# Build Powerful FrontEnd Workflows with PostCSS

Guide to writing/generating cutting edge CSS

# Key TakeAways

PostCSS - Deep Dive
- plugins you can use
- custom plugins

Workflow
- Essential Concepts
- Plugins to help with tasks

Drupal Theme
- Starter Template

# Introduction

WHAT - MICRO SERVICES
ARCHITECTURE - WHY - INSTALL

# Essential

WORKFLOW - PLUGINS
FUTURE CSS - PACKS

# Advanced

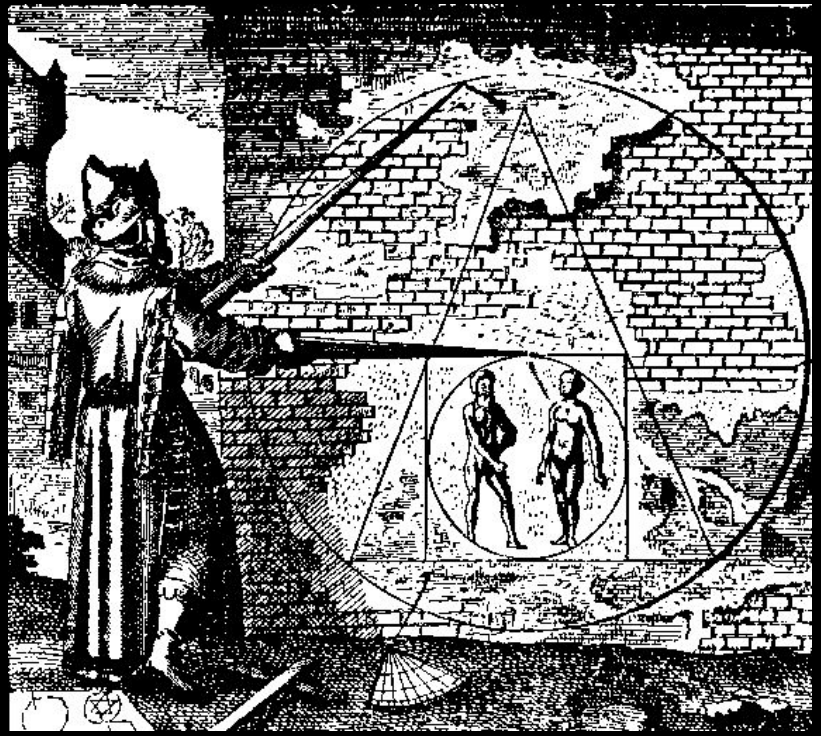
DRUPAL THEME - SMACSS
BEM - CUSTOM PLUGINS

# Introduction

what - why - how

What is it?
Microservice Architecture
Single Responsibility Principle
Installation
Gulp Script

# Philosopher's Stone



# Alchemy

# What is PostCSS?

Transformer!

