



Custom Components in React

Aleksander Chojnowski



Krok 1

Przygotowanie projektu React

```
npx create-react-app customcomp
```

```
cd customcomp
```

```
npm start
```

.../custcomp/src/App.js

```
import React from "react";
import "./App.css";

const handleShowEmoji = (event) => alert(event.target.innerHTML);
const emojis = [ {
  emoji: "😬",    name: "shy",
}, {
  emoji: "👉",    name: "point_right",
}, {
  emoji: "👈",    name: "point_left",
},
];

function App() {
  const greetingID = "greeting";
  return (
    <div className = "container" >
      <h1 id={greetingID}>Hello World!</h1>
      <ul>
        {emojis.map((item) => (
          <li key={item.name}>
            <button onClick={handleShowEmoji}>
              <span role="img" aria-label={item.name} id={item.name}>
                {item.emoji}
              </span>
            </button>
          </li>
        ))}
      </ul>
    </div>
  );
}
export default App;
```

.../custcomp/src/App.css

```
.container {
  display: flex;
  flex-direction: column;
  align-items: center;
}


button {
  font-size: 30px;
  border: none;
  background: none;
  cursor: pointer;
}

ul {
  display: flex;
  padding: 0;
  list-style: none;
}

li {
  margin: 0 20px;
  padding: 0;
}

#root {
  position: absolute;
  inset: 0;
  display: flex;
  justify-content: center;
  align-items: center;
}
```





Krok 2

Tworzenie niezależnego Componentu przy użyciu React Classes

Komponenty Reactowe to kontenery z elementami, których możemy używać w naszej aplikacji.

Mamy dwa typy komponentów: klasowe i funkcyjne.

Aby stworzyć nasz pierwszy komponent, tworzymy nowy plik w katalogu ./src i importujemy do niego Reacta i klasy komponentów

```
import React, { Component } from 'react';  
  
export default class Instructions extends Component {}
```

Dodajmy tekst do naszej instrukcji

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

class Instructions extends Component {
  render() {
    return <h4>Kliknij na emoji żeby je wyświetlić</h4>
  }
}

export default Instructions;
```

Dodajmy obraz do komponentu

```
import React, { Component } from "react";
import emoji from "../emoji.png";

class Instructions extends Component {
  render() {
    return (
      <div className= instructions>
        <img src={emoji} alt="crying emoji" />
        <h4>Kliknij na emoji żeby je wyświetlić</h4>
      </div>
    );
  }
}

export default Instructions;
```

```
.instructions {
  display: flex;
  flex-direction: column;
  align-items: center;
}

.instructions img {
  width: 100px;
}
```



Krok 3

Uporządkowanie struktury plików w projekcie

W katalogu ./src/ tworzymy katalog ./components do którego przenosimy wszystkie elementy będące komponentami oraz elementy potrzebne do ich obsługi
W naszym wypadku są to pliki:

- App.css
- App.js
- App.test.js
- emoji.png
- Instructions.js

Powinniśmy dostać błąd kompilacji

Compiled with problems:

ERROR in ./src/App.js 5:0-19

Module not found: Error: Can't resolve './App.css' in 'C:\Users\User\Desktop\PracDyp5\custcomp\src'

ERROR in ./src/App.js 6:0-42

Module not found: Error: Can't resolve './Instructions' in 'C:\Users\User\Desktop\PracDyp5\custcomp\src'

./src/index.js

```
import React from "react";
import ReactDOM from "react-dom";
import "./index.css";
import App from "./components/App";
import reportWebVitals from "./reportWebVitals";

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById("root")
);

reportWebVitals();
```


Przenosimy komponenty do indywidualnych katalogów

Pliki komponentu App.*
przenosimy do katalogu o
nazwie komponentu, czyli
./src/components/App

Tak samo postępujemy z
komponentami tj. pliki
komponentu Instructions
przenosimy do katalogu
./src/components/Instructions

```
C:\USERS\USER\DESKTOP\PRACDYP5\CUSTCOMP\SRC
├── App.js
├── index.css
├── index.js
├── reportWebVitals.js
├── setupTests.js
├── components
│   ├── App
│   │   ├── App.css
│   │   ├── App.js
│   │   └── App.test.js
│   └── Instructions
│       ├── emoji.png
│       └── Instructions.js
```

./src/index.js

```
import React from "react";
import ReactDOM from "react-dom";
import "./index.css";
import App from "./components/App/App";
import reportWebVitals from "./reportWebVitals";

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById("root")
);

reportWebVitals();
```

./src/components/App/App.js

```
import React from "react";
import "./App.css";
import Instructions from
  "../Instructions/Instructions.js";

const handleShowEmoji = (event) =>
  alert(event.target.innerHTML);

...

export default App;
```



Krok 4

Rozdzielenie styli do osobnych plików

Z pliku `./src/components/App/App.css` wycinamy style odpowiadające za stylowanie komponentu Instructions i wklejamy je do pliku

`./src/components/Instructions/Instructions.css`

W pliku `./src/components/Instructions/Instructions.js` importujemy style:

```
import './Instructions.css';
```

Krok 5

Tworzenie komponentu funkcyjnego

Aby zmienić nasz komponent klasowy na funkcyjny, w pliku `./src/components/Instructions/Instructions.js` wprowadzamy następujące zmiany

```
import React from "react";
import emoji from "../emoji.png";
import "../Instructions.css";

function Instructions() {
  return (
    <div className="instructions">
      <img src={emoji} alt="crying emoji" />
      <h4>Kliknij na emoji żeby je wyświetlić</h4>
    </div>
  );
}

export default Instructions;
```

Możemy również użyć funkcji strzałkowej

```
import React from "react";
import emoji from "./emoji.png";
import "./Instructions.css";

const Instructions = () => (
  <div className="instructions">
    <img src={emoji} alt="crying emoji" />
    <h4>Kliknij na emoji żeby je wyświetlić</h4>
  </div>
);

export default Instructions;
```