Lesson 9: Webpack

Wannes Fransen & Tom Eversdijk

UC Leuven

2020

What is webpack?

Core concepts

What is webpack?

Static module bundler for modern JavaScript (and more)

- *local scope* vs *global scope*

```
<script>
    var hello; //this is actually window.hello
</script>
```

```
<script>
    function myFunction() {
    // this is actually window.myFunction
    }
</script>
```

What is webpack?

Static module bundler for modern JavaScript (and more)

- *local scope* vs *global scope*

```
<script>
    var Module = (function() {
     // some code for your module
    }());
</script>
```

What is webpack?

Static module bundler for modern JavaScript (and more)

- *local scope* vs *global scope*

```
<script src="/js/slideshow.min.js"></script>
<script src="/js/underscore.min.js"></script>
// window._

<script src="/js/datepicker.min.js"></script>
<script src="/js/lodash.min.js"></script>
// window._ (OVERWRITE UNDERSCORE)
```

[LINK]

What is webpack?

Static module bundler for modern JavaScript (and more)

- *local scope* vs *global scope* $\Rightarrow$ *Module scope*
- Not limited to JavaScript, include all assets type (css, images, fonts, etc)
- speed optimization: chunks, HTTP/2, compression, code splitting

How does it work?

Internal dependency graph

- File depending on another file is a dependency
- Non-code assets can be provided as dependecies
- Recusively build a dependecy graph that includes every module your application needs
- Bundles modules to be loaded by the browser

What is webpack?

Core concepts

Entry

- Entry point to begin building internal dependency graph
- Default: *./src/index.js*

webpack.config.js

```
module.exports = {
    entry: './path/to/my/entry/file.js'
};
```

Output

- Output tells where to emit the bundles it creates and how to name these files.
- *Default: ./dist/main.js*

webpack.config.js

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

module.exports = {
  entry: './path/to/my/entry/file.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'my-first-webpack.bundle.js'
  }
};
```

Loaders

- Webpack only understands JavaScript & Json files
- Loaders allow to process other types of files and convert into valid modules
- **Test property**: identifies which files should be transformed
- **Use property**: identifies which loader should be used to do the transforming

Regex

```
Keep in mind that when using regex to match files,
you may not quote it. e.g. /\.txt$/ is
not the same as '/\.txt$/' or "/\.txt$/"
```

Loaders

webpack.config.js

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

module.exports = {
  output: {
    filename: 'my-first-webpack.bundle.js'
  },
  module: {
    rules: [
      { test: /\.txt$/, use: 'raw-loader' }
    ]
  }
}
```

Plugins

- Bundle optimization
- asset management
- injection of environment variables
- Webpack out of the box plugins [LINK]

Plugins

webpack.config.js

```
//installed via npm
const HtmlWebpackPlugin = require('html-webpack-plugin');
//to access built-in plugins
const webpack = require('webpack');

module.exports = {
  module: {
    rules: [
      { test: /\.txt$/, use: 'raw-loader' }
    ]
  },
  plugins: [
    new HtmlWebpackPlugin({template: './src/index.html'})
  ]
};
```

Mode

- Enable built-in optimizations
- Default: *production*
- Options: *development*, *production or none*

webpack.config.js

```
module.exports = {
  mode: 'production'
};
```