Learn this before React

JS

01

Introduction

In this tutorial we will explore top fundamental Javascript concepts necessary to know in order to have an effective first learning cycle of **React js / React Native**

Table of contents

- map() & filter()
- slice() & splice()
- concat()
- find() & findIndex()
- destructuring
- rest & spread operator
- promises

02

map and filter

Both are array methods and both return a new array when applying. Filter additionally eliminates items that don't match

```
const DATA = [
  {id: 1, title: 'first'},
  {id: 2, title: 'second'},
  {id: 3, title: 'third'},
  {id: 4, title: 'fourth'},
]

const uppperData = DATA.map(el=> el.title.toUpperCase())
console.table(uppperData)
```

(index)	Value
0	'FIRST'
1	'SECOND'
2	'THIRD'
3	'FOURTH'

```
const moduloData = DATA.filter(el => el.id % 2 === 0)
console.table(moduloData)
```

(index)	id	title
0	2	'second'
1	4	'fourth'

```
console.table(DATA)
```

(index)	id	title
0	1	'first'
1	2	'second'
2	3	'third'
3	4	'fourth'

03

slice and splice

method returns a new array with selected elements, while splice method changes the contents of an existing array

```
const charactersArr = [
  'Witcher',
  'Harry Potter',
  'Luke Sykwalker',
  'Tony Stark',
]

const copyArr = [...charactersArr]

copyArr.splice(0, 1);
console.log(copyArr)
```

```
['Harry Potter', 'Luke Sykwalker', 'Tony Stark']
```

```
copyArr.splice(copyArr.length, 1, 'Wonder Woman');
console.log(copyArr)
```

```
['Harry Potter', 'Luke Sykwalker', 'Tony Stark', 'Wonder Woman']
```

```
const selected = charactersArr.slice(0,2)
console.log(selected)
```

```
['Witcher', 'Harry Potter']
```

```
console.log(charactersArr)
```

```
['Witcher', 'Harry Potter', 'Luke Sykwalker', 'Tony Stark']
```

04

concat

This method returns a new array of merging two or more arrays.

```
const arr1 = [1, 2, 3, 4]
const arr2 = [10, 20, 30, 40]
const arr3 = [100, 200, 300, 400]

const mergedArr = arr1.concat(arr2, arr3)

console.log(mergedArr)
```

```
[1, 2, 3, 4, 10, 20, 30, 40, 100, 200, 300, 400]
```

05

find and findIndex

The find method returns the first element that satisfies the condition, while findIndex returns the index of that element

```
const DATA = [
  {id: 1, title: 'first'},
  {id: 2, title: 'second'},
  {id: 3, title: 'third'},
  {id: 4, title: 'fourth'},
]

const itemIdx = DATA.findIndex(el=> el.id === 2)
console.log(itemIdx)
```

```
1
```

```
const item = DATA.find(el=> el.id === 2)
console.log(item)
```

```
▶ {id: 2, title: 'second'}
```

06

destructuring

The destructuring assignment is a special syntax which enables unpacking (assigning) values from arrays or object properties directly into variables

```
const name = ['Luke', 'Skywalker']

const [firstName, lastName] = name

console.log(firstName, lastName)
```

```
Luke Skywalker
```

```
const jedi = {
  id: 1,
  name: 'Luke Skywalker',
  lightsaber: true,
  species: 'Human'
}

const {name:jediName, species, ...rest} = jedi
console.log(jediName)
console.group(species)
```

```
Luke Skywalker
Human
```

```
console.log(rest)
```

```
▶ {id: 1, lightsaber: true}
```

07

rest & spread operator

Rest parameter enables us to pass unspecified number of parameters to a function which will be placed into array, while the **spread operator** enables us to spread the content of a iterable (i.e. array) into individual elements

```
// SPREAD

const introduction = ['my', 'name', 'is', 'Luke', 'Skywalker']
const copyArr = [...introduction]
console.log(copyArr)
console.log(...copyArr)
```

```
▶ (5) ['my', 'name', 'is', 'Luke', 'Skywalker']
my name is Luke Skywalker
```

```
// REST

const getSize = (...args) => {
  return args.length
}

console.log(getSize(1, 5, 10))
console.log(getSize(10, 20, 40, 50, 60))
```

```
3
5
```

08

promises

In simple terms promises are used to handle asynchronous operations. Each promise can end as a success or failure having 3 possible statuses: pending, fulfilled or rejected. In the example below we handle promises with the async await syntaxt while fetching data from API

```
const fetchData = async() => {
  try {
    const response = await fetch('https://swapi.dev/api/people/');

    if (!response.ok) throw new Error(response.status);

    const result = await response.json();
    return result;
  }
  catch (e) {
    console.log(e)
  }
}
```