Webpack and React

Ustun Ozgur

2015-05-26 Tue

Webpack

- Bundlers
- Packs web related things (js, css, html) to one bundle file
- Similar to browserify
- More popular in React ecosystem
- Webpack supports ES6 modules, commonjs modules and amd modules.

CommonJS modules

- We will focus on this as it is the same as in node.
- Each module exports either a single item or an object

Single item exported

- `module.exports = foo` in foo.js
- Usage: `foo = require('./foo')` in bar.js in same dir.

Multiple items exported and required

- `module.exports.bar = bar; module.exports.baz = baz` in foo.js
- Usage: `foo = require('./foo'); foo.bar()` in second.js

Alternative syntax

- `module.exports = {bar: bar, baz: baz};`
- ES6 shortcut: `module.exports = {bar, baz};`
- For requiring ES6 shortcut:
  `{bar, baz} = require('./foo')` (destructuring)

React components

- Need a way to transform JSX to JS
- webpack has loaders
- either jsx-loader or babel-loader
- babel-loader is more comprehensive
- `npm install babel-loader --save`

webpack.config.js

- Need a config file to run webpack
- This specifies the entry and output files, and the loaders to use.

```
module.exports = {
  entry: './component.js',
  output: {path: 'build', filename: 'bundle.js'},
  module: {
    loaders: [
      { test: /\.js$/, exclude: /node_modules/,
        loader: "babel-loader"}
    ]
  }

}
```

- We can also require react, underscore etc and they will be bundled as well if they are installed as node\modules.

Production Settings

- Run with `webpack -p`
- File name is fixed by default. Append `[chunkname]`

```
module.exports = {
  entry: './component.js',
  output: {path: 'build',
           filename: 'bundle.[chunkhash].js'},
  module: {
    loaders: [
      { test: /\.js$/, exclude: /node_modules/,
        loader: "babel-loader"}
    ]
  }
}
```

Retrieving the output hash

- Run with `-j` flag to get a json summary.
- Pipe that through `jq` to get a specific data in the json output

```
webpack -j | jq ".assetsByChunkName.main"
```

- jq is at `http://stedolan.github.io/jq/`

Production and Dev in a Single File

```
module.exports = {
  entry: './component.js',
  output: {path: 'build',
           filename: (
             process.env.PROD ?
               'bundle.[chunkhash].js'
               : 'bundle')},
  module: {
    loaders: [
      { test: /\.js$/, exclude: /node_modules/,
        loader: "babel-loader"}
    ]
  }
}
```

Using this webpack config for dev and prod

- For dev: run `webpack`
- For prod: run
  `PROD=true webpack -p | jq ".assetsByChunkName.main" >`
- In your HTML, somehow embed the contents of `bundle_hash`

Further Tips for Development

- Watch mode: -w flag
- Development static server
- Install and run webpack-dev-server.
- Point to the following for websocket connection:
- `<script src="http://localhost:8080/webpack-dev-server.js`
- The bundle will be generated at `http://localhost:8080/bundle.js`
- Point to this url in your HTML.
- Make changes to your JS files and ensure that the page is refreshed automatically.

Exercise:

- Study the solution at `/solutions/01_hello_world_webpack`
- Go through the sample webpack example to make sure you understand how webpack works, and how its auto reloading dev server works.
- Make some changes to the hello world application and make sure that auto reload works as advertised.
- Componentize the todo application so that each component lives in a separate file and compile it to a single bundle file using webpack.