is a DOM API method - Account for around two-thirds of JavaScript Faults

Copyright: Karthik Pattabiraman, 2014
Bug Report Study: DOM

Want to retrieve this element

Text: “Hello world”
Bug Report Study: DOM-Related Faults

JavaScript code:

```javascript
var x = document.getElementById("elem");
```

DOM:

```
<div id="elem"
```

Inexistent ID

Will return null
DOM-Related Fault: Example

var elem, retrievedStr = [Retrieved via XHR];
var dotsInStr = retrievedStr.split(".").length;
if (dotsInStr == 0) {
    var prefix = "id_";
    elem = $("#" + prefix + retrievedStr);
}
else {
    elem = $(retrievedStr);
}

elem[0].focus()
DOM-Related Fault: Example

```javascript
var elem, retrievedStrStr = [Retrieved via XHR];
var dotsInStr = retrievedStrStr.split(".").length;
if (dotsInStr == 0) {
    var prefix = "id_";
    elem = $("#" + prefix + retrievedStrStr);
}
else {
    elem = $(retrievedStrStr);
}
elem[0].focus()
```

Retrieved string via XHR
DOM-Related Fault: Example

```javascript
var elem, retrievedStr = [Retrieved via XHR];
var dotsInStr = retrievedStr.split(".").length;
if (dotsInStr == 0) {
    var prefix = "id_";
    elem = $("#" + prefix + retrievedStr);
}
else {
    elem = $(retrievedStr);
}
elem[0].focus()
```

Find the number of dots in the string
DOM-Related Fault: Example

```javascript
var elem, retrievedStr = [Retrieved via XHR];
var dotsInStr = retrievedStr.split(".").length;
if (dotsInStr == 0) {
    var prefix = "id_";
    elem = $('"#" + prefix + retrievedStr);
}
else {
    elem = $(retrievedStr);
}

elem[0].focus()
```

If there are no dots, prepend “id_” to the string and access it via $(). Otherwise, leave it as is, and access it via $().
DOM-Related Fault: Example

var elem, retrievedStr = [Retrieved via XHR];
var dotsInStr = retrievedStr.split(".").length;
if (dotsInStr == 0) {
    var prefix = "id_";
    elem = $("#" + prefix + retrievedStr);
}
else {
    elem = $(retrievedStr);
}

elem[0].focus()

Retrieved string of “editor” would go here even though it has no dots, which would erroneously cause $() to use selector “editor”, which doesn’t match any elements.

UNDEFINED EXCEPTION!
DOM-Related Fault: Example

```javascript
var elem, retrievedStr = [Retrieved via XHR];
var dotsInStr = retrievedStr.split(".").length;
if (dotsInStr == 0) {
    var prefix = "id_";
    elem = $("#" + prefix + retrievedStr);
}
else {
    elem = $(retrievedStr);
}
elem[0].focus()
```

**BUG:** The assigned value should be `retrievedStr.split(".").length - 1`, as `length()` always returns at least 1.
Bug Report Study: Impact

• Impact Types – Based on Bugzilla [ICSE’11]
  • Type 1 (lowest impact), Type 5 (highest impact)

![Bar chart showing number of bug reports by impact type]

80% of highest impact faults are DOM-related

Copyright: Karthik Pattabiraman, 2014
Bug Report Study: Fix Times

- **Triage Time:** Time it took to assign or comment on the bug
- **Fix Time:** Time it took to fix the bug since it was triaged

![Graph showing average number of days for Triage Time and Fix Time, differentiated by DOM-related and non-DOM-related faults.](image-url)
Bug Report Study: Browser Specificity

Most JavaScript faults are not browser-specific
Bug Report Study: Summary

• Bug report study of 12 applications: JS faults
  – Over 300 bug reports analyzed; only fixed bugs considered

• DOM-related faults dominate JavaScript faults
  – Responsible for nearly two-thirds of all JavaScript faults
  – Responsible for 80% of highest impact faults
  – Take 50% longer time to fix for developers
  – Majority are not specific to web browser platform

• Need robust solutions for DOM-related faults
  – Fixing, Understanding and writing correct code

Copyright: Karthik Pattabiraman, 2014
This Talk

• Motivation and Approach

• Three approaches for measuring JS Reliability
  – Stack Overflow Reports [MSR 2014] – With F. Ocariza, A. Mesbah

• Conclusions and Next Steps
StackOverflow: Background

• Stack Overflow
  – QA website for programmers
  – Started in 2008
  – 4,125,638 questions asked from Jan’09 to Dec’12
  – 500,000+ questions related to web development

• Questions directly asked/answered by developers
  – Followed by discussion in comments
UIWebView intermittently denied access to html5 database storage

I have an iphone app that enables users to login via a native UIViewController on ios, that then fires up a UIWebView to display the main content. The webapp uses database storage to retain some of the content locally.

