JavaScript Errors in the Wild: An Empirical Study

Frolin S. Ocariza, Jr.¹
Karthik Pattabiraman¹
Benjamin Zorn²

¹ University of British Columbia (UBC), ²Microsoft Research (MSR)
Web 2.0 Application: Amazon.com

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

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 Clients
- Loose semantics
- DOM

Snapshot of iFeng.com: Leading media website in China

[Image of a diagram showing Document, Root element, Element, Attribute, Text, Error message: "An error occurred while processing this directive"]
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”
Contributions

- Devised methodology to collect and categorize JS error messages from web applications
- Characterized JS error messages that appeared in 50 top websites
- Analysis of the implications of results
  - Find possible ways to write more reliable JS code
  - Improve the quality of JS testing
  - Highlight JS characteristics that should be captured by static analysis tools
JavaScript Error Messages

- Any exception thrown by JS code is logged to JS console
Error Messages: Pros and Cons

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

- **Cons**
  - Error message may be benign
    - Still an indication of potential problem
  - May be incomplete
    - Publicly available JS bug reports very limited
Steps to Collect Error Messages

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

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

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

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

1. Description of error message
2. Line of code corresponding to error
3. File name 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](chart.png)

- **Web Application**
- **# of Errors**
- **Errors**
- Google, Yahoo, QQ, Amazon, WordPress, LinkedIn, Microsoft, 163, PayPal, Flickr, Apple, BBC, go.com, Youku, CNN, ESPN, MegaUpload, 4shared, About, Tumblr, CNET, Sogou, Hefg, ImageShack, NY Times
94% of errors fall into four predominant categories.

**Distribution of Error Messages**

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

• **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
Undefined Symbol Example

- **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
Null Exception Example

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

  ```javascript
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
Syntax Error Example

• 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
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
- Performed three executions in each testing speed

![Diagram showing testing speeds and their durations: Fast 0 ms, Medium 500 ms, Slow 1000 ms]
## Effect of Testing Speed: Results

<table>
<thead>
<tr>
<th>Error Message (shortened)</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permission Denied for view.atdmt.com to call <code>&lt;fname&gt;</code> on marquee.blogs.cnn.com</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</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>
Non-Deterministic Error: Example

- Tumblr page

`change_promo(promo) \rightarrow` expects `promo` to be within range 1-4

`setTimeout(change_promo, 5000) \rightarrow` no parameter specified, so random value for `promo`

No exception if `promo` within range
Exception if `promo` out of range
Effect of Testing Speed: Non-Determinism

- More than 70% of distinct errors are non-deterministic
Research Questions

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

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

How do errors correlate with static and dynamic characteristics of the app?
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
Dynamic Correlations: Eval Calls

- Low correlation
  - Compare: eval calls
    - correlate well with security
Static Correlations: JS Code Size

- Low correlation
  - JS reliability not tied very closely to code size
Research Questions: Answers

- Do errors occur in web apps and what categories do they fall in?
  - Average of four errors in each app. Errors fall into four well-defined categories.

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

- How do errors correlate with static and dynamic characteristics of the app?
  - Correlated with number of domains, # of domains with JS, Alexa rank, but not with eval calls and code size.

- Error occurrence by speed of testing? Majority of errors are non-deterministic?
Implications of 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
Conclusion and Future Work

- JavaScript code in production web apps → buggy!
- Study exposes JS reliability problems and analyzes errors
  - Data publicly available
  - http://ece.ubc.ca/~frolino/project/jser
- Future work
  - Better understanding of causes (root causes and correlations)
  - Currently: Fault localization
Backup Slides
JavaScript: “Good” or “Evil”? Versus

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

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