Member-only story

10 Javascript Exercises with Objects

Andrei Borisov · Follow

6 min read · May 14, 2020

Continuing the idea of ten exercises for arrays, I made a collection of tasks for objects. Like the previous one, this collection is oriented for junior and middle javascript developers.

For every task, I will provide a description, expected result, and solution. I do not claim every solution is the best approach to solve the exercises, but I hope it eventually would help you to improve your skills.

Also, I would like to mention that I won't handle all the error cases, like passing undefined, null, or wrong data types, I provide a basic solution, not writing a library for production.

You can start in your repository or clone mine. In that repository, you can find a full list of exercises and solutions. Also, you can easily check your solutions with pre-created tests. Link: https://github.com/andrewborisov/javascript-practice.

1. **isPlainObject** - Write a method that verifies an argument is a plain object, not an array or null

```
/**
 * Task description: Write a method that verifies an argument is a
plain object, not an array or null
 * Expected Result: True if object is plain, false otherwise.
     ({ a: 1 }) => true,
     ([1, 2, 3]) => false
 * Task complexity: 1 of 5
 * @param element - element to verify
 * @returns {boolean}
 */
export const isPlainObject = (element) => {
  throw new Error(`put your solution here ${element}`);
};

const data = { a: 1 };

console.log(isPlainObject(data)); // true
```

"IsPlainObject" function solution.

2. **MakePairs** - Write a method that returns a deep array like [[key, value]]

```
/**
  * Task description: Write a method that returns a deep array like
[[key, value]]
  * Expected Result: ({ a: 1, b: 2 }) => [['a', 1], ['b', 2]]
  * Task complexity: 1 of 5
  * @param {Object} object - Any object to transform into array
  * @returns {Array} - a deep array
*/
export const makePairs = (object) => {
```

```
  throw new Error(`put your solution here ${object}`);
};

const data = { a: 1, b: 2 };

console.log(makePairs(data)); // [['a', 1], ['b', 2]]
```

“MakePairs” function solution.

3. **Without** - Write a method that returns a new object without provided properties

```
/**
  * Task description: Write a method that returns new object without
provided properties
  * Expected Result: ({ a: 1, b: 2 }, 'b') => { a: 1 }
  * Task complexity: 2 of 5
  * @param {Object} object - Any object
  * @param {?} args - list of properties to remove from object
  * @returns {Object} - New object without listed values
*/
export const without = (object, ...args) => {
  throw new Error(`put your solution here ${object} ${args}`);
};

const data = { a: 1, b: 2 };

console.log(without(data, 'b')); // { a: 1 }
```

“Without” function solution.

4. **IsEmpty** - Write a method that makes a shallow check is object empty

```
/**
  * Task description: Write a method that makes a shallow check is
object empty
  * Expected Result: ({}) => true, ({ a: undefined }) => true,
      ({ a: 1 }) => false
  * Empty values: '', null, NaN, undefined
  * Task complexity: 2 of 5
  * @param {Object} object - Object with values of primitive data
types
  * @returns {boolean}
*/
export const isEmpty = (object) => {
```

```
  throw new Error(`put your solution here ${object}`);
};

const data = { a: 1, b: undefined };
const data2 = { a: undefined };

console.log(isEmpty(data)); // false
console.log(isEmpty(data2)); // true
```

“IsEmpty” function solution.

5. **IsEqual** - Write a method that makes a shallow compare of two objects

```
/**
  * Task description: Write a method that makes a shallow compare of
two objects
  * Expected Result: True if objects are identical, false if objects
are different ({ a: 1, b: 1 }, { a: 1, b: 1 }) => true
  * Task complexity: 2 of 5
  * @param {Object<string | number>} firstObj - Object with values of
primitive data types
  * @param {Object<string | number>} secondObj - Object with values of
primitive data types
  * @returns {boolean}
*/
export const isEqual = (firstObject, secondObject) => {
  throw new Error(`put your solution here ${firstObject}
${secondObject}`);
};

const data = { a: 1, b: 1 };
const data2 = { a: 1, b: 1 };
const data3 = { a: 1, b: 2 };

console.log(isEqual(data, data2)); // true
console.log(isEqual(data, data3)); // false
```

“IsEqual” function solution.

