JS Przetwarzanie struktur danych

JS Przetwarzanie struktur danych

Prowadząca

Anna Rodziewicz
Senior Front-end Developer

Plan gry

- Obiekty!
- Math & Date
- przetwarzanie obiektów
- kopiowanie

Obiekty

Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

Obiekt (czyli samochód) posiada pewne **właściwości**, takie jak: nazwa, kolor, liczba drzwi, moc silnika, oraz **czynności** (metody, funkcje) jakie może wykonać: może jechać, zwolnić, przyspieszyć, ...

Obiekt Array

Własności:

.length - zwraca długość tablicy

const arr = [1,2,3]

Metody:

.map,
.filter,
.reduce,
.push,
.pop

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

Obiekt String

Własności:

.length - zwraca długość napisu

const name = “a”

Metody:

.toUpperCase() / .toLowerCase()
.split(‘,’)
.indexOf(‘a’)
.replace(‘a’, ’b’)
.slice(x, y)
.charAt(x)

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

Zadanie

```
const rodzajeMiesa = ['wołowina',
'jagnięcina', 'drób',
'wieprzowina', 'ryby'];
```

// Napisz funkcję, która sprawdzi, czy danie w opisie nie zawiera mięsa

// Do każdego dania bezmięsnego dodaj klasę `'vegan'`

Zadanie

Napisz funkcję, która sprawdzi, czy podane zdanie jest **pangramem.**

Założenia 1: Nie używaj regexa
Założenie 2: W zdaniu nie występuje interpunkcja
Założenie 3: ...ale mogą wielkie i małe litery :)

Obiekt Date

Własności:

```
const date = new Date();
```

Metody:

.getDate()
.getMonth()
.getHours()
.getDay()
.getTime() //czas w milisekundach od 1 stycznia 1970

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

Obiekt Math

Własności:

Math.PI
Math.SQRT2

```
const random = Math.random();
```

Metody:

.random()
.min(1, 2, 3)
.max(1, 2, 3)
.floor(12.239)
.round(12.23845)

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

Zadanie

Stwórz metodę, która wybierze 6 **różnych** losowych liczb z przedziału 1 do 49

Zadanie

Stwórz metodę, która pozwoli sprawdzić, która pizza się najbardziej opłaca:

```
const pizzas = [
        {r: ‘15’, price: ‘35 PLN’ },
        {r: ‘20’, price: ‘42 PLN’ },
        {r: ‘30’, price: ‘55 PLN’ }
]
```

Obiekty

```
const cat = {

 name: 'Peru',

 meow: function() {
   console.log("Meow meow")
 }
};

console.log(cat.name);
cat.meow();
```

właściwość

metoda

Obiekty

```
cat.name = "Lion"
cat['name'] = "Lion"
```

odczyt / zapis właściwości

```
cat.meow()
cat['meow']()
```

wywołanie metody

Obiekty

```
cat.race = "Ragdoll"

cat['age'] = '3'
```

dynamiczne dodawanie właściwości

Zadanie

Zainicjuj zmienną person wskazującą na obiekt o właściwościach `name` (dowolny string) i metodzie `hello()`

Zaimplementuj funkcję `hello()` tak aby logowała tekst “Cześć [name]” używając wskaźnika this

Obiekty - iteracje

For ... in

Metoda pozwalająca na iterowanie po polach obiektu.

```
const object1 = { a: 1, b: 2, c: 3 };

for (let property in object1) {
    console.log(property);   // a b c
}
```

Obiekty - iteracje

For ... in

```
let  values = "";
const object1 = { a: 1, b: 2, c: 3 };

for (let property in object1) {
    values += object1[property];
}
console.log(values);  //  "123"
```

Zadanie

Stwórz funkcję, która jako argument przyjmuje obiekt i wypisze informacje o jego właściwościach.

Input: {name: ‘Peru’, age: ‘3Y’}

Output:
‘The name of object is Peru. The age of object is 3Y.’

Output*:
‘The name of object is Peru, the age of object is 3Y.’

Obiekty - iteracje

Object.values(...)

Metoda zwraca **tablicę wartości** pól z danego obiektu.

```
const object1 = {
 a: 'somestring',
 b: 42,
 c: false
};
console.log(Object.values(object1));   // ["somestring", 42,
false]
```

Zadanie

Z obiektu:

```
const obj = {
  'user1': ['Jan', 'Kowalski'],
  'user2': ['Monika', 'Nowak'],
  'user3': ['Krzysztof', 'Dąbek'],
  'user4': ['Marianna', 'Fiołek'],
  'user5': ['Zuzanna', 'Tata']
};
```

stwórz tablicę obiektów w postaci:

```
const usersArray = [{
    firstName: 'Jan',
    secondName: 'Kowalski'
}, {...}
...
]
```

Obiekty - iteracje

Object.keys(...)