A Workflow Tool - (post/pre)processor

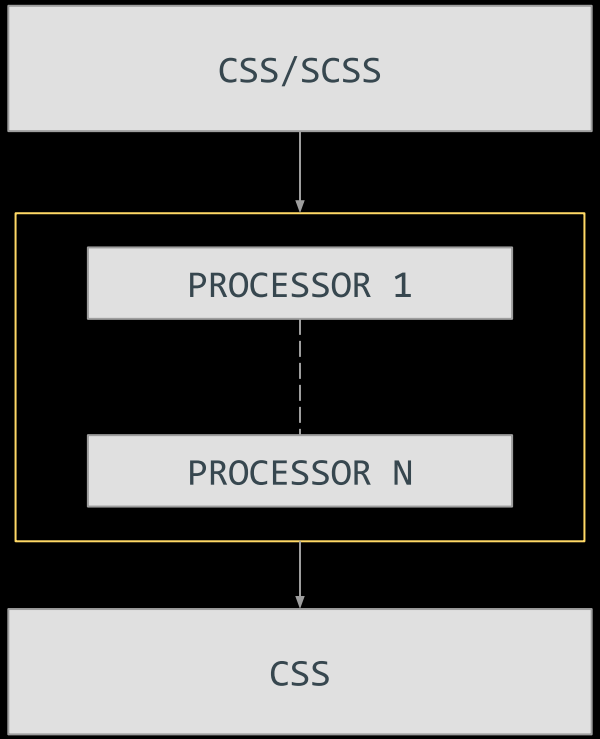
Language Extender

CSS Wayback/Time Machine

*architecture*

# Collection of API & Micro Services

# Architecture




POSTCSS

# What does PostCSS do?

Parses CSS
Creates Node Tree

Provides APIs for processors
Pipes between processors

# Plugin Design Principle

Single Responsibility Principle
Do one thing and do well

Node Modules
Written in JS

# 200+

You have a plugin for that! PostCSS.parts

# Why roll yours in PostCSS?

Choice - freedom to choose

Usage - tailor to your workflow

Modular/ lightweight hence faster

# Limitations of pre-processors

@extend across media queries

Automatically fix errors

Feature Requests and how it is handled

# Installation and First Task

# Required Tools

Node.js

NPM - Dependency Management System

Gulp/Grunt - The task runner

# Workflow Anatomy


CONSOLE RUN
MINIFY
EXPORT
WATCH

CSS PARSER
CONFIG PROCESSOR
PIPE

# Installation

Install Node

```
npm install gulp
npm install postcss
```

Gulp script to run PostCSS with the plugins.

# Gulp Task

```
gulp.task('css', function () {
  var autoprefixer = require('autoprefixer');
  var cssnext = require('cssnext');

  return gulp.src('./src/css/*.css')
    .pipe(postcss([
      autoprefixer,
      cssnext
    ]))
    .pipe(gulp.dest('./dist/css'));
});
```

# Essential PostCSS

Build custom workflow

Requirements of a Workflow?
Autoprefixer
Linting
futureCSS
Quantity Queries
Container Queries
CSS Modules
Packs

# Workflow Goals

## ENVIRONMENT

Mechanical Rules
Enforcement
Debug
Linting
Easy SetUp

## CODE HELPERS

Future Proofing
Fallback Support
Language Extensions
Shortcuts
Utilities

# Toolchain

## ENVIRONMENT

Mechanical Rules
Enforcement - stylelint
Debug - sourcemaps
Linting - stylelint
Easy SetUp - npm
I/O - import/cssnano

## CODE HELPERS

Fallback Support - oldie
Future Proofing - cssnext
Language Extensions precss
Shortcuts - short
Utilities - lot

# ENVIRONMENT

# Importer - postcss-import

*Before*

```
@import "normalize.css";
@import "css/app.css";

@import "css/mobile.css" (max-width: 780px);
```

*PostCSS - CSS*

```
/* content of normalize.css from module */
/* content of app.css */

@media (max-width: 780px) {
    /* contents of mobile.css */
}
```

# Linters - Stylelint

StyleLint
Mechanical Rules Enforcement
Over 100 Rules
Choose the ones you want

# Linters - Stylelint

```
var styleLintConfig = {
    "rules": {
        "color-no-invalid-hex": true,
        "declaration-colon-space-after": "always",
        "function-url-quotes": "double",
        "media-feature-colon-space-after": "always",
        "max-empty-lines": 2,
        "property-no-vendor-prefix": true,
        "rule-no-duplicate-properties": true,
        "string-quotes": "double",
        "selector-no-universal": true,
        "rule-no-shorthand-property-overrides": true,
        "indentation": 4,
    }
};
```

# Packers

cssnano
Does minify and some errors.

postcss-css-mqpacker
Concats all styles of same media query.

postcss-cachebuster
Busts assets cache using url params.

postcss-data-packer
embedded base64 to separate file.

# Source Maps

```
gulp.task('css', function () {
    var postcss    = require('gulp-postcss');
    var sourcemaps = require('gulp-sourcemaps');

    return gulp.src('src/css/*.css')
        .pipe( sourcemaps.init() )
        .pipe( postcss([require('autoprefixer') ]))
        .pipe( sourcemaps.write('.') )
        .pipe( gulp.dest('dist/') );
});
```

# Style Guides

postcss-style-guide, psg-theme-default

Provides KSS Style Living Guides
Uses Annotation to generate HTML

# Style Guides

```
/* @styleguide

@title Button

<button class="button button--red">Red
Button</button> */


var styleGuideConfig = {
  project: 'Drupal 8.0',
  dest: 'styleguide/index.html',
  showCode: true,
  themePath: './node_modules/psg-theme-default'
}
```

# CODE HELPERS

## Future CSS

# Future CSS

postcss-custom-media
postcss-custom-properties
postcss-extend
postcss-initial
postcss-media-minmax
cq-prolyfill

