Understanding Asynchronous Interactions In Full-Stack JavaScript

Saba Alimadadi, Ali Mesbah and Karthik Pattabiraman

ICSE 2016
saba@ece.ubc.ca

Sahand: http://github.com/saltlab/sahand
Stack Overflow

**JavaScript:**
Most popular language

<table>
<thead>
<tr>
<th></th>
<th>2016</th>
<th>2015</th>
<th>2014</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript</td>
<td>55.4%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQL (or SQL Server)</td>
<td>49.1%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td>36.3%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C#</td>
<td>30.9%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHP</td>
<td>25.9%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td>24.9%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td>19.4%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>15.5%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node.js</td>
<td>17.2%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AngularJS</td>
<td>17.9%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ruby</td>
<td>8.9%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Objective-C</td>
<td>8.5%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

GitHub

**JavaScript:**
Top languages on GitHub
Understanding JavaScript Apps

Whole Picture
Challenge 1. **Server-Side Callbacks**

- Asynchronous execution
- Callback hell

```javascript
fs.readdir(source, function(err, files) {
  files.forEach(function(filename, fileIndex) {
    gm(source + filename).size(function(err, values) {
      widths.forEach(function(width, widthIndex) {
        this.resize(w, h).write(newName, function(err) {
          
        })
      })
    })
  })
}) // example from callbackhell.com
```
Challenge 2. **Network Communications**
Challenge 3. Asynchronous **Client** Side

**Time**

**Interval**

**Response (timeout)**

**XHR**

**Server**
Summary of Challenges

• Server-side callbacks
• Network communication
• Asynchronous client side

Related work:

EMSE’13 UIST’14 ICSE’14, ECOOP’15
Our Approach: **Sahand**

1. Instrument automatically
2. Trace full-stack execution
3. Infer a behavioural model
4. Visualize the model
Behavioral Model

- **Nodes**
- **Lifelines of function executions**
- (A)Synchronous client/server events

- **foo()**
- Act
- Ina
- Act

- **bar()**
- Act

- **baz()**
- Sch
- Act
- Ina
- Act

- **event**

**Links** — Time, Type, Direction
Real Behavioural Models Are Complex
Visualization

Client-Side Analysis

<table>
<thead>
<tr>
<th>Function</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo()</td>
<td>active</td>
</tr>
<tr>
<td>bar()</td>
<td>active</td>
</tr>
</tbody>
</table>

Connecting client and server

<table>
<thead>
<tr>
<th>Function</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>baz()</td>
<td>active</td>
</tr>
<tr>
<td>app.js:45</td>
<td>scheduled</td>
</tr>
<tr>
<td>qux()</td>
<td>active</td>
</tr>
</tbody>
</table>

Server-Side Analysis
Visualization

**Client-Side Analysis**

<table>
<thead>
<tr>
<th>Function</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo()</td>
<td>active</td>
</tr>
<tr>
<td>bar()</td>
<td>active</td>
</tr>
</tbody>
</table>

**Server-Side Analysis**

<table>
<thead>
<tr>
<th>Function</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>baz()</td>
<td>active</td>
</tr>
<tr>
<td>app.js:45</td>
<td>scheduled</td>
</tr>
<tr>
<td>qux()</td>
<td>active</td>
</tr>
</tbody>
</table>

Events and DOM interactions

Timeouts

XHRs

Time — Temporal primitives — Time points

Event loop
Visualization

Client-Side Analysis

<table>
<thead>
<tr>
<th>Function</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo()</td>
<td>active</td>
</tr>
<tr>
<td>bar()</td>
<td>active</td>
</tr>
</tbody>
</table>

Function executions

Time — Temporal primitives — Time intervals

Server-Side Analysis

<table>
<thead>
<tr>
<th>Function</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>baz()</td>
<td>active</td>
</tr>
<tr>
<td>app.js:45</td>
<td>scheduled</td>
</tr>
<tr>
<td>qux()</td>
<td>active</td>
</tr>
</tbody>
</table>

Callbacks
Visualization

**Client-Side Analysis**

- `foo()`
  - active
  - active

- `bar()`
  - active

**Server-Side Analysis**

- `baz()`
  - active

- `app.js:45`
  - scheduled
  - active

- `qux()`
  - active

Time — Structure of time — Linear & Branching
Visualization

Client-Side Analysis

<table>
<thead>
<tr>
<th>Function</th>
<th>Status 1</th>
<th>Status 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo()</td>
<td>active</td>
<td>active</td>
</tr>
<tr>
<td>bar()</td>
<td>active</td>
<td></td>
</tr>
</tbody>
</table>

Time — Structure of time — Linear & Branching

Server-Side Analysis

<table>
<thead>
<tr>
<th>Function</th>
<th>Status 1</th>
<th>Status 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>baz()</td>
<td>active</td>
<td></td>
</tr>
<tr>
<td>app.js:45</td>
<td>scheduled active</td>
<td></td>
</tr>
<tr>
<td>qux()</td>
<td>active</td>
<td></td>
</tr>
</tbody>
</table>
Implementation: **Sahand**

- Express.js application
- Proxy -> dynamic instrumentation
- Esprima, Estraverse, Escodegen

[https://github.com/saltlab/sahand](https://github.com/saltlab/sahand)
Evaluation

Does using **Sahand** improve developers’ performance in program comprehension tasks?
Controlled Experiment

- *Sahand*’s effect on developers’ performance
- 12 Participants
- Object: full-stack JavaScript application
Controlled Experiment

• Design
  – Control: tool and expertise level
  – Measure: performance

• Procedure
  – Pre-questionnaire
  – Tutorial
  – Tasks
  – Post-questionnaire
Results Highlight

Using **Sahand**

3 times more accuracy

In the same time
Summary of Challenges

- Server-side callbacks
- Network communication
- Asynchronous client side

Related work:
- Zaidman et al. EMSE13
- Hirschman et al. UBT14
- Alimadadi et al. ICSE14, ECOOP15

Visualization

Client-Side Analysis

```plaintext
foo()  active
bar()  active
```

Connecting client and server

```plaintext
baz()  active
app.js:45 scheduled active
qux()  active
```

Server-Side Analysis

Results Highlight

Using Sahand
3 times more accuracy

In the same time

Control group
Experimental (Sahand) group

Saba Alimadadi

Hire Me!

Sahand: http://github.com/saltlab/sahand