React 3 – tablice, map(), filter()

Anna Gogolińska

Tablice

- Tablice/listy mogą przechowywać różne dane: napisy, liczby, obiekty.
  - const fruits = ["Jabłko", "Banan", "Gruszka"];
  - const numbers = [1, 2, 3, 4];
  - const students = [
    { name: "Anna", age: 20 },
    { name: "Jan", age: 22 },
    { name: "Maria", age: 19 }
    ];
- Tablice i listy w React możemy rozumieć że są tym samym.
- Zawsze można pobrać wielkość tablicy: **tablica.length**.

Tablice

- Aby uzyskać dostęp do elementu należy użyć jego numeru (od 0).
  - Jeśli element jest obiektem należy jeszcze podać pole (element obiektu) do którego się odwołujemy z użyciem . : element[nr].pole.
  - Pojedyncze elementy: colors[0], fruit[2].
  - Obiekty: students[0].name, students[2].age.

map() - key

- map() służy do przechodzenia po wszystkich elementach tablicy.
  - Używając map() definiujemy jak będziemy określać elementy tablicy.
  - Musimy dla każdego elementu tablicy wskazać co jest kluczem **key** tego elementu.
  - key to identyfikator elementu.
  - key nie jest dla nas – my go nie używamy, to coś czego potrzebuje sam React aby rozumieć który element jest który w tablicy. key pomaga zorientować się React'owi jakich zmian dokonuje się w tablicy, np.:
    - [1, 2, 3] – zmieniona na [1, 3]. Czy usunięto element o wartości 2 i element 3 stało się drugim elementem? Czy usunięto 3 a dla drugiego elementu zmieniono jego wartość z 2 na 3?

map() - key

- Jeśli element nie ma dobrej wartości dla key to używa się **index**.
  - index to numer elementu w tablicy.
  - Używa się go kiedy mamy tablicę pojedynczych elementów (liczb, napisów) lub obiektów, które nie mają jakiegoś typu identyfikatorów.
- Jeśli elementami tablicy są obiekty, które mają jakiś typ identyfikatorów – zazwyczaj pole id to można go użyć jako key.

map()

- Składnia:
  - Jako pierwszy argument dla map() podajemy jak będziemy określać element tablicy.
  - Jako kod w map() podajemy zazwyczaj jakiś element HTML który ma się stworzyć dla każdego elementu.
    - Jeśli używamy index (numer elementu w tablicy) jako key to dajemy go jako drugi argument w map():
      - **{ tablica.map((element, index) => (**

        **<li key={index}>{element}
**

)) }
    - Jeśli jako key używamy pole np. id z obiektu:
      - **{ tablica.map((element) => (**

        **<li key={element.id}>{element.pole}
**

)) }

map() – przykład

```
export default function App() {

  const numbers = [10, 20, 30];

  return (
    <div>
      {numbers.map((number, index) => (
        <p key={index}>Liczba: {number}</p>
      ))}
    </div>
  );

}
```

map() - przykład

```
export default function App() {

  const books = [
    { title: "Java", author: "Nowak", year: 2020 },
    { title: "Python", author: "Kowalski", year: 2019 },
    { title: "C++", author: "Wiśniewski", year: 2018 }
  ];

  return (
    <div>
      <h2>Książki</h2>
      <ul>
        {books.map((book, index) => (
          <li key={index}>
            {book.title} – {book.author} – {book.year}
          </li>
        ))}
      </ul>
    </div>
  );
}
```

map() – przykład

```
export default function App() {

  const products = [
    { id: 1, name: "Mysz" },
    { id: 2, name: "Klawiatura" },
    { id: 3, name: "Monitor" }
  ];

  return (
    <div>
      {products.map((product) => (
        <div key={product.id}>
          Produkt: {product.name}
        </div>
      ))}
    </div>
  );
}
```

filter()

