Serving Two Masters
An Empirical Study of Browser API Cooptation

Pete Snyder, Chris Kanich
University of Illinois at Chicago
Less Features \rightarrow More Features

Managed Memory \rightarrow Pointer Arithmetic
Outline

• Browser Complexity is Increasing
• Complexity is Often Not Useful
• Complexity is Harmful to Privacy
• Is Complexity is Harmful to Security?
1. Browser Complexity is Growing
Beyond the Web: Excavating the Real World Via Mosaic

THE MERCURY PROJECT

Ken Goldberg, Assistant Professor, Computer Science
Michael Marsch, Assistant Professor, Anthropology and
Steven Genter, M.S. Candidate, Computer Science
Juergen Roesman, Graduate Student, University of Dortmund, Germany
Nick Rothenberg, PhD Candidate, Visual Anthropology
Carl Sutter, Senior Programmer/Analyst, Center for Scholarly Technology
Jeff Wiegley, PhD Candidate, Computer Science
University of Southern California, Los Angeles, CA.
(To appear in the Second International WWW Conference, Chicago, IL, Oct 17-21, 1994)

Abstract

This paper describes a Mosaic server that allows users to “leave the Web” and interact with the real world. An interdisciplinary team of anthropologists, computer scientists and electrical engineers collaborated on the project, designing a system which consists of a robot arm fitted with a CCD camera and a pneumatic system. By clicking on an ISMAP control panel image, the operator of the robot directs the camera to move vertically or horizontally in order to obtain a desired position and image. The robot is located over a dry-earth surface allowing users to direct short bursts of compressed air onto the surface using the pneumatic system. Thus robot operators can “excavate” regions within the environment by positioning the arm, delivering a burst of air, and viewing the image of the newly cleared region. This paper describes the system in detail, addressing critical issues such as robot interface, security measures, user authentication, and interface design. We see this project as a feasibility study for a broad range of WWW applications.

Goals of the Project

WWW and Mosaic[1]-like servers provide a multi-media interface that spans all major platforms. Thousands of sites have been set up in the past year. Our goal with this project was to provide public access to a teleoperated robot, thus allowing users to reach beyond the digital boundaries of the WWW.

Such a system should be robust as it must operate 24 hours a day and it should be low in cost (we had an extremely limited budget). It is worth noting that the manufacturing industry uses the same criteria to evaluate robots for production. Thus our experience with...
1995: Netscape 2.0
THE ROAD TO ENLIGHTENMENT

Littering a dark and dreary road lay the past relics of browser-specific tags, Incompatible DOMs, and broken CSS support.

Today, we must clear the mind of past practices. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C, WAI and the major browser creators.

The css Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity. Learn to use the (yet to be) time-honored techniques in new and invigorating fashion. Become one with the web.

SELECT A DESIGN:

- Sample #1
  - by Dave Shea
- Sample #2
  - by Dave Shea
- Sample #3
  - by Dave Shea
- Sample #4
  - by Dave Shea
- Sample #5
  - by Dave Shea
1998: DOM1
1999: AJAX / XMLHttpRequest
Observations

- API growth started off very slow
- API growth was “document” centric
- “Broad” APIs
API Growth
<table>
<thead>
<tr>
<th>Year</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td>CSSOM View Module, Web Audio API, Proximity Events, Crypto Extensions, Touch Events, GeoLocation API, Pointer API, CSS Animations</td>
</tr>
<tr>
<td>2014</td>
<td>Calendar API, Messaging API, RDF Extensions, Progress events, Network Info API, Ambient Light API, HTML 5, WebCrypto API</td>
</tr>
<tr>
<td>2015</td>
<td>Encrypted Media Extensions, Web MIDI, Service Workers, Performance API, Raw Socket API, WebDriver API, SVG 2 API, WebRTC</td>
</tr>
</tbody>
</table>
2. Is This Complexity Useful?
Determining API “Usefulness”

- Measure how often APIs are called
- Decide whether those calls are “useful"
- Simulate real world web browsing
Measuring API Calls

- Selected 45 APIs and features
- Instrumented PhantomJS / WebKit
- Implemented missing APIs
“Usefulness” Oracle

- Subjective measure
- Ghostery and AdBlock+ filter rules
- Measure API usage pre-and-post filters
Simulated Browsing

- Alexa 10,000
- 10,000 random URLs
- 10,000 random Hosts
- “Random” sites taken from searching UNIX dictionary tri-grams on DDG
AJAX
DOM 1 + 2 APIs
# Rare APIs

<table>
<thead>
<tr>
<th>API Name</th>
<th>URLs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery API</td>
<td>21</td>
</tr>
<tr>
<td>Page Transition API</td>
<td>9</td>
</tr>
<tr>
<td>GeoLocation API</td>
<td>55</td>
</tr>
<tr>
<td>Shadow DOM</td>
<td>5</td>
</tr>
</tbody>
</table>
Non-used APIs

- IndexDB
- WebGL
- WebRTC
- Browser Name API
- Gamepad API
- SVG API
- Vibration API
- WebAudio API
- WebWorker API
GeoLocation API
3. Browser Complexity is Harmful to Privacy
Example: WebRTC

- Intent: Allow peer-to-peer applications
- Attack: Leaks local IP address
- Widely available (56.22%)
- Rarely used for intended purpose

<table>
<thead>
<tr>
<th>Browser</th>
<th>Version Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox</td>
<td>22</td>
</tr>
<tr>
<td>Chrome</td>
<td>23</td>
</tr>
<tr>
<td>Android Browser</td>
<td>40</td>
</tr>
<tr>
<td>Opera</td>
<td>30</td>
</tr>
</tbody>
</table>
Example: Crypto

