PostCSS

is for everybody

Hi, I'm Nathan


CSS

Remember 1996?

```
/* CSS level 1 */

DIV P            { font: small sans-serif }
.reddish H1      { color: red }
#x78y CODE       { background: blue }
DIV.sidenote H1 { font-size: large }
P {
  color: black;
  background: white;
  border: solid;
}
```

CSS Zen Garden

Dave Shea in 2003

```
/* css Zen Garden default style ... */
/* ... 2003, Dave Shea */
/* Added: May 7th, 2003 */
header h1 {
  background: transparent url(h1.gif) no-repeat top left;
  margin-top: 10px;
  display: block;
  width: 219px;
  height: 87px;
  float: left;

  text-indent: 100%;
  white-space: nowrap;
  overflow: hidden;
}
/* etc, etc, etc */
```

CSS is declarative

Read: *repetative*

```
.sidebar a { color: hotpink }

.widget .title { color: hotpink }
.widget .copy { font-size: 15px; }
.widget .cta { color: hotpink }

.header { font-family: Garamond, sans-serif; }
.fancy-content p { font-family: Garamond, sans-serif; }
.footer .about-us { font-family: Garamond, sans-serif; }
```

But we fixed that

with Sass

2005, Ruby

```
$accent: hotpink;
$dope-font: Garamond, sans-serif;

@mixin accent-type {
  font-family: $dope-font;
  font-style: italic;
}

.sidebar a { color: $accent }

.widget {
  .title { color: $accent }
  .copy { font-size: 15px; }
  .cta { color: $accent }
}

.header,
.fancy-content p,
.footer .about-us {
  @include accent-type;
}
```

then Less

2009, JavaScript

```
@accent: hotpink;
@dope-font: Garamond, sans-serif;

.accent-type () {
  font-family: @dope-font;
  font-style: italic;
}

.sidebar a { color: @accent }

.widget {
  .title { color: @accent }
  .copy { font-size: 15px; }
  .cta { color: @accent }
}

.header,
.fancy-content p,
.footer .about-us {
  .accent-type
}
```

