



JÖNKÖPING UNIVERSITY

School of Engineering

WEBPACK

Web Development with JavaScript and DOM

TWJK14 Spring 2017

Peter Larsson-Green

WEBPACK

Is a module bundler: <https://webpack.js.org/concepts/>

- Input: a single file with dependencies to other files.
- Output: a single file large file.

ES2015 IMPORT EXPORT

```
import sayHi from './file-b'  
  
sayHi()
```

file-a.js

```
export default function() {  
  console.log("Hi!")  
}
```

file-b.js

webpack file-a.js output.js

```
var sayHi = function() {  
  console.log("Hi!")  
}  
  
sayHi()
```

output.js

WEBPACK

Is distributed as an npm package.

1. Install node.js: <https://nodejs.org/en>
 - This also installs npm.
2. Create a new folder for your project: `mkdir my-app; cd my-app`
3. Setup npm usage: `npm init --yes`
 - This creates `package.json`.
4. Install the webpack package: `npm install webpack --save`
5. In `package.json`, add a script starting webpack:
 - `"build": "webpack input-file.js output-file.js"`
6. Run the script: `npm run build`

WEBPACK CONFIGURATION FILE

Without configuration file:

- `webpack main.js build.js`

With a configuration file:

- `webpack`

```
module.exports = {  
  entry: './main.js',  
  output: {  
    filename: 'build.js',  
    path: __dirname  
  }  
}
```

`webpack.config.js`

WEBPACK LOADERS

Files can be imported through loaders.

- A loader can change the content of the file before it is loaded.

```
export function sayHi() { console.log("Hi!") }
```

file-b.js

```
import { sayHi, length } from './add-length-loader!./file-b'
```

```
sayHi(); console.log(length)
```

file-a.js

```
export var length = 45
```

```
export function sayHi() { console.log("Hi!") }
```

```
module.exports = function(content) {
```

```
  return "export var length = "+content.length+"\n" + content
```

```
}
```

add-length-loader.js

WEBPACK LOADERS

Files can be imported through loaders.

- A loader can change the content of the file before it is loaded.

```
{"a": 1, "b": 2}
```

info.json

```
import info from './json-loader!./info.json'  
console.log(info.a)
```

file-a.js

```
export default {  
  "a": 1,  
  "b": 2  
}
```

```
module.exports = function(content) {  
  return "export default "+content  
}
```

json-loader.js

WEBPACK LOADERS

Files can be imported through loaders.

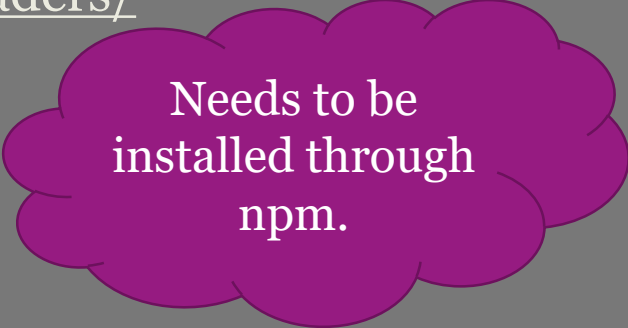
- A loader can change the content of the file before it is loaded.

Some existing loaders are listed at:

- <https://webpack.js.org/loaders/>

- E.g.:

- json-loader
- raw-loader
- coffee-loader
- babel-loader (with babel-core & babel-preset-es2015)




Needs to be
installed through
npm.

- Loaders can be configured in `webpack.config.js`.

WEBPACK LOADERS

Loaders can be configured in `webpack.config.js`.

```
module.exports = {  
  entry: './main.js',  
  output: {  
    path: __dirname,  
    filename: 'build.js'  
  },  
  module: {  
    rules: [RULE1, RULE2, RULE3, ...]  
  }  
}
```



```
{  
  test: /\.js$/,  
  loader: "babel-loader",  
  options: {  
    presets: ["es2015"]  
  }  
}
```

All ES6 syntax in `.js` files
will now be transpiled to
ES5 syntax.

BABEL PRESETS

Each feature babel can transpile is called a plugin.

- Available plugins: <https://babeljs.io/docs/plugins/#transform-plugins>

We need to tell babel which ones to use.

- Boring to list all of them.

A preset is a predefined set of these plugins.

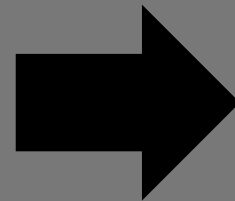
- Available presets: <https://babeljs.io/docs/plugins/#presets-official-presets>

WEBPACK SOURCE MAPS

Webpack and babel transform our code.

- Errors in original code will be discovered in the transformed code.
- Source maps to the rescue!
 - Maps the transformed code back to the original code.

```
module.exports = {  
  entry: './main.js',  
  output: {  
    path: __dirname,  
    filename: 'build.js'  
  },  
  devtool: "source-map"  
}
```



Generates
build.js and
build.js.map.

WEBPACK SOURCE MAPS

Webpack and babel transform our code.

- Errors in original code will be discovered in the transformed code.
- Source maps to the rescue!
 - Maps the transformed code back to the original code.
- Web browsers do not necessarily support source mapping.
- Creating source map takes time → build process slower.
 - Webpack offers different ways to generate source maps:
 - <https://webpack.js.org/configuration/devtool/>

WEBPACK PLUGINS

Webpack support plugins.

- Each plugin adds new functionality to webpack.
- List of common plugins: <https://webpack.js.org/plugins/>
 - E.g. UglifyjsWebpackPlugin: <https://webpack.js.org/plugins/uglifyjs-webpack-plugin/>
- Added through the configuration file:

```
module.exports = {  
  entry: './main.js',  
  output: {  
    path: __dirname,  
    filename: 'build.js'  
  },  
  plugins: [PLUGIN1, PLUGIN2, ...]  
}
```

WEBPACK WATCH MODE

In watch mode, webpack re-runs whenever you change a file.

- Start it with the `watch` flag:
 - `webpack --watch`
- Even cooler: use `webpack-dev-server` instead.
 - <https://webpack.js.org/guides/development/#webpack-dev-server>
- Supports *Hot Module Replacement*:
 - Code changes are pushed to the browser.
 - Perfect for CSS code (CSS is stateless).
 - Less suitable for HTML & JS.
 - Perfect for components from libraries/frameworks (React, Vue, ...).