Inspection is Overwhelming

There's a lot of code to examine
Entry points are not obvious
Information barriers present
Changes happen fast (JS/HTML)
Methods execute simultaneously

Curious user faces Information and Understanding Learning Barriers

[Ko, Myers, Aung, 2004]
Pilot Study

Setup
- 4 developers
- 20 mins
- Observe Strategy

Findings
- Find-all 60K LOC JS
- Copy-paste
- Poor tool utilization

Discussion
- Unravel the HTML, JS
- Bubble relevancy
- Hand-off inspection
- Detect libraries
<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FireCrystal</td>
<td>Web UI Replay with Causal JS</td>
</tr>
<tr>
<td>Oney, Myers 2009</td>
<td></td>
</tr>
<tr>
<td>Dynamic Web Breakpoints</td>
<td>DOM Changes set JS Breakpoints</td>
</tr>
<tr>
<td>Barton, Odvarko 2010</td>
<td></td>
</tr>
<tr>
<td>Theseus</td>
<td>Real-time JS Traces from Browser to Editor</td>
</tr>
<tr>
<td>Lieber, Brandt, Miller 2014</td>
<td></td>
</tr>
<tr>
<td>Scry</td>
<td>Web UI Replay with Causal JS/HTML/CSS and diffs</td>
</tr>
<tr>
<td>Burg, Ko, Ernst &lt;10m ago</td>
<td></td>
</tr>
</tbody>
</table>
Contributions of Unravel in Related Work

Unravel
Hibschman, Zhang (Right Now!)

JS HTML traces & libs while recording, then aggregate

Show me what happened first, inspect later

Be portable

Be extendable

Study how it’s used
Unravel [System]

**HTML Observations**
- Observe DOM
- Gather, Reduce, Filter
- Handoff Inspection

**JavaScript Traces**
- Trace Document API
- Gather, Reduce, Filter
- Handoff Inspection

**Library Detection**
- APIs
- Syntactic Sugar
- Shims, Polyfills
Live Demo!
Monitor all invocations of an API for `func` in `window.document`

```javascript
baseFunc = closure func
func = function(args) {
    capture args
    throw error
    catch error, get stack
    publish stack and args
    return baseFunc(args..context)
}
```
**User Study**

**Research Questions**
- How is strategy altered?
- What is the feature utilization?
- Which barriers were overcome?

**Task: How does a feature work (5)?**
- 15 minutes site A (Unravel)
- 15 minutes site B (Control)
- 15 minutes followup

**Milestones**
- M1: Key Source Code
- M2: Second Key Source
- M3: “Ah-ha” moment

**Users**
- 13 Junior & Senior developers
- Familiar with Chrome DevTools
Study Findings

53.4% Decrease in time to M1, not M2 or M3

Users looked through 80% less JavaScript

JavaScript Traces were the most helpful feature in user discussions

7 out of 13 users learned a new strategy during the exercise
<table>
<thead>
<tr>
<th>Discussion</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unravel helped users find a starting point of understanding the code</td>
<td>Only the client-side of the story</td>
</tr>
<tr>
<td>Overcome Design, Information, Understanding Barriers</td>
<td>Doesn’t deal with SVG animation well</td>
</tr>
<tr>
<td>Unravel drastically decreased the dependence on code search and inspection</td>
<td>Won’t capture WebGL transformations</td>
</tr>
<tr>
<td>Served both intermediate and expert developers equally well</td>
<td>Closed Document API references hide from Unravel</td>
</tr>
<tr>
<td></td>
<td>Minification leaves some variables a mystery while giving names to others</td>
</tr>
<tr>
<td></td>
<td>Only tested on a small number of users</td>
</tr>
</tbody>
</table>
Video of the Future!
Rapid Web Application Reverse Engineering via Interaction Recording, Source Tracing, and Library Detection

Joshua Hibschman  Haoqi Zhang
Spare Slides >>
Inject monitor agent before other scripts load.

Listen for events.

Aggregate reduce and filter on-the-fly.

Implementation [light dependencies]

Deep dive into the murky waters of script loading

By Jake Archibald
Published: June 5th, 2013
Comments: 57
Design Goal: Promote thick authentic learning

- Personally meaningful
- Relate to the real-world
- Think in modes of a discipline
- Assessment reflects the learning process

(Shaffer and Resnick, 1999)

How did they do that?

- Professional websites
- A programmer, user, tester, reverse engineer
- Integrate and reuse professional web techniques
Design Goal: Overcome Learning Barriers

(Ko, Myers, Aung, 2004)
Understanding Changes in Dynamic UI

Web Foraging
(Brandt et al 2009)

Timelapse
(Burg et al 2013)

Gliimpse
(Dragicevic et al 2011)

Scotty
(Eagan et al 2011)

Mimic
(Breslav et al 2014)