React: Base Features and Syntax

1. Using create react app

https://github.com/facebook/create-react-app

Installation:

If you've previously installed `create-react-app` globally via `npm install -g create-react-app`, we recommend you uninstall the package using `npm uninstall -g create-react-app`

1. `$npx create-react-app my-first-app`
2. `$cd my-first-app`
3. $npm start

You can also install SASS:

1. $npm install node-sass –save

https://scotch.io/starters/react/adding-sass-to-create-react-app-applications

2. Understanding the Folder Structure

a. package.json – You can find all dependencies and scripts
b. node_modules – holds all the dependencies and sub-dependencies of our project
c. public – the public folder is more interesting. It holds html file. We will never add more html pages in this project.

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

d. manifest.json – file created by create-react-app
e. src – it holds our components. It has index.js file which gets access to this root element in our DOM, in our html file.
   React.StrictMode - https://ru.react.js.org/docs/strict-mode.html. **StrictMode** is a tool for highlighting potential problems in an application

App.js

Remove logo. Delete **logo.svg** file.

```
import './App.css';


function App() {
  return (
    <div className="App">
      <h1>Hi, I'm a React App!</h1>
    </div>
  );
}
```

```

export default App;
```

App.css

```
.App {
  text-align: center;
}
```

3. Understanding Component Basics

Reference: **React 17 : Why it's so important?** https://dev.to/gouthamjm/react-17-why-it-s-so-important-il6

Refactor your code to class based components:

App.js

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {
  render(){
    return (
      <div className="App">
        <h1>Hi, I'm a React App!!</h1>
      </div>
    );
  }
}

export default App;
```

4. Understanding JSX

App.js

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {
  render(){
    // return (
    //   <div className="App">
  //     <h1>Hi, I'm a React App!!</h1>
  //   </div>
  // );
  return React.createElement('div', {className: 'App'}, React.createElement('h1', null, 'Hi, I\'m a React App!!'))
 }
}

export default App;
```

null is configuration.

The code we just wrote here is exact equivalent to our first return statement. It is JSX, that's way we import react. And it is look like html but it isn't.

5. Creating a Functional Components

In **src** folder create a **Person** folder. In the **Person** folder create **Person.js** file:

Person.js

```
const Person = () => {
  return <h2>Hi, I am a Person!</h2>
}

export default Person;
```

class-based solution

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

class Person extends Component {
 render(){
  return <h2>Hi, I'm a person!</h2>
 }
}

export default Person;
```

App.js

```
………
import Person from './Person/Person'

class App extends Component {
 render(){
  return (
   .........
    <Person />
   </div>
  );

 }
}

export default App;
```

6. Outputting Dynamic Content

App.js

```
class App extends Component {
 render(){
  return (
   <div className="App">
    <h1>Hi, I'm a React App!!</h1>
    <p>This is really working!</p>
    <Person />
    <Person />
    <Person />
   </div>
  );

 }
}
```

Person.js

```
const Person = () => {
return <h2>Hi, I am a Person and I am X {Math.floor(Math.random() * 30)} years old!</h2>
}

export default Person;
```

We can call function in JSX.

7. Working with Props.

App.js

```
<Person name="Dilshod" age="38"/>
<Person name="Gerhard" age="40"/>
<Person name="Asadbek" age="7"> My hobby is playing video games </Person>
```

Person.js

```
return <h2>Hi, I am a {props.name} and I am {props.age} years old!</h2>
```

Where is: My hobby is playing video games?

8. Understanding the “children” Prop.

Person.js

```
return (
    <div>
        <h2>Hi, I am a {props.name} and I am {props.age} years old!</h2>
        <h2>{props.children}</h2>
    </div>
)
```

9. Understanding and using State

App.js

```
class App extends Component {
  state = {
    persons: [
      {name: "Dilshod", age: 28},
      {name: "Gerhard", age: 40},
      {name: "Asadbek", age: 7}
    ]
  }


  render(){
    return (
      <div className="App">
        <h1>Hi, I'm a React App!!</h1>
        <p>This is really working!</p>
        <button>Swich Name</button>
        <Person name={this.state.persons[0].name} age={this.state.persons[0].age}/>
        <Person name={this.state.persons[1].name} age={this.state.persons[1].age}/>
        <Person name={this.state.persons[2].name} age={this.state.persons[2].age}>My hobby is playing video
games</Person>
      </div>
    );

  }
}
```

10. Handling Events with Methods and Manipulating the State

Resources: https://reactjs.org/docs/events.html#supported-events

Create a button to switch names. For this we need to create a method/function of the class. Use setState() method in order to update state.

App.js

```
class App extends Component {
  state = {
    persons: [
      {name: "Dilshod", age: 38},
      {name: "Gerhard", age: 40},
      {name: "Asadbek", age: 7}
    ],
    otherState: 'some other value' //it will not touch
  }

  switchNameHandler = () => {
    console.log('Was clicked!');
    this.setState({
      persons: [
        {name: "Dilshod Rahmatov", age: 38},
        {name: "Gerhard", age: 39},
        {name: "Asadbek", age: 7}
      ]
    })
  }

  render(){
    return (
      <div className="App">
        <h1>Hi, I'm a React App!!</h1>
        <p>This is really working!</p>
        <button onClick={this.switchNameHandler}>Swich Name</button>
        <Person name={this.state.persons[0].name} age={this.state.persons[0].age}/>
        <Person name={this.state.persons[1].name} age={this.state.persons[1].age}/>
        <Person name={this.state.persons[2].name}   age = {this.state.persons[2].age}> My hobby is playing video
games</Person>
      </div>
    );

