WEBPACK

- Open terminal and create a folder on the desktop.
- Open that folder and create a **package.json** file.

```
cd desktop
mkdir wp
cd wp
npm init -y
```

- Install webpack in this folder

```
npm install webpack
```

- NPM version 5+ will update package.json.

Source code

- Create a source folder where your original JS is stored.

```
mkdir source
```

- Create a **source/modules/utils.js** which exports a function and a constant.

```
function hyphenate(words) {
  return words.split(" ").join("-");
}

const company = "ACME Bagels Limited 2017";
export { hyphenate, company };
```

- Create **source/code.js** which imports and uses the function and constant.

```
import { hyphenate, company } from "./modules/utils" ;

console.log( hyphenate( company ));
```

- Create a destination folder where Webpack will write bundled code.
- Add a simple HTML file to the destination: **dest/index.html**

```
<html>
    <head>
        <title>Webpack</title>
    </head>
    <body>
        <script src="code.js"></script>
    </body>
</html>
```

- Webpack will bundle the two source .JS files together into a single .JS file in the dest folder.
- **Webpack** transpiles ES6 **module** syntax out of the box, without the need to install additional **transpilation** tools like **Babel**.
- Webpack will not alter any code other than import and export statements.

Run Webpack

- We can run Webpack without a configuration file on the command line.
- We need to include the full path to Webpack

```
./node_modules/.bin/webpack source/code.js dest/code.js
```

- This will generate dest/code.js
- Run dest/index.html in the browser. This should log output to the console.

```
ACME-Bagels-Limited-2017
```

- We can add the Webpack command to a package.json script.
- Within scripts we can reference locally installed npm packages by name instead of the full path

```
"scripts": {
  "build": "webpack source/code.js dest/code.js"
},
```

- We can then run this script

```
    npm run build
```

A Webpack configuration file

- We can create a configuration file **webpack.config.js**.
- It defines where the root .JS file in the source folder is, and the destination for the bundled files.

```
const path = require('path');

module.exports = {
    entry: './source/code.js',
    output: {
        filename: 'code.js',
        path: path.resolve(__dirname, 'dest')
    }
};
```

- We can update package.json to make Webpack read this config file.

```
"build": "webpack --config webpack.config.js"
```

- We run this the same way. Now Webpack reads the config file.

```
npm run build
```

- Webpack looks for the config file by default, so we can simplify the package.json script.

```
"build": "webpack"
```

- We can add debug and watch parameters to the command.
- Webpack will now run after every source code change we make.
- The Webpack config file can be refactored to move all the paths into a constant.

```
const path = require('path');
```

```
const paths = {
  DEST: path.resolve(__dirname, 'dest'),
  SOURCE: path.resolve(__dirname, 'source'),
};

module.exports = {
  entry: path.join(paths.SOURCE,'code.js'),
  output: {
    filename: 'code.js',
    path: paths.DEST
  }
};
```