then Stylus

2011, JavaScript

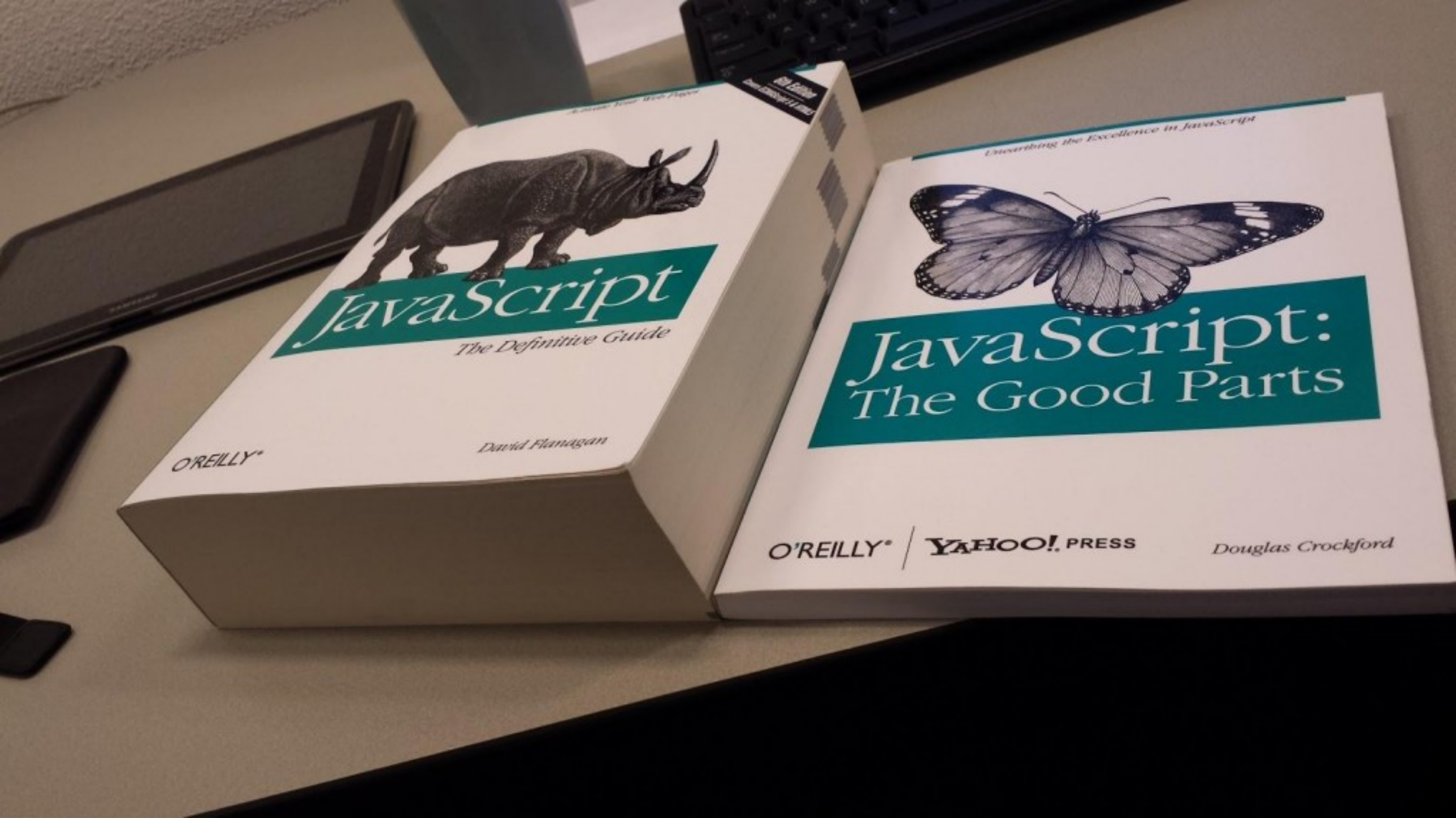
```
accent = hotpink
dope-font = Garamond, sans-serif

accent-type()
  font-family dope-font

.sidebar a
  color accent

.sidebar a
  color accent

.widget
  .title
    color accent
  .copy
    font-size 15px
  .cta
    color accent

.header,
.fancy-content p,
.footer .about-us
  accent-type()
```

