

3 JSowe mity

Filip Białek

Frontend dev @ VirtusLab

MIT: w JS występuje pass by reference

MIT: Typy proste
przekazywane przez wartość,
a obiekty przez referencję

```
2 const langs = ['Scala', 'JS', 'Python', 'Ruby'];
3
4 const getSortedArray = (arr) => {
5   return arr.sort();
6 };
7
8 const sortedLangs = getSortedArray(langs);
9
10 console.log(sortedLangs); // [ "JS", "Python", "Ruby", "Scala" ]
11
```

```
2 const langs = ['Scala', 'JS', 'Python', 'Ruby'];  
3  
4 const getSortedArray = (arr) => {  
5   return arr.sort();  
6 };  
7  
8 const sortedLangs = getSortedArray(langs);  
9  
10 console.log(sortedLangs); // [ "JS", "Python", "Ruby", "Scala" ]  
11  
12 console.log(langs);      // [ "JS", "Python", "Ruby", "Scala" ]
```

Przekazanie przez wartość:

- argument jest ewaluowany
- wynik ewaluacji przypisany jest do zmiennej lokalnej
- w rezultacie mamy dostęp do przekazanej wartości

Przekazanie przez referencję:

- do funkcji przekazana jest zmienna
- możemy zmodyfikować zmienną
- możemy przypisać do zmiennej nową wartość, przypisanie będzie widoczne poza ciałem funkcji

W JSie nie przekazujemy przez referencję, gdyż przypisanie nowej wartości nie będzie widoczne poza funkcją.


```
2 const ziutek = {
3   name: 'Ziutek',
4   age: 56,
5 };
6
7 const createCzesiek = (obj) => {
8   obj = {
9     name: 'Czesiek',
10    age: 67,
11  };
12  return obj;
13 };
14
15 const czesiek = createCzesiek(ziutek);
16
17 console.log(ziutek); // still ziutek
18 console.log(czesiek); // czesiek
```

W JSie mamy dwa typy zmiennych: proste i referencje

```
2 const ziutek1 = {  
3   name: 'Ziutek'  
4 };  
5  
6 const ziutek2 = {  
7   name: 'Ziutek'  
8 };  
9  
10 const ziutek3 = ziutek1;  
11  
12 console.log(ziutek1 === ziutek2); // false  
13  
14 console.log(ziutek1 === ziutek3); // true  
15
```

W JS przekazujemy przez
wartość, ale wartością jest
referencja.

W JS przekazujemy przez
wartość, ale wartością jest
referencja.

Pod zmienną i tak nigdy nie mamy
obiektu.

Przekazanie przez wartość
oznacza, że mamy do
czynienia z kopią referencji.

Call by sharing Barbara Liskov

We call the argument passing technique call by sharing, because the argument objects are shared between the caller and the called routine. [...] In particular it is not call by value because mutations of arguments performed by the called routine will be visible to the caller. And it is not call by reference because access is not given to the variables of the caller, but merely to certain objects.

Barbara Liskov - CLU manual

MIT: wszystko jest obiektem



in javascript everything is an object



[Wszystko](#)

[Grafika](#)

[Filmy](#)

[Wiadomości](#)

[Zakupy](#)

[Więcej](#)

[Ustawienia](#)

[Narzędzia](#)

Okolo 82 200 000 wyników (0,45 s)

Everything is an object in JavaScript | Go Make Things

<https://gomakethings.com/everything-is-an-object-in-javascript/> [Tłumaczenie strony](#)

16 lip 2018 - In JavaScript, everything is an object, even when it's something else. ... For example, `[1, 2, 3].length` works because arrays are also objects and `length` is a property of the Array object. `Function.arguments` is a property of the Function object. You can assign properties to basically anything in JS.

How is almost everything in Javascript an object? - Stack Overflow

<https://stackoverflow.com/questions/14822511/how-is-almost-everything-in-javascript-an-object/> [Tłumaczenie strony](#)

5 odpowiedzi

2 lut 2012 - That's right, in JavaScript, almost everything is an object. But these objects are bit different from what we see in Java, C++ or other conventional ...