Metoda zwraca **tablicę kluczy** z danego obiektu.

```
const object1 = {
 a: 'somestring',
 b: 42,
 c: false
};
console.log(Object.values(object1));   // ["a", "b", "c"]
```

Zadanie

Stwórz funkcję, której przekazując dwa obiekty, dopisze wartości drugiego obiektu jako klucze do pierwszego

Input:
obj1 = {name: ‘xxx’, family: ‘yyy’}
obj2 = {key: ‘zzz’}

Output:
obj1 = {name: ‘xxx’, family: ‘yyy’, ‘zzz’: null}

Obiekty - iteracje

Object.entries(...)

Metoda zwraca tablicę z elementami [klucz, wartość] z danego obiektu

```
const object1 = {
 a: 'somestring',
 b: 42,
 c: false
};
console.log(Object.entries(object1));
// [ [ 'a', 'somestring' ], [ 'b', 42 ], [ 'c', false ] ]
```

Obiekty

Object.hasOwnProperty(‘propertyName’)
Metoda do sprawdzania czy w danym obiekcie jest dana właściwość

```
const myObject = {
 a: 'somestring',
 b: 42,
 c: false
};

console.log(myObject.hasOwnProperty('b')); // true
console.log(myObject.hasOwnProperty('cos innego')); // false
```

Zadanie

Jeżeli obiekt nie ma wartości o nazwie “age” to ją dodaj i ustaw losową liczbę całkowitą.

Uwaga! Obiekt może posiadać ‘age’ równe null

Kopiowanie obiektów i tablic

```
let first = ['c', 'b', 'a']
let second = first;

>> first = ['c', 'b', 'a']
>> second = ['c', 'b', 'a']
```

Kopiowanie obiektów i tablic

```
first[0] = 'D'
```

```
>> first = ['d', 'b', 'a']
>> second = ?
```

Kopiowanie obiektów i tablic

```
first[0] = 'D'
```

```
>> first = ['d', 'b', 'a']
>> second = ['d', 'b', 'a']
```

shallow copy

===

płytka kopia

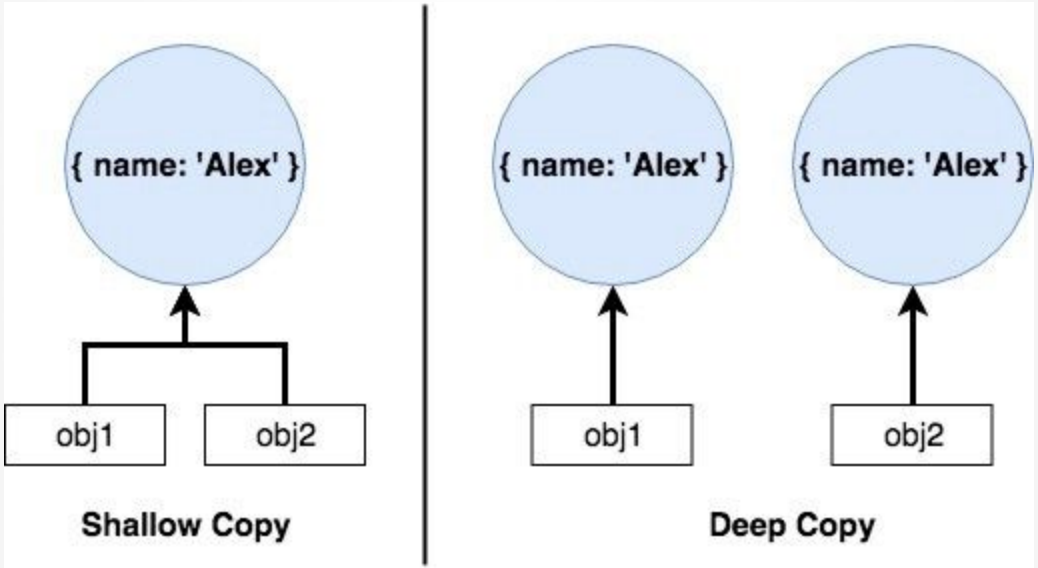
Kopiowanie obiektów i tablic

Co zrobić, aby:

```
>> first = ['d', 'b', 'a']
>> second = ['c', 'b', 'a']
```

deep copy
===
głęboka kopia

Kopiowanie obiektów i tablic



Kopiowanie obiektów i tablic

```
let first = ['c', 'b', 'a']
let second = JSON.parse(JSON.stringify(first));

>> first = ['c', 'b', 'a']
>> second = ['c', 'b', 'a']
```