# PostCSS Extend

```
.potato {
  color: white;
  outline: brown;
  font-family: sans-serif;
}

@media (width > 600px) {
  .potato:first {
    float: center;
  }

  .spud {
    @extend .potato;
    color: red;
    font-size: 4em;
  }
}
```

```
.potato {
  color: white;
  outline: brown;
  font-family: sans-serif;
}

@media (width > 600px) {
  .potato:first, .spud:first {
    float: center;
  }

  .spud {
    color: red;
    font-size: 4em;
    outline: brown;
    font-family: sans-serif;
  }
}
```

example from plugin page

# Container Queries

cq-prolyfill

```
.block:container(width > 200px) {
   background-image: url(“extra-large.jpg”);
}
```

# Quantity Queries

postcss-quantity-queries

```
li.error:at-least(2) {
  background-color: red;
}
```

:at-least(count)
:at-most(count)
:between(start,end)
:exactly(count)

# CSS Modules

postcss-initial

{all:initial}

# CODE HELPERS

## Language Extenders

# Pre/post compiler

## PRE COMPILER

write code in scss/less/stylus, gets converted into css.

precss

## POST COMPILER

write CSS Code, gets converted into CSS.

cssnext

# Pre/post compiler

## PRE COMPILER

```
@include display-flex;
```

mixins needs to be learned, their APIs

provides sass like mixins, functions etc.

## POST COMPILER

```
display: flex;
```

plain CSS, which is parsed by Autoprefixers

uses W3C css Variables

# CODE HELPERS

## Utility Plugins

# Utility Plugins

autoprefixer
pxtorem
postcss-sorting
perfectionist
postcss-font-pack
postcss-fontpath

# Utility Plugins - AutoPrefixer

```
.copyright-text {
     user-select: none;
}

.copyright-text { /*postprocessed css*/
    -webkit-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
     user-select: none;
}
```

refers caniuse.com to generate

does not generate for border-radius etc...

http://autoprefixer.github.io/

# Utility Plugins - pxtorem

```
h1 {
    margin: 0 0 20px;
    font-size: 32px;
    line-height: 1.2;
    letter-spacing: 1px;
}

h1 {
    margin: 0 0 20px;
    font-size: 2rem;
    line-height: 1.2;
    letter-spacing: 0.0625rem;
}
```

# Packs

Curated plugins

Takes care of interplay

Does heavy lifting

# Packs

cssnano - packer, optimizer for production
rucksack - new features and shortcuts
short - shorthand properties
precss - language extender
cssnext - language extender (W3C)

https://github.com/timaschew/postcss-compare-packs

# Advanced

Beyond the Basics

Write your own plugin
CSS Architecture
BEM / SMACSS Organization
Putting it all together

# PostCSS Boilerplate

postcss-plugin-boilerplate

Wizard to help creation
Clean git history
Write index.js and test.js
Document

# Examine a plugin

```
.foo:after {
    content: 'usd'
}

/*after PostCSS*/
.foo:after {
    content: '$'
}
```

# Anatomy of a Plugin

```
var postcss = require('postcss');
var currencyDB = require('typographic-currency-db');

module.exports = postcss.plugin('postcss-currency', function (opts) {
    opts = opts || {};

    return function (css) {
        css.walkDecls('content', function (decl) {
            var quote = decl.value.match(/'|"/);
            quote = quote ? quote[0] : '';

            for (var key in currencyDB) {
                var value = decl.value.replace(/['"]+/g, '').toUpperCase();
                if (value === key) {
                    decl.value = quote + currencyDB[key] + quote;
                }
            }
        });
    };
});
```

this is postcss-currency plugin

# CSS Architecture

# Architecture Challenges

Abstraction
Maintainability
Change Management

Debugging
Documentation

# SMACSS

Categorizing / Organizing CSS Rules

Base - base level CSS
Layout - layout based classes (l-)
Module - reusable modules
State - state of an item is-active
Theme - related to colors

Reduced dependency on structure
Thinking in Components

# Block Element Modifier

Rules to Name selectors

```
.block {}
.block__element {}
.block--modifier {}

.person {}
.person__hand {}
.person--female {}
.person--female__hand {}
.person__hand--left {}

//example from CSS Wizardy Blog
```

# Theming Process

Component Inventory - ppt
Component Library - code

Layout Pages
Composition of Components

Oddball Components
Context based alignment/changes

# End To End Workflow

CLASSY THEME
Configure with postCSS
Folder Structures
Plugins

Code Organization
Debugging - source maps
Linting

# DEMO & CODE WALKTHROUGH

# Summary

What is PostCSS
Architecture
Why PostCSS
How To start coding in PostCSS

Workflow Goals
PostCSS Tools that support
Useful Plugins

Write Plugin
SMACSS/BEM
All together

# Appendix - UseFul Links

PostCSS Page

Plugins List
PostCSS.parts

cssnext
precss

compare css packs

BEM
SMACSS