React 4 –zmiana wartości tablicy – tablica jako stan

Anna Gogolińska

Tablica jako stan

- Stan w React może przechowywać nie tylko liczby lub napisy, ale także **tablicę obiektów**.
- Składania jest taka sama jak w przypadku stanu, który jest liczbą czy napisem.
  - Mamy więc nazwę stanu, funkcję do zmiany wartości i wartość początkową w useState.
  - Bardziej rozbudowane jest jedynie podanie wartości początkowej – bo należy podać tu tablicę.

Przykład

```
const [students, setStudents] = useState([
  { id: 1, name: "Anna", points: 10 },
  { id: 2, name: "Jan", points: 15 },
  { id: 3, name: "Maria", points: 12 }
]);


...
<ul>
  {students.map((student) => (
    <li key={student.id}>
      {student.name} ({student.points})
    </li>
  ))}
</ul>
```

Przykład

```
const [students, setStudents] = useState([
  { id: 1, name: "Anna", points: 10 },
  { id: 2, name: "Jan", points: 15 },
  { id: 3, name: "Maria", points: 12 }
]);
```

Wyświetlanie

- Tablicę która jest w stanie wyświetla się tak samo jak „zwykłą” tablicę.
  - Używamy więc map().
- Wcześniej używaliśmy nazwy tablicy – teraz należy użyć nazwy stanu.

Zmiana stanu

- W React **nie zmieniamy elementu tablicy bezpośrednio**. Tworzymy nową tablicę i zapisujemy ją w stanie.
- Oczywiście kod zmiany tablicy piszemy w funkcji – funkcji strzałkowej.
  - Jeśli chcemy zmienić w ten sam sposób każdy element tablicy – nie potrzebujemy argumentu funkcji.
  - Funkcja może mieć jednak argument – można go użyć aby wybrać elementy które chcemy zmienić.
    - Często ten argument to id elementu który chcemy zmienić, a reszta ma być taka sama.

Zmiana stanu – każdy element tablicy

- W kodzie funkcji definiujemy nową tablicę.
  - Nowa tablica powstaje poprzez wykonanie map() na starej tablicy.
  - W map() dla każdego elementu pierwotnej tablicy piszemy jak na jego podstawie ma powstać nowy element. Ten nowy element będzie dodawany do tworzonej tablicy.
  - map() musi zwrócić ten nowy element więc należy napisać **return** nowyElement i zdefiniować ten nowy element.
  - Tworząc nowy element należy najpierw skopiować wartości pól pierwotnego elementu, składnia: **…element**
  - Potem napisać nową wartość dla wybranego pola (lub pól), składnia: **pole: nowa wartość**
- Tak stworzoną tablicę ustawiamy jako nową wartość dla stanu.

Przykład

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

  const [students, setStudents] = useState([
    { id: 1, name: "Anna", points: 10 },
    { id: 2, name: "Jan", points: 15 },
    { id: 3, name: "Maria", points: 12 }
  ]);

  const addPointToAll = () => {
    const newStudents = students.map((student) => {
      return {
        ...student,
        points: student.points + 1
      };
    });
    setStudents(newStudents);
  };
```

Zmiana stanu – jeden element tablicy

- Podobnie jak wcześniej piszemy funkcję strzałkową o prawie takiej samej postaci.
  - Funkcja ma mieć jednak argument – id wybranego elementu.
  - W ciele funkcji dodajemy warunek:
    - Jeśli id elementu przez który właśnie przechodzi map() jest równe id elementu który ma być zmieniony:
      - Zwracamy nowy, zmieniony element – tworzymy go jak w poprzednim przykładzie.
    - Jeśli id elementu przez który właśnie przechodzi map() jest inne to zwracamy element pierwotnej tablicy bez zmian.

Przykład

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

  const [students, setStudents] = useState([
    { id: 1, name: "Anna", points: 10 },
    { id: 2, name: "Jan", points: 15 },
    { id: 3, name: "Maria", points: 12 }
  ]);

const addPoint = (id) => {
  const newStudents = students.map((student) => {
    if (student.id === id) {
      return {
        ...student,
        points: student.points + 1
      };
    } else {
      return student;
    }
  });
  setStudents(newStudents);
};
```

Wywołanie funkcji

- Funkcję zmiany tablicy wywołujemy zazwyczaj jako reakcję na zdarzenie, zazwyczaj na onClick dla przyciski.
  - Jeśli funkcja nie ma argumentów to po prostu wpisujemy jej wywołanie.
  - Jeśli ma mieć jakiś konkretny argument to go podajemy.
  - Często mamy taką sytuację, że dla każdego elementu tablicy tworzymy jakieś elementy HTML plus przycisk który będzie zmieniał ten konkretny element.
    - Wtedy używamy map() aby stworzyć elementów strony.
    - Między innymi tworzymy też przycisk i definiując go, jako reakcję na naciśnięcie, podajemy funkcję do zmiany tablicy z argumentem równym id elementu dla którego tworzymy przycisk.

