Object.keys, values, entries

Let's step away from the individual data structures and talk about the iterations over them.

In the previous chapter we saw methods `map.keys()`, `map.values()`, `map.entries()`.

These methods are generic, there is a common agreement to use them for data structures. If we ever create a data structure of our own, we should implement them too.

They are supported for:

- `Map`
- `Set`
- `Array` (except `arr.values()`)

Plain objects also support similar methods, but the syntax is a bit different.

Object.keys, values, entries

For plain objects, the following methods are available:

- Object.keys(obj) – returns an array of keys.
- Object.values(obj) – returns an array of values.
- Object.entries(obj) – returns an array of `[key, value]` pairs.

...But please note the distinctions (compared to map for example):

	Map	Object
Call syntax	`map.keys()`	`Object.keys(obj)`, but not `obj.keys()`
Returns	iterable	"real" Array

The first difference is that we have to call `Object.keys(obj)`, and not `obj.keys()`.

Why so? The main reason is flexibility. Remember, objects are a base of all complex structures in JavaScript. So we may have an object of our own like `order` that implements its own `order.values()` method. And we still can call `Object.values(order)` on it.

The second difference is that `Object.*` methods return "real" array objects, not just an iterable. That's mainly for historical reasons.

For instance:

```
let user = {
  name: "John",
  age: 30
};
```

- `Object.keys(user) = ["name", "age"]`
- `Object.values(user) = ["John", 30]`
- `Object.entries(user) = [ ["name","John"], ["age",30] ]`

Here's an example of using `Object.values` to loop over property values:

```
let user = {
  name: "John",
  age: 30
};

// loop over values
for (let value of Object.values(user)) {
  alert(value); // John, then 30
}
```

Object.keys/values/entries ignore symbolic properties

Just like a `for..in` loop, these methods ignore properties that use `Symbol(...)` as keys.

Usually that's convenient. But if we want symbolic keys too, then there's a separate method Object.getOwnPropertySymbols that returns an array of only symbolic keys. Also, the method Reflect.ownKeys(obj) returns *all* keys.

Tasks

Sum the properties

importance: 5

There is a `salaries` object with arbitrary number of salaries.

Write the function `sumSalaries(salaries)` that returns the sum of all salaries using `Object.values` and the `for..of` loop.

If `salaries` is empty, then the result must be `0`.

For instance:

```
let salaries = {
  "John": 100,
  "Pete": 300,
  "Mary": 250
};

alert( sumSalaries(salaries) ); // 650
```

Open a sandbox with tests.

solution

```
function sumSalaries(salaries) {

  let sum = 0;
  for (let salary of Object.values(salaries)) {
    sum += salary;
  }

  return sum; // 650
}
```

Or, optionally, we could also get the sum using `Object.values` and `reduce`:

```
Object.values(salaries).reduce((a, b) => a + b) // 650
```

Open the solution with tests in a sandbox.

Count properties

importance: 5

Write a function `count(obj)` that returns the number of properties in the object:

```
let user = {
  name: 'John',
  age: 30
};

alert( count(user) ); // 2
```

Try to make the code as short as possible.

P.S. Ignore symbolic properties, count only "regular" ones.

Open a sandbox with tests.

solution

Open the solution with tests in a sandbox.

Previous lesson Next lesson

Share

Tutorial map

Comments

- You're welcome to post additions, questions to the articles and answers to them.
- To insert a few words of code, use the `<code>` tag, for several lines – use `<pre>`, for more than 10 lines – use a sandbox (plnkr, JSBin, codepen...)
- If you can't understand something in the article – please elaborate.

Anton A. Zabirko • a year ago

Flash cards for this lesson: https://quizlet.com/_3xx2dp

5 • Reply • Share ›

Andrew • 4 months ago

Why was it mentioned that values() method is not supported for array data type?

> Array (except arr.values())

I've tried it out and it works:

```
let myArr = ['a', 'b', 'c'];

for (let x of myArr.values()) {
  console.log(x);
}
```

2 • Reply • Share ›

Cristian • a month ago

//Sum the properties

let salaries = {
Cristian:100,
Leila:300,
Guada:380,
}

function sumSalaries(total, num) {
return total + num;
}

Object.values(salaries).reduce(sumSalaries, 0);

//Count properties

let user = {
name:"John",
age:30
}

Object.keys(user).length

Reply • Share ›

Santanu Bera • a month ago

array.values() works fine. It returns an Iterator.

Reply • Share ›

Shamil Mammadoff • 2 months ago

```
function sumSalaries(obj) {
let arr = Object.values(obj);

if(arr.length > 0) {
return arr.reduce((prev, cur) => prev + cur)
}

return 0;
}
```

Reply • Share ›

Nihal MacArth Agazade • 2 months ago

function sumSalaries(){
let salaries = {
"John": 100,
"Pete": 300,
"Mary": 250
};
return Object.values(salaries).reduce((a, b) => a + b);
}
alert(sumSalaries());

Reply • Share ›

JessieUni • 5 months ago

An optimization of the function **sumSalaries** using knowledge previously learnt:

```
function sumSalaries(salaries) {
  return Object.values(salaries).reduce((sum, currValue) => (sum + currValue), 0)
}
```

Reply • Share ›

JessieUni ➚ JessieUni • 5 months ago

Only after posting this did I saw that there are already plenty of comments using the same method. Everybody is doing a great job! :-)

2 Reply • Share ›

Victor Onlite • 6 months ago

task #1:

```
let sumSalaries = obj => Object.values(obj).reduce((acc, item)=> acc += item, 0);
```

Reply • Share ›

Ігор Царенко • 8 months ago

Here is my solution for second task:
let count = obj => {
return Object.keys(obj).length;
};

Reply • Share ›

Yasin • 9 months ago

```
function sumSalaries(salaries){
    let sum=0;
    for (let i of Object.values(salaries)){
        sum+=i}
```

```
    return (sum)}
```

Reply • Share ›

bo • 2 years ago

Why not use reduce on the array that is generated by Object.values(obj)?

Reply • Share ›

Ilya Kantor Mod ➚ bo • a year ago

You can! :)

1 Reply • Share ›

Avatar This comment was deleted.

Alexander Mesko ➚ Guest • 2 years ago

Probably just wants people to use Object methods to get a feel for them.

1 Reply • Share ›

Magar Hunter ➚ Guest • 10 months ago

because the purpose of the task is to make a solution from current lesson.

Reply • Share ›

Алексей Федорчук ➚ Guest • 2 years ago

Because in the example it is easier to read the code and we immediately see that the function returns.

Reply • Share ›

L0cutus • 23 days ago

Task#1

```
'use strict';

let salaries = {
"John": 100,
"Pete": 300,
"Mary": 250
};

function sumSalaries(salaries){

let sum = 0;
Object.values( salaries ).forEach( value => sum += value );

return sum;
}

alert( sumSalaries(salaries) ); // 650
```

Reply • Share ›

ALSO ON **JAVASCRIPT.INFO**

Promise

30 comments • 2 years ago

romul3003 — good article

Element size and scrolling

1 comment • 2 years ago

Jason Lee — I tried these occasions and found offsetParent is not null, and I also checked MDN and didn't find anything saying it should do so.

Currying and partials

14 comments • 2 years ago

nastya tikhomirova — This explanation for curring is the best I've seen!

JavaScript/DOM/Interfaces

7 comments • 8 months ago

kapalkat — Hey, I have a question regarding 'How we study' section, it is said that the course will last for 2 months, but how much time you