6. **Invoke** - Write a method that invokes an array method on a specific path

```
/**
  * Task description: Write a method that invokes an array method on a
specific path
```

```
  * Expected Result: ({ a: { b: [1, 2, 3] } }, 'a.b', splice, [1, 2])
=> [2, 3]
  * Task complexity: 3 of 5
  * @param {Object} object
  * @param {String} path - path in an object to property
  * @param {String} func - function to invoke
  * @param {Array} [args] - list of args
  * @returns {?}
*/
export const invoke = (object, path, func, args) => {
  throw new Error(`put your solution here ${object} ${path} ${func}
${args}`);
};

const data = { a: { b: [1, 2, 3] } }

console.log(invoke(data, 'a.b', 'splice', [1, 2])); // [2, 3]
```

“Invoke” function solution.

7. **IsEmptyDeep** - Write a method that makes a deep check is an object empty

```
/**
  * Task description: Write a method that makes a deep check is an
object empty
  * Empty values: '', null, NaN, undefined, [], {}
  * Expected Result: ({}) => true,
      ({ a: { b: undefined } }) => true,
      ({ a: { b: [] } }) => true
  * Task complexity: 3 of 5
  * @param {?} element - Object with values of any data types
  * @returns {boolean}
*/
export const isEmptyDeep = (element) => {
  throw new Error(`put your solution here ${element}`);
};

const data = { a: { b: undefined } };

console.log(isEmptyDeep(data)); // true
```

“IsEmptyDeep” function solution.

8. **IsEqualDeep** - Write a method that makes a deep compare of two objects

```
/**
  * Task description: Write a method that makes a deep compare of two
objects
  * Expected Result: True if objects are equal, false if objects are
different ({ a: 1, b: { c: 1 } }, { a: 1, b: { c: 1 } }) => true
  * @param {Object} firstObj - Object of any values
  * @param {Object} secondObj - Object of any values
  * @returns {boolean}
*/
export const isEqualDeep = (element) => {
  throw new Error(`put your solution here ${element}`);
};

const data = { a: 1, b: { c: 1 } };
const data2 = { a: 1, b: { c: 1 } };
const data3 = { a: 1, b: { c: 2 } };

console.log(isEqualDeep(data, data2)); // true
console.log(isEqualDeep(data, data3)); // false
```

“IsEqualDeep” function solution.

9. **Intersection** - Write a method that finds shallow intersections of objects

```
/**
  * Task description: Write a method that finds shallow intersections
of objects
  * Expected Result: ({ a: 1, b: 2 }, { c: 1, b: 2 }) => { b: 2 }
  * @param {Object<string | number>} firstObj - Object with values of
primitive data types
  * @param {Object<string | number>} secondObj - Object with values of
primitive data types
  * @returns {Object}
*/
export const intersection = (firstObject, secondObject) => {
  throw new Error(`put your solution here ${firstObject},
${secondObject}`);
};

const data = { a: 1, b: 2 };
const data2 = { c: 1, b: 2 };

console.log(intersection(data, data2)); // { b: 2 }
```

“Intersection” function solution.

10. **IntersectionDeep** - Write a method that finds all intersections of objects

```
/**
  * Task description: Write a method that finds all intersections of
objects
  * Expected Result: ({ a: 1, b: { c: 3 } }, { c: 1, b: { c: 3 } }) =>
{ b: { c: 3 } }
  * @param {Object} firstObj - Object with values of any data types
  * @param {Object} secondObj - Object with values of any data types
  * @returns {Object}
*/
export const intersectionDeep = (firstObject, secondObject) => {
  throw new Error(`put your solution here ${firstObject},
${secondObject}`);
};

const data = { a: 1, b: { c: 3 } };
const data2 = { c: 1, b: { c: 3 } };

console.log(intersectionDeep(data, data2)); // { b: { c: 3 } }
```

“IntersectionDeep” function solution.

I hope the tasks mentioned above could help you to learn something new or just have some fun.

