Basic Web Content Build System

- Introduction: Why a build system?
- Tasks to accomplish
    - CSS-Related
        - Process SASS/SCSS into CSS
        - Run Post CSS and plugins
            - Autoprefixer
            - (Optional) UNCSS: Trim the CSS Fat
            - CSS Nano minimizer
        - Lint CSS to make sure everything translated properly
        - (Optional) Extracting and inlining critical path CSS
    - Javascript-related
        - Transpile
            - Babel
            - Preset-env
                - Explain the difference between preset-env and yearly presets (preset-es2015 et al)
        - Concatenate
            - Is it still necessary? Yes if you choose not to use Webpack
        - Minimize
        - Lint
    - Image related
        - Imagemin for Image Compression
        - (Optional) Generating images for Responsive image sets
            - JPG
            - PNG
    - Webpack for asset bundling
    - Putting it all together
        - Gulp
        - NPM Scripts
        - Running from shell/terminal

It used to be that building content for the web all we had to do was write HTML, CSS and Javascript and upload it to the server. Now the stacks for each of HTML, CSS and Javascript have become complex enough where doing it by hand, while possible, becomes too complicated to do it consistently well.

That's where build systems come in. When implementing a build system you automate a lot of the typing and manual configuration for the tasks that you do most often when building your content.

The idea is to accomplish the following tasks:

- CSS-Related
    - (Optional) Process SASS into CSS
    - Run Post CSS and plugins
        - Autoprefixer
        - (Optional) UNCSS: Trim the CSS Fat
        - CSS Nano minimizer
    - Lint CSS to make sure everything translated properly
    - (Optional) Extracting and inlining critical path CSS
- Javascript-related
    - Transpile
        - Babel
        - Preset-env
            - Explain the difference between preset-env and yearly presets (preset-es2015 et al)
    - (Optional if not using Webpack) Concatenate
    - Minimize
    - Lint
- Image related
    - Imagemin for Image Compression
    - (Optional) Generating images for Responsive image sets
        - JPG
        - PNG
- Webpack for asset bundling

We'll explore creating a basic build systems using three different approaches: Gulp, NPM scripts and running the tools from the command line. This should cover a large percentage of build cases. If you want to use a different set of tools, we will assume that you know how to use them.

Common Requirements

The three approaches we'll try require Node.js. So before we start we need to install Node. The recommended way is to install NVM.

Using curl

```
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.11/instal
```

or Wget

```
wget -qO- https://raw.githubusercontent.com/creationix/nvm/v0.33.11/insta
```

The script clones the nvm repository to ~/.nvm and adds the commands necessary to run it to your shell's profile (~/.bash_profile, ~/.zshrc, ~/.profile, or ~/.bashrc).

```
export NVM_DIR='$HOME/.nvm'
['$HOME/.nvm'R/nvm.sh' ] && \. '$NVM_DIR/nvm.sh' ' ] && \. '# This loads n
```

To install the latest Long Term Support version, regardless of what version it is run the following commands from your shell or terminal.

The first command installs the latest version of the Carbon LTS line

The second creates a default alias for the version that you just installed

The last command uses the alias we created.

```
nvm install --lts=Carbon
nvm alias default 8.11.4
nvm use default
```

If you're using Windows 10, the suggested version of installing Node is to use the Windows Subsystem for Linux to install your favorite Linux distribution. You can then follow the instructions above.

Once node is installed create a folder that will be the root for all the projects.

```
mkdir build-projects
cd build-projects
```

Now we're ready to move to the first version.

Gulp

Gulp.js is an automation tool that will make your work easier by not making you repeat tasks over and over.

To run Node-based projects we must have a `package.json` file. To generate it run the following command.

```
npm init --yes
```

Then we install Gulp both globally and for the project. The first version will install the package globally and the second one will install it for the project as a development dependency.

```
npm install gulp-cli -g
npm install gulp -D
```

Finally create a `gulpfile.js` file. This will hold all the tasks we're about to create.

CSS-Related

SASS has become one of the main ways to write CSS. We'll leverage the gulp-sass to run the conversion. This plugin will use the Node bindings

PostCSS labels itself as a tool to transform CSS with Javascript. We will use two plugins:

- Autoprefixer
- UNCSS

We will use gulp-csslint to lint the resulting CSS to make sure that it transformed correctly

Installing the required packages

```
npm i -D gulp-sass gulp-sourcemaps gulp-postcss autoprefixer uncss gulp-cs
```

The tasks

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

const gulp = require('gulp');
const critical = require('critical').stream;

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

```
gulp.task('sass', function() {
  return gulp
    .src('./sass/**/*.scs'./sass/**/*.scss'cemaps.init())
    .pipe(
      sass({
        outputStyle: 'expanded',
      }).on('error', sass.logError),
    )
    .pipe(sourcemaps.write())
    .pipe(gulp.dest('./css'));
});
```

```
gulp.task('css', function () {
  const plugins = [
    autoprefixer({
      browsers: [
        'last 2 version'last 2 versions'
    cssnano(),
  ];
```

```
  return gulp.src('./src/*.css')
    .pipe(postcss(plugins))
    .pipe(gulp.dest('./dist'));
});
'./src/*.css'
```