Everything is an object?

4 odpowiedzi 3 paź 2013

How is a JavaScript string not an object?

2 odpowiedzi 19 maj 2019

Is 'Object' a function in JavaScript?

7 odpowiedzi 25 lut 2019

Everything in Javascript is a Function

2 odpowiedzi 8 lut 2013

[Więcej wyników z stackoverflow.com](#)

Is everything an object in Javascript? - Quora

<https://www.quora.com/Is-everything-an-object-in-Javascript> [Tłumaczenie strony](#)

5 odpowiedzi

No, but the things that aren't objects and can be worked upon (so number, string, boolean) will temporarily be wrapped as objects when read, which allows for a ...

Is Everything in JavaScript an Object! - CodeProject

<https://www.codeproject.com/Articles/1482251/Is-Everything-in-JavaScript-an-Object!> [Tłumaczenie strony](#)

23 wrz 2016 - This tip will try to explain why (almost) everything in JavaScript is an object.

What it really means when people say "Everything in ... - Radar

radar.oreilly.com/2014/05/what-it-really-means-when-people-say-everything-in-javascript-is-an-object/ [Tłumaczenie strony](#)

30 maj 2014 - When you begin programming with JavaScript you might run across books, tutorials, and people who say "Everything in JavaScript is an object.

Everything About Javascript Objects - Part 1 - Overflowjs

<https://overflowjs.com/posts/everything-about-javascript-objects-part-1/> [Tłumaczenie strony](#)

5 wrz 2019 - Nearly everything in JavaScript is an object other than six things that are not objects which are — ...

[JavaScript Objects - W3Schools](#)

meet.js

POWERED BY  VIRTUSLAB

```
1  
2 const someString = 'this is a string!';  
3 someString.indexOf('a');  
4  
5  
6 const someBoolean = false;  
7 someBoolean.valueOf();  
8  
9  
10 const someNumber = 42;  
11 someNumber.toString();
```

Tablice i funkcje są obiektami

Dygresja: **typeof**

Type	Result
Undefined	"undefined"
Null	"object" (see below)
Boolean	"boolean"
Number	"number"
BigInt	"bigint"
String	"string"
Symbol (new in ECMAScript 2015)	"symbol"
Host object (provided by the JS environment)	<i>Implementation-dependent</i>
Function object (implements <code>[[Call]]</code> in ECMA-262 terms)	"function"
Any other object	"object"

js

AB

8 typów danych:

- 7 prostych
- obiekty

Tylko obiekty mają metody

Boxing: typ prosty jest wrapowany przez obiekt, który dostarcza metod

```
2 "this is a string".toUpperCase();  
3  
4 new String('this is a string').toUpperCase();  
5
```

Podsumowując: w JS nie
wszystko jest obiektem nawet
jeżeli JS czasem tak mówi

MIT: domknięcia to specjalny rodzaj funkcji

MIT: domknięcia to funkcje zwracane przez inne funkcje

MIT: domknięcia to funkcje zdefiniowane wewnątrz innych funkcji

Proste domknięcie:

```
4 const two = 2;  
5  
6 const addTwo = (x) => {  
7   return x + two;  
8 }  
9  
10  
--
```

Domknięcie to funkcja wraz z
związaniami (bindings) wszystkich
zmiennych wolnych w ramach zakresu
leksykalnego.

```
1 const add = (x) => {  
2   let y = 0;  
3   return () => {  
4     y += x;  
5     console.log(y);  
6   }  
7 }  
8 const addTwo = add(2);  
9  
10 addTwo(); // 2  
11 addTwo(); // 4  
12 addTwo(); // 6
```


Closures are a poor man's object.

Module pattern

```
4 const module = (function () {  
5   const privatVal = 'this is private'  
6   let counter = 0; // private as well  
7   return {  
8     inc: () => {  
9       counter++;  
10    },  
11    getCounterValue: () => counter,  
12  }  
13 })();  
14  
15 module.getCounterValue(); // 0  
16 module.inc(); // 1  
17 module.inc(); // 2
```

Dzięki!

