webpack-examples (0.0.1)

Maksim Kostromin

Version 0.0.1, 2018-06-17 22:44:17 UTC

Table of Contents

Travis CI status:

Chapter 1. Introduction

Read reference documentation

webpack 4 examples:

- starter without config file
- basic starter
- add html-webpack-plugin
- add webpack-dev-server
- add css-loader and style-loader
- process html and using html / file loaders
- production build: minify everything, extract css…

Chapter 2. Quickstart

2.1. no config

using webpack without `webpack.config.js` *file*

```
mkdir starter-no-config
cd starter-no-config/

mkdir src dist
echo "console.log('hey!');" >> src/index.js
echo "<script src='./main.js'></script>" >> dist/index.html

npm init -y
npm i -g webpack webpack-cli
#npm i -DE webpack webpack-cli

webpack --mode=development
webpack --mode production
```

2.2. basic config

using webpack with basic config file: `config/webpack.dev.js`

```
const { resolve } = require('path');

module.exports = {
  entry: {
    main: './src/index.js'
  },
  mode: 'development',
  output: {
    filename: '[name]-bundle.js',
    path: resolve(__dirname, '../dist'),
  },
};
```

2.3. add HTML plugin

install html-webpack-plugin

```
npm i -DE html-webpack-plugin
```

update webpack condig

```
const HtmlWebpackPlugin = require('html-webpack-plugin');

module.exports = {
  // ...
  plugins: [
    new HtmlWebpackPlugin({
      template: './src/index.html',
      favicon: './src/favicon.ico',
    }),
  ],
};
```

2.4. add webpack-dev-server

update webpack condig

```
const { resolve, join } = require('path');

module.exports = {
  // ...
  devServer: {
    contentBase: join(__dirname, '../dist'),
    // show / overlay errors in browser
    overlay: true,
  },
};
```

install and run webpack-dev-server

```
npm i -DE webpack-dev-server
webpack-dev-server --config config/webpack.dev.js
```

2.5. add css-loader and style-loader

add some css styles

```css
body {
  margin: 0;
  background-color: #444;
}

/* lets centred content using flex */
#app {
  height: 100vh;
  display: flex;
  align-items: center;
  justify-content: center;
}

/* font styling */
h1 {
  color: white;
  font-size: 3em;
  font-family: Helvetica, sans-serif, 'DejaVu Sans', Arial;
  text-shadow: 0 0 25px white;
}
```

These loaders will apply in reverse order: first, `css-loader`, next: `style-loader`

install css-loader and style-loader

```
npm i -DE css-loader style-loader
```

update webpack condig

```js
module.exports = {
  // ...
  module: {
    rules: [
      {
        test: /\.css$/i,
        use: [
          { loader: 'style-loader' },
          { loader: 'css-loader' },
        ],
      },
    ],
  },
};
```

2.6. process html and using html / file loaders

prepare `./src/index.html`

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Webpack | Examples</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div id="app"></div>
  <script src="main-bundle.js"></script>
</body>
</html>
```

Few thinks are happening here: First, `html-loader` will do necessary linting of *.html files. Then `extract-loader` will tell webpack do not include it in result bundle.js file. And finally `file-loader` will put extracted content in output dir accordingly

remove useless plugin and install requires loaders: `html-loader`, `extract-loader` *and* `file-loader`

```
npm rm -DE html-webpack-plugin
npm i -DE html-loader extract-loader file-loader
```

update webpack condig

```
// const HtmlWebpackPlugin = require('html-webpack-plugin');

module.exports = {
  // ...
  /*
  plugins: [
    new HtmlWebpackPlugin({
      template: './src/index.html',
      favicon: './src/favicon.ico',
    }),
  ],
  module: {
    rules: [
      {
        test: /\.css$/i,
        use: [
          { loader: 'style-loader' },
          { loader: 'css-loader' },
          ],
        },
      ],
    },
    */
  module: {
    rules: [
      {
        test: /\.css$/i,
        use: [
          {
            loader: 'file-loader',
            options: {
              name: '[name].[ext]',
            },
          },
          { loader: 'extract-loader' },
          { loader: 'css-loader' },
        ],
      },
      {
        test: /\.html$/i,
        use: [
          {
            loader: 'file-loader',
            options: {
              name: '[name].[ext]',
            },
          },
          { loader: 'extract-loader' },
          { loader: 'html-loader' },
        ],
      },
      {
        test: /\.(ico)$/i,
        use: [
          {
            loader: 'file-loader',
            options: {
              name: '[name].[ext]',
            },
          },
        ],
      },
    ],
  },
};
```

finally combine all together in `./src/index.js`

```
require('./index.html');
require('./styles.css');
require('./favicon.ico');
```

I wont use that approach, I prefer `HtmlWebpackPlugin` and `ExtractTextWebpackPlugin`

2.7. using extract-text-webpack-plugin

install

```
npm rm -ED style-loader
npm i -DE extract-text-webpack-plugin@next css-loader
# also add less support
npm i -DE less-loader less
```

update webpack condig

```
const ExtractTextPlugin = require('extract-text-webpack-plugin');
const cssContent = new ExtractTextPlugin('[name]-[hash:8].css');
const lessContent = new ExtractTextPlugin('[name].less-[hash:8].css');

module.exports = {
  // ...
  module: {
    rules: [
      {
        test: /\.css$/i,
        use: cssContent.extract([
          'css-loader',
        ]),
      },
      {
        test: /\.less$/i,
        use: cssContent.extract([
          'css-loader',
          'less-loader',
        ]),
      },
    ],
  },
  plugins: [
    cssContent,
    lessContent,
  ],
};
```

There are better option available for using instead of extract text plugin: `mini-css-extract-plugin`

2.8. using mini-css-extract-plugin

install

```
npm rm extract-text-webpack-plugin
npm i -DE mini-css-extract-plugin
```

update webpack condig

```
const mode = process.env.NODE_ENV || 'production';
const isProduction = mode === 'production';
const MiniCssExtractPlugin = require('mini-css-extract-plugin');

module.exports = {
  mode,
  // ...
  module: {
    rules: [
      {
        test: /\.css$/i,
        use: [
          isProduction ? MiniCssExtractPlugin.loader : 'style-loader',
          {
            loader: 'css-loader',
            options: {
              importLoaders: 1,
            },
          },
          // 'postcss-loader',
          {
            loader: 'postcss-loader',
            options: {
              ident: 'postcss',
            },
          },
        ],
      },
    ],
  },
  plugins: [
    new MiniCssExtractPlugin({
      filename: '[name]-[hash:8].css',
    }),
  ],
};
```

update package.json

```
{
  "scripts": {
    "dev": "cross-env NODE_ENV=development webpack",
    "build": "cross-env NODE_ENV=production webpack",
    "start": "cross-env NODE_ENV=development webpack-dev-server --open"
  }
}
```

Chapter 3. Links

- Asciidoctor reference
- GitHub repo: lawwantsin/webpack-course
- GitHub
- Webpack documentation
- HTML webpack plugin documentation
- HTML webpack-dev-server documentation
- extract-loader
- extract-text-webpack-plugin