Przykład

```
  return (
    <div>

      <h2>Lista studentów</h2>

      <button onClick={addPointToAll}>
        Dodaj punkt wszystkim
      </button>

      <ul>
        {students.map((student) => (
          <li key={student.id}>
            {student.name} ({student.points})
          </li>
        ))}
      </ul>

    </div>
  );

}
```

Gdzie funkcja addPointToAll() jest:

```
const addPointToAll = () => {

    const newStudents =
                students.map((student) => {
      return {
        ...student,
        points: student.points + 1
      };
    });

    setStudents(newStudents);

  };
```

Przykład

```
  const [students, setStudents] = useState([ ….]);
  const [studentId, setStudentId] = useState("");

  const handleChange = (e) => {
    setStudentId(e.target.value);
  };
  return (
    <div>
      <input type="number" value={studentId} onChange={handleChange} />
      <button onClick={() => addPoint(Number(studentId))}> Dodaj punkt </button>
      <ul>
        {students.map((student) => (
          <li key={student.id}>
            {student.name} ({student.points})
          </li>
        ))}
      </ul>
    </div>
  );
}
```

Przykład

```
  const [students, setStudents] = useState([ ….]);
  const [studentId, setStudentId] = useState("");

  const handleChange = (e) => {
    setStudentId(e.target.value);
  };
  return (
    <div>
      <input type="number" value={studentId} onChange={handleChange} />
      <button onClick={() => addPoint(Number(studentId))}> Dodaj punkt </button>
      <ul>
        {students.map((student) => (
          <li key={student.id}>
            {student.name} ({student.points})
          </li>
        ))}
      </ul>
    </div>
  );
}
```

Gdzie addPoint() to funkcja:

```
 const addPoint = (id) => {
    const newStudents = students.map((student) => {
      if (student.id === id) {
        return {
          ...student,
          points: student.points + 1
        };
      } else { return student; }
    });
    setStudents(newStudents);
  };
```

Przykład

```
  const [students, setStudents] = useState([ ….]);
  const [studentId, setStudentId] = useState("");

  const handleChange = (e) => {
    setStudentId(e.target.value);
  };
  return (
    <div>
      <input type="number" value={studentId} onChange={handleChange} />
      <button onClick={() => addPoint(Number(studentId))}> Dodaj punkt </button>
      <ul>
        {students.map((student) => (
          <li key={student.id}>
            {student.name} ({student.points})
          </li>
        ))}
      </ul>
    </div>
  );
}
```

Number() – zmienia ma liczbę bo wartość pola rozumiana jest jako napis

Przykład

```
{students.map((student) => (
  <div key={student.id}>
    {student.name} ({student.points})
    <button onClick={() => addPoint(student.id)}>
      Dodaj punkt
    </button>
  </div>
))}
```

Gdzie addPoint() to funkcja:

```
const addPoint = (id) => {
  const newStudents = students.map((student) => {
    if (student.id === id) {
      return {
        ...student,
        points: student.points + 1
      };
    } else { return student; }
  });
  setStudents(newStudents);
};
```

Dodanie elementu do tablicy

- Należy oczywiście napisać funkcję, która będzie wywołana jako reakcja na zdarzenie – zapewne naciśnięcie przycisku.
- W tej funkcji trzeba stworzyć nowy element:
  - Składnia const nowy = {
    pole1: wartość1,
    pole2: wartość 2,
    …
    };
  - Dodać ten nowy element do tablicy:
    const nowaTab = [...tablica, nowy];
    - Operator … kopiuje wszystkie elementy tablicy.
  - Ustawić jako wartość stanu nową tablicę.

Przykład

```
const [students, setStudents] = useState([ …]);
const [name, setName] = useState("");
const [points, setPoints] = useState("");

const handleAddStudent = () => {
  const newStudent = {
    id: students.length + 1,
    name: name,
    points: Number(points)
  };
  const newStudents = [...students, newStudent];
  setStudents(newStudents);
};
return (
  <div>
    <input type="text" placeholder="Imię" value={name} onChange={(e) => setName(e.target.value)} />
    <input type="number" placeholder="Punkty" value={points} onChange={(e) =>
      setPoints(e.target.value)} />
    <button onClick={handleAddStudent}>Dodaj</button>
  </div>
  …
); }
```