rgi | GEOSPATIAL

Your Bundle is Bad and You Should Feel Bad

Luke Rees
Software Engineer
luke.rees@rgi-corp.com

16 May 2018

About Me

- Geography undergrad turned developer
- Full stack/Front-end developer @ Reinventing Geospatial for ~2 years
- Django, React, Redux and all the friends :)
- First time at FOSS4G (be nice plz)
- Purveyor of bad jokes

https://github.com/lukerees/FOSS4GNA-2018

webpack

A Brief Introduction. . .

Introduction to webpack

- What is webpack?
  - Webpack is a static module bundler for (modern) JavaScript applications
- Okay thanks, but what is a bundler?
  - A bundler “bundles” all (or most) of your application files into one common file
- Why do I want to use a bundler?
  - Because it allows you to link only one js file rather than serve out an entire folder of dependencies + core code

```
<script src="https://cdn.polyfill.io/v2/polyfill.min.js?features=requestAnimationFrame,Element.prototype.classList,URL"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.5/lodash.min.js"></script>
<script src="https://openlayers.org/en/v4.6.5/build/ol.js"></script>
<script src="https://api.tiles.mapbox.com/mapbox.js/plugins/turf/v2.0.0/turf.min.js"></script>
```

VS

```
<script src="build/bundle.js"></script>
```

Sweet Sweet Bundle

- @turf/turf@4.7.3
- babel-core@6.26.0
- babel-eslint@7.2.3
- babel-jest@20.0.3
- babel-loader@7.1.2
- babel-preset-react-app@3.1.0
- babel-runtime@6.26.0
- case-sensitive-paths-webpack-plugin@2.1.1
- css-loader@0.28.7
- dotenv@4.0.0
- eslint@4.10.0
- eslint-config-airbnb@16.1.0
- eslint-config-react-app@2.0.1
- eslint-loader@1.9.0
- eslint-plugin-flowtype@2.39.1
- eslint-plugin-import@2.8.0
- eslint-plugin-jsx-a11y@5.1.1
- eslint-plugin-react@7.4.0
- extract-text-webpack-plugin@3.0.2
- file-loader@1.1.5
- fs-extra@3.0.1
- html-webpack-plugin@2.29.0
- jest@20.0.4
- jsts@1.4.0
- jsts-es@1.5.4
- material-ui@0.19.4
- moment@2.19.1
- object-assign@4.1.1
- ol@4.4.2
- postcss-flexbugs-fixes@3.2.0
- postcss-loader@2.0.8
- promise@8.0.1
- raf@3.4.0
- react@16.0.0
- react-dev-utils@4.2.1
- react-dom@16.0.0
- style-loader@0.19.0
- sw-precache-webpack-plugin@0.11.4
- url-loader@0.6.2
- webpack@3.8.1
- webpack-dev-server@2.9.3
- webpack-manifest-plugin@1.3.2

app.css
app.js
index.css
index.js
rgi-logo.jpg

bundle.js

Core webpack Concepts

- Entry
- Output
- Loaders
- Plugins

https://github.com/lukerees/FOSS4GNA-2018

Entry

- The entry point tells webpack where to get started processing your application.
- From the entry point webpack recurses through your project files and identifies all dependencies used.
- This step determines what webpack with try to bundle up for you, and what it will leave out.
  - Unit test files, for example, are typically left out since nowhere in your application code would you ‘import’ from test file. Without an import webpack doesn’t even care the test files exist.
- Multiple entry points are permitted.
  - Could be used for loading polyfills or “vendoring” external dependencies into separate bundles

```
const config = {
  entry: './my_app/src/index.js'
}

module.exports = config;
```

Output

- The output defines what name your bundle will have and what directory it will be placed in.
- If you are using multiple entry points you will want to name your output files dynamically. Webpack has several options for injecting unique values for each corresponding entry (id, name, hash).

```
const config = {
  entry: './my_app/src/index.js',
  output: {
    path: './my_app/static/build',
    filename: 'bundle.js',
  }
}

module.exports = config;
```

