ReactJS Notes

1. Introduction to React.js

1.1. React Overview

What is React.js?

- **React** is a JavaScript library for building **user interfaces** (UI)
- Focused on creating **reusable components** - преизползване
- **all model, view and controller are in each component!!!** – всичко е наблъскано в компонента, но това всичко да е малко/за определено нещо. Иначе се нарушава логиката на ReactJS.
- Component е много умен template, който може да прави много неща
- Developed by **Facebook**

Features

- Open-source
- Declarative подход на писане
  - Design **simple** views for each **state** in your app
  - Easier to **debug**
- Component-Based
  - Encapsulated **components** that manage their **own** state
  - Keep **state** out of the **DOM**

- Isomorphic
  - JavaScript that runs on **both** client-side & server-side
  - **NextJS** спомага за приложения, които трябва да живеят на server-side страната
  - Better user experience
- Native support
  - Compose rich **mobile** UI in **Android**, **iOS**

Advantages

- Easy to learn
- Fast **performance**
- Use all **ES6** features
  - **Promises**, **Classes** and **Modules**
- Compatible with other **libraries**
- Great **error reporting**

1.2. React Installation

Packages, Setup, Structure

CDN using

https://legacy.reactjs.org/docs/getting-started.html
https://legacy.reactjs.org/docs/add-react-to-a-website.html

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="https://unpkg.com/react@18/umd/react.development.js" crossorigin></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"
crossorigin></script>
    <title>Document</title>
</head>
```

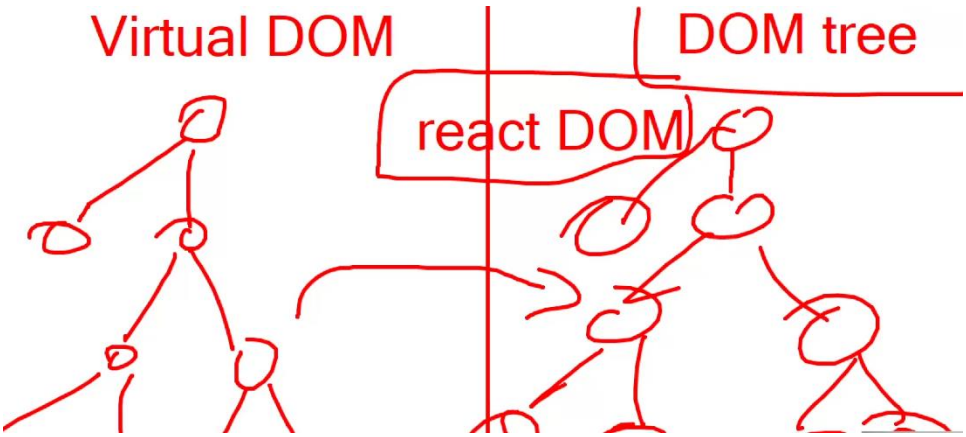
How React work

1. Дисплейва първо Hello from server като на Preview в dev tools Chrome на network на index.html се показва само Hello from server
2. Клиента получава response от сървъра, като на свой ред се изпълнява кода от JS скрипта, което дисплейва и втория елемент

```
<body>
    <h2>Hello from server</h2>
    <div id="root"></div>

    <script>
        const rootDomElement = document.getElementById('root');
        console.dir(rootDomElement);
        const rootReactElement = ReactDOM.createRoot(rootDomElement);

        const headingReactElement = React.createElement('h1', {}, 'Hello from React');
        console.log(JSON.parse(JSON.stringify(headingReactElement)));
        rootReactElement.render(headingReactElement);
    </script>
</body>
```



React DOM библиотеката взема всичко от Virtual DOM (React) и го пренася в добре познатото ни DOM tree

Самият Virtual DOM React елемент е супер лек с малко полета

```
_owner: null
▸ _store: {}
▸ [[Prototype]]: Object
```

А стандартния DOM tree елемент е доста тежък с мега много много много атрибути

