# React I

Martin Ledvinka

martin.ledvinka@fel.cvut.cz

Winter Term 2017

## Contents

1 Deployment and build
- Build Process
- Build Tools

2 Modules

3 Tasks

## Deployment and build

## Deployment

How is the web content served by our application?

- The configuration is in `WebAppConfig`,
- Resource handlers map URLs to locations of static content (JS, CSS),
    - More efficient access,
- We enable default servlet configuration,
- Configure Jackson message converter for serialization/deserialization of request data,
- `index.html` is in the root of war and is served by default by the application server.

## Tools Used in React Apps

- NodeJS,
  - $\approx$ JVM,
  - JavaScript application server, runtime for executing JavaScript applications,
  - Our case – run the build tools only,
- npm,
  - $\approx$ Maven,
  - Used for dependency management in JS,
  - And for execution of build tools,
  - `package.json` – configuration file,
    - *dependencies* – packed into the target bundle,
    - *devDependencies* – used only during development.

## Build Process

The following applies to both ear-rt and ear-setup.
Execute them from `${project.basedir}/src/main/webapp`.
Section *scripts* in `package.json`.

1. `npm install` – downloads all JS dependencies declared in `package.json`,
2. `npm run build` – builds production version of the JS UI,
3. `npm run build-dev` – builds non-minified version of the JS UI,
4. `npm start` – starts file watcher which rebuilds development version of the JS UI when changes in source code are detected. Used during development and debugging.

## Build Tools

- Sections *devDependencies* and *scripts* in `package.json`.

### Browserify

- Recursive analysis of module imports in JS code,
- Builds a bundle of all the discovered modules,
- Bundle is then served in HTML page in a single `script` tag.

### Babel

- JS compiler, enables use of the latest JS syntax in older-browser compatible way,
- Used as transformation step in *Browserify* (*babelify*).

## Build Tools II

### Watchify

- Watches for changes in source files and reruns *Browserify* when a change is detected.

### Uglify

- JS code minification,
- Reduces resulting bundle size.

### clean-css

- CSS optimization and minification,
- Reduces CSS file size.

## Modules

## CommonJS Modules

CommonJS

- Specification of modules in JS,
- No module specification built in JS until ES6,
- Implemented by NodeJS - function `require`,
- And object `module.exports`,
- Works for:
  - External libraries,
  - Local files.

## ES6 Modules

- EcmaScript 6 comes with its own definition of modules,
- `import`, `export` keywords,
- Not compatible with CommonJS,
- Still not supported by NodeJS and older browsers,
- Babel translates into CommonJS `require` and `module.exports`.

**For Interested**

*Compare the source codes of ear-rt and the compiled* `bundle.js` *to see the difference.*

## Tasks

## Tasks

Working with ear-rt, zip downloaded from CourseWare.

### Together

1. Connect the `Tags` component with its store.
2. Add the `Tags` component to `Report`.
3. Implement new tag creation.

### You

1. Implement rendering of tags of a report.
2. Implement adding tags to a report.
3. Implement removal of tags from a report.

Solution will be available as usual at `https://gitlab.fel.cvut.cz/ear/reporting-tool-seminars`, branch WS2017-seminar-9-solution.

## Resources

- https://blog.risingstack.com/node-js-at-scale-module-system-commonjs-require/
- https://hackernoon.com/node-js-tc-39-and-modules-a1118aecf95e
- https://developer.mozilla.org/cs/docs/Web/JavaScript/Reference/Statements/import
- https://github.com/reflux/refluxjs
- https://facebook.github.io/flux/