Even faster web mapping
Michael Neutze
a little recap
Too many bytes

<table>
<thead>
<tr>
<th>Category group</th>
<th>Avg width</th>
<th>Hits</th>
<th>% hits</th>
<th>Avg PNG</th>
<th>% bw</th>
<th>Avg SVGZ</th>
<th>% bw</th>
<th>SVG bloat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flags</td>
<td>30px</td>
<td>101,119</td>
<td>39%</td>
<td>570</td>
<td>8%</td>
<td>10,173</td>
<td>24%</td>
<td>18x</td>
</tr>
<tr>
<td>Icons</td>
<td>32px</td>
<td>46,108</td>
<td>18%</td>
<td>1,681</td>
<td>11%</td>
<td>7,277</td>
<td>8%</td>
<td>4x</td>
</tr>
<tr>
<td>Logos</td>
<td>39px</td>
<td>49,676</td>
<td>19%</td>
<td>2,222</td>
<td>15%</td>
<td>7,511</td>
<td>9%</td>
<td>3x</td>
</tr>
<tr>
<td>Maps</td>
<td>192px</td>
<td>7,301</td>
<td>3%</td>
<td>25,599</td>
<td>25%</td>
<td>199,171</td>
<td>34%</td>
<td>8x</td>
</tr>
<tr>
<td>All</td>
<td>48px</td>
<td>257,035</td>
<td>100%</td>
<td>2,856</td>
<td>100%</td>
<td>16,669</td>
<td>100%</td>
<td>6x</td>
</tr>
</tbody>
</table>

3/4 of SVG views are simple images such as flags, logos, and other icons, usually rendered.
Map madness! WTF?
Measuring performance

● Look at your source

```xml
<g id="g2854" transform="matrix(0.522274, 0, 0, 0.522274, -504.923, 94.5014)" />
<g id="g2858" transform="matrix(0.522274, 0, 0, 0.522274, -504.923, 94.5014)" />
...
<g id="g2966" transform="matrix(0.522274, 0, 0, 0.522274, -504.923, 94.5014)" />
...
```

● The fastest element to render is the one that isn’t in your source
Measuring performance

• find the hot-spots in your app
• fight the urge to guess
Don't Touch the DOM

- DOM manipulation is generally expensive
- Precalculate values
- suspendRedraw
Why speed matters

- People hate to wait
- Fast loading = Good experience
- Higher search ranking
- Less bandwidth usage
- Profit
Basic Principles

- Reduce http requests
- Reduce filesize of assets being transferred:
  - optimize compression of bitmap images (incl. Smush.it™)
  - minify JavaScript files (e.g. YUI Compressor, Google Closure Tools)
- Use gzip compression on the server (e.g. Apache’s `mod_deflate` module)
- Aggressive caching (far future expire headers)
SVG embedded within HTML (because we can)
Building Blocks of a Thematic Map
How the blocks are built
for (i=0; i<(maxRegion); i++) {
    var newLine = document.createElementNS(svgns, 'line');
    newLine.setAttributeNS(null, 'x1', barWidth*i+10);
    newLine.setAttributeNS(null, 'y1', 100);
}
Backwards compatibility takes its toll

SVG Web provides Flash version from the same SVG markup for Internet Explorer 6, 7 & 8
Demo

http://vis.uell.net/svgopen/11/atlas.html
Loading the election map

Using the Firebug extension for Firefox
# Assets by filesize

<table>
<thead>
<tr>
<th>File</th>
<th>Usage</th>
<th>Size (gziped)</th>
</tr>
</thead>
<tbody>
<tr>
<td>svg.js</td>
<td>JavaScript library SVG Web for compatibility with legacy versions of Internet Explorer</td>
<td>26.1 KB</td>
</tr>
<tr>
<td>map.svgz</td>
<td>Geometry (boundary paths) of the electoral districts, derived from a shape file</td>
<td>17.1 KB</td>
</tr>
<tr>
<td>static_data.js</td>
<td>Election results of previous elections as well as demographic data</td>
<td>7.0 KB</td>
</tr>
<tr>
<td>programme.js</td>
<td>Program logic, e.g. reading of the data, statistics, classification methods, colouring of the map</td>
<td>6.3 KB</td>
</tr>
<tr>
<td>atlas.html</td>
<td>Main HTML file, contains results table, key and dropdown menus</td>
<td>3.1 KB</td>
</tr>
<tr>
<td>realtime_data.txt</td>
<td>Updated results from the current election throughout election night – loaded via XMLHttpRequest</td>
<td>2.3 KB</td>
</tr>
</tbody>
</table>
Aggressive caching