```
console.dir(rootDomElement);
```

```
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools
▾ div#root
  ▸ __reactContainer$fvjv2nzqj8: FiberNode {tag: 3, key: null, elementType: null, type: null, stateNode: Fib
    _reactListening6wzw7bxs65x: true
    accessKey: ""
    align: ""
    ariaAtomic: null
    ariaAutoComplete: null
    ariaBrailleLabel: null
    ariaBrailleRoleDescription: null
    ariaBusy: null
    ariaChecked: null
    ariaColCount: null
    ariaColIndex: null
    ariaColSpan: null
    ariaCurrent: null
    ariaDescription: null
    ariaDisabled: null
    ariaExpanded: null
    ariaHasPopup: null
    ariaHidden: null
    ariaInvalid: null
    ariaKeyShortcuts: null
    ariaLabel: null
    ariaLevel: null
    ariaLive: null
    ariaModal: null
    ariaMultiLine: null
    ariaMultiSelectable: null
    ariaOrientation: null
    ariaPlaceholder: null
    ariaPosInSet: null
    ariaPressed: null
```

Nested elements without JSX

```
<body>
    <div id="root"></div>
```

```
    <script>
        const rootDomElement = document.getElementById('root');
        console.dir(rootDomElement);
        const rootReactElement = ReactDOM.createRoot(rootDomElement);

        const headingReactElement = React.createElement('h1', {}, 'Hello from React');
        const secondHeadingReactElement = React.createElement('h2', {}, 'Some slogan here');
const headerReactElement = React.createElement('header', {}, headingReactElement,secondHeadingReactElement);
        rootReactElement.render(headerReactElement);
    </script>
</body>
```

```
<div id="root"> event
  <header>
    <h1>Hello from React</h1>
    <h2>Some slogan here</h2>
  </header>
</div>
```

Create React App

- Less to learn - **instant reloads** help you focus on development
- Only one dependency - no complicated version mismatches
- No Lock-In - under the hood **Webpack**, **Babel**, **ESLint**
- Install the React app creator (one-time global install)

https://create-react-app.dev/docs/getting-started - трета документация за React, по-скоро документация за инсталатора

React App Creator - Install and Run the React App Creator

- Run the React app creator - a toolchain of several libraries.

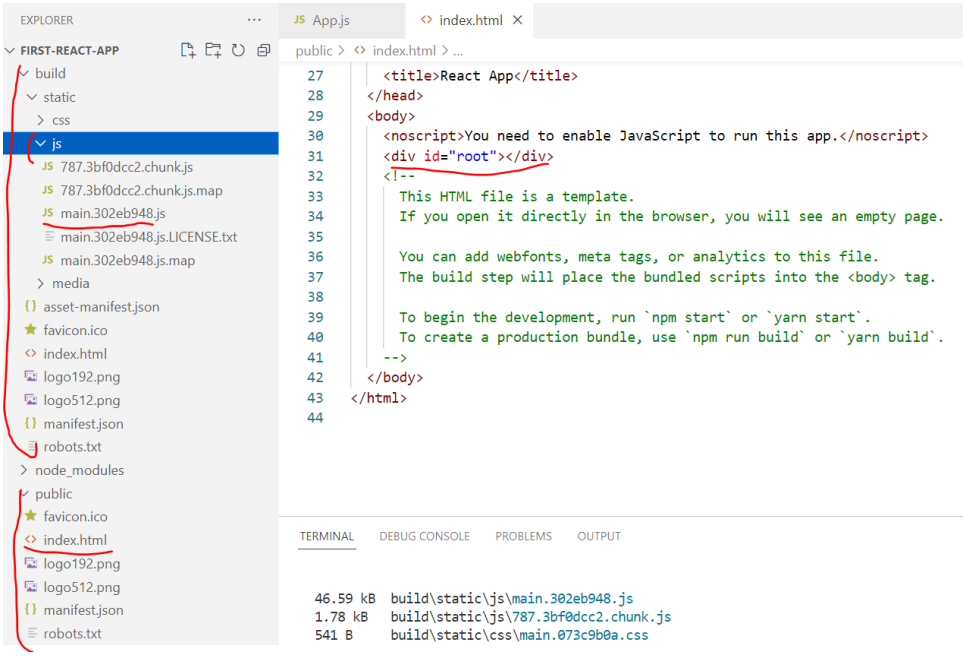
Easily we can start our react application. In the past a lot of efforts to start the react - many libraries separately to install
npx create-react-app my-app създава папка my-app в текущата директория, в която се намираме
npx create-react-app . текущата директория инсталирай
Взима най-новата версия на react app creator и я изпълнява

development mode Starts your React app in from the command line
cd my-app
npm start

- Browse your app from http://localhost:3000
- ReactJS работи **по-ефективно от hot-reloading**! В ReactJS се нарича **hot-module-replacement**! Браузърът няма да презареди целия html, а само променените части/модули
- npm start го стартирва така, че JSX кода се трансформира в ReactJS код и в последствие до обикновен JS код, който браузъра чете, и всичко това става в паметта и не се вижда

production mode Starts your React app in from the command line

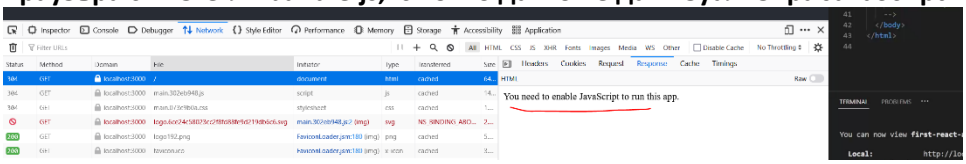
Ако искаме да видим как се транспилира JSX кода, то изпълняваме до т.н. **статичен код**:
npm run build
И ни се появява познатата папка **build**



Със следните 2 команди можем да го заредим на браузъра след това.

The build folder is ready to be deployed.
You may serve it with a static server:
npm install -g serve
serve -s build

Браузъра си изтегля bundle.js, с който да може да визуализира JavaScript кода.



package.json out of the box:

