Kódím.cz

/ Webové aplikace / Digitální akademie Web / React

11 Async/await a ošetření chyb

Ošetření chybových stavů API a použití async/await místo then()

Předchozí
Opakování: LeviExpress, druhá část

Nemá následující lekci

Ošetření chyb

Než se začneme zabývat ošetřením chyb, podívejme se naposledy na krásný přehledný kód, který předpokládá, že svět je ideální a k žádné chybě nemůže dojít:

```
const fetchData = () => {
  fetch('https://random.kodim.app/api/diceroll')
    .then((resp) => resp.json())
    .then((data) => setRoll(data.result.number));
};
```

Stavový kód, který nám vrátil server, nám prohlížeč ukládá do objektu `Response`. To je ten objekt, který dostaneme jako parametr naší funkce, kterou předáváme prvnímu `then()` po volání `fetch()`. Konkrétně číselný stav je uložen v property `status`. Pokud si v naší funkci zpracovávající odpověď pojmenujeme parametr s odpovědí (Response) jako `resp`, jak jsme si v tomto kurzu zvykli, stavový kód načteme pomocí `resp.status`.

Nyní se musíme dle stavového kódu rozhodnout, zda vše dopadlo dobře a zpracujeme data z odpovědi, nebo se něco nepovedlo a chceme uživateli zobrazit nějakou hezkou chybovou zprávu pomocí stavu `errorMessage`. Tentokrát použijeme endpoint s příznakem `?act=shaky`, který není tak sebejistý a přiznává, že se občas něco nepovede a vrátí serverovou chybu.

```
const fetchData = () => {
  fetch('https://random.kodim.app/api/diceroll?act=shaky')
```

```
    .then((resp) => {
      if (resp.status === 200) {
        setErrorMessage(null);
        return resp.json();
      }

      if (resp.status === 500) {
        setErrorMessage('Server vrátil chybu.');
      }
    })
    .then((data) => {
      if (data !== undefined) {
        setRoll(data.result.number);
      }
    });
};
```

Nezapomeňte na to, že v této první funkci obvykle říkáme, že se z odpovědi mají vzít data a zpracovat jako JSON, tj. převést z JSONu na JavaScriptový objekt. Tedy obvykle z funkce vracíme výsledek volání `resp.json()`. Tohle v naší funkci zůstalo v podobě `return resp.json()` ve větvi zpracovávající stavový kód `200`.

Nafoukla se i funkce v druhém `then`. Tam je potřeba zkontrolovat, že vůbec nějaká data máme, protože pokud nastala chyba, funkce v předchozím `then` nevrátí nic (přesněji, vrátí `undefined`). V takovém případě neděláme nic, protože stejně nemáme co zobrazit.

Ošetření fatálních chyb

Chyby, které nám posílá server, máme (aspoň trochu) ošetřené. Nyní je na čase postarat se o případy, kdy se komunikace se serverem vůbec nezdaří, např. protože náš uživatel cestuje amazonským pralesem mimo mobilní signál nebo prostě zakopne o síťový kabel.

Výpadky internetu nemusí být jednoduché otestovat. Abychom nemuseli vytahovat z počítače síťový kabel nebo balit notebook do alobalu kvůli WiFi, můžeme v prohlížeči v Dev Tools zajít na kartu *Network* kde v rozbalovací nabídce pravděpodobně máte vybráno *No throttling*. V této nabídce můžete omezovat rychlost připojení prohlížeče, abyste si tak nasimulovali pomalé připojení k internetu. A je tam také volba *Offline*, která způsobí, že se (jen) prohlížeč od internetu úplně odpojí. Pozor na to, že se odpojí celý prohlížeč a nebude fungovat ani `localhost` – tj. svou aplikaci si musíte načíst, když máte internet v prohlížeči

zapnutý, pak přepnete mód na *Offline*, zkusíte volání API, zjistíte, že máte něco špatně, opravíte to, zapnete internet, obnovíte stránku, přepnete do *Offline* a tak pořád dokola.

Pro ošetření fatálních chyb slouží funkce `catch()`, která se často píše na konec za všechny `then()`. Funkce uvnitř `catch()` se zavolá ve chvíli, kdy nastala fatální chyba. Všechny funkce uvnitř `then()` se při fatální chybě přeskočí.

```
const fetchData = () => {
  fetch('https://random.kodim.app/api/diceroll?act=shaky')
    .then((resp) => {
      if (resp.status === 200) {
        setErrorMessage(null);
        return resp.json();
      }

      if (resp.status === 500) {
        setErrorMessage('Server vrátil chybu.');
      }

      if (resp.status === 503) {
        setErrorMessage('Server je přetížen.');
      }
    })
    .then((data) => {
      if (data !== undefined) {
        setRoll(data.result.number);
      }
    })
    .catch((error) => {
      console.error('Chyba komunikace se serverem:', error.message);
      setErrorMessage('Chyba komunikace se serverem');
    });
};
```

Předchozí
Stavové kódy HTTP odpovědi

Následující
Cvičení: Ošetření chyb

Kódím.cz

Verze 2.0.0-beta.7