Loaders

- Loader allow you to bundle non standard files and convert syntax.
- What are some useful loaders?
  - babel-loader: Write in ES6 and JSX (React) and transpile to ES5.
  - css-loader & style-loader: Import your stylesheets directly in js.
  - url-loader: Include images as inline strings in bundle rather than separate assets.

```
module: {
  rules: [
      {
          test: /\.js$/,
          exclude: [/node_modules\/(?!jsts)/, /staticfiles/,
          use: {
              loader: 'babel-loader',
              options: {
                  presets: ['es2015', 'react']
              }
          }
      },
      {
          test: /\.css$/,
          use: [
              { loader: 'style-loader' },
              { loader: 'css-loader' }
          ]
      },
      {
          test: /\.(svg|png|jpg|gif)$/,
          use: {
              loader: 'url-loader',
              options: {
                  limit: '100000'
              }
          }
      },
  ],
}
```

Plugins

- Plugins allow for custom behavior during module processing
- There are a wide range of really good plugins out there, or you can create your own
- The use of plugins takes webpack from convenient to very powerful.
- What are some useful plugins?
  - DefinePlugin: Set global env variables
  - UglifyJsPlugin: Minification and “tree shaking”
  - CommonChunksPlugin: Extract duplicated modules
  - BundleAnalyzerPlugin: More on this later . . .

```
plugins: [
  new webpack.DefinePlugin({'process.env.NODE_ENV': "'production'"}),
  new webpack.optimize.UglifyJsPlugin({
      compress: { warnings: false },
      sourceMap: false,
  }),
  new webpack.optimize.CommonsChunkPlugin({
      name:'node-modules',
      filename: 'node-modules.js',
      minChunks(module, count) {
          var context = module.context;
          return context && context.indexOf('node_modules') >= 0;
      }
  }),
  new BundleAnalyzerPlugin({
      analyzerMode: 'static',
      reportFilename: 'report.html'
  }),
]
```

Lessons learned

What to do and what not to do

Don’t Dev my Prod

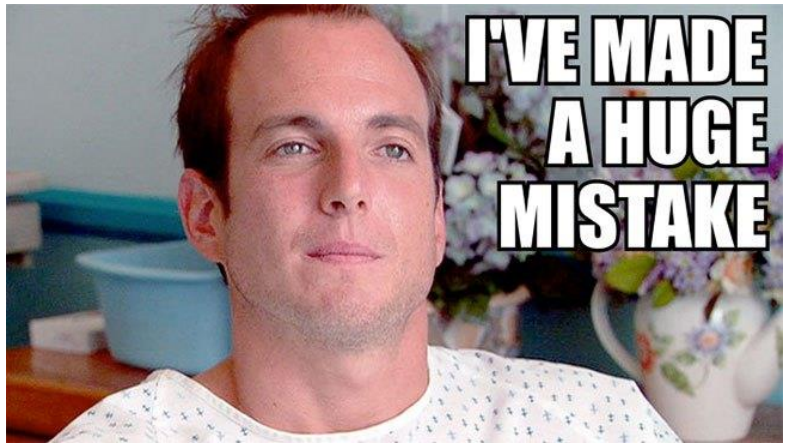
- Production and development configurations can look very different
  - Do you want to use devtool?
  - How about the devServer?
  - Are you minifiying for production?
  - Do you need to pass in a production environment variable?
- Use env variable or different configuration files.

```
"build": "PROD=1 node_modules/webpack/bin/webpack.js",
```

OR

```
webpack.config.js
webpack.dev.config.js
```

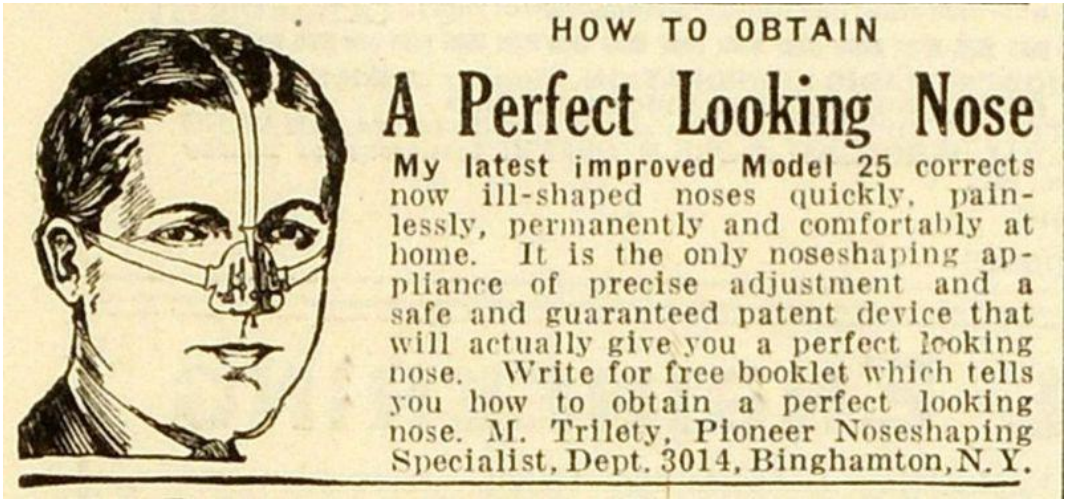
Deploying dev build to prod:



- Some libraries can be built for prod or dev specifically

It Hurts to be Beautiful

- Use UglifyJS or another minification library!
- Tree shaking is essential
  - Reduce size by eliminating unused code
- Minification has minimal impact, but it can add up
  - Using the mangle option in UglifyJS can help



https://github.com/lukerees/FOSS4GNA-2018

Don’t Be Mr. Worldwide

- Use *proper* ES6 import syntax
- Don’t do this

```
const _ = require('lodash');
const equal = _.isEqual(something, someOtherThing);
```

```
import _ from 'lodash';
const equal = _.isEqual(something, someOtherThing);
```

- You might be able to do this

```
import { isEqual } from 'lodash';
const equal = isEqual(something, someOtherThing);
```

- Do this

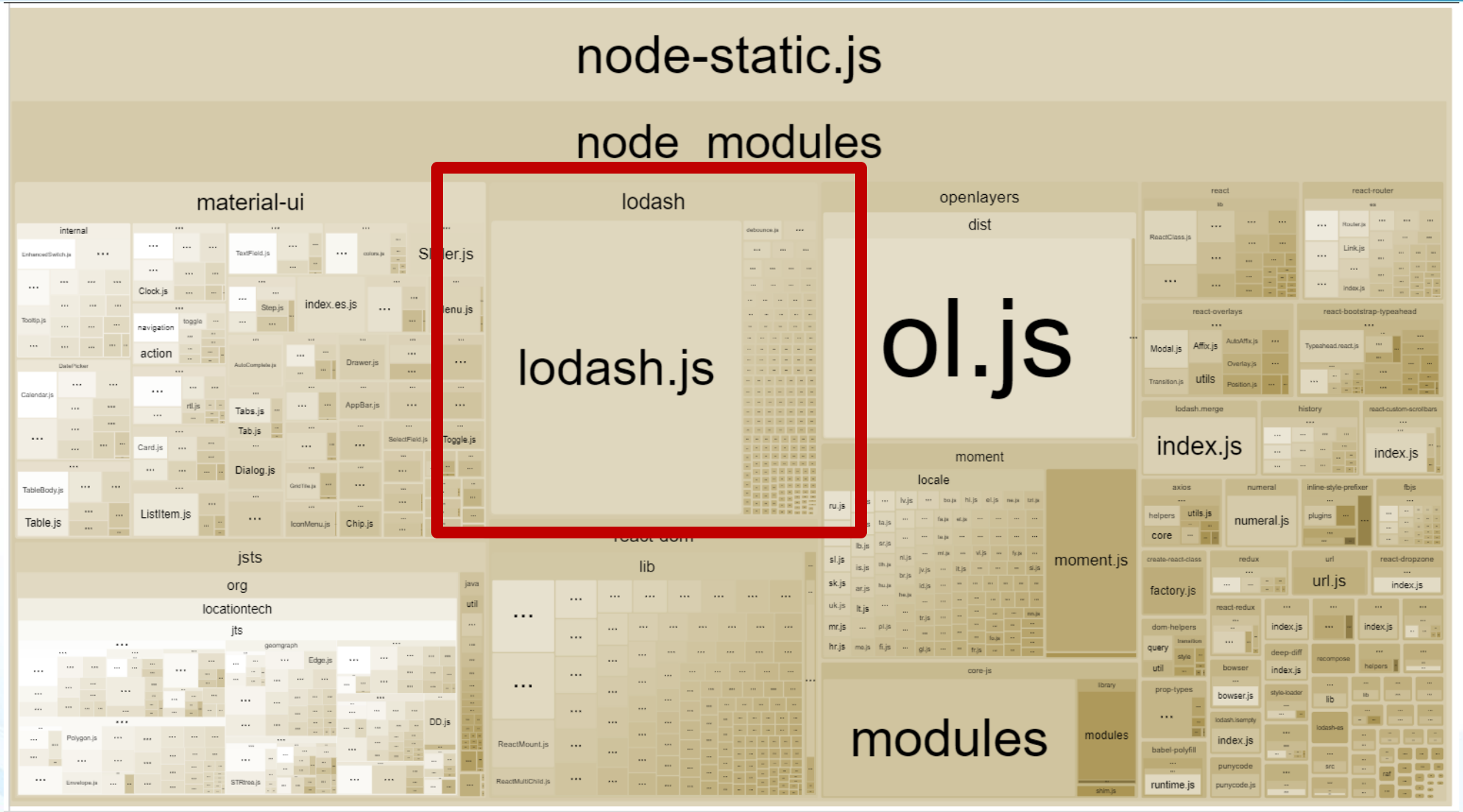
```
import isEqual from 'lodash/isEqual';
const equal = isEqual(something, someOtherThing);
```