- **Intent:** Allow applications to perform crypto operations
- **Use:** Generates persistent random identifiers
- **Widely available (70.24%)**
- **Rarely used for intended purpose**

<table>
<thead>
<tr>
<th>Browser</th>
<th>Version Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox</td>
<td>38</td>
</tr>
<tr>
<td>Chrome</td>
<td>31</td>
</tr>
<tr>
<td>Android Browser</td>
<td>4.4</td>
</tr>
<tr>
<td>Opera</td>
<td>30</td>
</tr>
<tr>
<td>IE</td>
<td>11</td>
</tr>
<tr>
<td>iOS</td>
<td>7.1</td>
</tr>
</tbody>
</table>
Methodology

• Load and measure each URL
• Reload and remeasure with Ghostery
• Big differences in API usage -> privacy-harmful APIs
CSSOM API (Document)
Crypto API
Storage API
## “Non-User Serving” APIs

<table>
<thead>
<tr>
<th>API</th>
<th>Pages #</th>
<th>Ghost #</th>
<th>Ghost %</th>
<th>ABP #</th>
<th>ABP %</th>
<th>Both #</th>
<th>Both %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSSOM (Doc)</td>
<td>249</td>
<td>18</td>
<td>92.8</td>
<td>34</td>
<td>86.3</td>
<td>1</td>
<td>99.6</td>
</tr>
<tr>
<td>Crypto</td>
<td>7,713</td>
<td>1,123</td>
<td>85.4</td>
<td>38</td>
<td>99.5</td>
<td>27</td>
<td>99.6</td>
</tr>
<tr>
<td>Language</td>
<td>16,909</td>
<td>2,242</td>
<td>86.7</td>
<td>2,072</td>
<td>87.7</td>
<td>1,131</td>
<td>93.3</td>
</tr>
<tr>
<td>&lt;iframe&gt;Injection</td>
<td>12,110</td>
<td>3,202</td>
<td>73.6</td>
<td>4,464</td>
<td>63.1</td>
<td>1,351</td>
<td>88.8</td>
</tr>
<tr>
<td>Page Visibility</td>
<td>729</td>
<td>228</td>
<td>68.7</td>
<td>81</td>
<td>88.9</td>
<td>86</td>
<td>88.2</td>
</tr>
<tr>
<td>Websocket</td>
<td>225</td>
<td>99</td>
<td>56.0</td>
<td>58</td>
<td>74.2</td>
<td>43</td>
<td>80.9</td>
</tr>
<tr>
<td>Plugin Detection</td>
<td>18,116</td>
<td>5,870</td>
<td>67.6</td>
<td>4,133</td>
<td>77.2</td>
<td>3,512</td>
<td>80.6</td>
</tr>
<tr>
<td>Battery API</td>
<td>21</td>
<td>17</td>
<td>19.0</td>
<td>4</td>
<td>81.0</td>
<td>6</td>
<td>71.4</td>
</tr>
<tr>
<td>Storage</td>
<td>12,357</td>
<td>5,499</td>
<td>55.5</td>
<td>5,496</td>
<td>55.5</td>
<td>3,817</td>
<td>69.1</td>
</tr>
</tbody>
</table>
## “User Serving” APIs

<table>
<thead>
<tr>
<th>API</th>
<th>Pages #</th>
<th>Ghost #</th>
<th>Ghost %</th>
<th>ABP #</th>
<th>ABP %</th>
<th>Both #</th>
<th>Both %</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOM 1 (creating)</td>
<td>23,304</td>
<td>22,651</td>
<td>2.8</td>
<td>21,409</td>
<td>8.1</td>
<td>21,266</td>
<td>8.7</td>
</tr>
<tr>
<td>DOM 1 (querying)</td>
<td>23,659</td>
<td>22,965</td>
<td>2.9</td>
<td>21,705</td>
<td>8.3</td>
<td>21,580</td>
<td>8.8</td>
</tr>
<tr>
<td>AJAX</td>
<td>20,016</td>
<td>19,027</td>
<td>4.9</td>
<td>16,153</td>
<td>19.3</td>
<td>16,303</td>
<td>18.6</td>
</tr>
<tr>
<td>Canvas API</td>
<td>2,095</td>
<td>1,949</td>
<td>7.0</td>
<td>1,676</td>
<td>20.0</td>
<td>1,694</td>
<td>19.1</td>
</tr>
<tr>
<td>User Agent</td>
<td>23,439</td>
<td>21,195</td>
<td>9.6</td>
<td>19,602</td>
<td>16.4</td>
<td>18,870</td>
<td>19.5</td>
</tr>
<tr>
<td>&lt;audio&gt;</td>
<td>307</td>
<td>292</td>
<td>4.9</td>
<td>247</td>
<td>19.5</td>
<td>242</td>
<td>21.2</td>
</tr>
<tr>
<td>Blob API</td>
<td>308</td>
<td>287</td>
<td>6.8</td>
<td>233</td>
<td>24.4</td>
<td>238</td>
<td>22.7</td>
</tr>
<tr>
<td>&lt;svg&gt;</td>
<td>860</td>
<td>798</td>
<td>7.2</td>
<td>520</td>
<td>39.5</td>
<td>527</td>
<td>38.7</td>
</tr>
<tr>
<td>History API</td>
<td>576</td>
<td>490</td>
<td>14.9</td>
<td>374</td>
<td>35.1</td>
<td>349</td>
<td>39.4</td>
</tr>
</tbody>
</table>
4. Is Complexity harmful to Security?
@todo

- Status quo violates “principle of least privilege”
- Gathering data from open bug databases
- Lots of hand labeling involved…
- On going…
5. Conclusions
Conclusions

- Browsers are growing in complexity quickly
- Mismatch between user intent and web author intent
- Mismatch between need and capability
- Harms privacy, might harm security
Thanks!