Example .htaccess file for an Apache Webserver

ExpiresActive On
ExpiresDefault "access plus 300 seconds"
ExpiresByType text/javascript "access plus 1 week 1 hour"
ExpiresByType application/x-javascript "access plus 1 week 1 hour"
ExpiresByType image/svg+xml "access plus 1 year 1 hour"
Measuring Performance with the YSlow Extension

<table>
<thead>
<tr>
<th>TYPE</th>
<th>SIZE (KB)</th>
<th>GZIP (KB)</th>
<th>URL</th>
<th>EXPIRES (Y/M/D)</th>
<th>RESPONSE TIME (ms)</th>
<th>ETAG</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>9.2K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>js</td>
<td>147.1K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>js</td>
<td>107.8K</td>
<td>26.7K</td>
<td><a href="http://vis.ueell.net/svgweb/r1320/svg.js">http://vis.ueell.net/svgweb/r1320/svg.js</a></td>
<td>2012/2/9</td>
<td>4</td>
<td>&quot;6a002b3-1a51a-498e413fa9f80&quot;</td>
<td></td>
</tr>
<tr>
<td>js</td>
<td>17.6K</td>
<td>7.1K</td>
<td><a href="http://vis.ueell.net/svgopen/11/static_data.js">http://vis.ueell.net/svgopen/11/static_data.js</a></td>
<td>2011/8/20</td>
<td>5</td>
<td>&quot;6ba10b7-44f9-4aa5fa2653780&quot;</td>
<td></td>
</tr>
<tr>
<td>js</td>
<td>21.7K</td>
<td>6.4K</td>
<td><a href="http://vis.ueell.net/svgopen/11/programme.js">http://vis.ueell.net/svgopen/11/programme.js</a></td>
<td>2011/8/20</td>
<td>5</td>
<td>&quot;6ba10b5-54e9-4aa5fc20ba240&quot;</td>
<td></td>
</tr>
<tr>
<td>image</td>
<td>38.0K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>image</td>
<td>38.0K</td>
<td>17.5K</td>
<td><a href="http://vis.ueell.net/svgopen/11/map.svgz">http://vis.ueell.net/svgopen/11/map.svgz</a></td>
<td>2012/8/12</td>
<td>6</td>
<td>&quot;6ba10b8-4481-4aa5fa2653780&quot; smush.it</td>
<td></td>
</tr>
<tr>
<td>xhr</td>
<td>6.2K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# How browsers cache .svgz

## Table 1: Cache Behavior for .html and .js Files

<table>
<thead>
<tr>
<th>Name</th>
<th>Path</th>
<th>Method</th>
<th>Status Text</th>
<th>Type</th>
<th>Size Transfer</th>
<th>Time Latency</th>
<th>Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>atlas.html</td>
<td>/svgopen/11</td>
<td>GET</td>
<td>200 OK</td>
<td>text/html</td>
<td>8.99KB, 3.39KB</td>
<td>55ms, 45ms</td>
<td></td>
</tr>
<tr>
<td>svg.js</td>
<td>/Svgweb/1320</td>
<td>GET</td>
<td>304 Not Modified</td>
<td>text/plain</td>
<td>105.28KB, 212B</td>
<td>50ms, 50ms</td>
<td></td>
</tr>
<tr>
<td>static_data.js</td>
<td>/svgopen/11</td>
<td>GET</td>
<td>304 Not Modified</td>
<td>text/plain</td>
<td>17.24KB, 206B</td>
<td>68ms, 68ms</td>
<td></td>
</tr>
<tr>
<td>programme.js</td>
<td>/svgopen/11</td>
<td>GET</td>
<td>304 Not Modified</td>
<td>text/plain</td>
<td>21.23KB, 206B</td>
<td>79ms, 79ms</td>
<td></td>
</tr>
<tr>
<td>map.svgz</td>
<td>/svgopen/11</td>
<td>GET</td>
<td>200 OK</td>
<td>image/svg+xml</td>
<td>37.18KB, 17.45KB</td>
<td>27ms, 19ms</td>
<td></td>
</tr>
<tr>
<td>realtime_data.txt</td>
<td>/svgopen/11</td>
<td>HEAD</td>
<td>200 OK</td>
<td>text/plain</td>
<td>6B, 2.67KB</td>
<td>60ms, 68ms</td>
<td></td>
</tr>
<tr>
<td>realtime_data.txt</td>
<td>/svgopen/11</td>
<td>GET</td>
<td>200 OK</td>
<td>text/plain</td>
<td>6.07KB, 2.67KB</td>
<td>3ms, 1ms</td>
<td></td>
</tr>
</tbody>
</table>