Don’t be this guy ^

Bundle Analyzers are

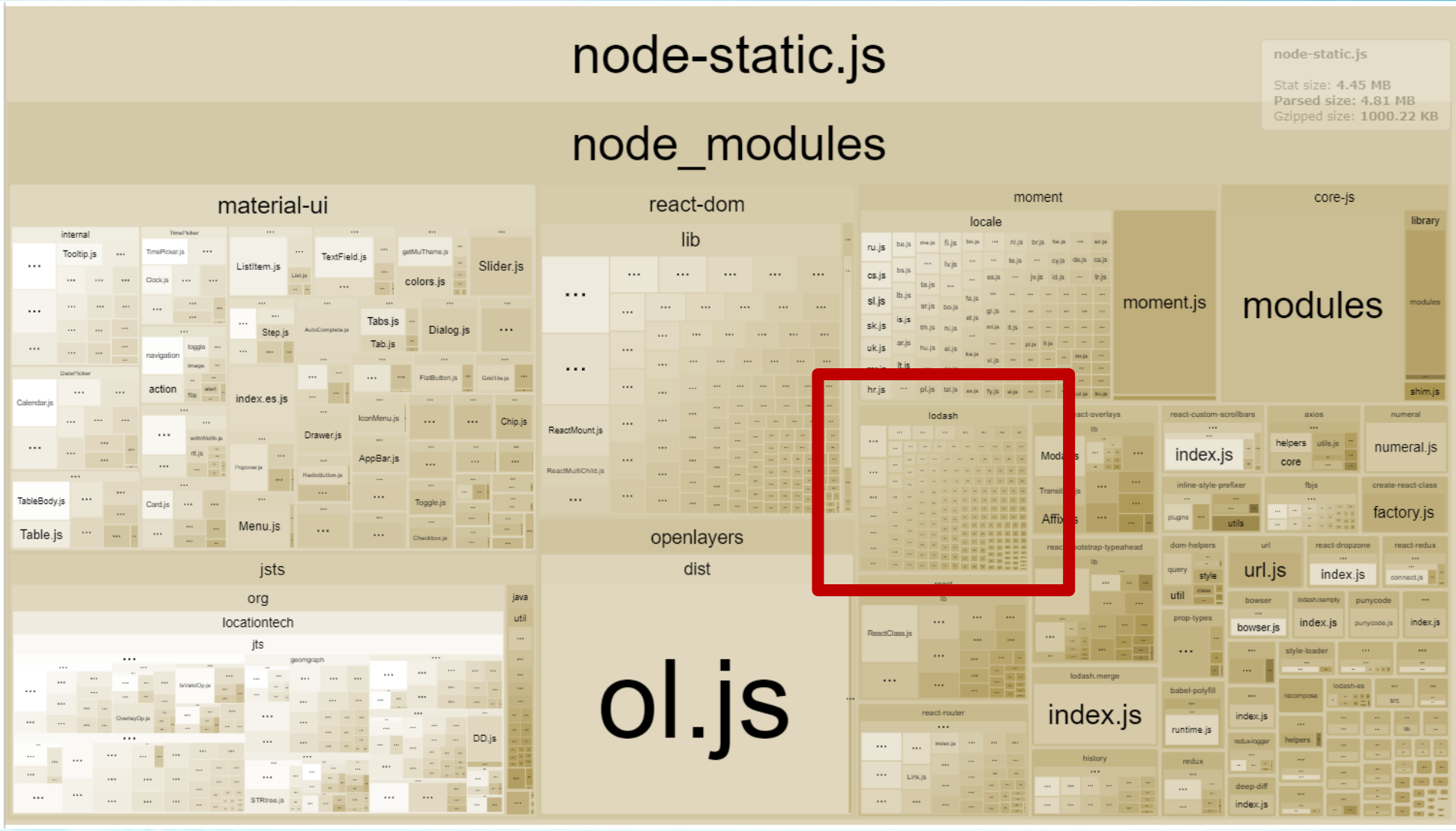
- Not a requirement, but a strong suggestion
  - Be me
  - Go to meetup
  - Hear speaker mention webpack-bundle-analyzer
  - Next day at work install analyzer
  - MFW global lodash import

