11 Async/await a ošetření chyb

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

Předchozí
Opakování: LeviExpress, druhá část

Nemá následující lekci

Bonus: Paralelní zpracování více požadavků

Klíčová slova `async/await` umožňují psát kód, který se lépe čte. Důležité je ale uvědomovat si, co se pod tímto kódem skrývá a že ve skutečnosti si to prohlížeč sám převede na volání funkce `then()`. Existují však případy, kdy `await` nechceme použít a budeme potřebovat pracovat přímo s `Promise` objekty. Je to v situacích, když chceme pracovat s více `Promise` objekty najednou.

Pokud bychom chtěli poslat dva nezávislé požadavky na server a udělali bychom to pomocí následujícího kódu, bude se nejprve čekat na dokončení prvního požadavku a teprve pak se pošle druhý požadavek. Představme si, že nechceme hodit jednou kostkou, ale chceme hodit dvěma kostkami najednou – a abychom je lépe rozlišili, jedno bude klasická šestiboká kostka, druhé bude dvanáctistěn.

```
const nacistData = async () => {
  const resp6 = await fetch('https://random.zkusmo.eu/random-6');
  const data6 = await resp6.json();
  setKostka1(data6.randomNumber);

  const resp12 = await fetch('https://random.zkusmo.eu/random-12');
  const data12 = await resp12.json();
  setKostka2(data12.randomNumber);
};
```

Toto čekání je ale zbytečné a zbytečně zpomaluje aplikaci – obou výsledků se dočkáme později, než by bylo nutné. Lepší by bylo poslat oba požadavky najednou a počkat, než budou oba dva splněné. Mohli bychom kód přepsat zpátky pomocí

`then()`, ale to by nám umožnilo reagovat na dokončení každého požadavku zvlášť, ale nemohli bychom čekat na to, až budou splněné oba dva požadavky (resp. museli bychom při skončení jednoho požadavku kontrolovat, zda už nebyl dokončet i ten druhý, a opačně).

Naštěstí existuje způsob, jak zpracovat požadavky paralelně a čekat na dokončení všech požadavků najednou. Tímto způsobem je použití statické metody `Promise.all()`. `Promise.all()` přijímá pole `Promise` objektů a vrací novou `Promise`, která se splní až když se splní všechny původní promisy.

Příklad použití:

```
const promises = [
  fetch('https://random.zkusmo.eu/random-6'),
  fetch('https://random.zkusmo.eu/random-12'),
];
const [resp6, resp12] = await Promise.all(promises);
const data6 = await resp6.json();
const data12 = await resp12.json();
setKostka1(data6.randomNumber);
setKostka2(data12.randomNumber);
```

`Promise` má i další statické metody, které umožňují pracovat s více `Promise` dalšími způsoby – např. počkat na první splněnou `Promise` (např. pokud pošlete požadavky na předpověď počasí na několik různých serverů a zobrazíte první odpověď, která dorazí).

Předchozí
Použití try-catch pro ošetření fatálních chyb

Verze 2.0.0-beta.7