Speed up your map with Closure-Compiler

Chad Killingsworth
Assistant Director of Web & New Media
What slows down your map?

• Large amounts data (100s of markers, polylines, etc)
• JavaScript size and complexity
• Network Latency (especially on mobile devices)
Impact of JavaScript Code Size

- On the iPhone 3G, it takes around 20ms to parse each KB of JavaScript.
- Faster mobile devices and desktop browsers experience a lesser impact.
JavaScript Compressors

- Dean Edwards Packer
- YUI Compressor
- JSMIn
- Closure-Compiler
Advantages of Closure-Compiler

- Full compiler – not just a compressor
- Helps identify type errors
- Optimized for servers which Gzip JavaScript source
- Many code optimizations including:
  - Constant folding
  - Function inlining
  - Dead code elimination
  - Loop optimizations
  - Variable renaming/reuse
Closure-Compiler Modes

• Whitespace Only: minimal compression benefits
• Simple Optimizations: performs safe optimizations
• Advanced Optimizations: best final code size and highest number of optimizations. Requires some changes to most source code.
# Real World Comparison

<table>
<thead>
<tr>
<th></th>
<th>Original Source</th>
<th>YUI Compressor</th>
<th>Packer</th>
<th>JSMIn</th>
<th>Closure (Simple Mode)</th>
<th>Closure (Advanced Mode)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size % (bytes)</td>
<td>100% (40,875)</td>
<td>62.5% (25,546)</td>
<td>67.78% (27,706)</td>
<td>68.72% (28,089)</td>
<td>55.51% (22,689)</td>
<td>38.92% (15,909)</td>
</tr>
<tr>
<td>Gzip Size</td>
<td>20.37% (8,325)</td>
<td>15.18% (6,204)</td>
<td>15.94% (6,514)</td>
<td>16.11% (6,584)</td>
<td>14.34% (5,863)</td>
<td>12.47% (5,096)</td>
</tr>
</tbody>
</table>
Ways to use Closure-Compiler

- Web Application: http://closure-compiler.appspot.com/
- Web Service API
- Downloadable JAR application
- Custom compiler build
Example Usage

Java -jar compiler.jar --compilation_level SIMPLE_OPTIMIZATIONS --js map_uncompressed.js --js_output_file map.js
What to avoid when using Closure-Compiler

- JavaScript “with” statement
- JavaScript “eval” statement
- String representations of function or variable names
Using Advanced Optimizations

- Requires annotating your JavaScript with JSDoc tags

```javascript
/**
 * @param {string} Input
 * @return {string}
 */

function SafeString(Input) {
    return Input.replace(/[^_0-9a-z]/gi, "_")
}
```
Using Advanced Optimizations

- Assumes that ALL of the JavaScript on your page is declared and used within your source file. References to other libraries must be defined by as an extern.

```javascript
google.maps.event.addListenerOnce(map, 'tilesloaded',
    function () {
        document.getElementById("LoadingDiv").style.display = "none";
    });
```
Using Advanced Optimizations

- Prohibits mixing dotted and string references to an object property reference

```javascript
/**
 * @param {number} lat
 * @param {number} lng
 * @return {Object.<string,*>}
*/
function LocationInfo(lat, lng) {
    var a = {};
    a.latLng = new google.maps.LatLng();
    a.latLngString = a["latLng"]toUrlValue();
    return a;
}
```
Using Advanced Optimizations

• Removes unreferenced code. Requires exporting functions that are called elsewhere.

```javascript
/** @this {Element} */
function imgMouseOver() {
   this.src = this.src.replace(/\..*\./i, "_over.png");
}
window["imgMouseOver"] = imgMouseOver;
```
Closure-compiler Information

• Website
  http://code.google.com/closure/compiler/

• Discussion group
  http://code.google.com/closure/compiler/community.html

• Web application
  http://closure-compiler.appspot.com/

• Blog
  http://closuretools.blogspot.com/
About Me

• Blog
  http://blogs.missouristate.edu/web/author/chk790/

• Twitter
  @ChadHikes