Javascript-related

installing the plugins

```
npm i -D gulp-babel babel-core babel-preset-env
npm i -D gulp-concat
npm i -D gulp-eslint
npm i -D gulp-uglify-es gulp-rename
```

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

const concat = require('gulp-concat');

const eslint = requier('eslint');

const rename = require('gulp-rename');
const uglify = require('gulp-uglify-es').default;
```

```
gulp.task('transpile', () => {
  gulp
    .src('src/app.js')
    .pipe(
      babel({
        presets: ['env'],
      }),
    )
    .pipe(gulp.dest('dist'));
});
```

```
gulp.task('scripts', function() {
  return gulp
    .src('./lib/*.js')
    .'./lib/*.js''all.js'))
    .pipe(gulp.dest('./dist/'));
});
```

```
gulp
  .src(['**/*.js', '!node_modules/**'])
 '!node_modules/**' .pipe(
    eslint.result(result => {
      // Called for each ESLint result.
      console.log(`ESLint result: ${result.filePath}`);
      console.log(`# Messages: ${result.messages.length}`);
      console.log(`# Warnings: ${result.war`ESLint result: ${result.filePa
    }),
  );
```

```
gulp.task('uglify', function() {
  return gulp
    .src('lib/bundle.js')
 'lib/bundle.js'('bundle.min.js'))
    .pipe(uglify('bundle.min.js'/* options */))
    .pipe(gulp.dest('lib/'));
});
```

Image related

```
npm i -D imagemin imagemin-mozjpeg imagemin-webp
```

Imagemin for Image Compression

```
gulp.task('imagemin', () => {
  return gulp
```

```
    .src('src/images')
    .pipe(
      imagemin({
        progressive: true,
        svgoPlugins: [{ removeViewBox: false }, { cleanupIDs: false }],
        use: [mozjpeg()],
      }),
    )
    .pipe(gulp.dest('static/images'))
    .pipe(
      $.size({
        pretty: true,
        title: 'imagemin',
      }),
    );
});
```

Generating images for Responsive image sets

```
gulp.task('processImages', () => {
  return gulp
    .src(['src/images/**/*.{jpg,png'src/images/**/*.{jpg,png}'/*.png'])
    .pipe(
      $.responsive({
        '*': [
          {
            // image-small.jpg is 200 pixels wide
            width: 200,
            rename: {
              suffix: '-small',
              extname: '.jpg',
            },
          },
          {
            // image-small@2x.jpg is 400 pixels wide
            width: 200 * 2,
            rename: {
```

```
      suffix: '-small@2x',
      extname: '.jpg',
    },
  },
  {
    // image-large.jpg is 480 pixels wide
    width: 480,
    rename: {
      suffix: '-large',
      extname: '.jpg',
    },
  },
  {
    // image-large@2x.jpg is 960 pixels wide
    width: 480 * 2,
    rename: {
      suffix: '-large@2x',
      extname: '.jpg',
    },
  },
  {
    // image-extralarge.jpg is 1280 pixels wide
    width: 1280,
    rename: {
      suffix: '-extralarge',
      extname: '.jpg',
    },
  },
  {
    // image-extralarge@2x.jpg is 2560 pixels wide
    width: 1280 * 2,
    rename: {
      suffix: '-extralarge@2x',
      extname: '.jpg',
    },
  },
  {
    // image-small.webp is 200 pixels wide
```

```
      width: 200,
      rename: {
        suffix: '-small',
        extname: '.webp',
      },
    },
    {
      // image-small@2x.webp is 400 pixels wide
      width: 200 * 2,
      rename: {
        suffix: '-small@2x',
        extname: '.webp',
      },
    },
    {
      // image-large.webp is 480 pixels wide
      width: 480,
      rename: {
        suffix: '-large',
        extname: '.webp',
      },
    },
    {
      // image-large@2x.webp is 960 pixels wide
      width: 480 * 2,
      rename: {
        suffix: '-large@2x',
        extname: '.webp',
      },
    },
    {
      // image-extralarge.webp is 1280 pixels wide
      width: 1280,
      rename: {
        suffix: '-extralarge',
        extname: '.webp',
      },
    },
```

```
          {
            // image-extralarge@2x.webp is 2560 pixels wide
            width: 1280 * 2,
            rename: {
              suffix: '-extralarge@2x',
              extname: '.webp',
            },
          },
          {
            quality: 80,
            progressive: true,
            skipOnEnlargement: false,
            withMetadata: true,
          },
        ],
      }).pipe(gulp.dest('dist/images')),
    );
});
```

Webpack for asset bundling

```
const webpack = require('webpack-stream');

gulp.task('default', function() {
  return gulp
    .src('src/app.js')
    .pipe(webpack())
    .pipe(gulp.dest('dist/'));
});
```

Default task to tie it all together

What the gulpfile looks together

NPM Scripts

Command Line Tools