React – część II, formularze, wyświetlanie warunkowe, obrazki

Anna Gogolińska

Formularze

- Elementy formularza, które znamy:
  - Pole tekstowe (text, number, …), textarea, select, checkbox, radio.
- W React formularze działają podobnie jak w HTML, ale wartości pól najczęściej zapisujemy w **stanie komponentu**.
  - Label - <label **htmlFor**="name">Imię</label>
    <input id="name" type="text" />

onChange

- Zdarzenie zmiany wartości pola formularza – onChange.
  - Jako reakcję tak jak w przypadku onClick przekazujemy funkcję.

  <input type="text" onChange={**handleChange**} />
  - W tym przypadku funkcji przekazujemy argument – obiekt samego zdarzenia (podobnie jak można w JS).
  - Zdarzenie zwyczajowo oznaczamy zmienną **event** lub **e**.

Argument event

- Zazwyczaj cały event (e) nas nie interesuje, a chcemy pozyskać z niego informacje o polu, które zostało zmienione.
  - e.target – źródło zdarzenia, czyli pole formularza, które zostało zmienione i dla którego wywołano funkcję.
  - e.target.value – wartość z pola formularza. Stosuje się dla input, textarea, select, radio.
  - e.target.checked – dla checkbox'a i radio, true lub false w zależności czy zaznaczony.

Wartość pola – stan

- Zazwyczaj wartość pola formularza (lub czy zaznaczone) jest przypisywana jako stan.
  - Wtedy funkcja obsługująca zdarzenie wywołuje funkcję ustawiającą wartość stanu.
  - Argumentem dla funkcji ustawiającej wartość stanu jest wartość pola formularza (lub czy zaznaczone).

Input

```
import { useState } from "react";

export default function App() {

  const [name, setName] = useState("");

  const handleChange = (e) => {
    setName(e.target.value);
  };

  return (
    <div>
      <input type="text" onChange={handleChange} />
      <p>Wpisałaś: {name}</p>
    </div>
  );
}
```

Select

```
import { useState } from "react";
export default function App() {

  const [country, setCountry] = useState("");

  const handleChange = (e) => {
    setCountry(e.target.value);
  };

  return (
    <div>
      <select onChange={handleChange}>
        <option value="pl">Polska</option>
        <option value="de">Niemcy</option>
        <option value="fr">Francja</option>
      </select>
      <p>Wybrany kraj: {country}</p>
    </div>
  );
}
```

Radio i checkbox

- Radio:
  - Dla radio można sprawdzić checked ale nie jest to używane.
  - Zazwyczaj robi się kilka radio z jednym name, różnymi value i tą samą funkcją do obsługi onChange.
  - Każdy z nich odpowiada jednemu, wspólnemu stanowi.
  - Wtedy w funkcji używa się event.target.value i to zwróci value tego radio, które jest zaznaczone i to ustawia się jako wartość dla stanu.
- Checkbox:
  - Dla checkbox można użyć też value ale zazwyczaj nie jest to używane.
  - Każdy checkbox ma zazwyczaj własną funkcję do obsługi zaznaczenia i własny, odpowiadający stan.
  - W funkcji używa się event.target.checked aby ustawić stan odpowiedni dla danego checkboxa.

Radio

```
import { useState } from "react";
export default function App() {

  const [gender, setGender] = useState("");

  const handleChange = (e) => {
    setGender(e.target.value);
  };

  return (
    <div>
      <input type="radio" id="female" name="gender" value="female" onChange={handleChange} />
      <label htmlFor="female">Kobieta</label>     <br />
      <input type="radio"  id="male"  name="gender" value="male" onChange={handleChange} />
      <label htmlFor="male">Mężczyzna</label>
      <p>Wybrana opcja: {gender}</p>
    </div>
  );
}
```

Checkbox

```
import { useState } from "react";
export default function App() {

  const [java, setJava] = useState(false);
  const [python, setPython] = useState(false);

  const handleJavaChange = (e) => setJava(e.target.checked);
  const handlePythonChange = (e) => setPython(e.target.checked);

  return (
    <div>
      <input type="checkbox" id="java" onChange={handleJavaChange} />
      <label htmlFor="java">Java</label> <br />
      <input type="checkbox" id="python" onChange={handlePythonChange} />
      <label htmlFor="python">Python</label>
      <p>Java: {java.toString()}</p>
      <p>Python: {python.toString()}</p>
    </div>
  );   }
```

onSubmit

- Zdarzenie wywoływane przy wysyłaniu formularza – czyli jako reakcja na naciśnięcie <input type="submit">
- Reakcję na to zdarzenie dodajemy do znacznika <form>
  - Jak przy innych zdarzeniach podajemy jako reakcję funkcję strzałkową.
- W funkcji która jest reakcją na zdarzenie należy podać jako argument event i wywołać na nim preventDefault() aby strona się nie przeładowała.

Przykład

```
import { useState } from "react";

export default function App() {

  const [name, setName] = useState("");
  const handleChange = (e) => setName(e.target.value);

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("Wpisano:", name);
  };

  return (
    <div>
      <h2>Formularz</h2>
      <form onSubmit={handleSubmit}>
        <label>Imię: </label>
        <input type="text" value={name} onChange={handleChange} />
        <input type="submit" value="Wyślij" />
      </form>
      <p>Wpisałaś: {name}</p>
    </div>
  );
}
```

Obrazki

- Jeśli mamy obrazek na dysku i chcemy go wyświetlić na stronie:
  - Obrazek (obrazki) umieszczamy w folderze src/assets
  - Potem trzeba je zaimportować do pliku App.jsx
    - Podczas tego importu nadajemy nazwę dla obrazka.
    - Jest to u góry pliku:
      import obrazek from "./assets/plik.png";
  - Potem używając tej nazwy możemy podać ją źródło obrazka.
    ****

Obrazki

- Drugi sposób (łatwiejszy):
  - Obrazki umieszczamy nie w src/assets ale w folderze public.
  - Potem można podać w src od razu nazwę pliku (lub ścieżkę jeśli utworzymy podkatalog).
    - Przed nazwą dodajemy „/”
    -
    -

Wyświetlanie warunkowe

- W React często chcemy wyświetlić element tylko wtedy, gdy spełniony jest jakiś warunek.
  - Do tego można użyć operatora logicznego **&&**.
  - Składnia: {warunek && element}
  - Jeśli warunek jest prawdziwy to element będzie widoczny. Jeśli nie jest prawdziwy to element się nie wyświetli.
  - Strona odświeża się sam jak zmieni się jakiś stan, więc jeśli w warunku użyty jest jakiś stan to warunek będzie na nowo sprawdzony i element się pokaże albo ukryje jeśli wartość warunku się zmieniła.

Wyświetlanie warunkowe

```
import { useState } from "react";
import kot from "./assets/kot.png";

export default function App() {

  const [showImage, setShowImage] = useState(false);

  const handleChange = (e) => {
    setShowImage(e.target.checked);
  };

  return (
    <div>
      <input type="checkbox" id="image" onChange={handleChange} />
      <label htmlFor="image">Pokaż obrazek</label>
      {showImage && <img src={kot} alt="Kot" width="200" />}
    </div>
  );
}
```