Școala
informală
de IT

Components

First component – install dependencies

Gitbash integrated terminal

```
npx create-react-app counter-app
cd counter-app/
npm run start
npm install bootstrap
```

index.js

```
import 'bootstrap/dist/css/bootstrap.css';
```

First component – create component file

Gitbash integrated terminal

```
cd src/
mkdir components
cd components/
touch Counter.jsx
```

First component – create initial code

src/components/counter.jsx -> use rcc shortcut (react class component)

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

export default class Counter extends Component {
  render() {
    return (
      <h1>Counter Component</h1>
    )
  }
}
```

First component – use newly created component

src/App.js

```
import './App.css';
import Counter from './components/Counter';

function App() {
  return (
    <div className="App">
      <Counter />
    </div>
  );
}

export default App;
```

First component – embed multiple elements

src/components/counter.jsx -> React component must be wrapped in a container -> div or React.Fragment component

```
import React, { Component } from 'react';
export default class Counter extends Component {
   render() {
      return (
         <div>
            <h1>Counter App</h1>
            <button>Increment</button>
            <button>Decrement</button>
         </div>
      );
   }
}
```

First component – create state

src/components/counter.jsx -> the state object should contain one or more properties

```
state = { count: 0 };

render() {
   return (
      <div>
         <button>Increment</button>
         <span>{this.state.count}</span>
         <button>Decrement</button>
      </div>
   );
}
```

First component – JSX, class method, desctructuring

src/components/counter.jsx

```
<span>{this.formatCount()}</span>

formatCount() {
    const { count } = this.state;
    return count === 0 ? 'Zero' : count;
}
```

First component – setting attributes

src/components/counter.jsx

```
state = { count: 0,
          imageUrl: 'https://picsum.photos/200'
};

<img src={this.state.imageUrl} />
```

First component – using bootstrap classes

src/components/counter.jsx

```
<button className="btn btn-dark btn-sm">
      +
</button>
<span className="badge bg-info m-2">
   {this.formatCount()}
</span>
<button className="btn btn-dark btn-sm">
     -
</button>
```

First component – applying custom styles

```
styles = {
    fontSize: 20,
};
```

```
<span style={this.styles} className="badge bg-info m-2">
    {this.formatCount()}
</span>
```

```
<span style={{ fontSize: 18 }} className="badge bg-info m-2">
    {this.formatCount()}
</span>
```

First component – conditional classes

```
getCounterClasses() {
    let classes = 'badge m-2 bg-';
    const { count } = this.state;
    classes += count === 0 ? 'warning' : 'info';
    return classes;
}
```

```
<span className={this.getCounterClasses()}>{this.formatCount()}</span>
```

First component – Handling events

- all react elements have properties based on standard DOM events but in camel case format → onClick, onKeyDown, onKeyPress etc.

```
handleIncrement() {
    // this - undefined
    console.log(this.state.count);
}

render() {
    return (
        <div>
            <button className="btn btn-dark btn-sm">-</button>
            <span className={this.getCounterClasses()}>{this.formatCount()}</span>
            <button onClick={this.handleIncrement} className="btn btn-dark btn-sm">
                +
            </button>
        </div>
    );
}
```

First component – Binding events handlers

- all react elements have properties based on standard DOM events but in camel case format → onClick, onKeyDown, onKeyPress etc.

```
constructor() {
    super();
    this.handleIncrement = this.handleIncrement.bind(this);
}

handleIncrement() {
    // this - undefined
    console.log(this.state.count);
}
```

```
return (
    <div>
        <button className="btn btn-dark btn-sm">-</button>
        <span className={this.getCounterClasses()}>{this.formatCount()}</span>
        <button
            onClick={this.handleIncrement.bind(this)}
            className="btn btn-dark btn-sm"
        >
            +
        </button>
    </div>
```

```
handleIncrement = () => {
    console.log(this.state.count);
};

render() {
    return (
        <div>
            <button className="btn btn-dark btn-sm">-</button>
            <span className={this.getCounterClasses()}>{this.formatCount()}</span>
            <button onClick={this.handleIncrement} className="btn btn-dark btn-sm">
                +
            </button>
        </div>
    );
}
```

```
render() {
    return (
        <div>
            <button className="btn btn-dark btn-sm">-</button>
            <span className={this.getCounterClasses()}>{this.formatCount()}</span>
            <button
                onClick={() => this.handleIncrement()}
                className="btn btn-dark btn-sm"
            >
                +
            </button>
        </div>
    );
}
```

First component – Updating the state

- the state object is imutable, so it cannot be modified directly, in react class components, the state can be modified using setState() method

```
handleIncrement = () => {
    this.setState({ count: this.state.count + 1 });
};

handleDecrement = () => {
    this.setState({ count: this.state.count - 1 });
};

render() {
    return (
        <div>
            <button onClick={this.handleDecrement} className="btn btn-dark btn-sm">
                -
            </button>
            <span className={this.getCounterClasses()}>{this.formatCount()}</span>
            <button onClick={this.handleIncrement} className="btn btn-dark btn-sm">
                +
            </button>
        </div>
    );
}
```