JavaScript | Coding | Javascript Object | Junior Developer | Coding Exercise

Senior Software engineer at EPAM

More from Andrei Borisov

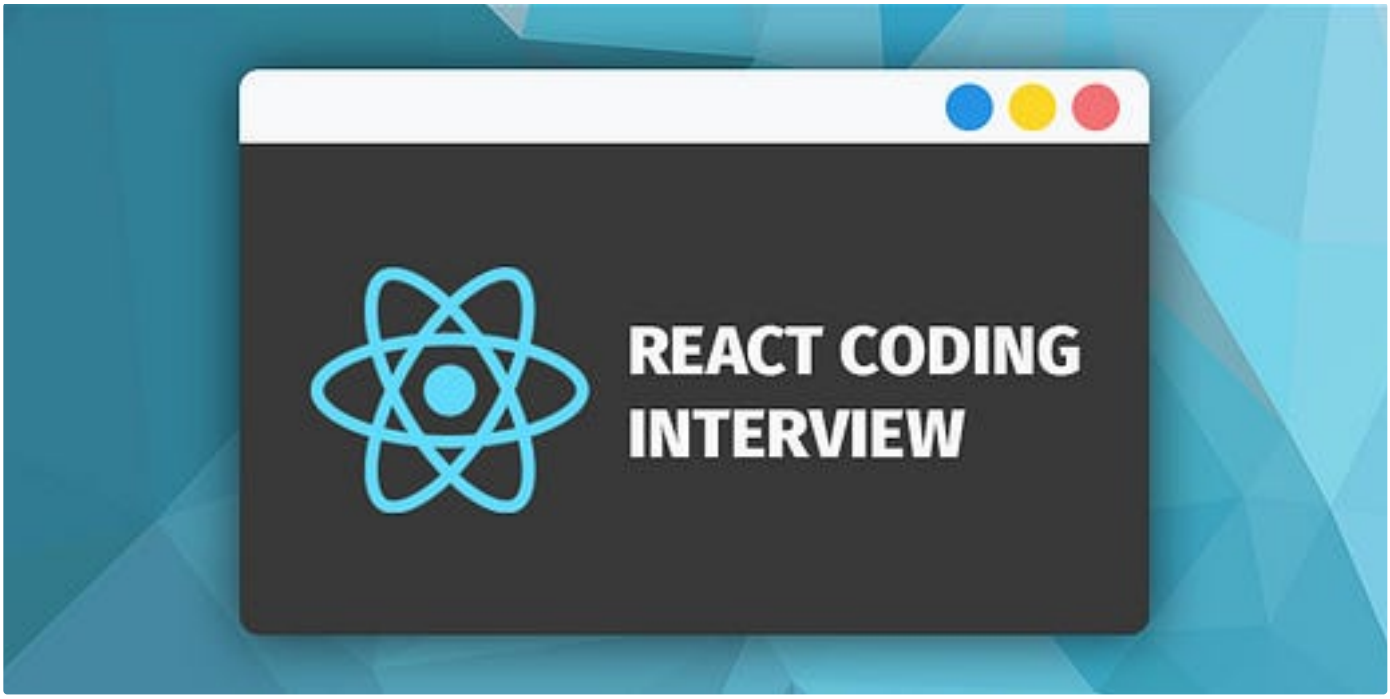
Andrei Borisov

10 Javascript Exercises with Arrays

Basically, this article could be useful for junior and middle Javascript software engineers. I prepared ten exercises with arrays, which I...

· 6 min read · Apr 30, 2020

396



Andrei Borisov in Geek Culture

React coding interview task

In this article, I would like to share a task for a junior or middle front-end developer on ReactJS.

· 7 min read · Aug 2, 2021

Andrei Borisov

Создание монорепозиториев на NX

Создание и настройка монорепозитория при использовании библиотеки Nx. Создание React приложения на Nx. Разработка кастомных схем через Nx.

· 6 min read · Apr 23, 2020

4

Andrei Borisov

Что нужно знать front-end разработчику, чтобы переехать в США?

