RoR - SASS & Webpack

Overview

By default, Ruby on Rails 6.0 comes with an implementation of SassC which is used to complie the .scss files into css for the browser to use. However, in order to use the new module functions of sass, such as @use, Dart Sass must be added to webpacker since it is the only complier which supports the function.

# CSS

Compile CSS with Webpacker

In general, it is good practice to have all CSS and JS loaded into one file, which can be downloaded on the initial page visit and then cached on the client machine for quicker access. Controller specific scripts and styles can still be added, however styles should be designed per feature instead of per controller.

By default webpacker only compiles javascript, however it can also compile CSS with specific compilers, such as Dart Sass. Tell webpacker to compile CSS:

# config/webpacker.yml

extract\_css: true

Move Stylesheets to the Correct Location

Stylesheets will now be compiled and included by webpacker so long as they are in the correct location. The new location is defined in the webpacker.yml as 'app/javascript/packs by default.

Create the webpacker application sass file:

app/javascript/packs/application.scss

Add application.scss to Application Layout

Add the application.scss pack tag to the default application view layout:

<%= stylesheet\_pack\_tag 'application', media: 'all', 'data-turbolinks-track': 'reload' %>

Add the new pack to the webpacker manifest:

# config/webpacker.yml

Add Modern Sass

Add Sass features not yet supported by SassC such as module imports using the @use rule:

# app/javascript/packs/styles/base/\_globals.scss

$default-background: red;

# app/javascript/packs/styles/application.scss

@use 'styles/base/globals';

$background-color: globals.$default-background;

body {

background-color: $background-color;

}

When compiling, this will cause errors in webpacker since its sass-loader does not support the new features.

Add Dart Sass

Add sass to yarn:

yarn add sass

Add configuration to webpack environment:

# config/webpack/environment.js

const { environment } = require('@rails/webpacker')

// Get the actual sass-loader config and set loader to dart sass

const sassLoader = environment.loaders.get('sass')

const sassLoaderConfig = sassLoader.use.find( e => e.loader == 'sass-loader' )

sassLoaderConfig.options.implementation = require('sass')

module.exports = environment

Webpack should now load .scss and .sass files using sass (Dart Sass) instead of node-sass (LibSass).

# Fonts

Overview

Fonts can be automatically compiled by the rails default webpack, however it can be more precise to compile them seperately and select the output location.

Remove Default Font Loading

Remove the default asset loading of font files from the static\_assets\_extensions list in the webpack configuration file:

# config/webpacker.yml

static\_assets\_extensions:

- .jpg

- .jpeg

- .png

- .gif

- .tiff

- .ico

- .sv

# Remove

# - .eot

# - .otf

# - .ttf

# - .woff

# - .woff2

Add URL Rewriter

Since the @font-face src uses relative urls to reference the fonts, and those urls will changes after webpack compling, url rewriting is required. In order to use url rewriting in webpack add the resolve-url-loader to webpack:

yarn add resolve-url-loader

Then add to webpack environment before the sass loader:

// resolve-url-loader must be used before sass-loader

environment.loaders.get('sass').use.splice(-1, 0, {

loader: 'resolve-url-loader'

});

Add Font Loader

Fonts are loaded in webpack using the file-loader. Add a custom font loader module export, which tests for typical font extensions and saves the files in :

# config/webpack/loaders/fonts.js

module.exports = {

test: /\.(woff(2)?|eot|otf|ttf)$/,

use: {

loader: 'file-loader',

options: {

name: '[name]-[contenthash].[ext]',

outputPath: 'css/fonts/',

publicPath: (url) => `fonts/${url}`,

}

}

}

Then add to the webpack environmnet before the sass loader:

// Allow Webpack to read font files

const fonts = require('./loaders/fonts')

environment.loaders.prepend('fonts', fonts)

Fonts will now be placed in packs/css/fonts, and the src url(<font-path>) rewritten to match the new location of the asset.