```
{
  "name": "first-react-app",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
```

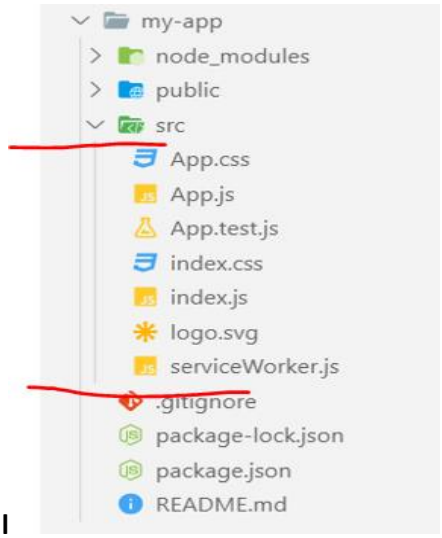
```
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",          development mode
    "build": "react-scripts build",          production mode
    "test": "react-scripts test",           unit/интеграционни  тестове out of the box
    "eject": "react-scripts eject"         веднъж eject-не ли се, не можем да го inject-нем наново
  },
  "eslintConfig": {                    за намиране на грешки в нашия код
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  }
}
```

Finding Information

- Visit the **official website -** https://reactjs.org/
- Documentation - https://reactjs.org/docs/installation.html
- Online sandbox - https://codesandbox.io/ - ако искаме да споделим част от кода на някого в изолирана среда

React App Structure

- **package.json** - project configuration
  - Module name, dependencies, build actions



- **index.html**
  - App main HTML file

- **index.js**
  - App main JS file (startup script)

- **App.js**, **App.css**, **App.test.js**
  - React component "**App**"

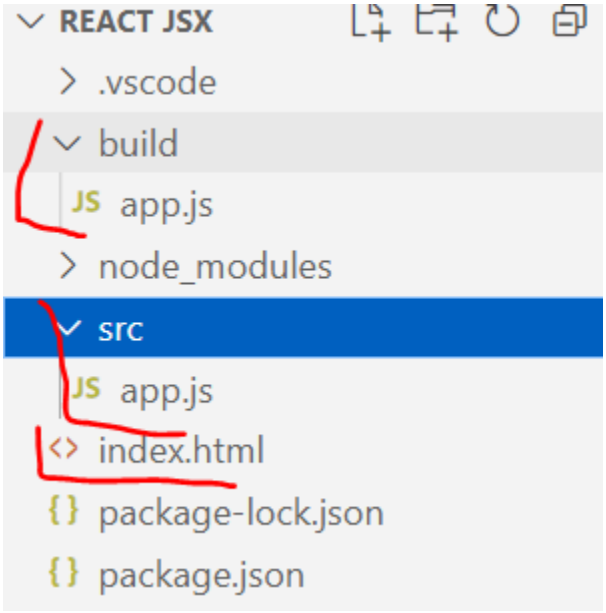
1.3. JSX Syntax

JSX installing

Гледаме и двете документации (старата и новата)
https://legacy.reactjs.org/docs/add-react-to-a-website.html (the old one)
https://react.dev/learn (new one)

with CDN
`script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>`

without CDN
Step 1: Run `npm init -y` (if it fails, here's a fix)
Step 2: Run `npm install babel-cli@6 babel-preset-react-app@3`
Step 3: npx babel --watch ./src --out-dir ./build --presets react-app/prod



След третата команда каквото напишем в ./src/app.js то автоматично отива и в ./build/app.js

JSX Overview

- **JSX** is React's JavaScript **superset language** – пишем друг синтаксис, и той се превежда до обикновен **JS**
  - Has all of JavaScript's **features** and more
- Unique approach to **mixing HTML and JS**
- Compiles to **plain JavaScript**
- **Браузърите не разбират JSX, затова трябва компилатора babel**