- filter() służy do wybierania tylko niektórych elementów tablicy.
  - filter() zwraca nową, przefiltrowaną tablicę.
  - Pisząc filter() jako argument funkcji definiujemy jak będziemy oznaczać elementy tablicy.
  - Jako kod filter() piszemy warunek – będzie on sprawdzany dla każdego elementu tablicy.
    - W warunku oczywiście powinien się znaleźć element tablicy.
    - W warunkach używać należy === i !== (bezpieczniejsze jeśli chodzi o typy bo ich nie zmienia).
  - Jeśli dla danego elementu warunek jest prawdziwy to element trafia do wynikowej tablicy. Jeśli warunek jest fałszywy to nie trafia.

filter() – przykład

- const numbers = [1, 2, 3, 4, 5, 6];
  const evenNumbers = numbers**.filter((number) => number % 2 === 0);**

- const fruits = ["Jabłko", "Banan", "Gruszka"];
  const selected = fruits**.filter((fruit) => fruit !== "Banan");**

- const students = [
  { name: "Anna", age: 20 },
  { name: "Jan", age: 22 },
  { name: "Maria", age: 19 }
  ];
  const adults = students**.filter((student) => student.age >= 21);**

filter() i map()

- Możemy obie rzeczy połączyć.
  - Najpierw wpisujemy filter() aby wybrać tylko niektóre elementy.
  - Potem na wynikowej tablicy z filter() wywołujemy map().
  - Możemy najpierw stworzyć wynikową tablicę (u góry kodu).
  - Możemy też wywołać na raz filter() i map(). Ma to trochę „dziwną” składnię:
    - **{ tablica.filter(…).map(…) }**

Przykład

```
export default function App() {
  const numbers = [1, 2, 3, 4, 5, 6];

  const evenNumbers = numbers.filter((number) => number % 2 === 0);

  return (
    <div>
      <h2>Liczby parzyste</h2>
      <ul>
        {evenNumbers.map((number, index) => (
          <li key={index}>{number}</li>
        ))}
      </ul>
    </div>
  );
}
```

Przykład

```
export default function App() {

  const students = [
    { id: 1, name: "Anna", age: 20 },
    { id: 2, name: "Jan", age: 22 },
    { id: 3, name: "Maria", age: 19 }
  ];
  const adults = students.filter((student) => student.age >= 20);

  return (
    <ul>
      {adults.map((student) => (
        <li key={student.id}>
          {student.name} ({student.age})
        </li>
      ))}
    </ul>
  );
}
```

Przykład – to samo co poprzedni inny zapis

```
export default function App() {

const students = [
  { id: 1, name: "Anna", age: 20 },
  { id: 2, name: "Jan", age: 22 },
  { id: 3, name: "Maria", age: 19 }
];

return (
  <ul>
    {students
      .filter((student) => student.age >= 20)
      .map((student) => (
        <li key={student.id}>Imię: {student.name} (wiek: {student.age})
        </li>
      ))}
  </ul>
);
}
```

Przykład

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

export default function App() {

 const tasks = [ { name: "Napisz zadanie domowe", active: true },
  { name: "Umyj samochód", active: false },
  { name: "Zrób zakupy", active: true },
  { name: "Posprzątaj pokój", active: false }  ];

 const [showActive, setShowActive] = useState(true);

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

 const filteredTasks = tasks.filter((task) => task.active === showActive);

 return (
  <div>
   <h2>Lista zadań</h2>
   <input type="checkbox" id="active" checked={showActive}  onChange={handleChange}  />
   <label htmlFor="active">Pokaż aktywne zadania</label>
   <ul>
    {filteredTasks.map((task, index) => (
     <li key={index}>{task.name}</li>
    ))}
   </ul>  </div>
 ); }
```

Przykład

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

export default function App() {

 const tasks = [ { name: "Napisz zadanie domowe", active: true },
  { name: "Umyj samochód", active: false },
  { name: "Zrób zakupy", active: true },
  { name: "Posprzątaj pokój", active: false }  ];

 const [showActive, setShowActive] = useState(true);

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

 const filteredTasks = tasks.filter((task) => task.active === showActive);

 return (
  <div>
   <h2>Lista zadań</h2>
   <input type="checkbox" id="active" checked={showActive}  onChange={handleChange}  />
   <label htmlFor="active">Pokaż aktywne zadania</label>
   <ul>
    {filteredTasks.map((task, index) => (
     <li key={index}>{task.name}</li>
    ))}
   </ul>  </div>
 ); }
```