

React pre začiatočníkov

IV. Zoznamy

Ing. Slavomír Kožár

3. 4. 2022



Obsah

- ToDo projekt
 - Form
 - Renderovanie poľa komponentov
 - .map()
 - keys
 - immutable state
 - .filter()
 - .concat()
-
- ToDo tasks

ToDo Projekt

<https://github.com/slavokozaar/react-pre-zaciatocnikov-todo-project/>

```
$ git clone https://github.com/slavokozaar/react-pre-zaciatocnikov-todo-project.git
```

```
$ cd react-pre-zaciatocnikov-todo-project
```

```
$ npm install
```

```
$ npm start
```

ToDo App



To Do:

1. formulár
2. vytvorenie nového tasku

Awesome Todo list

Add

all
active
completed

<input type="checkbox"/>	Completed Task	
<input type="checkbox"/>	Todo Task	

© 2022

Form

```
function handleSubmit (e) {  
    e.preventDefault();  
}
```

vypnutie default funkcie formulára

```
<form  
    onSubmit = {handleSubmit}    udalosť vytvorená pri odoslaní formulára  
>  
    <input ... />  
    <button type="submit">Add</button>  
</form>
```

Pole (Array) komponent

```
const array = [  
    <p>hello</p>,  
    <p>world</p>  
];
```

```
return (  
    <div>  
        {array}  
    </div>  
);
```

Pole (Array) komponent

Staticky zadefinované pole komponent nedáva zmysel...

```
return (  
    <div>  
        <p>hello</p>  
        <p>world</p>  
    </div>  
);
```

Pole (Array) komponent

Bežne máme pole dát, ktoré potrebujeme konvertovať na pole komponent....

```
const array = [  
  'hello',  
  'world'  
];
```

```
const array = [  
  <p>hello</p>,  
  <p>world</p>  
];
```


Array.prototype.map()

Metóda **map()** vytvorí **nové pole** vyplnené **výsledkami** volania poskytnutej funkcie na **každom prvku** vo volajúcom poli.

```
map(callbackFn)
```

```
callbackFn(element) { /* ... */ }
```

```
callbackFn(element, index) { /* ... */ }
```

```
callbackFn(element, index, array) { /* ... */ }
```

Array.prototype.map()

V praxi sa často používajú arrow funkcie...

```
.map( (element) => ( /* ... */ ) )
```

```
.map( (element, index) => ( /* ... */ ) )
```

```
.map( (element, index, array) => ( /* ... */ ) )
```

Array.prototype.map()

Metóda **map()** vytvorí **nové pole** vyplnené **výsledkami** volania poskytnutej funkcie na **každom prvku** vo volajúcom poli.

```
const array = [1, 4, 9, 16];
```

```
const map1 = array.map(x => x * 2);
```

```
console.log(map1);
```

[2, 8, 18, 32]

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

Pole komponent

```
const array = ['hello', 'world'];

return (
  <div>
    {
      array.map((element) => (
        <p>{element}</p>
      ))
    }
  </div>
);
```

ToDo - tasky

Zobrazenie taskov v ToDo aplikácii z dátovej štruktúry:

```
[  
  {  
    "text": "Nainštaluj si Node.js",  
    "active": true  
  }, {  
    "text": "Naštuduj prezentácie z Kurz pre začiatočníkov",  
    "active": true  
  }  
  ...  
]
```

Key(s)

Prvky vo vnútri poľa by mali mať **unikátne kľúče**.

Kľúče pomáhajú React-u identifikovať, **ktoré položky** sa **zmenili**, boli **pridané** alebo **odstránené**...

```
const array = ['hello', 'world'];  
...  
{  
  array.map((element, index) => (  
    <p key={index}>{element}</p>  
  ))  
}
```

Pole v state

```
const [array, setArray] = useState([1, 2, 3, 4]);
```

```
<button  
  onClick={() => {  
    array.push(5) ;  
    setArray(array) ;  
  }}  
>
```

Pole v state - demo

<https://github.com/slavokozar/react-pre-zaciatocnikov-array-demo>

Immutable

```
let a = [1, 2, 3];  
console.log(a);
```

[1, 2, 3]

```
let a = b;
```

```
b.push(4);
```

```
console.log(b);
```

[1, 2, 3, 4]

```
console.log( a === b )
```

true

Immutable

React spustí **render** iba v prípade, že sa hodnota **state zmení**.

Ak v state potrebujeme uložiť zloženú štruktúru (pole, objekt) musíme s týmito dátami pracovať v tzv. immutable režime.

Neupravujeme pôvodnú dátovú štruktúru, ale vytvárame kópiu so zmenou.

Immutable

```
let a = [1, 2, 3];  
console.log(a);
```

[1, 2, 3]

```
let b = Array.from(a);  
b.push(4);  
console.log(b);
```

[1, 2, 3, 4]

```
console.log( a === b )
```

false

Array.prototype.concat()

Metóda **concat()** sa používa na **zlúčenie** dvoch alebo viacerých **polí**. Táto metóda nezmení existujúce polia, ale namiesto toho **vytvorí nové pole**.

```
const array1 = ['a', 'b'];  
const array2 = ['c', 'd'];  
const array3 = array1.concat(array2) ;
```

```
console.log(array3) ;           ['a', 'b', 'c', 'd']
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/concat

ToDo App

- pridanie nového tasku pomocou funkcie `.concat()`
- nový task pridaný v momente odoslania formulára

Array.prototype.filter()

Metóda **filter()** vytvorí **nové pole** so všetkými **prvkami**, ktoré prejdú **testom** implementovaným poskytnutou **funkciou**.

```
const words = ['a', 'bb', 'ccc', 'dddd', 'eeee', 'ffffff'];
```

```
const result = words.filter((word) => ( word.length > 3 ));
```

```
console.log(result);           ['dddd', 'eeee', 'ffffff']
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter

ToDo App

- vymazanie úlohy zo zoznamu taskov
- task vymazaný kliknutím na tlačidlo


```
<button  
  type="button"  
  className="btn btn-sm btn-link text-danger"  
>  
  <svg></svg>  
</button>
```

map, filter, reduce










```
[🐮, 🥔, 🐔, 🌽].map(cook) ⇒ [🍔, 🍟, 🍗, 🍿]  
[🍔, 🍟, 🍗, 🍿].filter(isVegetarian) ⇒ [🍟, 🍿]  
[🍔, 🍟, 🍗, 🍿].reduce(eat) ⇒ 💩
```


.map(=>) => 

.filter() => 

.every() => false

.some() => true

.fill(, 1) => 

.find() => 

.reduce((acc, cur)=>acc+cur)=> 

Congratulations!



homm.fun

meme-arsenal.ru

ĎAKUJEME !

Ing. Slavomír Kožár

slavomir.kozar@innovis.sk

www.innovis.sk

