Loaders & Plugins

webpack
MODULE BUNDLER

SoftUni Team

Technical Trainers

Software University

SoftUni Foundation

Software University

http://softuni.bg

Table of Contents

Webpack Loaders

Why Do We Use Loaders?

- Webpack by itself **only** knows **JavaScript**
- To pack any other type of resources like **.css** or **.scss** or **.ts**, webpack needs help
- Loaders are the **node-based** utilities built for webpack to help **compiling** and/or **transforming** a given type of resource that can be bundled as a javascript module

What is a Loader?

- Loaders provide an easy way to **intercept** our **dependencies** and **preprocess** them before they get bundled
- Webpack supports a large variety of formats through **loaders**. It supports a couple of JS module formats out of the box.
- You always set up a loader, or loaders, and connect those with your directory structure.

Loader

```
const baseConfig = {
 module: {
   rules:[
        {
          test: /*RegEx*/,
          use:[
            {
               loader: /*loader name*/,
                options: /*optional config object*/
            }
          ]
        }
    ]
  }
};
```

Loader Evaluation Order

- Webpack's loaders are always **evaluated** from:
  - **right to left**
  - **bottom to top**

```
{ test: /\.css$/,
  use: ["style-loader", "css-loader"], },
```

```
{
  test: /\.css$/,
  use: "style-loader",
},
{
  test: /\.css$/,
  use: "css-loader",
},
```

Passing Parameters to Loaders

SoftUni Foundation

- It's preferable to go through **use**:

```
{
  test: /\.js$/,
  include: PATHS.app,

  use: [
    {
      loader: "babel-loader",
      options: {
        presets: ["env"],
      },
    },
    // Add more loaders here
  ],
},
```

CSS Loader and Style Loader

SoftUni Foundation

- In order to load CSS, you need **css-loader** and **style-loader**
- **css-loader**
  - Goes through possible **@import** and **url()** lookups within the **matched files** and treats them as a regular import
  - If an **@import** points to an **external resource**, css-loader skips it
- **style-loader**
  - Injects the styling through a **style** element
  - It also implements the **Hot Module Replacement**

Loading CSS Configuration (1)

SoftUni Foundation

- css-loader and style-loader **installation**

```
npm install css-loader style-loader --save-dev
```

- Adding the plugin to your **webpack config**

```
import css from 'file.css';
```

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

Setting Up the Initial CSS

- For the webpack to know that the **file.css** is our dependency, we need to **import** the file in our **dependency tree**

```
import './../styles/file.css';
```

- Execute **npm start** and browse to **http://localhost:8080** if you are using the **default port** and open up **file.css**
- Since **inlining** CSS isn't a good idea for production usage, it makes sense to use **MiniCssExtractPlugin** to generate a **separate** CSS file

MiniCssExtractPlugin

SoftUni Foundation

- Aggregates **multiple** CSS files into **one**
- It comes with a **loader** that handles the **extraction** process
- Picks up the **result** aggregated by the loader and **emits** a **separate** file
- Comes with **overhead** during the compilation phase
- It doesn't work with **Hot Module Replacement** (HMR) yet, but the plugin is used only for production

```
npm install --save-dev mini-css-extract-plugin
```

Setting Up MiniCssExtractPlugin (1)

- Add the configuration below to the beginning of your configuration

```
const MiniCssExtractPlugin = require("mini-css-extract-plugin");

exports.extractCSS = ({ include, exclude, use = [] }) => {
  // Output extracted CSS to a file
  const plugin = new MiniCssExtractPlugin({
    filename: "[name].css",
  });
//Continues in the next slide
```

uses the name of the entry where the CSS is referred

Setting Up MiniCssExtractPlugin (2)

SoftUni Foundation

```
return {
    module: {
      rules: [
        {
          test: /\.css$/,
          include,
          exclude,
          use: [
            MiniCssExtractPlugin.loader,
          ].concat(use),
        },
      ],
    },
    plugins: [plugin],
  };
};
```

URL Loader

SoftUni Foundation

- **url-loader** is the perfect option for **development** purposes, as you don't have to care about the **size** of the resulting bundle
- It comes with a **limit** option that can be used to defer **image** generation to **file-loader** after an **absolute** limit is reached
  - This way you can **inline** small files to your JavaScript bundles while generating **separate files** for the bigger ones
  - When **limit** option is used, **url-loader** passes possible additional options to **file-loader** making it possible to configure i ts behavior further

