Hello, World!

High Order Components / Render Props

Kaj Białas

LEOCODE — SOFTWARE ENGINEER

infoShare ACADEMY — FRONT-END TRAINER

infoShare ACADEMY

Wzorce projektowe

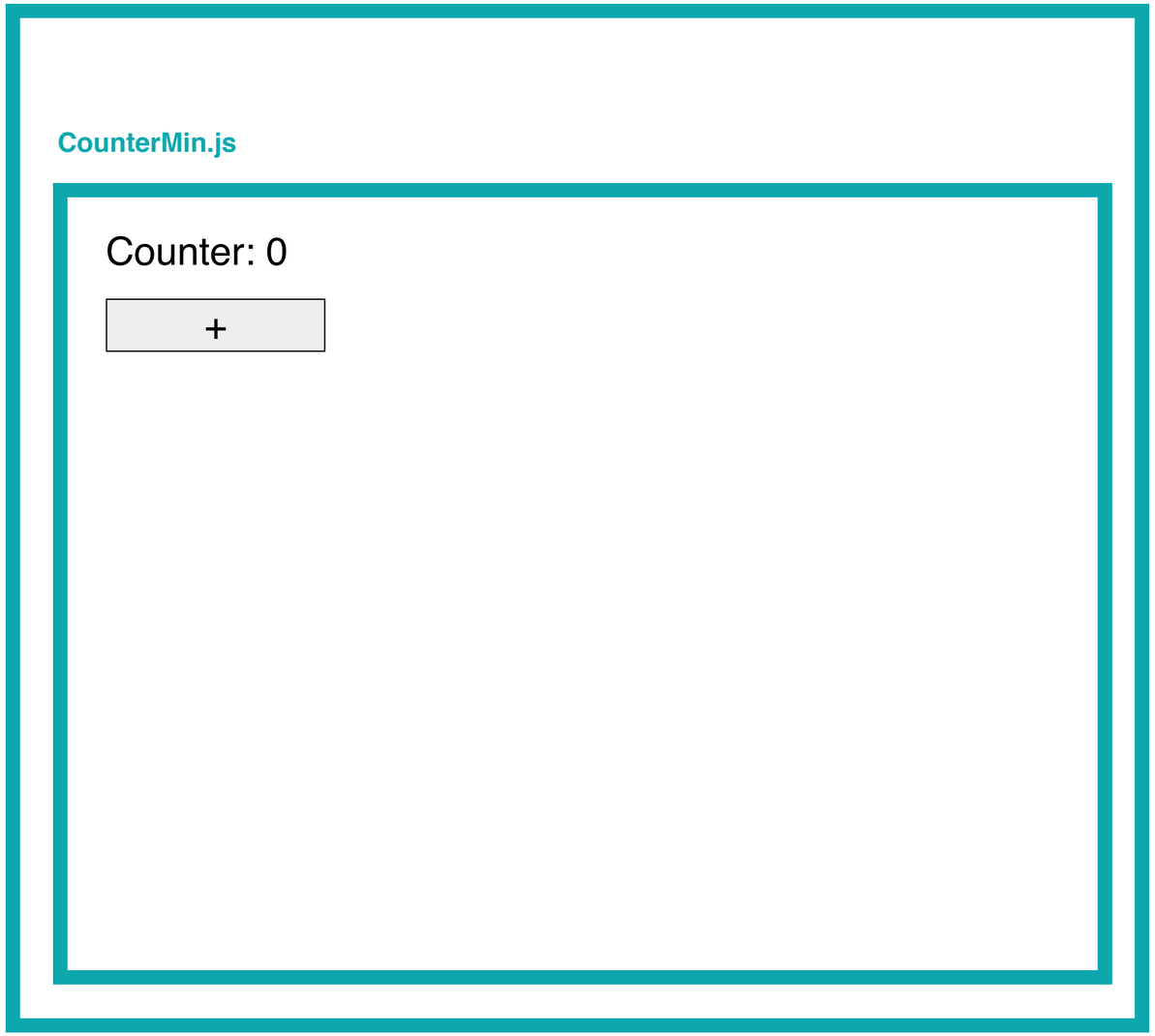
High Order Components (HOC)