Bundle Analyzers cont.



https://github.com/lukerees/FOSS4GNA-2018

Bundle Analyzers cont.



https://github.com/lukerees/FOSS4GNA-2018

The Dependency of My Dependency is My Dependency

- You should be aware of not only your immediate dependencies, but also the entire dependency tree!
- If I run "npm install --save ol", how many packages am I actually installing?
  - a. One
  - b. Five
  - c. Eight
  - d. Who cares your gonna use it either way

```
test@1.0.0 C:\dev\test
`-- ol@4.6.5
  +-- pbf@3.1.0
  | +-- ieee754@1.1.11
  | `-- resolve-protobuf-schema@2.0.0
  |   `-- protocol-buffers-schema@2.2.0
  +-- pixelworks@1.1.0
  `-- rbush@2.0.1
    `-- quickselect@1.1.1
```

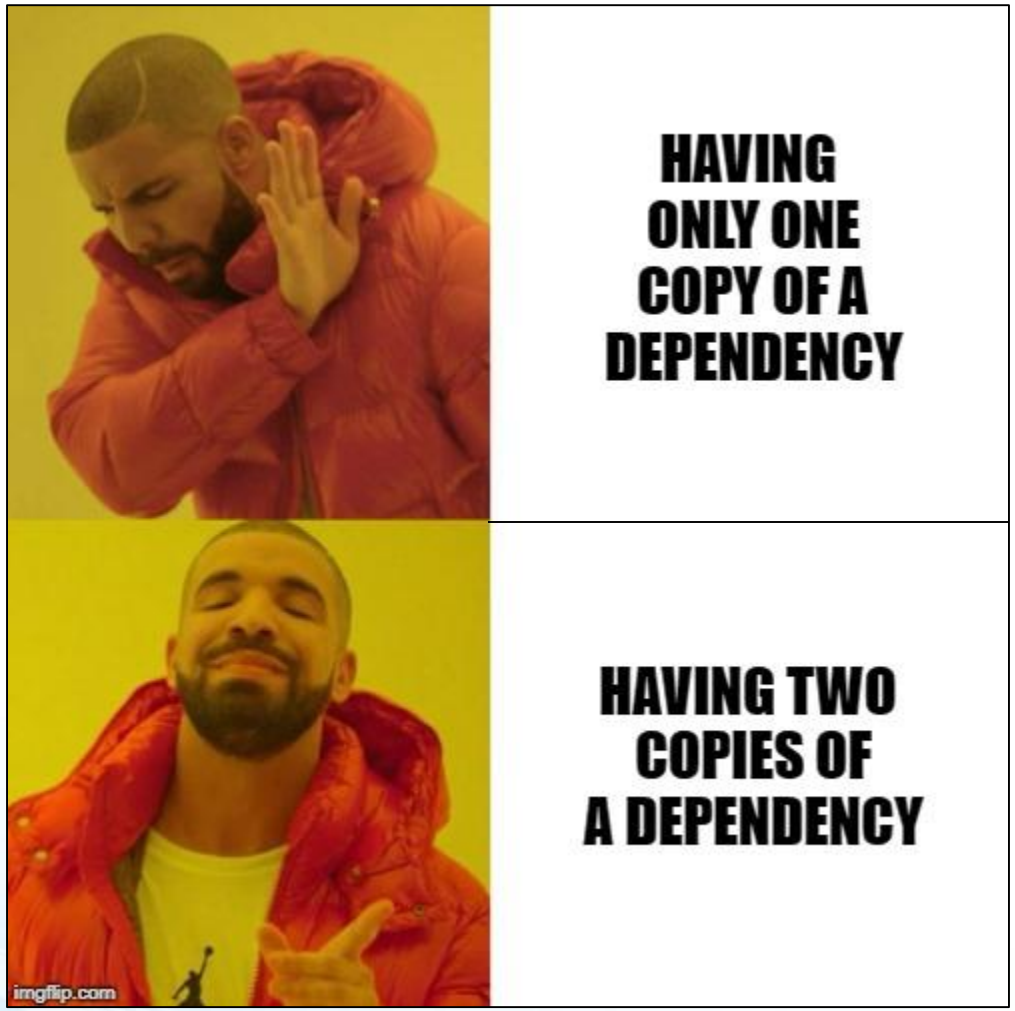
- Do you like JSTS?