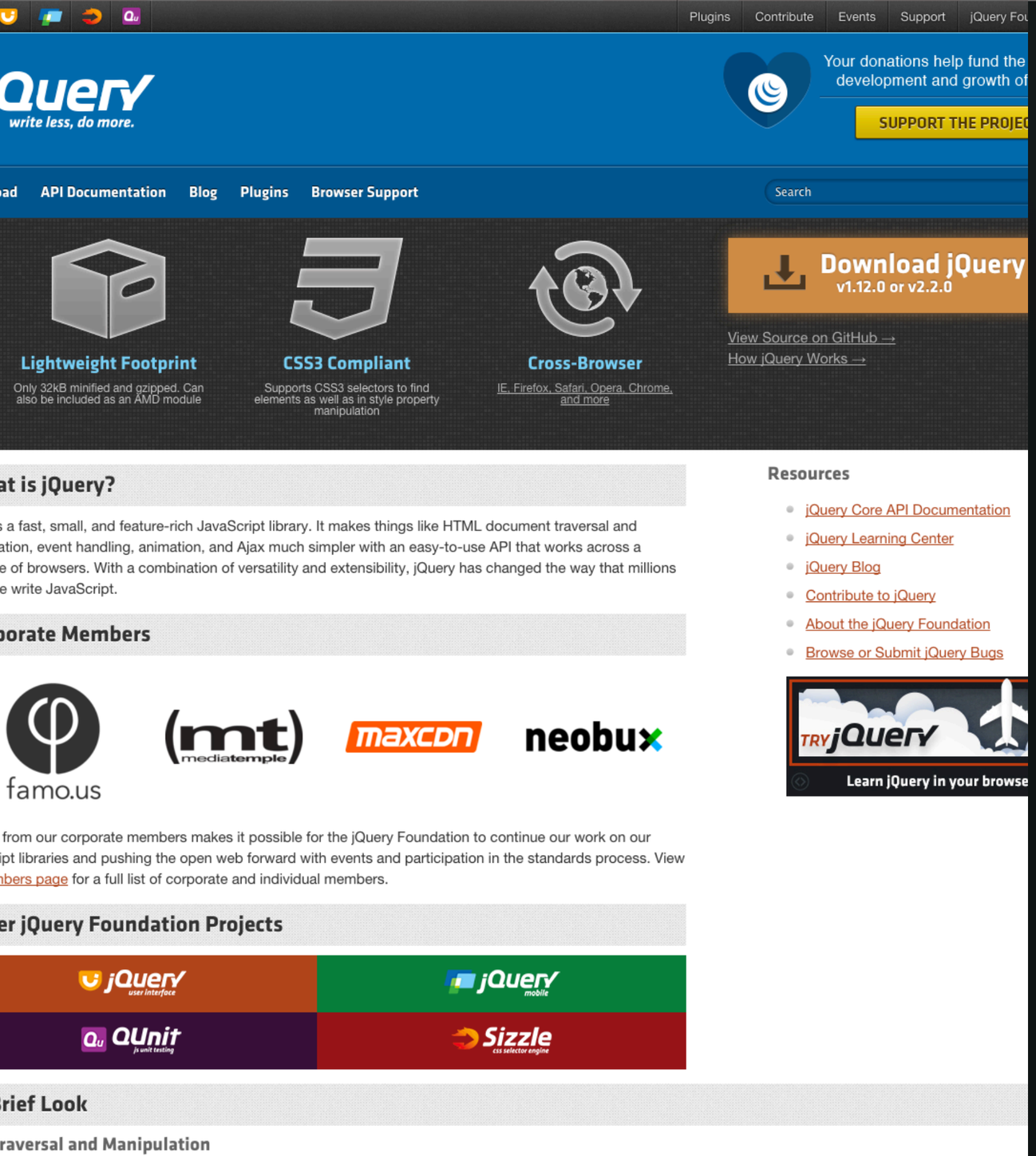
So problem solved, right?