High Order Components

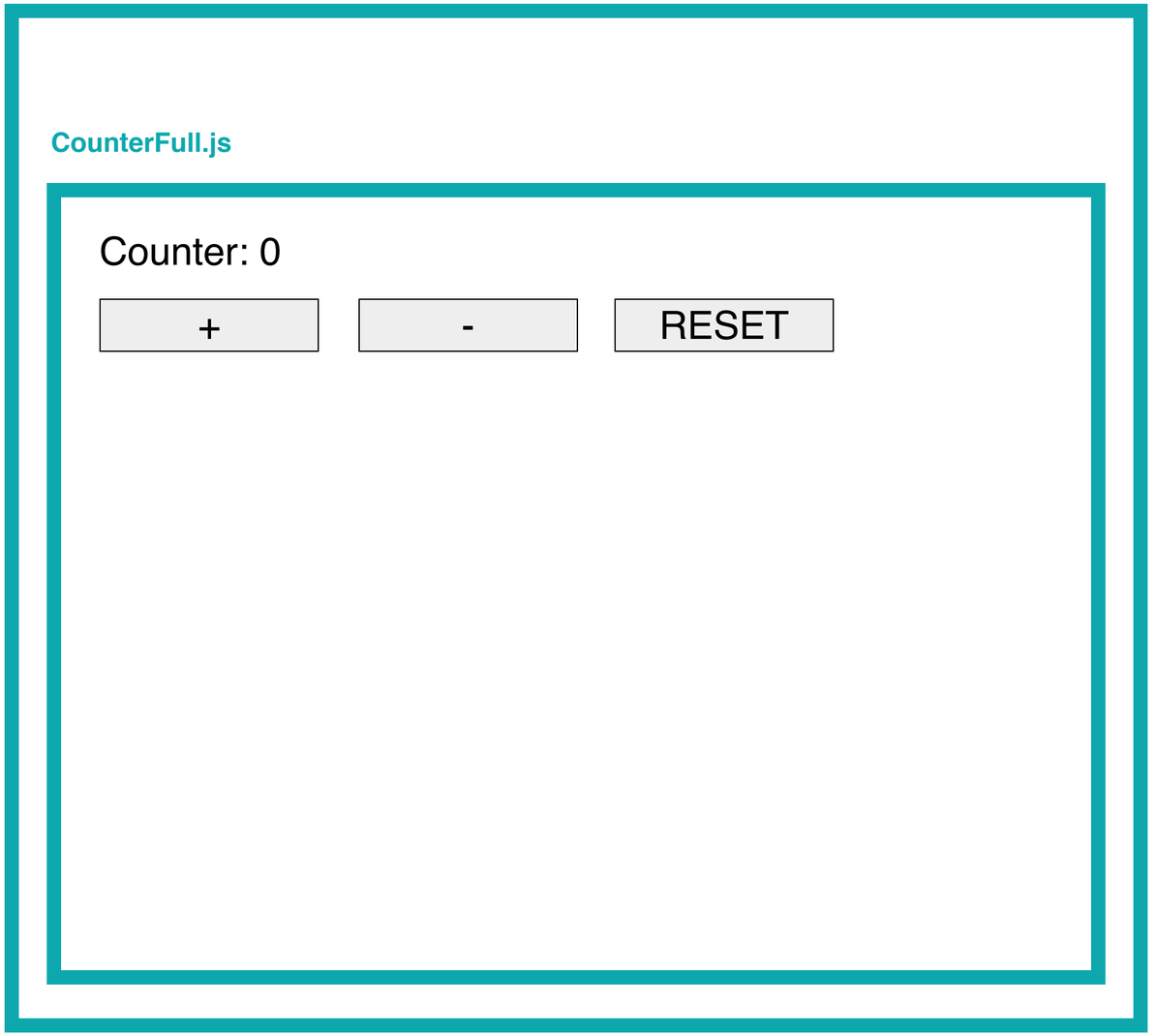
Wzorzec projektowy aplikacji opartej o bibliotekę React, ułatwiający tworzenie uniwersalnej logiki.

Polega na tworzeniu komponentu wyższego rzędu, który może obsługiwać uniwersalną logikę.

