POSTCSS

What the *#$ is it?!?

PREPROCESSORS

Hopefully you're using one.

PREPROCESSORS

Give us some pretty powerful features to supercharge CSS

- Partials
- Variables
- Mixins
- Extends
- Nesting
- Loops
- Colour Goodies

PREPROCESSORS

Some of the reasons we love preprocessors:

- Reduce Repitition (DRY)
- Save time
- Make code consistent and easier to manage
- Helps keep things organised
- Pretty easy to setup
- Pretty easy to use
- Makes CSS much easier to work with on big projects

PREPROCESSORS ARE AWESOME

They help us work faster and smarter

AUTOPREFIXER

Automagically adds vendor prefixes to your css

```
.example {
    display: flex;
    transition: all .5s;
}
```

```
.example {
    display: -webkit-box;
    display: -webkit-flex;
    display: -ms-flexbox;
    display: flex;
    -webkit-transition: all .5s;
    transition: all .5s;
}
```


MAGIC

OR IS IT?

Autoprefixer npm package file

```
{
  "name": "autoprefixer",
  "version": "6.3.1",
  "description": "Parse CSS and add vendor prefixes to CSS rules using values from the Can I Use website",
 …
 "license": "MIT",
  "repository": {
    "type": "git",
    "url": "git+https://github.com/postcss/autoprefixer.git"
  },
  "dependencies": {
    "postcss-value-parser": "^3.2.3",
    "normalize-range": "^0.1.2",
    "num2fraction": "^1.2.2",
    "browserslist": "~1.1.1",
    "caniuse-db": "^1.0.30000387",
    "postcss": "^5.0.14"
  },
  "devDependencies": {
    "vinyl-source-stream": "1.1.0",
    "gulp-json-editor": "2.2.1",
...
```

YOU ALREADY USE POSTCSS

Autoprefixer is a PostCSS plugin.

POSTCSS COMPLIMENTS SASS

But that's not all it does...

IT CAN REPLACE SASS

Using PostCSS as your CSS Preprocessor

Any application that **can** be written in Javascript,
will eventually be written in Javascript.

Atwood's Law

http://blog.codinghorror.com/the-principle-of-least-power/

Variables

```
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
    font: 100% $font-stack;
    color: $primary-color;
}


// results in the following css
body {
    font: 100% Helvetica, sans-serif;
    color: #333;
}
```

postcss-advanced-variables

```
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
    font: 100% $font-stack;
    color: $primary-color;
}


// results in the following css
body {
    font: 100% Helvetica, sans-serif;
    color: #333;
}
```

Mixins

```
@mixin border-radius($radius) {
    border-radius: $radius;
}

.box {
    @include border-radius(10px);
}


// results in the following css
.box {
    border-radius: 10px;
}
```

postcss-mixins

```
@define-mixin border-radius $radius {
    border-radius: $radius;
}

.box {
    @mixin border-radius 10px;
}


// results in the following css
.box {
    border-radius: 10px;
}
```

Sass For

```
@for $index from 1 through 3 {
    .col-#{$index} {
        width: #{$index+'0%'};
    }
}


// results in the following css
.col-1 {
    width: 10%;
}
.col-2 {
    width: 20%;
}
.col-3 {
    width: 30%;
}
```

postcss-advanced-variables

```
@for $index from 1 to 3 by 1 {
    .col-$index {
        width: $(index)0%;
    }
}


// results in the following css
.col-1 {
    width: 10%;
}
.col-2 {
    width: 20%;
}
.col-3 {
    width: 30%;
}
```

Each

```
@each $icon in (foo, bar, baz) {
    .icon-#{$icon} {
        background: url(icons/#{$icon}.png);
    }
}


// results in the following css
.icon-foo {
    background: url('icons/foo.png');
}
.icon-bar {
    background: url('icons/bar.png');
}
.icon-baz {
    background: url('icons/baz.png');
}
```

postcss-advanced-variables

```
@each $icon in (foo, bar, baz) {
    .icon-$(icon) {
        background: url(icons/$icon.png);
    }
}

// results in the following css
.icon-foo {
    background: url('icons/foo.png');
}
.icon-bar {
    background: url('icons/bar.png');
}
.icon-baz {
    background: url('icons/baz.png');
}
```

YOU GET THE IDEA...

But Chris, that seems like a lot of plugins to have to manage?