  }
}
```

11. Passing Method References Between Components

1.
App.js

```
<Person
          name={this.state.persons[1].name}
          age={this.state.persons[1].age}
          click={this.switchNameHandler}
          />
```

Person.js

```
<h2 onClick={props.click}>Hi, I am a {props.name} and I am {props.age} old!</h2>
```

2. Bind()

App.js

```
switchNameHandler = (newName) => {
    console.log('Was clicked!');
    this.setState({
      persons: [
        {name: newName, age: 38},
        {name: "Gerhard", age: 39},
        {name: "Asadbek", age: 7}
      ]
    })
  }
```

```
<button onClick={this.switchNameHandler.bind(this, "Dilshod!!")}>Swich Name</button>
    <Person
     name={this.state.persons[0].name}
     age={this.state.persons[0].age}/>
    <Person
     name={this.state.persons[1].name}
     age={this.state.persons[1].age}
     click={this.switchNameHandler.bind(this, "Bob!")}
     />
```

12. Adding two way binding

Add new class method: nameChangedhandler()

```
<input type="text" onChange={props.changed} value={props.name} />
 nameChangedHandler = (event) => {
   this.setState({
     persons: [
       {name: "Dilshod", age: 38},
       {name: event.target.value, age: 40},
       {name: "Asadbek", age: 7}
     ]
   })
 }
```

```
<Person
     name={this.state.persons[0].name}
     age={this.state.persons[0].age}/>
    <Person
     name={this.state.persons[1].name}
     age={this.state.persons[1].age}
     click={this.switchNameHandler.bind(this, "Bob!")}
     changed={this.nameChangedHandler}
     />

    <Person
     name={this.state.persons[2].name}
     age={this.state.persons[2].age}>My hobby is playing video games</Person>
```

Person.js

```
<input type="text" onChange={props.changed} value={props.name} />
```

We cannot change other input values, because of we cannot update them.

13. Adding Styling with Stylesheets

Create a Person.css file.

Person.css
```
.Person {
   width: 60%;
   margin: 16px auto;
   border: 1px solid #eee;
   box-shadow: 0 2px 3px #ccc;
   padding: 16px;
   text-align: center;
}
```

Person.js
```
import './Person.css';
```

```
<div className='Person'>
```

14. Working with Inline Styling

App.js
```
render(){
   const style = {
     backgroundColor: 'white',
     font: 'inherit',
     border: '1px solid blue',
     padding: '8px',
     cursor:"pointer"
   }


   return (
     <div className="App">
       <h1>Hi, I'm a React App!!</h1>
```

```
    <p>This is really working!</p>
    <button>
     style={style}
     onClick={this.switchNameHandler.bind(this, "Dilshod!!")}>Swich Name</button>
```

Assignment:

1. Create TWO new components: UserInput and UserOutput
2. UserInput should hold an input element, UserOutput two paragraphs
3. Output multiple UserOutput components in the App component (any paragraph texts of your choice)
4. Pass a username (of your choice) to UserOutput via props and display it there
5. Add state to the App component (=> the username) and pass the username to the UserOutput component
6. Add a method to manipulate the state (=> an event-handler method)
7. Pass the event-handler method reference to the UserInput component and bind it to the input-change event
8. Ensure that the new input entered by the user overwrites the old username passed to UserOutput
9. Add two-way-binding to your input (in UserInput) to also display the starting username
10. Add styling of your choice to your components/ elements in the components - both with inline styles and stylesheets

Finished files.

App.js

```
/* eslint-disable react/require-render-return */
import React, {Component} from 'react';
import './App.css';
import Person from './Person/Person'

class App extends Component {
 state = {
  persons: [
   {name: "Dilshod", age: 38},
   {name: "Gerhard", age: 40},
   {name: "Asadbek", age: 7}
  ],
  otherState: 'some other value' //it will not touch
 }

 switchNameHandler = (newName) => {
  console.log('Was clicked!');
  this.setState({
   persons: [
    {name: newName, age: 38},
    {name: "Gerhard", age: 39},
    {name: "Asadbek", age: 7}
   ]
  })
 }

 nameChangedHandler = (event) => {
  this.setState({
   persons: [
    {name: "Dilshod", age: 38},
    {name: event.target.value, age: 40},
    {name: "Asadbek", age: 7}
   ]
  })
 }

 render(){
  const style = {
   backgroundColor: 'white',
      font: 'inherit',
      border: '1px solid blue',
      padding: '8px',
      cursor:"pointer"
    }


    return (
      <div className="App">
        <h1>Hi, I'm a React App!!</h1>
        <p>This is really working!</p>
        <button
          style={style}
          onClick={this.switchNameHandler.bind(this, "Dilshod!!")}>Swich Name</button>
        <Person
          name={this.state.persons[0].name}
          age={this.state.persons[0].age}/>
        <Person
          name={this.state.persons[1].name}
          age={this.state.persons[1].age}
          click={this.switchNameHandler.bind(this, "Bob!")}
          changed={this.nameChangedHandler}
          />

        <Person
          name={this.state.persons[2].name}
          age={this.state.persons[2].age}>My hobby is playing video games</Person>
      </div>
    );

  }
}

export default App;
```

Person.js

```
import './Person.css';

const Person = (props) => {
return (
    <div className='Person'>
        <h2 onClick={props.click}>Hi, I am a {props.name} and I am {props.age} years old!</h2>
        <h2>{props.children}</h2>
        <input type="text" onChange={props.changed} value={props.name} />
    </div>
)
}

export default Person;
```

Person.css

```
.Person {
    width: 60%;
    margin: 16px auto;
    border: 1px solid #eee;
    box-shadow: 0 2px 3px #ccc;
    padding: 16px;
    text-align: center;
}
```