Home.js http://localhost:300/



Counter.js http://localhost:300/counter



Home.js

http://localhost:300/

CounterMin.js

Counter: 0

+

```
1  class CounterMin extends Component {
2      state = {counter: 0};
3
4      incrementCounter = () => this.setState({counter: this.state.counter + 1});
5
6      render(){
7          return (
8              <div>
9                  Counter: {this.state.counter}
10                 <button onClick={this.incrementCounter}>+</button>
11             </div>
12         )
13     }
14 }
```

Counter.js

http://localhost:300/counter

CounterFull.js

Counter: 0

+ - RESET

```
1  class CounterMin extends Component {
2      state = {counter: 0};
3
4      incrementCounter = () => this.setState({counter: this.state.counter + 1});
5
6      decrementCounter = () => this.setState({counter: this.state.counter - 1});
7
8      resetCounter = () => this.setState({counter: 0})
9
10     render(){
11         return (
12             <div>
13                 Counter: {this.state.counter}
14                 <button onClick={this.incrementCounter}>+</button>
15                 <button onClick={this.decrementCounter}>-</button>
16                 <button onClick={this.resetCounter}>RESET</button>
17             </div>
18         )
19     }
20 }
```

Home.js http://localhost:300/

Counter.js http://localhost:300/counter

CounterHOC.js

state
incrementCounter
decrementCounter
resetCounter

render()

CounterMin.js

Counter: 0

+

CounterFull.js

Counter: 0

+ - RESET

CounterHOC.js

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

function counterHoc(WrappedComponent) {
  return class extends Component {
    state = {counter: 0};

    incrementCounter = () => this.setState({counter: this.state.counter + 1});

    render() {
      return (
        <WrappedComponent counter={this.state.counter} incrementCounter={this.incrementCounter} />
      );
    }
  }
}

export default counterHoc;
```

Counter.js

```
import React from 'react';
import CounterHoc from './CounterHoc';

function Counter(props) {
  return (
    <div>
      Counter: {props.counter}
      <button onClick={props.incrementCounter}>+</button>
    </div>
  )
}

export default CounterHoc(Counter);
```

High Order Components

- jest to funkcja przyjmująca jako parametr komponent
- zwracany jest nowy komponent, posiadający swoją logikę oraz renderujący komponent otrzymany w parametrach

Issue #0

- Odtwórz logikę przedstawionej aplikacji, wykorzystując wzorzec **High Order Component**.

Home.js **http://localhost:300/**

CounterMin.js

Counter: 0

+

Counter.js **http://localhost:300/counter**

CounterFull.js

Counter: 0

+ - RESET

withLocalStorage.js

```
1  import React, { Component } from 'react';
2
3  function withLocalStorage(WrappedComponent) {
4    return class extends Component {
5      load = (key) => localStorage.getItem(key);
6      save = (key, data) => localStorage.setItem(key, data);
7
8      render() {
9        return (
10         <WrappedComponent load={this.load} save={this.save} />
11       );
12     }
13   }
14 }
15
16 export default withLocalStorage;
```

Issue #1

- Wykorzystując aplikację z **Issue #0** zadbaj żeby aktualna wartość licznika była zapamiętana i wyświetliła się po odświeżeniu strony.
- Wykorzystaj mechanizm **localStorage**.

Wzorce projektowe

Render Props

Render Props

Wzorzec projektowy aplikacji opartej o bibliotekę React, ułatwiający tworzenie uniwersalnej logiki.

Polega na przekazaniu renderowanego komponentu, jako props w komponencie z uniwersalną logiką.

Home.js

http://localhost:300/

CounterMin.js

Counter: 0

+

```
1  class CounterMin extends Component {
2      state = {counter: 0};
3
4      incrementCounter = () => this.setState({counter: this.state.counter + 1});
5
6      render(){
7          return (
8              <div>
9                  Counter: {this.state.counter}
10                 <button onClick={this.incrementCounter}>+</button>
11             </div>
12         )
13     }
14 }
```

Counter.js

http://localhost:300/counter

CounterFull.js

Counter: 0

+ - RESET