POSTCSS PLUGIN PACKS

Bundle multiple PostCSS plugins together as a single plugin

precss

PreCSS is a tool that allows you to use Sass-like markup in your CSS files.

postcss-partial-import: W3C and Sass-like imports
postcss-mixins: Sass-like mixins
postcss-advanced-variables: Sass-like variables and methods
postcss-custom-selectors: W3C custom selectors
postcss-custom-media: W3C custom media queries
postcss-custom-properties: W3C custom variables
postcss-media-minmax: W3C < <= >= > media queries
postcss-color-function: W3C color methods
postcss-nesting: W3C nested selectors
postcss-nested: Sass-like nested selectors
postcss-atroot: place rules back up to the root
postcss-property-lookup: reference other property values
postcss-extend: W3C and Sass-like extend methods
postcss-selector-matches: W3C multiple matches pseudo-classes
postcss-selector-not: W3C multiple not pseudo-classes

https://github.com/jonathantneal/precss

PUTTING IT ALL TOGETHER

Using PostCSS in your build process

Using PostCSS with Gulp

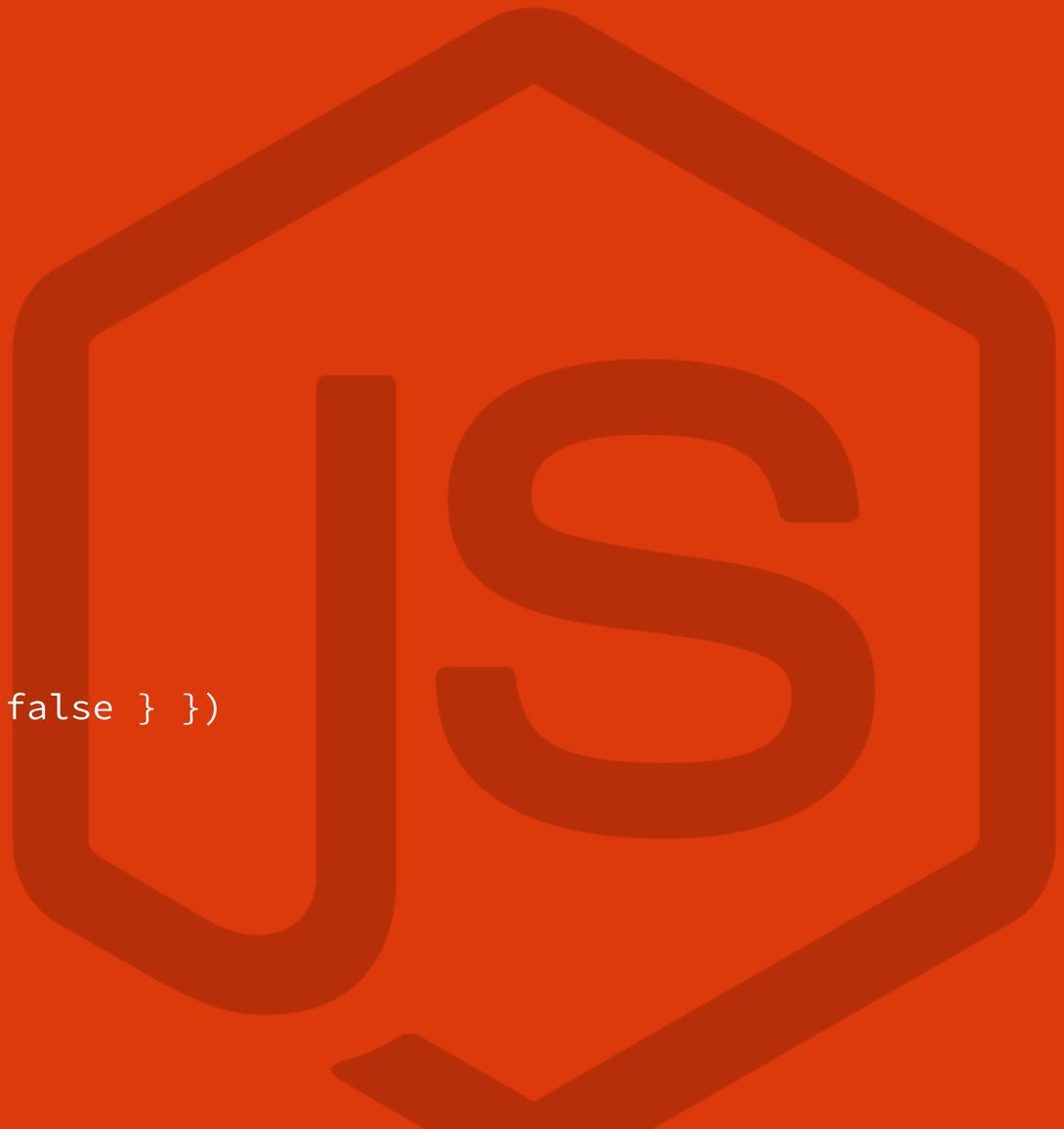
```
// specify the plugins we want to use
var gulp = require('gulp');
var postcss = require('gulp-postcss');
var sourcemaps = require('gulp-sourcemaps');
var cssnano = require('gulp-cssnano');

// let's do this…
gulp.task('default', function () {
    return gulp.src('src/**/*.css')
        .pipe(sourcemaps.init())
        .pipe(postcss([
            require('precss'),
            require('autoprefixer')
        ]))
        .pipe(cssnano())
        .pipe(sourcemaps.write('.'))
        .pipe(gulp.dest('build/'));
});
```

