Kódím.cz

/ Webové aplikace / Digitální akademie Web / React

# 5 Formulářové prvky, efekty

Zapojíme do našich React aplikací formulářové prvky a ukážeme si, jak pomocí efektů volat API.

Předchozí
**Události, stav**

Následující
**Komunikace dítě rodič**

Díky tomu, že jsme se v minulé lekci naučili pracovat se stavem, otvírají se nám v Reactu mnohé nové možnosti.

## Formulářové prvky

Formulářové prvky jako textová políčka, zaškrtávací tlačítka apod. jsou jedním z hlavních způsobů, jak získat vstup od uživatele. V čistém JavaScritpu jsme zvyklí získávat hodnoty z těchto prvků tak, že je vybereme pomocí `querySelector` a použijeme například vlastnost `value`.

```
<form>
  <label>Zadej svůj věk: <input id="age-input" type="text" /></label>
  <button type="submit">Odeslat</button>
</form>
```

```
const ageInput = document.querySelector('#age-input');
console.log(ageInput.value);
```

V Reactu však k DOM elementům na stránce přístup nemáme. Hodnotu uvnitř textového políčka si tak musíme uložit do stavu.

Představme si jednoduchou komponentu, kde uživatel zadá svůj věk podobně jako v příkladu výše.

```
const AgeField = () => (
  <label>
    Zadej svůj věk: <input type="text" />
  </label>
);
```

Hodnotu uvnitř elementu `input` si chceme uložit do stavu pokaždé, když dojde k její změně. Vytvoříme si proto stav `age` a budeme jej měnit v reakci na událost `onChange`.

```
const AgeField = () => {
  const [age, setAge] = useState('');

  return (
    <label>
      Zadej svůj věk:
      <input type="text" onChange={(e) => setAge(e.target.value)} />
    </label>
  );
};
```

Tímto postupem se snažíme provázat obsah políčka s hodnotou ve stavu. Kdykoliv uživatel obsah políčka změní, my na to zareagujeme změnou stavu `age`. Tomuto principu se anglicky říká data binding.

Pozor však, že náš data binding zatím funguje pouze jedním směrem, tedy *změna políčka ⟶ změna stavu*. Pokud se z nějakého důvodu změní hodnota ve stavu `age`, obsah políčka se zatím neaktualizuje.

## Obousměrný data binding

V praxi téměř vždy budeme chtít takzvaný two-way (obousměrný) data binding. To zařídíme jednoduše tak, že hodnotu ve stavu vždy nastavíme jako hodnotu políčka.

```
const AgeField = () => {
  const [age, setAge] = useState('');

  return (
```

```
      <label>
        Zadej svůj věk:
        <input type="text" value={age} onChange={(e) => setAge(e.target.val
      </label>
    );
};
```

Takto zajistíme provázanost i druhým směrem, tedy *změna stavu → změna políčka*. Nyní, když změníme obsah políčka, změní se nám stav. A když naopak změníme stav, změní se nám obsah políčka.

Následující
**Cvičení: Formulářové prvky**