Setting Up url-loader

- To load **.jpg** and **.png** files while inlining files below 25kB, you would have to set up a loader

```
npm install url-loader --save-dev
```

```
{
  test: /\.(jpg|png)$/,
  use: {
    loader: "url-loader",
    options: {
      limit: 25000,
    },
  },
},
```

File Loader

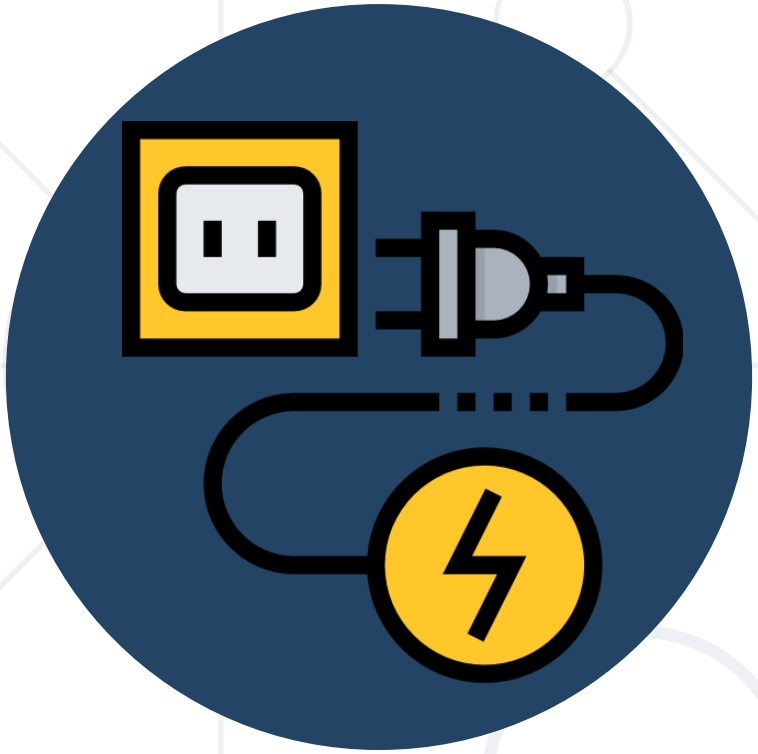
- If you want to **skip** inlining altogether, you can use **file-loader** directly
- The following setup **customizes** the resulting **filename**

```
npm install file-loader --save-dev
```

```
  {
    test: /\.(jpg|png)$/,
    use: {
      loader: "file-loader",
      options: {
        name: "[path][name].[hash].[ext]",
  . . .
```

Some Other Interesting Loaders

- Files
  - raw-loader, val-loader, url-loader etc.
- JSON
  - json-loader, json5-loader, cson-loader
- Transpiling
  - Babel-loader, script-loader, buble-loader etc.
- Templating
  - handlebars-loader, html-loader etc.
- Styling
  - Style-loader, css-loader, less-loader etc.
- Other



Plugins

What is a Webpack Plugin?

- Webpack itself has been implemented as a **collection of plugin s**
- They serve the purpose of doing **anything else** that a loader cannot do
- You have access to webpack's **compiler and compilation** processes
- Plugins allow you to intercept webpack's execution through **hooks**
- Plugins can have plugins

Some Interesting Webpack Plugins

SoftUni Foundation

- **CopyWebpackPlugin** - copies individual files or entire directories to the build directory
- **BabelMinifyWebpackPlugin** - minification with babel-minify
- **HtmlWebpackPlugin** - easily create HTML files to serve your bundles
- **IgnorePlugin** - exclude certain modules from bundles
- Other

Copy Plugin

SoftUni Foundation

- Copies individual **files** or entire directories, which already exist, to the build directory
- Installation

```
npm install copy-webpack-plugin --save-dev
```

- Usage

```
const CopyPlugin = require('copy-webpack-plugin');
module.exports = {
  plugins: [
    new CopyPlugin([
      { from: 'source', to: 'dest' },
      { from: 'other', to: 'public' },
. . .
```

Clean Plugin

- A webpack plugin to **remove/clean** your **build** folder(s)
- By default, this plugin will remove **all** files inside webpack's **output.path** directory, as well as all **unused** webpack assets after every **successful** rebuild
- Installation

```
npm install --save-dev clean-webpack-plugin
```

Clean Plugin Usage

SoftUni Foundation

```
const { CleanWebpackPlugin } = require('clean-webpack-plugin');
const webpackConfig = {
    plugins: [
        new CleanWebpackPlugin(),
    ],
};
module.exports = webpackConfig;
```