**7 requests | 26.78KB transferred**

## Table 2: Cache Behavior for .xml Files

<table>
<thead>
<tr>
<th>Name</th>
<th>Path</th>
<th>Method</th>
<th>Status Text</th>
<th>Type</th>
<th>Size Transfer</th>
<th>Time Latency</th>
<th>Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>atlas.html</td>
<td>/svgopen/11</td>
<td>GET</td>
<td>304 Not Modified</td>
<td>text/html</td>
<td>235B, 9.08KB</td>
<td>38ms, 37ms</td>
<td></td>
</tr>
<tr>
<td>svg.js</td>
<td>/Svgweb/1320</td>
<td>GET</td>
<td>304 Not Modified</td>
<td>application/xhtml+xml</td>
<td>241B, 105.28KB</td>
<td>22ms, 22ms</td>
<td></td>
</tr>
<tr>
<td>static_data.js</td>
<td>/svgopen/11</td>
<td>GET</td>
<td>304 Not Modified</td>
<td>application/xhtml+xml</td>
<td>238B, 17.24KB</td>
<td>27ms, 27ms</td>
<td></td>
</tr>
<tr>
<td>programme.js</td>
<td>/svgopen/11</td>
<td>GET</td>
<td>304 Not Modified</td>
<td>application/xhtml+xml</td>
<td>238B, 21.23KB</td>
<td>32ms, 32ms</td>
<td></td>
</tr>
<tr>
<td>map.svgz</td>
<td>/svgopen/11</td>
<td>GET</td>
<td>(from cache)</td>
<td>image/svg+xml</td>
<td>5ms, 5ms (from cache)</td>
<td>5ms, 5ms</td>
<td></td>
</tr>
<tr>
<td>realtime_data.txt</td>
<td>/svgopen/11</td>
<td>HEAD</td>
<td>200 OK</td>
<td>text/plain</td>
<td>2.69KB, 0B</td>
<td>21ms, 21ms</td>
<td></td>
</tr>
<tr>
<td>realtime_data.txt</td>
<td>/svgopen/11</td>
<td>GET</td>
<td>(from cache)</td>
<td>text/plain</td>
<td>1ms, 1ms (from cache)</td>
<td>1ms, 1ms</td>
<td></td>
</tr>
</tbody>
</table>

**7 requests | 3.62KB transferred | 163ms (onload: 96ms, DOMContentLoaded: 100ms)**
Measuring performance

- Look at your source

  `<g id="g2854" transform="matrix(0.522274, 0, 0, 0.522274, -504.923, 94.5014)"/>
  `<g id="g2858" transform="matrix(0.522274, 0, 0, 0.522274, -504.923, 94.5014)"/>
  ...
  `<g id="g2966" transform="matrix(0.522274, 0, 0, 0.522274, -504.923, 94.5014)"/>
  ...

- The fastest element to render is the one that isn’t in your source
Map madness!

WTF?

Wikimedians love to err on the side of too much detail! For raster images this is fantastic -- a
Level of Detail

Tools and workflows to optimize path data
Optimizing paths: Illustrator
Effects of path oversimplification

Original

Simplified
gaps highlighted
Optimizing paths: Inkscape
Precision settings: Illustrator

- SVG Profiles: SVG 1.1
- Fonts: Type: Adobe CEF, Subsetting: None (Use System Fonts)
- Images: Location: Embed
- Preserve Illustrator Editing Capabilities
- CSS Properties: Presentation Attributes
- Decimal Places: 3, Encoding: Unicode (UTF-8)
- Original File: 140 KB
- 2 Decimal Places: 121 KB
- 1 Decimal Place: 101 KB
Precision settings: Inkscape
@erikdahlstrom

Erik Dahlström

just discovered that Inkscape can save "optimized svg" (aka clean & web-ready), with viewBox etc... File > Save as > Optimized svg

28 Sep via BitlBee  ⭐ Favorite  ✅ Undo Retweet  ⬅️ Reply

Retweeted by mac
Optimized SVG via Inkscape
Inkscape’s Optimized SVG Output requires lxml

Not installed by default on MacOSX

Inkscape has received additional data from the script executed. The script did not return an error, but this may indicate the results will not be as expected.

The fantastic lxml wrapper for libxml2 is required by inkex.py and therefore this extension. Please download and install the latest version from http://cheeseshop.python.org/pypi/lxml/, or install it through your package manager by a command like: sudo apt-get install python-lxml
Adjacency issues are not the fault of Illustrator or Inkscape
Whenever possible start from a shapefile
MapShaper, a free, interactive online Shapefile editor
Converting Shapefiles to SVG

perl ogis2svg.pl --input yourshapename --output yourshapename.svg --roundval 0.1
Future Improvements?

• Tile based vector maps

• Loading detail on demand
  (cf. progressive loading of jpeg)

• Should a future SVG path element be able
to represent shared boundaries?

Michael Neutze
http://vis.uell.net/