Using PostCSS with Grunt

```
module.exports = function(grunt) {
    grunt.initConfig({
        postcss: {
            options: {
                map: {
                    inline: false
                },
                processors: [
                    require('precss')(),
                    require('autoprefixer')(),
                    require('cssnano')()
                ]
            },
            dist: {
                src: 'src/main.css',
                dest: 'build/main.css'
            }
        }
    });

    grunt.loadNpmTasks(‘grunt-postcss');
    grunt.registerTask('default', ['postcss']);
}
```

Using PostCSS with Node

```
// load node modules
var fs = require('fs');
var postcss = require('postcss');

// set our variables to keep things organised
var INPUT = 'src/main.css';
var OUTPUT = 'build/main.css';
var CSS = fs.readFileSync('src/main.css');

// let's do this…
postcss([
    require('precss'),
    require('autoprefixer'),
    require('cssnano')
])
.process(CSS, { from: INPUT, to: OUTPUT, map: { inline: false } })
.then(function (result) {
    fs.writeFileSync('build/main.css', result.css);
    if (result.map) {
        fs.writeFileSync(OUTPUT + '.map', result.map);
    }
});
```



AND THERE'S MORE...

ECMAScript 6  BABEL  Javascript

TRANSPILERS FTW

Write future-proof code, that works in any browser.

TOMORROW'S CODE, TODAY

Sass? Where we're going, we don't need Sass.

CSS4 ANYONE?

https://googlechrome.github.io/samples/css-custom-properties/index.html

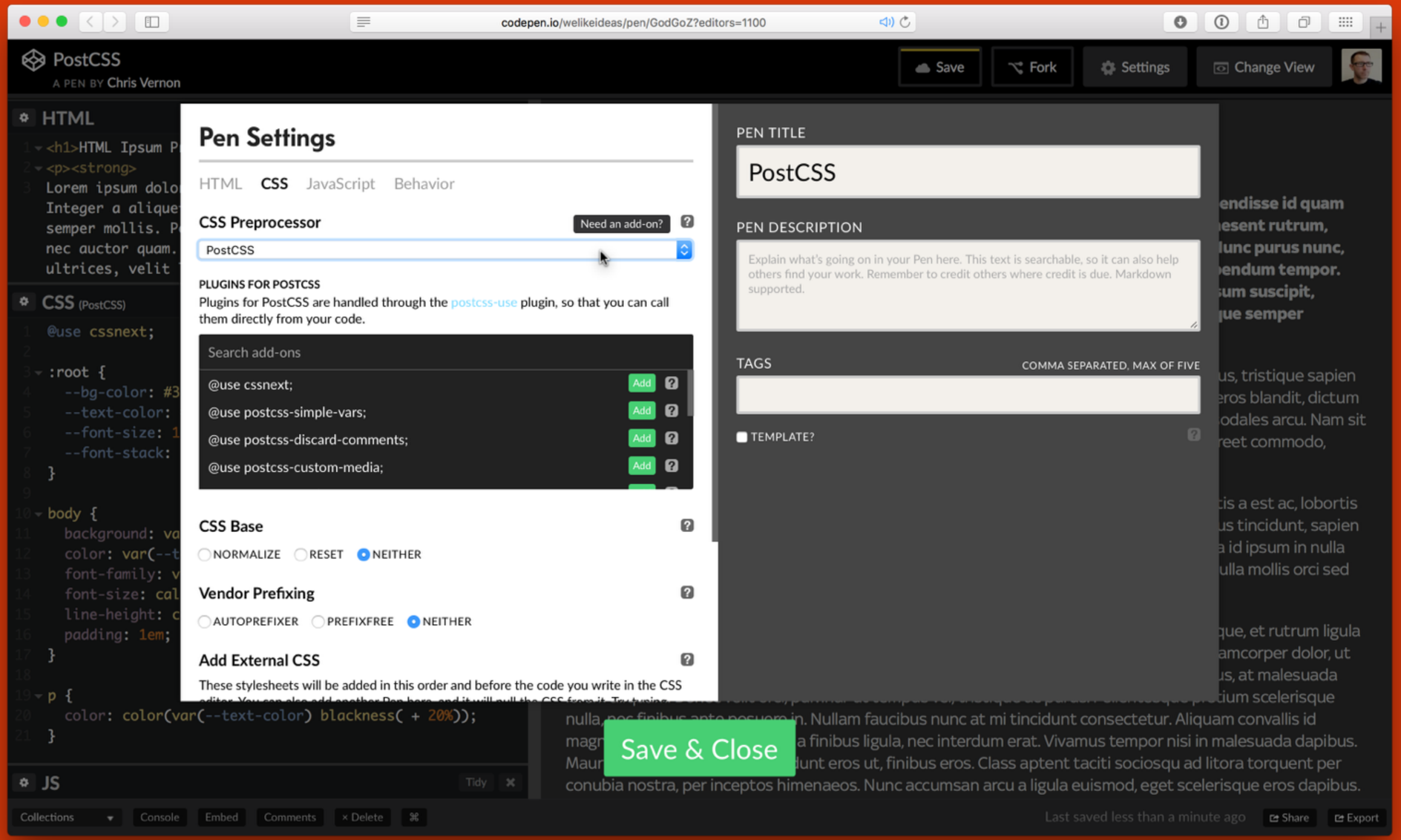
cssnext

```
:root {
    --bg-color: #333;
    --color: #fff;
    --font-size: 1rem;
    --font-stack: Helvetica, Sans-serif;
}

body {
    background: var(--bg-color);
    color: var(--color);
    font-family: var(--font-stack);
    font-size: calc(var(--font-size) * 1.2);
    line-height: calc(var(--font-size) * 1.5);
    padding: 1em;
}

p {
    color: color(var(--color) blackness(+20%));
}
```