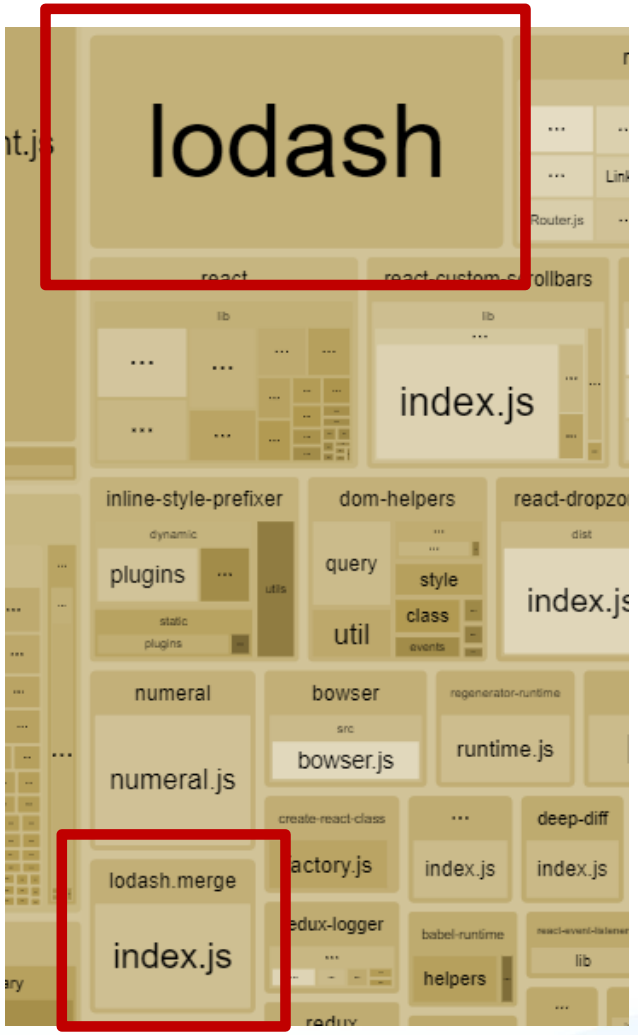
  How about Turf?

  Why not have both by installing Turf :p

- Just because I only install and import lodash.merge doesn't mean I don't have all of lodash bundled . . .

The Dependency of My Dependency is My Dependency





https://github.com/lukerees/FOSS4GNA-2018

Modernity in Development (ooh that sounds fancy)

- Not all libraries will be a perfect fit for your modern webapp :/
  - Openlayers has two libraries: openlayers and ol.
  - One exports individual ES6 modules, the other must be used as a global variable.

    (remember that thing about not being Mr. Worldwide?)
  - Some libraries may need additional processing
  - JSTS had to be transpiled using the babel-loader

The End

Questions? Comments? Painfully obvious mistakes?