Kopiowanie obiektów i tablic

Co zrobić, aby:

```
>> first = ['d', 'b', 'a']
>> second = ['c', 'b', 'a']
```

Kopiowanie obiektów i tablic

```
const wildCat = JSON.parse(JSON.stringify(cat))
```

```
>> cat = {name: 'Peru',
    meow: function() {
        console.log("Meow meow")
    }
};
>> wildCat = ?
```

Kopiowanie obiektów i tablic

```
const wildCat = JSON.parse(JSON.stringify(cat))
```

```
>> cat = {name: 'Peru',
    meow: function() {
        console.log("Meow meow")
    }
};
>> wildCat = {name: 'Peru'}
```

Kopiowanie obiektów i tablic

```
let wildCat = {}
Object.assign(wildCat, cat)
```

Object.assign(CEL, ...ZRODLA)

```
>> cat = {name: 'Peru',
     meow: function() {
          console.log("Meow meow")
     }
};
>> wildCat = {name: 'Peru',
     meow: function() {
          console.log("Meow meow")
     }
};
```

Zadanie

Stwórz nowy obiekt `plant`, który będzie miał dokładnie te same właściwości co `wildPlant` oraz `homeDecoration`

Destrukturyzacja

Dekompozycja (zwana też **destrukturyzacją**) to nowa funkcjonalność w ES6, która polega na wyciąganiu zmiennych z jakiejś struktury - np. z obiektów lub tablic.

```
const cat = {
 name: 'Peru',
 age: '3Y',
 race: 'Ragdoll'
};
```

Destrukturyzacja

```
const cat = {
 name: 'Peru',
 age: '3Y',
 race: 'Ragdoll'
};

const {name, age} = cat;

>> name = 'Peru'
>> age = '3Y'
```

Destrukturyzacja

```
const cat = {
 name: 'Peru',
 age: '3Y',
 race: 'Ragdoll'
};

const {name: catName, age: catAge} = cat;

>> catName = 'Peru'
>> catAge = '3Y'
```

Destrukturyzacja

```
const cat = {
 name: 'Peru',
 age: '3Y',
 race: 'Ragdoll'
};

const {name, ...catWithoutName} = cat;

>> name = 'Peru'
>> catWithoutName = {age: '3Y', race: 'Ragdoll'};
```

Destrukturyzacja tablic

```
const cats = ['Ragdoll', 'Maine Coon', 'Bengal', 'Snowshoe']

const {catRagdoll, ...otherCats} = cats;

>> catRagdoll = 'Ragdoll'
>> otherCats = ['Maine Coon', 'Bengal', 'Snowshoe']
```

Operator spread | rest

```
Object.assign(CEL, ...ZRODLA)
```

Operator spread umożliwia rozwinięcie wyrażenia. Składnia rozwinięcia pozwala na rozwinięcie wyrażenia w miejscach, w których potrzebne jest wiele argumentów (do wywołań funkcji), wiele elementów (do literałów tablicowych) lub wiele zmiennych ().

Operator spread

```
const name = ‘Ania’;

>> [...name] = [‘A’, ‘n’, ‘i’, ‘a’]

const prices = [1, 2, 5, 60, 22, 10]

Math.min[prices] >> NaN
Math.min[...prices] >> 1
```

Operator spread

```
const smallArray = [1, 2, 3];
const
 bigArray = [4, 5, 6];

smallArray.push(...bigArray); >> [1, 2, 3, 4, 5, 6]

 [1, 2, 3, ...bigArray, 7, 8]
```

Zadanie

const weekTemperature = [7, 8, 4, 3, 2, 2.5, 0, -1, -1.5, -2, -2.5, 0, 2, 3.5, 5.5, 9, 10.5, 4, 11, 11.5, 10, 9.5, 8, 9, 7.5, 9, 11, 11.5, 12, 9, 7, 6.5, 4, 3, 5, 3.5, 3, 2, 2.5, 3, 2, 1, 2, 1.5, 0, -2]

Stacja meteorologiczna badała temperaturę przez cały tydzień. Znajdź najwyższą zanotowaną wartość.

Policz średnią zmierzoną temperaturę.

Kontakt

Dziękuję!

Ania Rodziewicz
aerodziewicz@gmail.com