A short Intro to React

How to get started?

- Download & install node.js at nodejs.org
- Open window of terminal and create an application with the command: **npx create-react-app *appname***
- In the terminal: **cd *appname* and open code editor**
- Run the server by typing in the terminal: **npm start**

JSX

- JSX - stands for JavaScript Syntax Extension
- In simple words- it enables writing HTML tags in JavaScript and placing them in the Document Object Model (DOM) without the use of createElement (see example below)
- Babel has the capability to translate JSX into JavaScript
- Example from babel.io :

```jsx
<>
<div className='text-muted'>Hello World</div>
<Button className="primary">Click here</Button>
</>
```

JSX

```js
React.createElement(React.Fragment, null,
React.createElement("div", {
  className: "text-muted"
}, "Hello World"), React.createElement(Button, {
  className: "primary"
}, "Click here"));
```

JS

JSX part 2

- It’s not required for React, but it makes coding in React easier & nicer
- It looks similar to html but it actually is JS

```
const hello = <h1>Hello world</h1>
```

- Requires one parent element

```
function App() {
  return (
    <div className="App">
      <h1>Hello world</h1>
      <button>Click here</button>
    </div>
  );
}
export default App;
```

OK

```
function App() {
  return (
      <h1>Hello world</h1>
      <button>Click here</button>
  );
}
export default App;
```

Not OK

Virtual DOM

- First of all what is DOM? DOM stands for “Document Object Model” and in simple words is a tree data structure of elements which is created when a certain page is loaded

- Similar to actual DOM - Virtual DOM is a tree data structure that lists React elements

- When taking some action in an application (i.e. adding elements), instead of direct manipulation of the DOM, React uses a different approach - it creates virtual DOM, which takes care of this manipulation, before making changes in the actual DOM.

- Thanks to virtual DOM in the actual DOM we update only the elements that changed

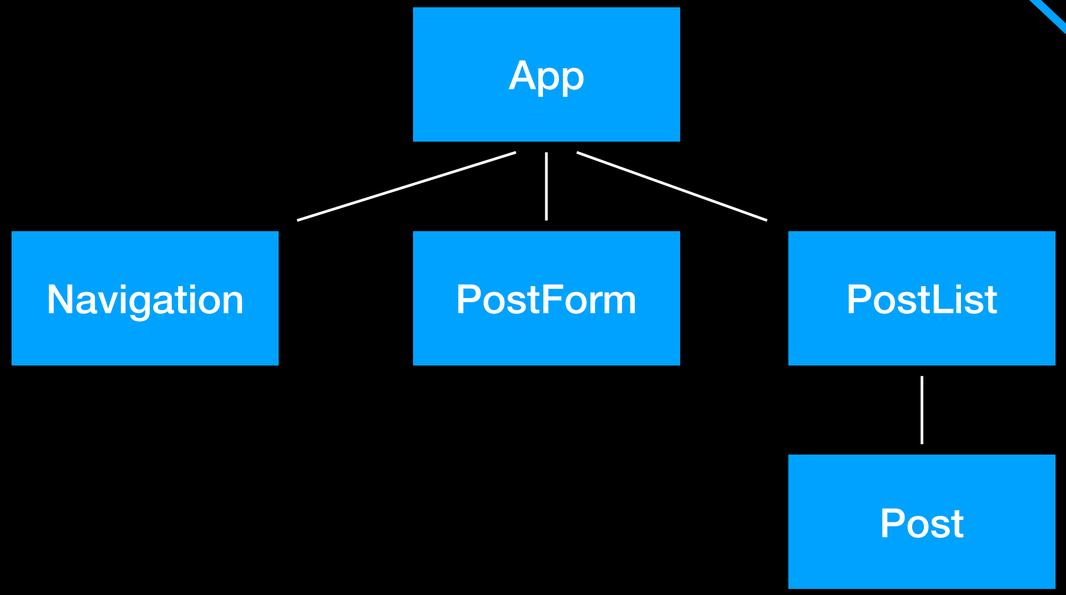
- Virtual DOM approach is faster, more efficient and interactive

Components

- Main element of the user interface (UI) is a component
- Each component has it’s own structure and logic
- We can distinguish two types of components:
  - Class components
  - Functional components

Components part 2

We can create components inside components - our application will consists of many components, but the main component (in our case App) will be responsible for the user interface with the use of ReactDOM.render()




```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

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

// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
```

Props vs State

- Props: is a shorthand for properties and enables passing data between components. It’s immutable

- State: appears in stateful components. It stores properties data of a Class Component, and allows modifications via setState method

Class components

```jsx
import React, { Component } from 'react';

class ClassComponent extends Component {
    state = {
        status: "Hello, I'm a Class Component"
    }
    render() {
        return (
            <React.Fragment>
                {this.state.status}
            </React.Fragment>
        );
    }
}

export default ClassComponent;
```

Stateful component with access to Lifecycle methods (explained later)

Functional components

```
import React from 'react';

const FunctionalComponent = () => {
    return (
        <h1>Hello, I am a Functional Component</h1>
    );
}

export default FunctionalComponent;
```

This type of component doesn’t have it’s state. It’s basically a javascript function which can take in props as an argument and returns an React element

Popular events

To perform certain actions in our application we can use events

- onClick
- onChange
- onFocus
- onMouseMove
- onSelect
- onScroll
- onKeyDown

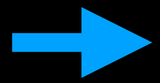
```
import React, { Component } from 'react';

class ClassComponent extends Component {
    state = {
        status: "Hello, I'm a Class Component"
    }

    handleStateChange = () => {
        const new_status = "Bye bye from CC"
        this.setState({ status: new_status })
    }

    render() {
        return (
            <React.Fragment>
                {this.state.status}
                <br />
                <button onClick={this.handleStateChange}>Click me</button>
            </React.Fragment>
        );
    }
}

export default ClassComponent;
```

Popular Lifecycle methods

- Class component has access to lifecycle methods

STAGES: Mounting | Updating | Unmounting

- **constructor** - method called automatically while creation object from a Class and is called before component is mounted



```
constructor(props) {
    super(props);
    this.state = {
        status: "Hello, I'm a Class Component"
    };
}
```

- **render** - the only required method that returns the content of the component
- **componentDidMount** - perfect method to get the data, takes place after rendering (the DOM exists)
- **componentDidUpdate** - called after a certain component update in the DOM
- **componentWillUnmount** - called when a certain component will be removed from the DOM.

Promises & fetching data

- With the use of fetch method and componentDidMount lifecycle method we can simply get the data from our api

- In react fetch method creates a promise which can trigger either “then” method if the promise is completed or “catch” when it’s not

```
fetch(`http://apiexample.com/crypto`)
.then(response => response.json())
.then(res=> console.log(res))
.catch(err => console.log(err))
```

- For fetching data we will be using a 3rd party JS library called axios that converts the data automatically into JSON (we will use the “then” method only once) and is easier to work with

Thank you