Sass, Less, and Stylus all have

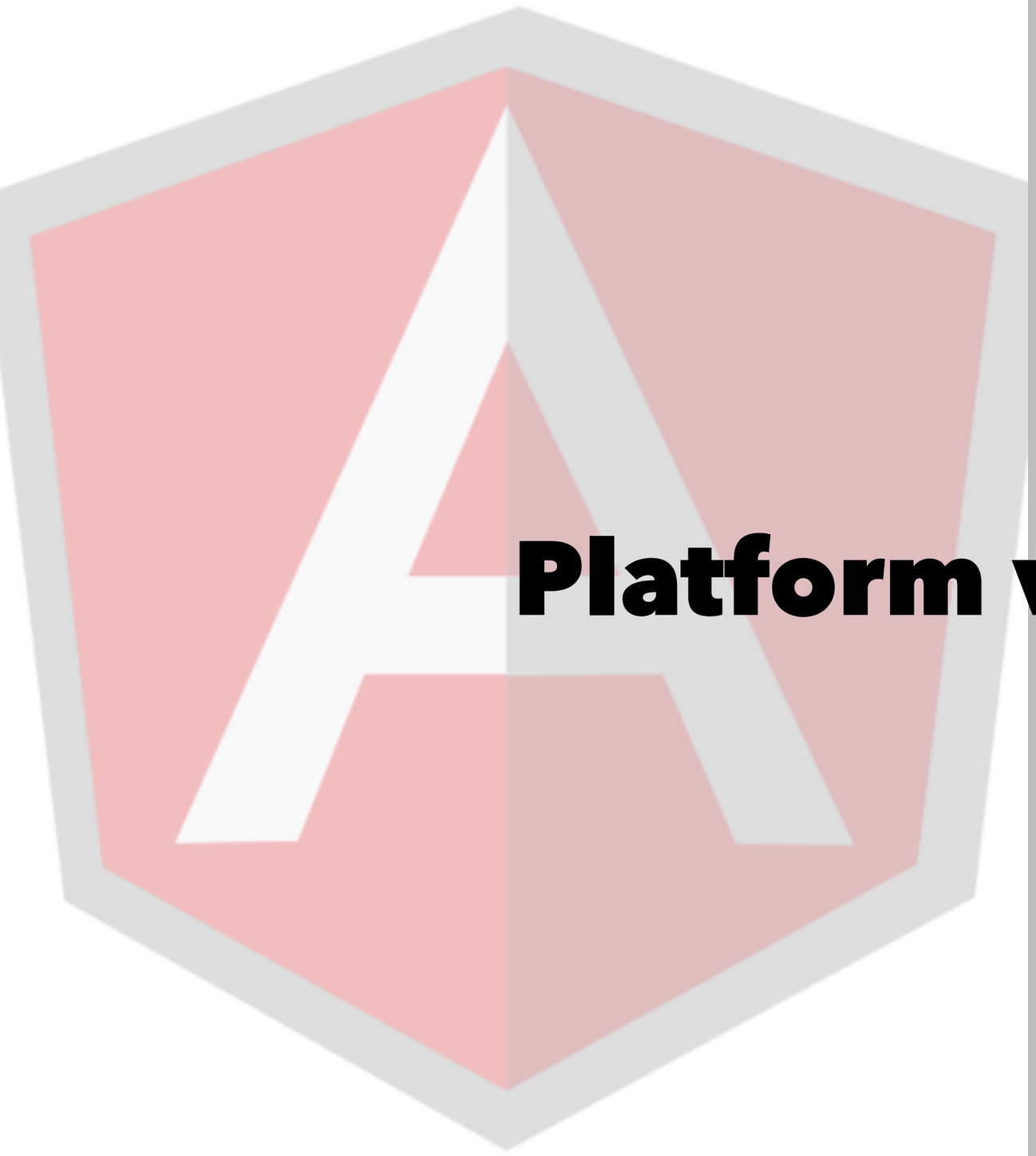
- Variables
- Inheritance
- Mixins
- Importing
- Nesting