However, every now and then, the webapp will fail to load when attempting to access the database with the following message appearing in the logs.

...... sandboxd[3203] : APPNAME(3201) deny file-write-create /Databases.db
StackOverflow: NLP Analysis

- Filter web-related questions based on tags provided
- Analyzing the text provided in the questions and answers (Latent Dirichlet Allocation)
StackOverflow Datasets

- **JavaScript**
- **CSS**
- **HTML5**

Copyright: Karthik Pattabiraman, 2014
RQ1: Categorization of topics of discussion

![Bar chart showing category share for various topics]
RQ2: Temporal trends over time
RQ3: Prevalence of web in mobile development
RQ4: Technical challenges

I'd like to store a JavaScript object in HTML5 `localStorage`, but my object is apparently being converted to a string.

I can store and retrieve primitive JavaScript types and arrays using `localStorage`, but objects don't seem to work. Should they?

I have noticed I am getting a "CSS Explosion". It is becoming difficult for me to decide how to best organize and abstract data within the CSS file.
StackOverflow: Summary of Findings

- **Finding 1:** Though cross-browser issues dominated in the past, they have declined sharply since 2012.

- **Finding 2:** DOM and Canvas interactions consistently dominate

- **Finding 3:** Mobile web application development is on the rise, compared to traditional web application development

- **Finding 4:** Even expert programmers are confused by APIs and documentation
StackOverflow: Implications

• **Finding 1, 2 (Categorization and temporal trends)**
  – *Researchers* can shift their focus away from cross browser issues to DOM and Canvas related ones.

• **Finding 4 (Prevalence of mobile applications)**
  – Need better tools for mobile web development

• **Finding 5 (Technical Challenges)**
  – Can guide *standardization communities* to focus on areas that need improvement.
This Talk

• Motivation and Approach

• Three approaches for measuring JS Reliability
  – Stack Overflow Reports [MSR 2014] – With K. Bajaj and A. Mesbah

• Conclusion and Next Steps
Conclusions

• Web 2.0 applications’ reliability is challenging

• Measure the reliability of web applications
  – [ISSRE’11]: Based on error messages in real web apps
  – [ESEM’13]: Based on bug reports in web apps
  – [MSR’14]: Based on StackOverflow questions

• Need to improve web applications’ reliability –
  Use of empirical data to drive improvements
Next Steps (Part 2 of tutorial)

- AutoFlox (with Frolin Ocariza): ICST 2012
  - Fault localization

- Vejovis (with Frolin Ocariza): ICSE 2014
  - Fault Repair

- Clematis (with Saba Alimadi, Sheldon Sequira): ICSE 2014
  - Program Understanding

- Dompletion (with Kartik Bajaj): ASE’14
  - Code completion
AutoFlox [Ocariza – ICST 2012]

- **AutoFlox**: Automatic fault localization tool for JS
  - Find origin of the null value
  - i.e., find the *direct DOM access*

```javascript
var toggle = 1;
var x = "hlelo_";
var y = "world";
var elem = document.getElementById(x + y);
var dis = "";
if (toggle == 1) {
    dis = "block";
} else {
    dis = "inline";
}
 elem.style.display = dis;
```

Direct DOM Access
(This is where the NULL value came from)
Vejovis [Ocariza – ICSE’14]

• **Vejovis**: automatic repair of DOM-related errors
  
  – Starts at direct DOM access found by AutoFlox
  
  – Provide fix suggestions based on common fix patterns

```javascript
var toggle = 1;
var x = "hlelo_";
var y = "world";
var elem = document.getElementById(x + y);
var dis = "";
if (toggle == 1) {
    dis = "block";
}
else {
    dis = "inline";
}
elem.style.display = dis;
```

Copyright: Karthik Pattabiraman, 2014
Clematis [Alimadadi – ICSE’14]

- Challenge: Web applications are complex, and consist of DOM interactions, AJAX messages and timeouts
- Difficult to trace the links between events and JS code
- Clematis allows users to visualize causal dependencies between events and code, and the DOM

Copyright: Karthik Pattabiraman, 2014
Dompletion [Bajaj - ASE’14]

• Provide code-completion suggestions for programmers for DOM-JavaScript interaction
  – Based on analysis of JavaScript code and DOM