```
1  class CounterMin extends Component {
2      state = {counter: 0};
3
4      incrementCounter = () => this.setState({counter: this.state.counter + 1});
5
6      decrementCounter = () => this.setState({counter: this.state.counter - 1});
7
8      resetCounter = () => this.setState({counter: 0})
9
10     render(){
11         return (
12             <div>
13                 Counter: {this.state.counter}
14                 <button onClick={this.incrementCounter}>+</button>
15                 <button onClick={this.decrementCounter}>-</button>
16                 <button onClick={this.resetCounter}>RESET</button>
17             </div>
18         )
19     }
20 }
```

Home.js http://localhost:300/

Counter.js http://localhost:300/counter

CounterRenderProps.js

state
incrementCounter
decrementCounter
resetCounter
props

render() => props.component

CounterMin.js

Counter: 0

+

CounterFull.js

Counter: 0

+ - RESET

CounterWrapper.js

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

class CounterWrapper extends Component {
  state = { counter: 0};

  incrementCounter = () => this.setState({counter: this.state.counter + 1});

  render() {
    return this.props.render({
      counter: this.state.counter,
      incrementCounter: this.incrementCounter,
    })
  }
}

export default CounterWrapper;
```

Counter.js

```
import React from 'react';

function Counter(props) {
  return (
    <div>
      Counter {props.counter}
      <button onClick={props.incrementCounter}>+</button>
    </div>
  )
}

export default Counter;
```

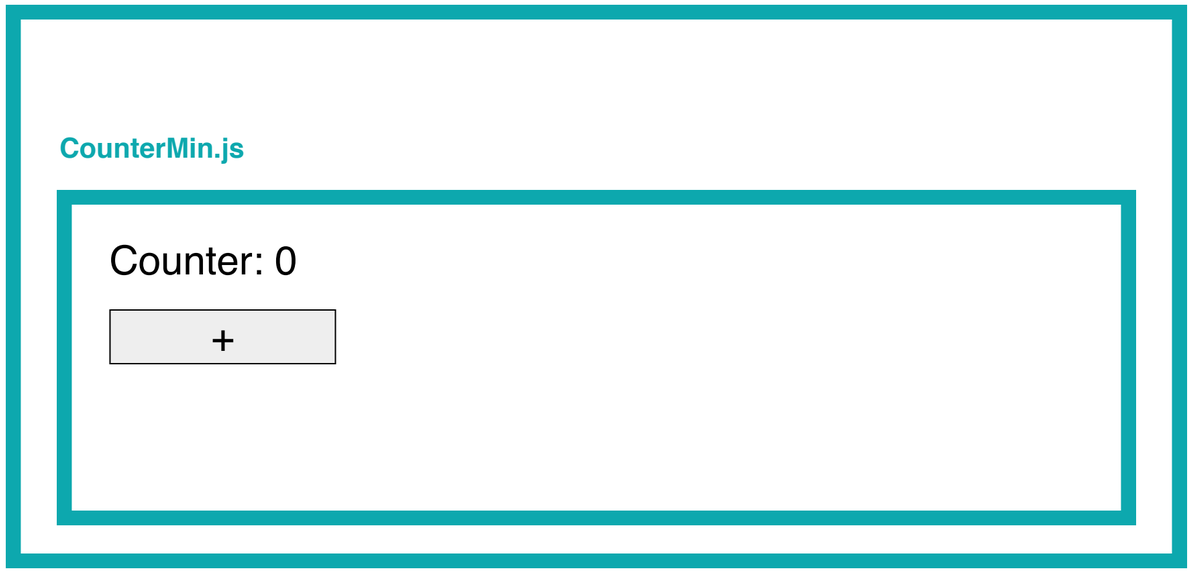
App.js - użycie

```
import React from 'react';
import Counter from './Counter';
import CounterWrapper from './CounterProps';

function App() {
  return (
    <div className="App">
      <CounterWrapper
        render={({counter, incrementCounter}) => <Counter counter={counter} incrementCounter={incrementCounter} />}
      />
    </div>
  );
}

export default App;
```

Issue #2

- Odtwórz logikę przedstawionej aplikacji, wykorzystując wzorzec **RenderProps**.

Home.js **http://localhost:300/**



Counter.js **http://localhost:300/counter**