Webpack 5 Boilerplate Template

Demo

- Demo page demonstrating building - SASS, JavaScript, Images, Fonts, HTML

Table of Contents

Features

- **Simple setup** instructions
    - Start development of a project right away with **simple**, **configured**, **linter enabled**, **browser synced** asset files.
- Configuration per **environment**
    - `development` - `sourcemaps`, `browser synced developmentment server`
    - `production` - `minification`, `sourcemaps`
- Configurable **browsers versions support**. It uses `browserslist` - just specify the browsers you want to support in the `package.json` file for `browserslist`:

```
"browserslist": [
    "last 2 versions",
    "> 5%"
]
```

- The built CSS / JavaScript files will respect the **configured supported browser versions** using the following tools:
    - `autoprefixer` - automatically adds vendor prefixes to CSS rules
    - `babel-preset-env` - smart preset that allows you to use the latest JavaScript without needing to micromanage which syntax transforms (*and optionally, browser polyfills*) are needed by your target environment(s).
- Demo project files to be used as a reference and **example demo** building of:
    - *JavaScript*
    - *SASS / PostCSS*
    - *HTML* templates
    - *Images* (*CSS backgrounds and image tags*)
    - *Fonts*
- Support for **assets optimization** for production environment with ability to configure:
    - **Code Minification** of *JavaScript* and *CSS* processed files.
    - **Optimize Assets Loading** - inline and embed **images** / **fonts** files having file size below a *configurable* threshold value.
    - **Images Optimisation** - optimize `jpeg`, `jpg`, `png`, `gif`, `svg` filesize and loading type via `imagemin`. Plugin and Loader for webpack to optimize (*compress*) all images using `imagemin`. Do not worry about size of images, now they are always optimized/compressed.
- Support for **source code syntax style and formatting linters** that analyze source code to flag any programming errors, bugs, stylistic errors or suspicious constructs:

    - **SASS/PostCSS syntax cheker** - you can change or add additional rules in `.sasslintrc` file. Configuration options can be found on `sass-lint` documentation.
    - **JavaScript syntax checker** - following the `airbnb` style, you can review and configure the rules in `.eslintrc` file. Configuration options can be found on `eslint` documentation.
- Latest Webpack 5 - *JavaScript* module bundler.
- Latest SASS/PostCSS compiler based on Dart `sass`.
- Latest Babel 7 (`@babel/core`) - JavaScript compiler - *Use next generation JavaScript, today.*
- Integration with Travis CI
    - Demo deployment available to GitHub pages
- Configured and ready to use **Webpack Dev Server** plugin for faster local development - `webpack-dev-server`
- Integration with Webpack Bundle Analyzer - *Visualize size of webpack output files with an interactive zoomable treemap.*

Requirements

- `node` : `^12 || >=14`
- `npm`

Setup

Installation

1. Choose and download the latest template release from List of Releases.
2. Extract the release archive to a new directory, rename it to your project name and browse the directory.
3. Install all dependencies using `npm` *clean install* command.

```
$ npm ci
```

> More on the clean install npm command can be read here `npm ci`

> You can still use `npm install` in cases the `npm ci` raises system error due to specific platform incompatibilities.

Define Package Metadata

- Amend `package.json` file and optionally specify:
    - `name` - Name of your project. A name can be optionally prefixed by a scope, e.g. `@myorg/mypackage`.
    - `version` - Specify and maintain a version number indicator for your project code.
    - `author` - Your organisation or just yourself. You can also specify `contributors`.
    - `description` - Short description of your project.
    - `keywords` - Put keywords in it. It's an array of strings.
    - `repository` - Specify the place where your code lives.
    - `license` - Announce your code license, figure out the license from Choose an Open Source License .

- `browserslist` - Specify the supported browsers versions - you can refer to full list of availalbe options.

Configuration

Environment Configuration

- Edit the `configuration/environment.js` if you want to specify:
  - **`server`**: configure development server, specify `host`, `port`. Refer to the full development server configuration options for `webpack-dev-server`.
  - **`limits`**: configure file size thresholds for assets optimizations.
    - Image/Font files size in bytes. Below this value the image file will be served as Data URL (*inline base64*).
  - **`paths`**: `src` or `dist` directories names and file system location.

Additional `webpack` configuration

You can additionally configure `webpack` for specific environment:

- `development` - `configuration/webpack.dev.config.js`
- `production` - `configuration/webpack.prod.config.js`
  - Note that if you prefer to build and deploy `sourcemap` files:

> You should configure your server to disallow access to the Source Map file for normal users!

Development

Assets Source

- **SASS/PostCSS** files are located under `src/scss/`
- **JavaScript** files with support of latest ECMAScript *ES6 / ECMAScript 2016(ES7)/ etc* files are located under `src/js/`
- **Image** files are located under `src/images/`
- **Font** files are located under `src/fonts/`
- **HTML** files are located under `src/`
  - It will **automatically** build **all HTML files** placed under `src/` directory, no need to manually configure each template anymore!

Build Assets

One time build assets for development

```
$ npm run build
```

Build assets and enable source files watcher

```
$ npm run watch
```

This command is suitable if you develop with external web server.

> **Note:** File watching does not work with *NFS* (*Windows*) and virtual machines under *VirtualBox*. Extend the configuration in such cases by:

```
module.exports = {
  //...
  watchOptions: {
    poll: 1000 // Check for changes every second
  }
};
```

Start a development server - reloading automatically after each file change.

```
$ npm run dev
```

Production

Build Assets

Optimize assets for production by:

```
$ npm run production
```

Get Built Assets

- *CSS* files are located under `/dist/css/`
- *JavaScript* files with support of *ES6 / ECMAScript 2016(ES7)* files are located under `/dist/js/`
- *Images* are located under `/dist/images/`
    - Images part of the *design* (*usually referenced in the CSS*) are located under `/dist/images/design/`
    - Images part of the *content* (*usually referenced via* `<img>` *tags*) are located under `/dist/images/content/`
- *Fonts* are located under `/dist/fonts/`
- *HTML* files are located under `/dist/`

Run Code Style Linters

SASS

```
$ npm run lint:sass
```

JavaScript

```
$ npm run lint:js
```

Additional Tools

Run Assets Bundle Analyzer

```
$ npm run stats
```

> This will open the visualisaion on the default configuraiton URL `localhost:8888`, you can change this URL or port following the package documentation.

Continuous Integration

This boilerplate template contains integration with Travis CI. The build system runs all linting scripts and deploys production optimized pages to *GitHub* pages upon push to the `master` branch. However, note that this deployment flow only works for *Project Pages*, as User and Organization pages only support the master branch flow.

For more information on how to set up alternative deployment processes, check out the Travis CI documentation on deployment. The service can deploy to dozens of cloud providers, including Heroku, AWS, and Firebase.