React – część I, podstawy, useState, zdarzenia

Anna Gogolińska

Czym jest React?

- React to biblioteka JavaScript służąca do tworzenia interfejsów użytkownika.
- Pozwala budować aplikacje jako zestaw małych elementów zwanych komponentami.
- Każdy komponent odpowiada za fragment strony (np. formularz, lista, galeria).
- React automatycznie aktualizuje widok gdy zmieniają się dane.

Jak wygląda projekt React

- Najważniejszy plik to **App.jsx** – tutaj znajduje się główny komponent aplikacji.
- Folder src zawiera kod aplikacji.
- Folder public może zawierać obrazy lub inne pliki statyczne.
  - src/App.css – style (np. padding 20px).
  - src/index.jsx – start aplikacji, tu można dołączyć Bootstrap.
  - public/assets – obrazki (najłatwiej na egzamin)

Komponent w React

- Komponent to funkcja JavaScript, która zwraca JSX.
  - JSX wygląda podobnie do HTML, ale jest częścią JavaScript.
- Komponent opisuje jak ma wyglądać fragment strony.
- Na egzaminie najczęściej wystarczy jeden komponent: App.

Przykład komponentu

```
export default function App() {
  return (
    <div>
      <h1>Moja aplikacja</h1>
      <p>To jest prosty komponent.</p>
</div>
  );
}
```

JSX

- JSX wygląda jak HTML, ale ma drobne różnice:
  - class → className (nadawanie klasy stylu)
  - for w label → htmlFor (etykieta dla pola formularza)
  - W JSX możesz wstawiać JS w { }
  - var → const (do tworzenia zmiennych)
  - każdy znacznik musi być zamknięty:
, <img/>, <input /> (spacja nie jest konieczna).

JSX

```
export default function App() {
  const name = 'Ala';
  return (
    <div className="container">
      <label htmlFor="x">Imię</label>
      <input id="x" className="form-control" />
      <p>Cześć, {name}!</p>
       Napis jakiś <br/>
       Napis jakiś
    </div>
  );
}
```

Czym jest stan?

- Stan (**state**) to dane, które mogą się zmieniać w czasie działania aplikacji.
- Przykłady: tekst wpisany w formularzu, liczba kliknięć, zaznaczony checkbox.
- React przechowuje stan i odświeża widok gdy się zmieni.

Przykład useState

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

export default function App() {
  const [count, setCount] = useState(0);

  const increase = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <p>Licznik: {count}</p>
      <button onClick={() => increase()}>Kliknij</button>
    </div>
  );
}
```

Stan

- Składnia const [count, setCount] = useState(0);
  - Tworzymy **zmienną** oznaczającą wartość **stanu**, jej zmiana powoduje automatyczne odświeżenie widoku – tu zmienna to *count*
    - Możemy w kodzie komponentu wpisać:
      Licznik: {count}
  - Tworzymy funkcję, której używać będziemy do zmiany wartości stanu – tu *setCount*
    - Wywołanie np. setCount(5) zmieni wartość count na 5.
  - Ustawiamy wartość początkową przez *useState,* tu na wartość 0 (można też ustawić na napis - useState(""), tablicę - useState([]), obiekt - useState({})).

Przykład useState

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

export default function App() {
  const [count, setCount] = useState(0);

  const increase = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <p>Licznik: {count}</p>
      <button onClick={ () => increase() }>Kliknij</button>
    </div>
  );
}
```

Zdarzenia w React

- Zdarzenia pozwalają reagować na działania użytkownika.
- Najczęstsze zdarzenia to onClick, onChange i onSubmit.
  - onClick reaguje na kliknięcie przycisku, onChange reaguje na zmianę wartości pola formularza (więcej na następnej lekcji), onSubmit reaguje na wysłanie formularza (więcej na następnej lekcji).
- Reakcje na zdarzenia definiujemy jako **funkcje strzałkowe** (o nich za chwilę).
- Reakcje na zdarzenie otaczamy { } (a nie " ").