And we tried to use *ALL* the features for a while

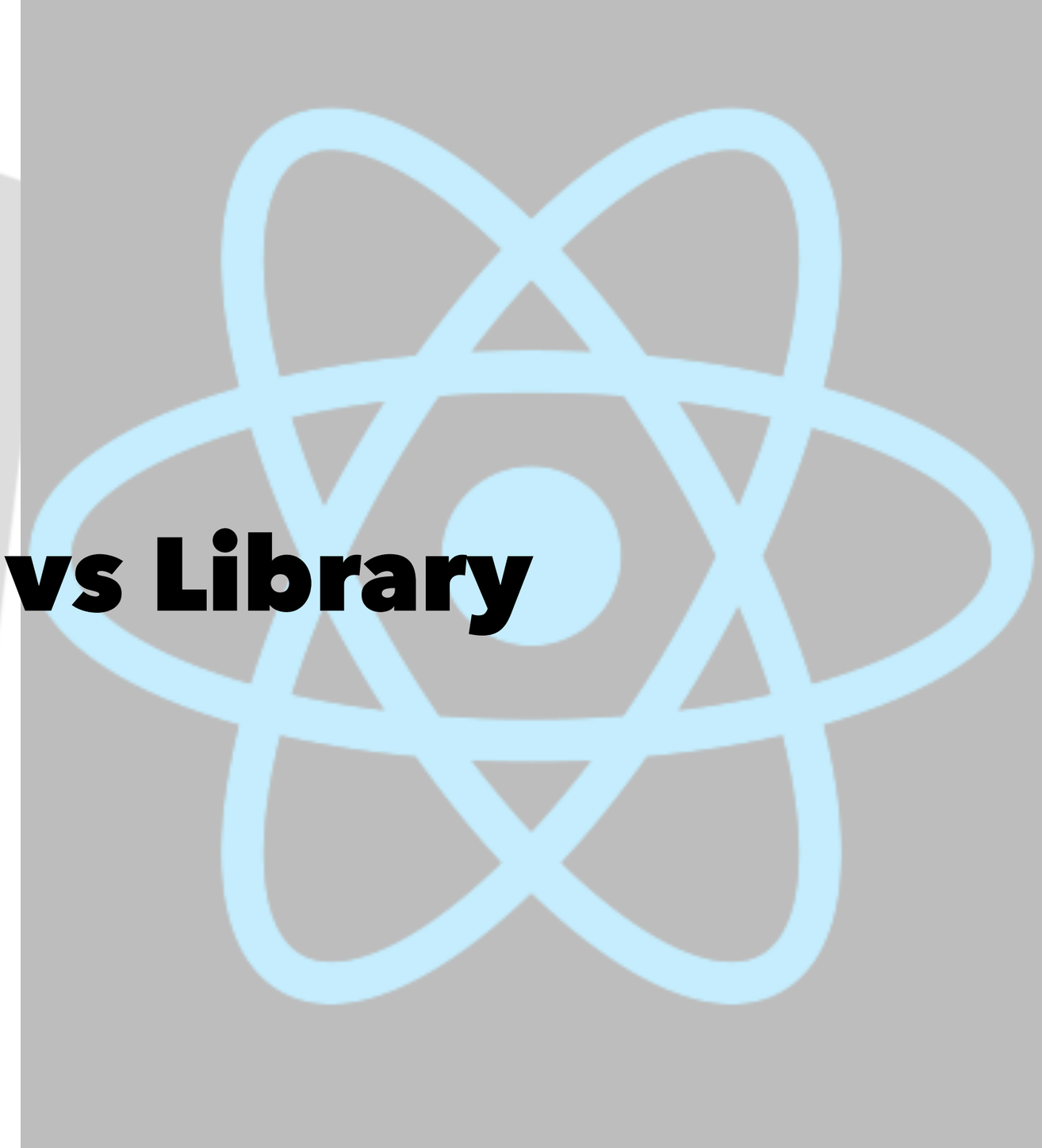

JavaScript
The Definitive Guide
David Flanagan
O'REILLY
Unearthing the Excellence in JavaScript
JavaScript:
The Good Parts
O'REILLY
YAHOO! PRESS
Douglas Crockford


Plugins
Contribute
Events
Support
Query
write less, do more.
Your donations help fund the development and growth of
SUPPORT THE PROJEC
API Documentation
Blog
Plugins
Browser Support
Search
Download jQuery
v1.12.0 or v2.2.0
View Source on GitHub →
How jQuery Works →
Lightweight Footprint
Only 32kB minified and gzipped. Can also be included as an AMD module
CSS3 Compliant
Supports CSS3 selectors to find elements as well as in style property manipulation
Cross-Browser
IE, Firefox, Safari, Opera, Chrome, and more
at is jQuery?
a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and
ation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a
e of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions
e write JavaScript.
porate Members
famo.us
mediatemple
maxCDN
neobux
from our corporate members makes it possible for the jQuery Foundation to continue our work on our
ipt libraries and pushing the open web forward with events and participation in the standards process. View
nbers page for a full list of corporate and individual members.
er jQuery Foundation Projects
jQuery user interface
jQuery mobile
QUnit
Sizzle
Brief Look
raversal and Manipulation
Resources
jQuery Core API Documentation
jQuery Learning Center
jQuery Blog
Contribute to jQuery
About the jQuery Foundation
Browse or Submit jQuery Bugs
TRY jQuery
Learn jQuery in your browse