```
<div className="red">Children Text</div>
```
този ред не е HTML!!! **Думата class е запазена, затова използваме className**

Превежда се до следния JS код:

```
React.createElement("div",
  { className: "red" },
  "Children Text"
);
```

Думата for също е запазена, затова използваме htmlFor

```
<label htmlFor="name" className="col-sm-3 control-label">Name<sup>*</sup></label>
```

Или в нашия пример

```
const rootDomElement = document.getElementById('root');
console.dir(rootDomElement);
const rootReactElement = ReactDOM.createRoot(rootDomElement);
```

заместваме този синтаксис

```
// const headingReactElement = React.createElement('h1', {}, 'Hello from React');
// const secondHeadingReactElement = React.createElement('h2', {}, 'Some slogan here');
// const headerReactElement = React.createElement('header', {}, headingReactElement,
secondHeadingReactElement);
// console.log(JSON.parse(JSON.stringify(headingReactElement)));
```

с този по-лесен JSX синтаксис:

```
const headerElement = (
    <header>
        <h1>Hello from React</h1>
        <h2>Slogan here</h2>
    </header>
);
```

и накрая пак рендерираме

```
rootReactElement.render(headerReactElement);
```

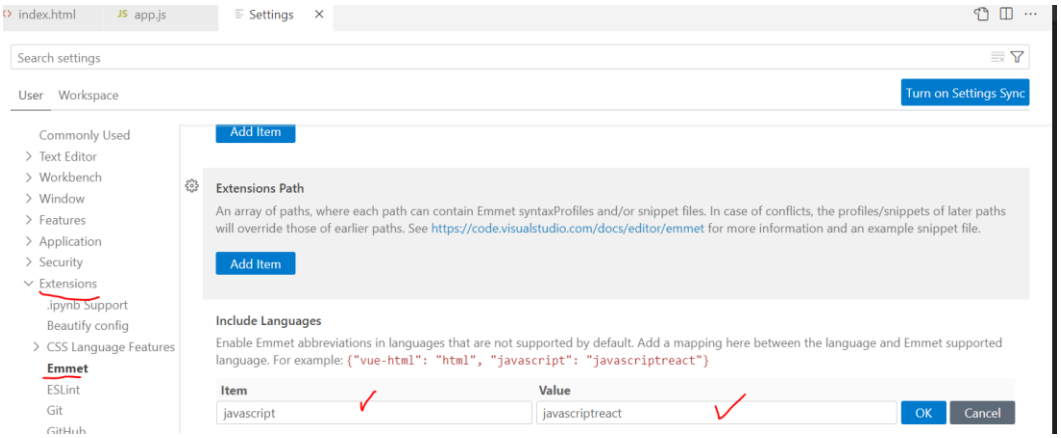
В build папката се създава същия app.js, но разписан с чист React синтаксис

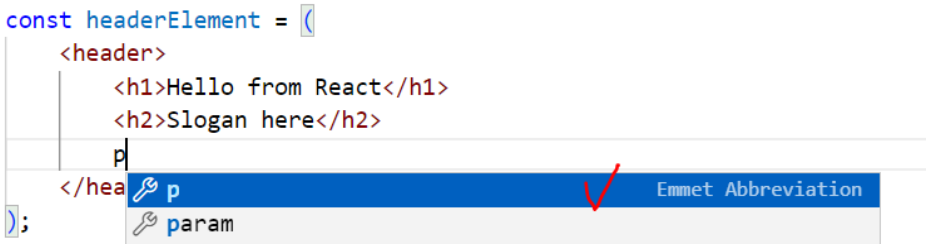
```
var headerElement = React.createElement(
    'header',
    null,
    React.createElement(
        'h1',
        null,
```

