3 Zobrazování seznamů

Ukážeme si, jak v React komponentách zobrazovat seznamy.

Předchozí
Podmíněné zobrazení

Následující
Události, stav

Seznamy v JSX

Na konci předchozí části jsme se dotknuli způsobu, jak v Reactu zobrazit nějaký seznam. Celá myšlenka tkví v tom, že React nám velmi přimočaře umožňuje zobrazit pole JSX elementů.

Představme si následující pole obsahující `li` elementy.

```
const dayElements = [
  <li>pondělí</li>,
  <li>úterý</li>,
  <li>středa</li>,
  <li>čtvrtek</li>,
  <li>pátek</li>,
];
```

Všimněte si, že jde o normální JavaScriptové pole, které jako svoje prvky obsahuje JSX elementy. Každý JSX element je hodnota, není tedy žádný problém mít pole takovýchto hodnot.

Pokud takové pole máme už připravené, můžeme elementy v něm obsažené zobrazit uvnitř nějakého rodiče prostě tak, že proměnnou `dayElements` do tohoto rodiče vložíme.

```
const App = () => (
  <>
    <h1>Pracovní dny</h1>
```

```
    <ol className="days">{dayElements}</ol>
  </>
);
```

Kdyby byl v proměnné `dayElements` uložen řetězec, budeme mít uprostřed `ol` seznamu prostě kousek textu. Jelikož však máme v `dayElements` pole JSX elementů, React je jednoduše zapojí jako děti našeho seznamu.

Když tento kód spustíme, React nám do konzole prohlížeče vypíše varování.

```
Warning: Each child in a list should have a unique "key" prop.
```

Pro tutu chvíli jej můžeme ignorovat. Později si vysvětlíme, co přesně znamená a jak se k němu postavit.

Použití map

V předchozím případě jsme pole JSX elementů měli připravené dopředu. V praxi jej však chceme vyrobit z nějakých dat. Máme například názvy dní v týdnu jako řetězce v poli.

```
const days = ['pondělí', 'úterý', 'středa', 'čtvrtek', 'pátek'];
```

Z tohoto pole chceme vyrobit pole JSX elementů pro náš seznam. K tomu nám stačí použít funkci `map`, kterou jsme v první části lekce tak poctivě trénovali.

```
const days = ['pondělí', 'úterý', 'středa', 'čtvrtek', 'pátek'];

const dayElements = days.map((day) => <li>{day}</li>);

const App = () => (
  <>
    <h1>Pracovní dny</h1>
    <ol className="days">{dayElements}</ol>
  </>
);
```

Tento kód bude hezky fungovat. Z hlediska profesionáních vývojářů je však zbytečně ukecaný. Proměnnou `dayElements` používáme pouze jednou, takže můžeme na místě jejího použití rovnou zavolat náš mapovací kód.

```
const days = ['pondělí', 'úterý', 'středa', 'čtvrtek', 'pátek'];

const App = () => (
  <>
    <h1>Pracovní dny</h1>
    <ol className="days">
      {
        days.map((day) => <li>{day}</li>)
      }
    </ol>
  </>
);
```

Pokud vám kód výše stále přijde srozumitelný, je zde příležitost jej udělat ještě malinko kompaktnější. V praxi se často setkáte s takovýmo formátováním.

```
const days = ['pondělí', 'úterý', 'středa', 'čtvrtek', 'pátek'];

const App = () => (
  <>
    <h1>Pracovní dny</h1>
    <ol className="days">
      {days.map((day) => (
        <li>{day}</li>
      ))}
    </ol>
  </>
);
```

Tady už je potřeba si dát zatraceně dobrý pozor na jednotlivé závorky a dobře se orientovat v tom, co která znamená.

Předchozí Následujicí

Cvičení: Zkracování a map

Složitější komponenty

Kódím.cz

Verze 2.0.0-beta.7