Contents

webpack

Problem and solution

Multiple js files are hard to manage. A sophisticated web application usually has a large number of js files to include. The usage of `<script>` tags is not enough to handle this.

Luckily, webpack is here to help. It compiles js files, including their dependencies, into a single js file. Webpack can do the same to css files and even image assets.

simple example with npm dependencies

See `simple-npm-deps` branch for demo.

```
webpack {source_file} {bundle_file}: simply handles the dependencies elegantly
```

simple example with local dependencies

See `simple-local-deps` branch for demo.

```
module.exports = {object_to_export}: the simplest way to export
require('{file_path'): the simplest way to require
```

Production mode

- `webpack -p` : production mode, with javascript minification

Watch mode

- `webpack --watch` : watch mode, automatically rebuilds on change

Bail mode

By default, webpack doesn't exit with status 1 (meaning error) on errors. But bail mode is there to do the right thing.

- `webpack --bail` : bail mode, exit status is 0 for no success and 1 for error

  ./node_modules/webpack/bin/webpack.js --bail if [$? -ne 0]; then echo "webpack build failed. Aborting.." exit 1 fi echo "webpack build succeeded"

webpack-dev-server

Problem and solution

Running **npm start** every time after a change is annoying!

Use **webpack-dev-server**. It saves us the need to on npm start and run it again.

demo and explanation

- `npm i`
- `npm start`
- visit `http://localhost:8080/`
- update `src.js`
- go back to browser, refresh `http://localhost:8080/` and notice the update

Curiously, notice that npm start doesn't actually provide an output file, and it is safe to physically delete **bundle.js** file (I have already done that). But http://localhost:8080/bundle.js is accessible.

This means **webpack-dev-server** automatically generate and serve bundle.js.

json loader for webpack

https://github.com/webpack/json-loader/

Installation

```
npm install json-loader --save
```

Usage

```
var json = require("json!./file.json");
// => returns file.json content as json parsed object
```