Netcracker

# Front-end development course

## JavaScript

Lecture #8

# Old school interfaces

jQuery and AJAX brought what is called WEB 2.0

3 big milestones in web development:

- XMLHttpRequest (originally introduced in 2000)
- jQuery (2007)
- React.js and especially Virtual DOM concept (2013)

# React

ReactJS - is a relatively small JavaScript library, it's main idea is to build element (or components) tree, manage and update it when state and properties changes. That tree compiles to React's inner representation and other libraries (like react-dom, react-pdf , redocx, etc) can be used to render that representation to actual view.

React initially was introduced in 2013 by Facebook (original video from conference).

Funny conference talk on building UI in general.

# JSX

JSX - is a syntax extension to JavaScript, it's looks like XML or may remind you a template language,  but with more advanced features. It's recommended to use it with React to describe what the UI should look like.

```
const element = <h1>Hello, world!</h1>;
```

JSX is invented to ***separates concerns*** with loosely coupled units called "components" that contain both markup and logic.

You can embed any expression in JSX using curly braces and also JSX is itself an expression, it compiles just to function call so i can be used whenever expression is expected.

You can play with JSX syntax online here.

# JSX

Both JSX elements and elements created with *React.createElement* represents DOM node as objects, and essentially are plain old JS objects like those:

```
const element = {
  type: 'h1',
  props: {
    className: 'greeting',
    children: 'Hello, world'
  }
};
```

From that time and later we will call it *React elements*.

React elements are immutable and represents snapshot of the UI at a certain point in time.

# React components

React components are main building blocks of your application. Components let you split the UI into independent, reusable pieces, and think about each piece in isolation. Components accepts inputs called ***"props"*** and returns React elements.

Components can be implemented in functional and class styles.

All attributes from React components is being parsed as a single object and passes as ***props*** to components.

Component ***props*** are read-only and you must never modify your inputs. Try to always use "pure" functions, their behaviour is more predictable and less buggy.

React has a rule "all React components must act like pure functions with respect to their props".

We shouldn't be afraid to divide UI into small components, take a look at Reddit and try to visually divide home page to small set of components.

# Components state

One way to change view in React app is just invoke ReactDOM.render() every time some data changes.

Another (and most correct) way is to use ***state*** and update it using ***setState*** method.

State also makes component really reusable and encapsulated.

Three rules for working with the state:

- **Do not modify state directly**, instead use ***setState()***
- **State updates may be asynchronous.** React may multiple ***setState()*** calls into single update, so do not rely on current **state** and **props** (good example)
- **State updates are merged.** React merges the object you provide into the current state (but merge is shallow)

# Q&A

# Thank You