```
        'Hello from React'
    ),
    React.createElement(
        'h2',
        null,
        'Slogan here'
    )
);
```

След третата команда каквото напишем в ./src/app.js то автоматично отива и в ./build/app.js

Настройка на html emmet за ReactJS





lorem също работи

JSX Syntax

Не можем да имаме в JSX 2 елемента един до друг. Т.е. трябва да бъдат вложени в един по-горен елемент/например div.

```
const headerReactElement = (
    <header className="container">
        <h1>Hello from React</h1>
        <h2>Slogan here</h2>
        <p>Mrrrrrrrr</p>
```

```
        <button>Click</button>
    </header>
);
```

```
<div className="red">Children Text</div>
```

```
<MyCounter count={3 + 5} />  expression
```

```
let gameScores = {
    player1: 2,
    player2: 5
};
```

Custom component and passing a variable to it

```
<DashboardUnit index="2" onClick={() => {}>
    <h1>Scores</h1>
    <Scoreboard className="results" scores={gameScores} />
</DashboardUnit>
```

Коментари в JSX е по този начин (**Ctrl + /**)

```
{/* <!-- End: Footer --> */}
```

JSX Rules and Principles

- Standard elements use lowercase names
  - **div, span, form, input,** …
- Custom components always use **Pascal** case
  - **M**yCustomComponent, **G**reeting, **S**coreBoard, …
- Component name cannot be an expression
  - Use a variable instead
- There must be a **root element** - да бъдат сглобени в един главен елемент
- More info at: https://reactjs.org/docs/jsx-in-depth.html https://react.dev/

Compilation

- JSX compiles to function calls thanks to Babel

```
<div className="red">Children Text</div>
```
този ред не е HTML!!! **Думата class е запазена, затова използваме className**

Превежда се до следния JS код:

```
React.createElement("div",          div html tag
  { className: "red" },             properties object
  "Children Text" [, ...]                   list of children
);
```

1.4. Composition - Definition and Advantages

Composition of components

- React components can be nested, like DOM elements

Пример 1:

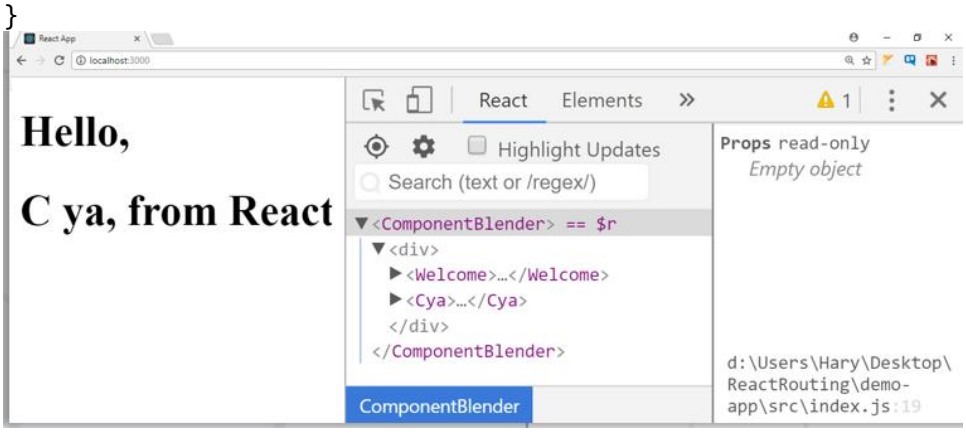
```
ReactDOM.render(<ComponentBlender />,
    document.getElementById('root'));
```

Функции, които връщат JSX

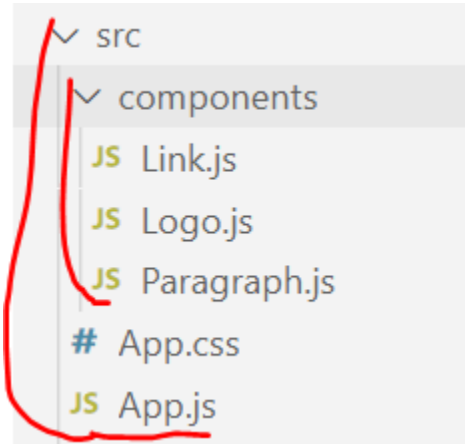
```
function Welcome() {
    return <h1>Hello, from React</h1>;
}

