Configuration Scripts and Transpilers

webpack
MODULE BUNDLER

SoftUni Team

Technical Trainers

Software University

SoftUni Foundation

CC BY NC SA

Software University

http://softuni.bg

Have a Question?

SoftUni Foundation

sli.do

#webpack

Table of Contents

Babel Environment Preset

Loading JavaScript

SoftUni Foundation

- Webpack **processes** ES2015 module definitions by default and **transforms** them into **code**
- It does **not** transform specific syntax, such as **const**
- The resulting code can be **problematic** especially in the **older** browsers
- The problem can be worked around by processing the code through **Babel** (famous JavaScript compiler)

Using Babel with Webpack

SoftUni Foundation

- During **development**, it can make sense to **skip** processing if you are using **language features**
- Processing through Babel becomes almost a **necessity** when you compile your code for **production**
- You can use Babel with webpack through **babel-loader** or through **babel-webpack-plugin**
- Connecting Babel with a project allows you to **process** webpack configuration through it
- To achieve this, name your webpack configuration using the **webpack.config.babel.js** convention

Setting Up babel-loader

- It takes the code and **turns** it into a format older browsers can **understand**

```
npm install babel-loader @babel/core --save-dev
```

```
...
module: {
    rules: [
      {
        test: /\.js$/,
        include,
        exclude,
        use: "babel-loader",
...
```

Setting Up babel-loader

SoftUni Foundation

- If you are using a modern browser for development, you can consider processing only the production code through Babel
- Even though you have Babel installed and set up, you are still missing one bit: **Babel configuration**

Setting Up .babelrc

- @babel/preset-env - preset that enables the required plugins based on the optional environment definition you pass to it

```
npm install @babel/preset-env --save-dev
```

- To make Babel aware of the preset, you need to put it in a **.babelrc** file

```
{
  "presets": [
    [
      "@babel/preset-env",
      {
         "modules": false
      }
...
```

Options - Targets

- Use to tell Babel which **browsers** your app **supports**

```
{
  "targets": "> 0.25%, not dead"
}
```

- **Compatible** with **browserslists**

useBuiltIns option

SoftUni Foundation

- **useBuiltIns** configures how @babel/preset-env **handles polyfills**
- When either the usage or entry options are used, **@babel-preset-env** will add direct references to **core-js** modules as bare **imports** (or requires).
- This means **core-js** will be **resolved relative** to the **file** itself and needs to be **accessible**.
- Since **@babel/polyfill** was **deprecated** in 7.4.0, we recommend directly adding **core-js** and setting the version via the **corejs option**

useBuiltIns: 'entry' property

SoftUni Foundation

- Only use **import "core-js";** or **import "regenerator-runtime/runtime";**
- Once in your whole app. If you are using **@babel/polyfill**, it **already includes** both **core-js** and **regenerator-runtime**:
  - Importing it twice will **throw** an **error**.
  - Multiple imports or requires of those packages might cause **global collisions** and other **issues** that are hard to trace.
- We recommend creating a **single entry file** that only contains the **import statements**.

useBuiltIns: 'usage'

SoftUni Foundation

- Adds **specific imports** for polyfills when they are **used** in each file. We take advantage of the fact that a **bundler** will **load** the same **polyfill** only **once**
- If environment **supports** it

```
var a = new Promise();
```

- If environment **doesn't support** it

```
import "core-js/modules/es.promise";
var a = new Promise();
```

React Preset

- @babel/preset-react
  - Installation

```
npm install --save-dev @babel/preset-react
```

  - Usage

```
{
  "presets": ["@babel/preset-react"]
}
```

TypeScript Preset

Setting Up TypeScript

SoftUni Foundation

- Microsoft's TypeScript is a compiled language that follows a similar setup as Babel
- The neat thing is that in addition to JavaScript, it can emit type definitions
- Stronger typing is valuable for development as it becomes easier to state your type contracts
- You can use TypeScript with webpack using the following loaders:
  - ts-loader
  - awesome-typescript-loader

ts-loader

- Installation

```
npm install ts-loader --save-dev
```

- You also need TypeScript if you don't have it already

```
npm install typescript --save-dev
```

- Configuration
  - Create or update webpack.config.js
  - Add a **tsconfig.json** file

webpack.config.js

SoftUni Foundation

```
module.exports = {
  mode: "development",
  devtool: "inline-source-map",
  entry: "./app.ts",
  output: {
    filename: "bundle.js"
  },
  resolve: {
    // Add `.ts` and `.tsx` as a resolvable extension.
    extensions: [".ts", ".tsx", ".js"]
  },
  module: {
    rules: [
      // all files with a `.ts` or `.tsx` extension will be handled by
`ts-loader`
      { test: /\.tsx?$/, loader: "ts-loader" }
...
```

tsconfig.json

SoftUni Foundation

- The tsconfig.json file **controls** TypeScript-related options so that your IDE, the tsc command, and this loader all **share** the same **options**

```
{
  "compilerOptions": {
    "sourceMap": true
  }
}
```

ESLint

SoftUni Foundation

- A tool for **identifying** and **reporting** on **patterns** found in ECMA Script/JavaScript code, with the goal of **making code** more **consistent** and **avoiding bugs**
- You can install ESLint using npm

```
npm install eslint --save-dev
```

- You need **.eslintrc** file for configurations
- In order to use it with **webpack** you should use **eslint-loader**

eslint-loader

SoftUni Foundation

- Install

```
npm install eslint-loader --save-dev
```

- Usage

```
module.exports = {
  // ...
  module: {
    rules: [
      {
        test: /\.(js|jsx)$/,
        exclude: /node_modules/,
        use: ['babel-loader', 'eslint-loader'],
...
```

Live Exercise

Summary

SoftUni Foundation

- Babel **gives** you **control over** what **browsers** to support
  - It can **compile** ES2015+ **features** to a **format** the older browser understand
  - allows you to use **experimental** language features
- **babel-preset-env** is valuable as it can **choose** which **features** to **compile** and which **polyfills** to **enable** based on your **browser** definition
- Besides Babel, webpack supports other solutions like **TypeScript**

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

XSsoftware

SoftwareGroup
doing it right

NETPEAK

SmartIT

SUPER HOSTING .BG

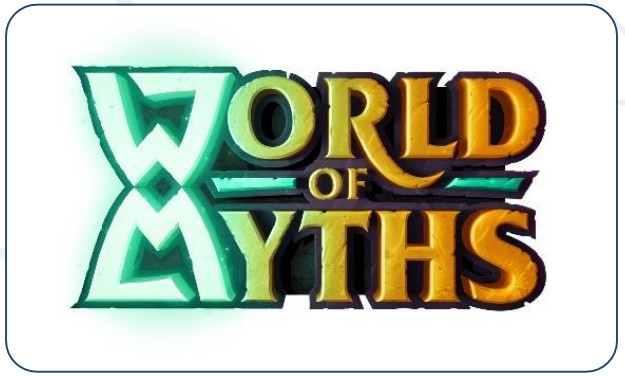
INDEAVR
Serving the high achievers

INFRAGISTICS®

LIEBHERR

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

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