microjs
I need ...
Tweet
Tweet-Templ
function t(s,d){for(var p in d)s=s.replace(new RegExp('{'+p+'}','g'),d[p]);return s;}
0.1 kB
SubtleLocationProxy
14 1
Proxy the location of one window, frame or iframe to the hash of another and
0.8 kB
Histogram
34 6
Provides a histogram data structure from a PNG/JPEG/GIF image path. NodeJS, AMD module and vanilla JS support
0.8 kB
blob-util
82 9
Utilities for working with Blob objects in the browser
3.1 kB
rounding.js
0 0
Exact rounding with a choice of rounding algorithms
1.1 kB
imago.js
26 2
It's a wonderful image library! With the purpose to facilitate the manipulation of images, imago.js enables you to perform
2.1 kB
Metamorph.js
253 28
Metamorph.js is a library that allows you to create a string of HTML, insert it into the DOM, and
1.6 kB
Simple, lightweight routing for web browsers



Platform vs Library



What if we don't want to use all the features of a Preprocessor?

How do we add features?

The evolution of Preprocessors

TJ had an idea

... Rework caters to a different audience, and provide you the flexibility to craft the preprocessor you want, not the choices the author(s) have force on you.

– TJ Holowaychuk

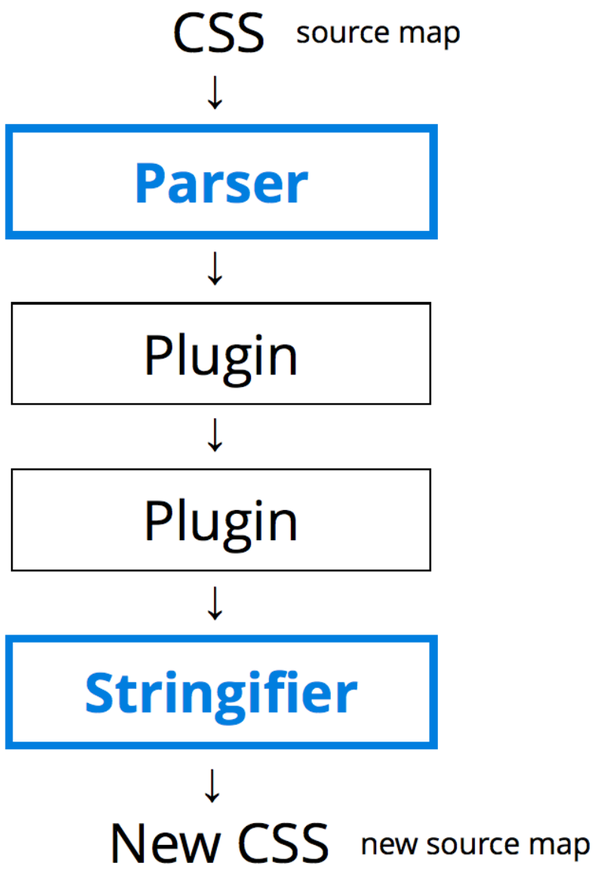
modular-css-preprocessing-with-rework

Andrey took it and ran with it

Because Rework was a proof of concept, it was the first generation of modular CSS processing. With PostCSS we have a better parser, a better API, and better source map support.

– Andrey Sitnik

So how does it work?



So how does it work?

Using PostCSS

```
const postcss = require('postcss');

postcss([ plugin1, plugin2 ])
    .process(css)
    .then( result => {
      /* do something with result.css */
    });
```

So how does it work?

writing a plugin

```
// a PostCSS plugin
function (css) {
  css.walkRules( rule => {
      rule.walkDecls( decl => {
        // manipulate decl.value here
      });
  });
};
```

Replacing Sass

- postcss-simple-vars
- postcss-nested
- postcss-mixins
- postcss-custom-media
- postcss-color-function

Take it to the future!