function Cya() {
    return <h1>C ya, from React</h1>;
}

function ComponentBlender() {
    return (
        <div>
            <Welcome />
            <Cya />
        </div>
    );
}
```



Пример 2:



App.js

```
function App() {
  return (
    <div className="App">
      <header className="App-header">
        <Logo />
        <Paragraph />
        <Link />
      </header>
    </div>
  );
}
```

Link.js

```
const Link = function () {
    return (
        <a
            className="App-link"
            href="https://reactjs.org"
            target="_blank"
            rel="noopener noreferrer"
        >
            More info
        </a>
    );
}
export default Link;
```

Logo.js

```
import logo from '../logo.svg';

export default function Logo() {
```

```
    return <img src={logo} className="App-logo" alt="logo" />
}
```

Paragraph.js

```
const Paragraph = () => {
    return <p>Some text 3</p>;
}
export default Paragraph;
```

И т.н.

```
function App() {
  return (
    <div>
      <Navigation />
      <Header/>
      <div className="container">
        <EventInfo />
        <Speakers />
      </div>

      <Tickets />
```

Component Syntax

- Names always **start with uppercase**
- Tags must be **closed**
- If there are no children - as a rule **always** use **self-closing tags** if there are no children
- Information is passed via **props**

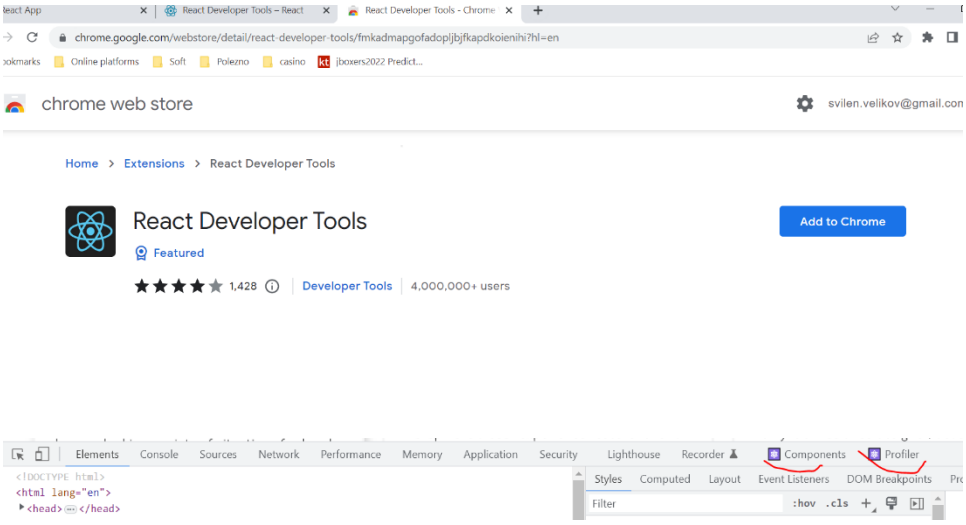
```
<Dropdown> A dropdown list <UserHead name='homeHeader' />
    <Menu>
        <MenuItem>Do Something</MenuItem>
        <MenuItem>Do Something Fun!</MenuItem>
        <MenuItem>Do Something Else</MenuItem>
    </Menu>
</Dropdown>
```

Advantages

- Encapsulate logic
- **Separate** your code
    - Easier to **maintain** and **debug**
    - Allows **reusability**
- Components are neat/спретнати/чисти = достатъчно малки също

1.5. ReactDev Tools in the browser

https://reactjs.org/link/react-devtools



44. Other

https://github.com/softuni-practice-server/softuni-practice-server

commonJS модулна система (например module.exports) - да не я използваме в ReactJS
ES6 module system (например export default) - само нея да използваме в ReactJS

какво е named export