http://cssnext.io

{ } css

```
body {
    background: #333;
    color: #fff;
    font-family: Helvetica, Sans-serif;
    font-size: 19px;
    font-size: 1.2rem;
    line-height: 24px;
    line-height: 1.5rem;
    padding: 1em;
}

p {
    color: rgb(212, 213, 213);
}
```



http://codepen.io/welikeideas/pen/GodGoZ

stylelint

```
src/main.css
⚠  Invalid option value "space" for rule "indentation" [stylelint]
1:8 ⚠  Unexpected invalid hex color "#e056ef0" (color-no-invalid-hex) [stylelint]
13:15  ⚠  Expected single space after ":" (declaration-colon-space-after) [stylelint]
13:20  ⚠  Unexpected empty line (max-empty-lines) [stylelint]
```

- Part of the build process using PostCSS
- Enforce consistant CSS styling
- Avoid errors in your stylesheets
- Unopinionated, configure however you want per project
- Maintain consistency when working as a team

http://stylelint.io

postcss-spiffing

```
/* Your well-spelt CSS */
body {
  background-colour: grey;
  transparency: 0.3;
  text-align: centre;
  text-transform: capitalise;
  border: 1px solid grey;
}

span {
  font-weight: plump;
}

.frame {
  background-photograph: url('/queen.png') !please;
}
```

https://github.com/HashanP/postcss-spiffing

{ } css

```
body {
  background-color: gray;
  opacity: 0.3;
  text-align: center;
  text-transform: capitalize;
  border: 1px solid gray;
}

span {
  font-weight: bold;
}

.frame {
  background-image: url('/queen.png') !important;
}
```

postcss-lolcat-stylesheets

```
/* Lolcat aka Lolspeak CSS Properties */
.ohai {
    posishun: relativ;
    bakground-color: chawklit;
    bakground-image: none;
    font-pplz: Helvetica, Arial;
    color: silvr;
    lettr-spacin: 2px;
    paddin-rite: 30px;
}
```

https://github.com/sandralundgren/postcss-lolcat-stylesheets

{ } css

```
.ohai {
    position: relative;
    background-color: chocolate;
    background-image: none;
    font-family: Helvetica, Arial;
    color: silver;
    letter-spacing: 2px;
    padding-right: 30px;
}
```

POSTCSS

- Powerful, modular system written in Javascript
- Only use what you need to
- Compliment or use instead of SASS/LESS
- Reduce build complexity/dependencies
- 3–30 x faster than other preprocessors
- Write your own plugins (using Javascript)
- Apply what you learn elsewhere (Javascript FTW)
- Thriving community

POSTCSS RESOURCES

PostCSS

https://github.com/postcss/postcss

Plenty more info on their Github repo

@PostCSS

https://twitter.com/postcss

Twitter accounts posts lots of goodies

PostCSS Playground

https://sneakertack.github.io/postcss-playground/

Lets you build a simple PostCSS processor and see what's happening to the css that is being transformed by each plugin.

PostCSS Build Examples

https://github.com/welikeideas/postcss-build

Repo with examples of how to build PostCSS

PostCSS.parts

http://postcss.parts

Searchable catalogue of PostCSS plugins

PostCSS Deep Dive

http://webdesign.tutsplus.com/series/postcss-deep-dive--cms-889

In-depth guide on all things PostCSS

QUESTIONS?

@welikeideas

@christopher.vernon