Use CSSNext

PostCSS-cssnext is a PostCSS plugin that helps you to use the latest CSS syntax today

CSSNext

- custom properties and `var()`
- custom media queries and media query ranges
- custom selectors
- `:any-link`, `:matches`, `:not`

Plugins for Fallbacks

- **cssgrace** - provide fallbacks to IE7
- **autoprefixer** - never type a vendor prefixes again

cssgrace

```
.icon {
    opacity: 0.6;
    display: inline-block;
}
```

cssgrace

```
.icon {
    opacity: 0.6;
    filter: alpha(opacity=60);
    display: inline-block;
    *display: inline;
    *zoom: 1;
}
```

autoprefixer

```
.flexy {
  display: flex;
}

a {
  color: hotpink;
  transition: color 0.15s ease;
}
```

autoprefixer

```
.flexy {
  display: -webkit-box;
  display: -webkit-flex;
  display: -ms-flexbox;
  display: flex;
}

a {
  color: hotpink;
  -webkit-transition: color 0.15s ease;
  transition: color 0.15s ease;
}
```

Plugins for
Optimization and Analysis

- **doiuse** - Lint CSS against caniuse database
- **postcss-colorblind** - check your colors
- **colorguard** - maintain a consistent color palette
- **cssnano** - minify css
- **postcss-stats** - get stats from cssstats.com

Plugins for
Pointless fun and Lolz

- **postcss-trolling** - inject pranks into your codebase
- **postcss-lolcat-stylesheets** - code like an angsty teen?

Need more plugins?

Create your own!

- Plugin Tutorials
- AST Explorer
- Well documented API

So let's hook it up

If you live in a perfect world with no tools

```
const postcss = require('postcss');

// node read files

postcss([
    require('autoprefixer'),
    require('postcss-import'),
    require('postcss-cssnext')
  ])
  .process(css)
  .then( result => {
    // node write files
  });
```

But you don't live in a world without build tools

So, Grunt

```
module.exports = function(grunt) {
  grunt.initConfig({
    postcss: {
      options: {
        processors: [
          require('autoprefixer')({browsers: ['last 4 versions']}),
          require('postcss-cssimport')(),
          require('postcss-cssnext')()
        ]
      },
      dist: {
        src: 'src/style.css',
        dest: 'dest/style.css'
      }
    }
  });
  grunt.loadNpmTasks('grunt-postcss');
};
```

And Gulp,

```
const postcss = require('gulp-postcss');

gulp.task('css', () => {
  gulp.src('src/style.css')
    .pipe(postcss([
      require('autoprefixer')({ browsers: ['last 4 versions'] }),
      require('postcss-cssimport'),
      require('postcss-nested')
    ]))
    .pipe(gulp.dest('dist/style.css'));
});
```

And also support for

- CLI: postcss-cli
- HTML: posthtml-postcss
- Stylus: poststylus
- Rollup: rollup-plugin-postcss
- Broccoli: broccoli-postcss
- Connect/Express: postcss-middleware
- *And various others, this list is too long*

But perhaps, you don't want to rewrite everything?

Use PostCSS with Sass

```
const gulp = require('gulp');
const postcss = require('gulp-postcss');
const sass = require('gulp-sass');

gulp.task('css', () => {
  return gulp.src('./src/*.scss')
    .pipe(sass().on('error', sass.logError))
    .pipe(postcss([ /* plugins */ ]))
    .pipe(gulp.dest('./dest'));
});
```

Use PostCSS with Less

```
const gulp = require('gulp');
const postcss = require('gulp-postcss');
const less = require('gulp-less');

gulp.task('css', () => {
  return gulp.src('./src/*.less')
    .pipe(less())
    .pipe(postcss([ /* plugins */ ]))
    .pipe(gulp.dest('./dest'));
});
```

Use PostCSS with Stylus

```
const gulp = require('gulp');
const stylus = require('gulp-stylus');
const poststylus = require('poststylus');

gulp.task('css', () => {
  return gulp.src('./src/*.styl')
    .pipe(stylus({
        use: [
            poststylus([ /* plugins */ ])
        ]
    }))
    .pipe(gulp.dest('./dest'));
});
```

The way of the future

- PostCSS is faster
- Lighter (less LOC) and more flexible
- The strength of modularity
- A natural evolution to authoring CSS
- *at least start using* ***Autoprefixer***

I'm on the Internet

- nathanstilwell.com
- @nathanstilwell
- google.com/#q=nathan+stilwell

Thanks for listening

Questions?