React and React Components

Hello!

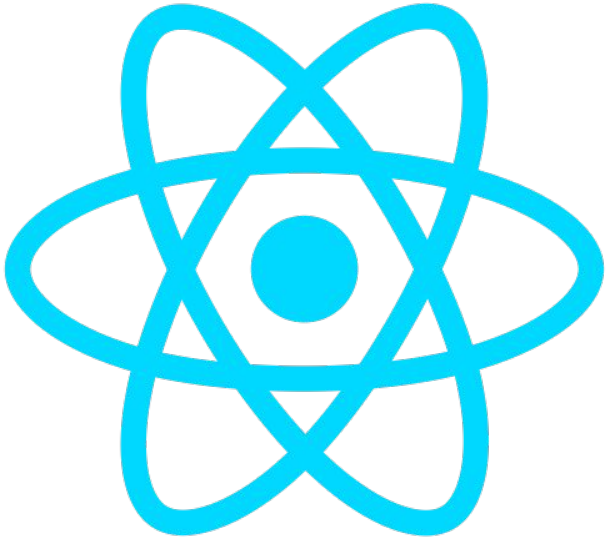
I am Mateusz Choma

I am a scientific mind, passionate of technology, an engineer "squared" - a graduate of two universities in Lublin :)
As well, I am a JS developer, entrepreneur and owner of small software house - Amazing Design.

1.
React

React
About

- React was developed by Facebook
- first version was shown in 2013
- there were some license-related issues with patents restriction in case of lawsuit
- license-related issues ended in 2017 with React 16.x.x when Facebook publish nwest and previous version under MIT license



React
About

React is a JavaScript library for building user interfaces.

- it provides us a component based approach for building applications
- it is responsible for rendering the app in the DOM
- React is implemented in such a way to make the rendering of out UI as fast as possible by using Virtual Dom

More info - https://reactjs.org

React
Virtual DOM (VDOM)

Virtual DOM is in memory DOM-like tree that React uses for comparing and searching for changes.

React firstly make diffs between previous and new VDOM trees, and then find the best way to render this changes to the real DOM.

“React implements a browser-independent DOM system for performance and cross-browser compatibility.” - *quote from* *React Docs*

P.S. This is not a React idea - but React popularize it.

React
Virtual DOM (VDOM)

“The Virtual DOM is an abstraction of the HTML DOM. It is lightweight and detached from the browser-specific implementation details. Since the DOM itself was already an abstraction, the virtual DOM is, in fact, an abstraction of an abstraction.”

- quote from an article on http://reactkungfu.com/

React
JSX

JSX is a name for syntax that React uses for describing HTML structure.
JSX seems like HTML but it compiles to pure JavaScript!
As you see it depends on React object because it calls it's createElement method!

```
const App = () => (
  <div>
    <h1>Hello</h1>
  </div>
)
```

```
const App = () => (
  React.createElement(
    'div',
    {},
    React.createElement(
      'h1',
      {},
      'Hello'
    )
  )
)
```

React
React and React DOM

React and ReactDOM are objects provided by the library that contain some methods we will use to build React App.

In every file when you works with JSX or component that is a class extending React.Component you must

```
import React from 'react'
```

ReactDOM is used only to render certain React component/s into a container in real DOM.

```
ReactDOM.render(<h1>Hello, world!</h1>, document.getElementById('root'))
```

React
Rendering in react

Rendering is an operation when React builds new VDOM structure by calling components or components render methods.

Each call returns small fragment of new VDOM. React makes all VDOM tree, compare with previous and make changes on real DOM.

React
Component

React components let you split the UI into independent, reusable pieces.

React components can be:

- a **function**
- a **ES6 class** defined by extending React.Component

React
Component - props and state

Component can get **props** when it is called. Every component have **props**! Props are similar to HTML attributes when component is called in JSX.

Components created, by class can also have **state**.

Components with state are called *“smart components”*.

Components without state are called *“dumb components”*.

React
Function component

```
// “normal” function
function App(props) {
  return <div><h1>{props.title}</h1></div>
}

// arrow function - preferred way !
const App = props => <div><h1>{props.title}</h1></div>

// component call
const result = <App title="Hello world" />
```

React
Function component

In function component props are passed as an argument.

props is an object that have property named as the attribute

Calling object like this:

```
<App title="Hello world" />
```

will pass to component

```
props === {title: "Hello world"}
```

React
Class component

Class component **must extends React.Component** class and **must have at least render method!**
Props are available in props property of the class, so we can access in through `this.props .`

```
class App extends React.Component {
    render() {
      return <div><h1>{this.props.title}</h1></div>
    }
}
```

React
React.createClass - DEPRECATED !!!

Before ES6 React provides **createClass** method, and it was used to create components using class. It is **DEPRECATED NOW**, but sometimes you can find it in old libraries.

```
const OldComponent = React.createClass({
    render() {
        return <h1>My old component</h1>
    }
})
```

React
JSX - embedding expression

We can embed **ANY** JavaScript expression in JSX using { } brackets!

```
const name = 'Johny Bravo'

const Comp1 = () => <h1>Hello, {name}!</h1>

const Comp2 = () => <p>2+2 = <strong>{2+2}</strong></p>
```

React
JSX - embedding expression

If we want to use conditions inside JSX we can use ternary operator - because it is an expression not a statement and it returns value!

```
const isVisible = true
const App = () => (
    <div>
        isVisible ?
            <h1>Hello!</h1>
            :
            null
    </div>
)
```

React
JSX - additional rules

When we want to render nothing we should use **null!**

All components should contains only one element without siblings!

This element can contains more elements but component should returns only one parent - element!

React
JSX - camelcase naming conventions

All props in React components should be written in camelCase. Only aria-* and data-* can be written with dash - HTML-like.

JSX supports all of HTML attributes.

There are some props that behaves differently than in HTML:

- className - to add CSS class (class is a keyword in JS)
- onChange - works as expected unlike in browsers DOM
- style - accepts a JS object with camelCase properties rather than CSS string

More -> https://reactjs.org/docs/dom-elements.html