Babel

JavaScript Compiler

ES6 Browser Support

- Newest versions:
  - chrome: 97% , FireFox 97%, Safari 99%, IE 96%, Opera 97%,
- More then 12% on desktop are still using IE11 (www.netmarketshare.com)
- IE11 ES6 support = 11%
- Around 30% of users on desktop and mobile will have less then 20% support for ES6

Problem

- ES6 is cool and we don’t want to write in ES5
- We want our app to run for all users
- ES6 is not supported for all users
- Example: Modules are currently only working in the new chrome and safari
- We want to also to use experimental features not in ES8
    - example decorators are at stage 2

Solution

- Add build process to create our app
- During the build process compile our ES8 + Experimental code and turn it into ES5

Introducing Babel

- Babel is a popular compiler for JavaScript
- Like similar compiler it has 3 stages
  - parse the code
  - transform it
  - generate new code
- By default babel doesn’t do anything to the code (parsing and generating the same code again)
- transforming the code is done with **plugins**
- it’s common to use more than one plugin, for this case we have **presets** which contain a group of **plugins**

babel-cli

- babel is written with node
- there is babel-core which can be used to transform code from your node project
- there is babel-cli which you can use to transform code from your command line
- for this course we will use babel-cli to compile our code
- install babel-cli with npm
- after that you will have **babel** command in your command line
- usage:
    - **babel <src> --out-file <dest>**

Plugins/Presets

- without plugins or presets babel doesn’t do much
- we have to install plugin/presets from npm
- we have to tell babel to use those plugins/presets
- babel configuration is in a file called **.babelrc**
- **babel-preset-env** includes presets for es2015, es2016, es2017
- **babel-preset-stage-0** includes experimental features from stage-0, 1, 2, 3 (like decorators)
- **babel-preset-react** - includes support for jsx used with react
- **babel-plugin-transform-decorators-legacy -** this is a plugin which will allow us to use decorator syntax
- loading multiple presets the order will be from last to first
- Let’s install those presets and compile our file with babel again

webpack + babel

- Our project build is done with **webpack**
- We want the babel compilation process to be part of our webpack build
- babel and webpack play nice together and we can tell webpack to process files with js extension with **babel-loader**
- We need to install **babel-loader** with **npm**
- **{test: /\.js$/, loader: ‘babel-loader’}**
- let’s try and install **webpack** and process the file using **webpack** and **babel-loader**

Summary

- With babel our es6 project can be compatible with older browsers
- we can also use experimental features
- we will use webpack and babel for our react project
- for the rest of the course we will copy the configuration we did to the next lessons in react