HTML Plugin

SoftUni Foundation

- The HtmlWebpackPlugin **simplifies creation** of HTML files to serve your webpack bundles
- This is especially useful for webpack bundles that **include** a **hash** in the filename which changes every compilation
- Installation

```
npm install --save-dev html-webpack-plugin
```

HTML Plugin Usage

SoftUni Foundation

The plugin will **generate** an HTML5 file for you that includes **all** your webpack **bundles** in the **body** using script tags

```
let HtmlWebpackPlugin = require('html-webpack-plugin');
let path = require('path');
module.exports = {
  entry: 'index.js',
  output: {
    path: path.resolve(__dirname, './dist'),
    filename: 'index_bundle.js'
  },
  plugins: [new HtmlWebpackPlugin()]
};
```

Ignore Plugin

- IgnorePlugin **prevents generation** of modules for **import** or **req uire** calls matching the **regular expressions** or **filter functions**
- Using regular expressions
  - **resourceRegExp**: A RegExp to test the resource against
  - **contextRegExp**: (optional) A RegExp to test the context (directory) against

```
new webpack.IgnorePlugin({resourceRegExp, contextRegExp});
// old way, deprecated in webpack v5
new webpack.IgnorePlugin(resourceRegExp, [contextRegExp]);
```

Source Maps

Source Maps

- Source maps provide a **mapping** between the original and the transformed source code
- If you are using webpack 4 and the new **mode** option, the tool will generate source maps **automatically** for you in **development** mode
- To see how webpack handles source maps, see source-map-visualization by the author of the tool
- Webpack can generate both inline or separate source map files

Inline and Separate Source Maps

SoftUni Foundation

- **Inline source maps** are valuable during development due to better performance
- **Separate source maps** are handy for production use as it keeps the bundle size small
- By disabling source maps, you are performing a sort of obfuscation
- **Hidden source maps** give stack trace information only

Inline Source Map Types

- Webpack provides multiple inline source map variants
  - **devtool: "eval"**- eval generates code in which **each module** is **w rapped** within an **eval** function
  - **devtool: "cheap-eval-source-map"**- goes a step **further** and it includes base64 encoded version of the code as a data url. The result contains **only line data** while losing column mappings
  - **devtool: "cheap-module-eval-source-map"**-is the same idea, except with higher quality and lower performance
  - Other

Separate Source Maps

SoftUni Foundation

- Webpack can also generate production **usage friendly** source maps
- These end up in separate files ending with **.map** extension and are loaded by the browser **only when required**
- This way your users get **good performance** while it's **easier** for you **to debug** the application
- source-map is a reasonable default here. Even though it **takes l onger** to generate the source maps this way, you get the **best quality**

Separate Source Map Types

SoftUni Foundation

- **devtool: "cheap-source-map"**
  - The result is going to miss column mappings.
  - Source maps from loaders, such as css-loader, are not going to be used
- **devtool: "cheap-module-source-map"**
  - The same as previous except source maps from loaders are simplified to a single mapping per line

Live Exercise

Summary

- Loaders
  - A **loader** definition consists of **conditions** and **actions**
  - can be **synchronous** or **asynchronous**
  - accept **input** and produce **output** based on it
- Plugins
  - Can **intercept** webpack's **execution**
  - Can be **combined** with loaders
  - Have access to **compiler** and **compilation**
- Source Maps

Questions?

SoftUni

Software University | SoftUni Svetlina | SoftUni Creative | SoftUni Digital | SoftUni Foundation | SoftUni Kids

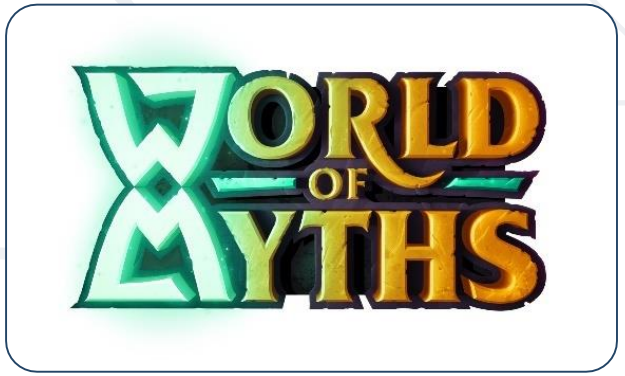
https://softuni.bg/trainings/2469/webpack-4-august-2019

SoftUni Diamond Partners

SoftUni Foundation

SoftUni Organizational Partners

SoftUni Foundation



Trainings @ Software University (SoftUni)

SoftUni Foundation

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

SoftUni Foundation

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license