Jak pisaliśmy to w JS?

```
<script>
function klik() {
  console.log("Kliknięto przycisk");
}
</script>

<button onclick="klik()">Kliknij</button>
```

Reakcja na zdarzenie

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

export default function App() {
  const [count, setCount] = useState(0);

  const increase = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <p>Licznik: {count}</p>
      <button onClick={ () => increase() }>Kliknij</button>
    </div>
  );
}
```

Reakcja na zdarzenie onClick – otoczona jest ona { } i ma formę funkcji strzałkowej.

Funkcja strzałkowa

- W React zamiast „zwykłych” funkcji używa się funkcji strzałkowych. Mają one formę:
  (argmenty) => { kod }
- Możemy definiować swoje funkcje używając składni funkcji strzałkowych i potem je wywoływać.
  - Jak definiujemy swoją funkcję to zaczynamy ją od **const** tak samo jak dla zmiennych czy stanów (nie function).
- Reakcja na zdarzenie zazwyczaj jest opisywana jako funkcja strzałkowa. Wtedy argumenty są narzucone przez zdarzenie – np. **onclick nie ma argumentów**.
- Będziemy używać również innych elementów które wymagają składni funkcji strzałkowych i które będą wymuszać określone argumenty (o tym później).

Funkcje strzałkowe

- Określenie funkcji zaczynamy od argumentów.
- Jeśli funkcja nie ma argumentów piszemy (), jeśli ma argumenty to np. (n), (x, y):
  - Opis reakcji na zdarzenie:
    - onClick={**()** => increase()}
  - Własna funkcja:
    - const increase = **()** => ….
    - const increase = **(n)** => {
- Potem jest strzałka.

Funkcja strzałkowa

- Po strzałce piszemy kod funkcji, który się ma wykonać.
  - Jeśli jest tylko jedna instrukcja w funkcji to ją piszemy:
    - Reakcja na zdarzenie:
      - onClick={() => **increase()**}
    - Własna funkcja:
      - const increase = () => **setCount(count + 1)**;
  - Jeśli jest więcej instrukcji to trzeba otoczyć je klamrami:
    - Reakcja na zdarzenie:
      - onClick={ () => **{**
console.log("Klik");
increase();
} }
    - Własna funkcja:
      - const increase = () => **{**
console.log("klik");
setCount(count + 1);
};

Reakcja na zdarzenie

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

export default function App() {
  const [count, setCount] = useState(0);

  const increase = () => {
      console.log("klik");
      setCount(count + 1);
  };

  return (
    <div>
      <p>Licznik: {count}</p>
      <button onClick={ () => increase() }>Kliknij</button>
    </div>
  );
}
```

- W reakcji na naciśnięcie przycisku wywołujemy funkcję **increase()**, która jest zdefiniowana jako funkcja strzałkowa. Jest tylko jedna instrukcja więc nie trzeba pisać { }.
- Funkcja increase() wypisuja na konsolę „klik” i ustawia nową wartość dla zmiennej count poprzez wywołanie funkcji setCount().
- setCount() jest zdefiniowana jako funkcja która modyfikuje wartość stanu count.

Reakcja na zdarzenie

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

export default function App() {
  const [count, setCount] = useState(0);

  const increase = () => {
      console.log("klik");
      setCount(count + 1);
  };

  return (
    <div>
      <p>Licznik: {count}</p>
      <button onClick={increase}>Kliknij</button>
    </div>
  );
}
```

- Jeśli całą reakcją na zdarzenie jest tylko wywołanie jednej funkcji bez argumentów to można wpisać tylko jej nazwę zamiast całej składni strzałkowej.

Reakcja na zdarzenie

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

export default function App() {
  const [count, setCount] = useState(0);

  const increase = (n) => {
      setCount(count + n);
  };

  return (
    <div>
      <p>Licznik: {count}</p>
      <button onClick={ () => increase(10) }>Zwiększ o 10</button>
    </div>
  );
}
```