React

routing, UI components & iterating

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.
Routing

Routing
What is it?

Routing is displaying various content depends on route that users is visiting.

Route depends on path.

Path is a part of URL address that lies behind /, and starts from /, so / is the main route in an app.

Routing
react-router

```
import {BrowserRouter, Route} from 'react-router-dom'
<BrowserRouter>
  <div>
    <Route exact path="/" component={Home} />
    <Route path="/about" component={AboutMe} />
    <Route path="/contact" component={Contact} />
    <Route path="/projects" component={Projects} />
  </div>
</BrowserRouter>
```

Routing
react-router

react-router have BrowserRouter component that can displaying only that routes that math current path.

To Route components we can pass components that we want to display when path is matched through component prop or provide render method through render prop.

More ->
https://reacttraining.com/react-router/web/guides/quick-start

Routing
react-router exact path

Path can be matched in multiple Routes at once. For example path in browser ‘/dashboard’ matches

```
<Route path=”/dashboard”>
```
and
```
<Route path=”/”>
```

because /dashboard contains /.

When we want to display route on exactly that path we must add prop **exact**.

Routing
Routes with parameters

Sometimes we want to display the same component but with different data inside (e.g. product view - data depends on uid passed inpath)

```
<Route path=”/product/:uid” component={Product}>
```

In this case uid passed in path will be accessible in Product component under:

props.match.prams.uid

We can name params as we want and place multiple params in route!

Routing

Routes with parameters - withRouter

When we want to use params from path in a component nested in component that is passed as component prop in Route component we can use withRouter higher order component, that will find params form the closest Route to our component and pass the props.match to our component.

Code looks like wrapping component in function before exporting:

```
import {withRouter} from 'react-router'

const NestedInRoute = () => ({ ... })

export default withRouter(NestedInRoute)
```

2.
UI components

UI components
What is it?

UI components are components that don’t provide any logic to our app - there only provide nice styled UI elements.

We can off course build our own UI components, that should be reusable and unified.

There are multiple libraries that have already builded beautiful and functional components.

We recommend you Material UI - http://material-ui.com

3.
Iterating & rendering tips

Iterating & rendering tips

What is iterating in React?

In React JSX we can use whole arrays (pure JS ones) to render content.

That arrays can contain everything that can be rendered so they can contain:

- numbers
- strings
- components
- nulls, undefineds and booleans (not rendered)

They can’t contain objects because objects are not valid React childs

More -> https://reactjs.org/docs/lists-and-keys.html

Iterating & rendering tips

Power of iterating

```
const uniqueNumbers = [1, 2, 3]

const App = () => (
  <ul>
    { uniqueNumbers }
  </ul>
)

const App = () => (
  <ul>
    { uniqueNumbers.map(number => <li key={number}>{number}</li>}
  </ul>
)
```

Iterating & rendering tips

Key property when iterating

Keys help React identify which items have changed, are added, or are removed. Keys should be given to the elements inside the array to give the elements a stable identity.

The best way to pick a key is to use a string **that uniquely identifies a list item among its siblings**. Most often you would use IDs from your data as keys.

Index from array are not best option (but better than no key ;)) !

Iterating & rendering tips

Array function when iterating

Array functions that returns new array are incredible good in helping to render what we precisely wants to render!

Using **.map** we can transform array with data into array of React components that wraps that data.

Using **.filter** we can get rid of that parts of data that doesn't matches out criteria (e.g. stored in component state), and then make components form results using **.map**.

Iterating & rendering tips

Preparing elements in variables and render them

```
const numbers = [1, 2, 3, 4, 5]

class App extends Component{
    render(){
        const listItems = numbers.map((number) =>
          <li>{number}</li>
        )

        return (
            <div>{listItems}</div>
        )
    }
}
```

Iterating & rendering tips

Conditional rendering

We can use any expression in JSX so ternary operator is the best option for conditional rendering!

```
render() {
  const isLoggedIn = this.state.isLoggedIn

  return (
    <div>
      The user is
      <b>
        {isLoggedIn ? 'currently' : 'not'}
      </b> logged in.
    </div>
  );
}
```

Iterating & rendering tips
props.children

Every component have special prop called children. This is an array of components that was passed to component between its opening and cloesing tag.

With this property we can decide what to do with children! For example we can display them or not!

```
const ConditionalDiv = () => (
    <div>
        {props.display ? props.children : null}
    </div>
)
```