REACT - DAY ONE

TOOLCHAIN

Bartosz Cytrowski — infoShare Academy

NVM

Helps you manage multiple NodeJS environments on one machine.

https://github.com/creationix/nvm

NODEJS

It is a platform on which we can run JavaScrip. It has access to wide range of operating system capabilities.

https://nodejs.org/en/

NPM

Node Packaged Modules (a big repository)

https://www.npmjs.com/

Node Package Manager (a tool)

https://docs.npmjs.com/

YARN

A tool solving the same problems as `NPM` yet faster.

https://yarnpkg.com/lang/en/

WEBPACK

A tool for bundling web application. It allows us to develop an application using multiple linked files and it merges them before sending to the browser.

https://webpack.js.org/

WEBPACK DEV SERVER

A `Webpack` extension working as a HTTP server with live-reload capabilities - refreshes the browser when we change our code.

https://webpack.js.org/configuration/dev-server/

BABEL

A JavaScript transpiler capable of transforming modern JS syntax into older one allowing us to write code using ES6, 7, 8, 9 etc.

https://babeljs.io/

ESLINT

A static code analyser which detects common code problems without running it.

https://eslint.org/

AUTOPREFIXER

A tool that adds browser-prefixes to your CSS rules.

https://github.com/postcss/autoprefixer

CRA (CREATE-REACT-APP)

Bootstrapping script that allows you to start building React app without thinking about the whole configuration process. **Everyting you need to know is written in the docs**.

READ THOSE EVERY SINGLE DAY

https://facebook.github.io/create-react-app/docs/documentation-intro

MODULES, IMPORTING AND EXPORTING

Every single file creates a closure so we no longer have to care about accidentally creating global variables.

We name those files `modules`

We can define which value can be accessed from the outside of given file by exporting it using `export default`:

```
const sum = (a, b) => a + b

export default sum
```

The code above makes it possible to import the `sum` function in other file so that it can be used there.

```
// math.js
const sum = (a, b) => a + b

export default sum
```

You can import sum in other file multiple times (it's impractical). You define the name for it during import:

```
// index.js
import sum from './math'
import dodaj from './math'
import blah from './math'

console.log(sum(1, 2)) // -> 3
console.log(dodaj(1, 2)) // -> 3
console.log(blah(1, 2)) // -> 3
```

We can export more than one value from given file using `named exports`

```
// math.js
export const sum = (a, b) => a + b
export const multiply = (a, b) => a * b
```

And then we can import them by name or create an alias for given name

```
import { sum } from './math' // notice the curly brackets
```

```
import { sum as addTwoValues } from './math'
import { sum, multiply } from './math'
import { sum as fizz, multiply as buzz } from './math'
```

The `import` and `export` parts of the file are being resolved by `Webpack` during source code processing.

`Webpack` builds a dependency tree and tries to order every `module` in a way that every definition required by our code is available before execution.

Just imagine keeping houndreds of `<script>` tags in order.

MODERN JS FEATURES

Our toolchain uses `Babel` to transpile our source code into code that can be understood by the browser.

It means we can use basically every feature of ES6, 7, 8, 9 to write our code - even if it is not yet standardised.

`Babel` is being triggerd by the `Webpack` as soon as it finds a file with `js` or `jsx` extension during the compilation time.

This behavior can be customized since `Webpack` and `Babel` are huge software monsters on our command 😈

To learn what you can do with ES2015 (ES6) go to the docs:
https://babeljs.io/docs/en/learn

THAT'S ALL..

...YOU HAVE TO KNOW ABOUT THE TOOLS FOR NOW