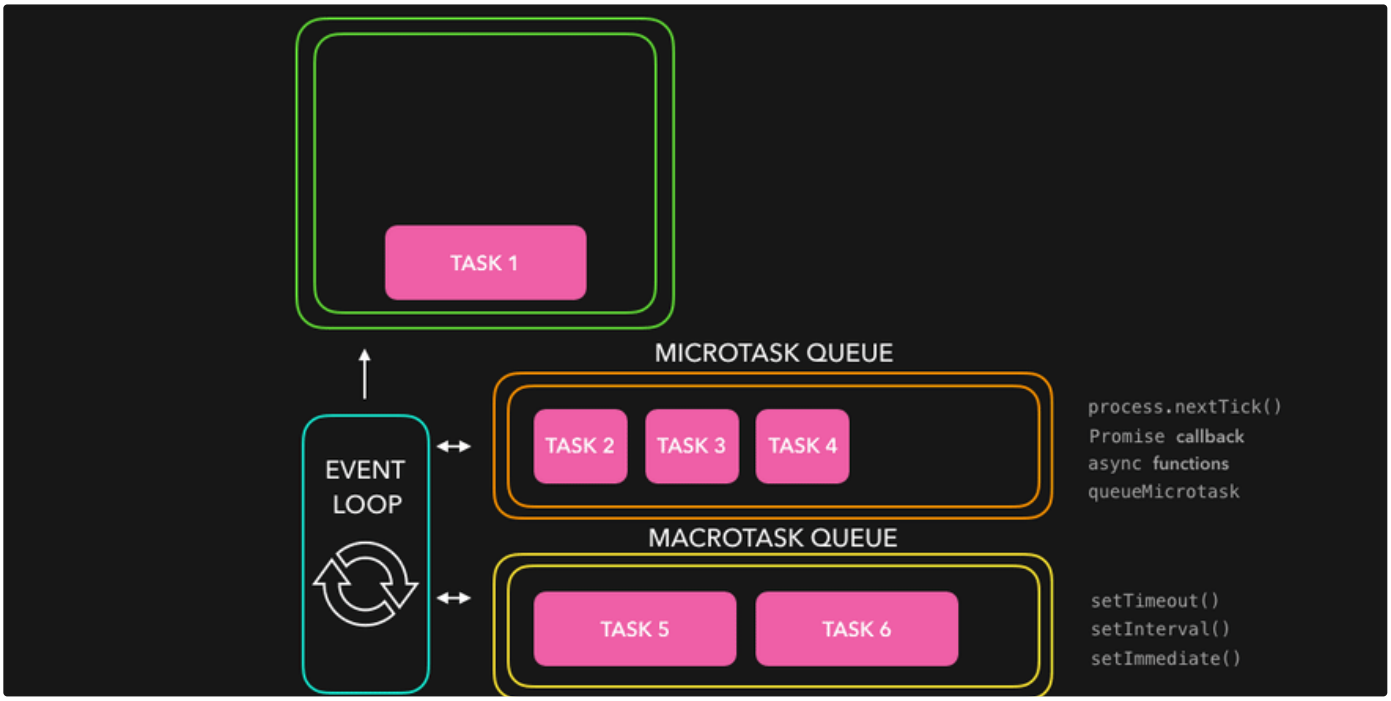
В это статье я расскажу о навыках, которые нужны javascript разработчику для переезда в США.

· 4 min read · Jul 13, 2020

53

See all from Andrei Borisov

Recommended from Medium



Jeswanth Reddy in Version 1

Difference Between Promise and Async/Await

If you're reading this, you probably understand how the promise and async/await are different in the execution context.

2 min read · May 12

196 2

Avinash Kumar

Slice and Splice in JavaScript?

slice()-: Slice is used to get a new array by selecting a sub-array from a given array and does not change the original array.

2 min read · Aug 10

3

Lists

Stories to Help You Grow as a Software Developer

19 stories · 295 saves

General Coding Knowledge

20 stories · 235 saves

The New Chatbots: ChatGPT, Bard, and Beyond

13 stories · 91 saves

Generative AI Recommended Reading

52 stories · 170 saves

Cihan in Interesting Coding

Advanced One-Liner Codes in JavaScript in Detail

Discover the power of concise and efficient code with comprehensive guide to advanced one-liner codes in JavaScript.

· 10 min read · Mar 17

Rabi Siddique in Level Up Coding

💻 15 JavaScript Techniques to Enhance Your Code

As a popular programming language for web development, JavaScript offers a lot of features and functionalities. Whether you're a seasoned...

7 min read · Mar 14

Emma Delaney

Top 10 tricky JavaScript questions that I used to ask in interviews

Some tricky interview scheduling questions you will face. These questions seem easy, but there is something fishy about them. So today I'm...

4 min read · May 24

Abidullah

Mastering JavaScript Shorthand: Array Spread and Object Destructuring

Discover the power of JavaScript shorthand with array spread and object destructuring. Simplify code, boost productivity, and write elegant...

2 min read · Aug 9

8

See more recommendations