

| Przetwarzanie Struktur Danych

Operatory porównania

20 == '20' → true

20 === '20' → false

20 != '20' → false

20 !== '20' → true

Wartości fałszywe

- False
- Undefined
- Null
- NaN
- 0
- ""

Typeof false ➡ 'boolean'

Obiekt Math

Wbudowany obiekt zawierający własności i metody związane z funkcjami i stałymi matematycznymi.

Math.**abs**(-10) // zwraca wartość absolutną (dodatnią)

Math.**round**(24.2342) // zaokrąglenie

Math.**ceil**(24.3) // zaokrąglenie w górę

Math.**floor**(24.89) // zaokrąglenie w dół

Math.**max**(3, 4, 5, 1, 0) // zwraca największy argument

Math.**min**(22, 233, 1, 56) // zwraca najmniejszy argument

Math.**round**(Math.random() * 10)

// .random zwraca losową liczbę w zakresie od 0 do 1

Obiekt Date

Tworzy instancje Date,
które pozwalają na pracę z danymi opisującymi datę i czas.

Tworzenie dat:

```
var date = new Date(); // zwraca obiekt z aktualną datą
```

```
new Date(2017, 3, 28, 12, 23, 4) // tworzy obiekt z datą  
// rok, miesiąc, dzień, godziny, minuty, sekundy
```

```
new Date(2222); // ilość milisekund od 1 stycznia 1970
```

```
date.toString() // konwertuje obiekt z datą do ciągu znaków(string)
```

Obiekt Date

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

```
date.getFullYear();
```

```
date.getMonth();
```

```
date.getDate();
```

```
date.getHours();
```

```
date.getMinutes();
```

```
date.getMilliseconds();
```

```
date.getDay(); // dzień tygodnia
```

```
date.getTime();// ilość milisekund od 1 stycznia 1970
```

Metody te mają odpowiedniki z **set**

Number

var num = 2 typeof num ➡ number

num.toString() // zwraca liczbę zapisaną jako string
(255).toString(16) // 'ff',

num.toFixed(x) // x to ilość miejsc po przecinku

MAX_VALUE	Returns the largest number possible in JavaScript
MIN_VALUE	Returns the smallest number possible in JavaScript
NEGATIVE_INFINITY	Represents negative infinity
NaN	Represents a "Not-a-Number" value
POSITIVE_INFINITY	Represents infinity

String

var **name** = 'Ania' typeof name ➡ string

name.**length** // długość stringa

name.**charAt**(x) // zwraca element ciągu o indexie x

name.**indexOf**('nia') // sprawdza czy „nia” znajduje się w **name**

name.**replace**('nia', 'nna') // zamienia ciąg znaków

name.**slice**(x, y) // wycinamy treść od indexu x do indexu y

name.**substr**(x, y) // wycinamy treść długości y zaczynając od indexu x

name.**split**(',') // dzieli ciąg znaków

name.**toUpperCase**()

name.**toLowerCase**()

TABLICE



Arrays - tablice

```
var myTable = [ ];
```

```
var array = new Array('jeden', 'dwa', 3);
```

```
var liczby = [ 10 , 11, 12 ];
```

```
var cars = [ 'Saab', 'Volvo', 'BMW' ];
```

Arrays - indexy

Do elementów tablicy dostajemy się po indeksie

```
var cars = [ 'Saab', 'Volvo', 'BMW' ];
```

```
cars [ 0 ] === 'Saab,
```

```
cars [ 1 ] === 'Volvo,
```

```
cars [ 2 ] === 'BMW'
```

Arrays - zapis

```
var cars = [ 'Saab', 'Volvo', 'BMW' ];
```

```
cars [ 2 ] = 'Niemiecki wóz,
```

```
cars [ 3 ] = 'Trabant,
```

```
cars [ 4 ] = 'Czarna Wołga'
```

Array.length – długość tablicy

```
var a = [ ];
```

```
var b = [ "Freeze" ];
```

```
var c = [ "Batman", "Robin", "Freeze", "Riddler" ]
```

```
a.length == 0
```

```
b.length == 1
```

```
c.length == 4
```

Tablice wielowymiarowe

```
var table = [ [11, 12], [21, 22], [31, 32] ];
```

table[1]  [21, 22]

table[1][0]  21

Array.concat - sklejanie tablic

```
var a = [ "Batman", "Robin" ];  
var b = [ "Freeze", "Riddler" ];  
var c = a.concat(b)
```

```
a == [ "Batman", "Robin" ];  
b == [ "Freeze", "Riddler" ];  
c == [ "Batman", "Robin", "Freeze", "Riddler" ];
```

Array - **pop**, **push**, **shift**, **unshift**

unshift('Catwoman')

push('Catwoman')

["Batman", "Robin", "Freeze", "Riddler"];

shift()

pop()

Array – **shift()**

`arr.shift()` - Usuwa pierwszy element z tablicy zwracając go. Metoda ta zmienia długość tablicy.

```
var heroes = ['Batman', 'Robin', 'Gordon'];
```

```
var firstHero = heroes.shift();
```

```
firstHero    ==    'Batman'  
heroes       ==    ['Robin', 'Gordon'];
```

Array – pop()

`arr.pop()` - Usuwa ostatni element z tablicy zwracając go. Metoda ta zmienia długość tablicy.

```
var heroes = ['Batman', 'Robin', 'Gordon'];
```

```
var lastHero = heroes.pop();
```

```
lastHero    ==    'Gordon'  
heroes      ==    ['Batman', 'Robin'];
```

Array – **unshift()**

`arr.unshift()` - Dodaje jeden lub więcej elementów na początek tablicy i zwraca jej nową długość.

```
var heroes = ['Batman', 'Robin', 'Gordon'];  
var newHeroes = heroes.unshift('Rob', 'Gor');
```

```
heroes      == ['Rob', 'Gor', 'Batman', 'Robin', 'Gordon'];  
newHeroes   == 5
```

Array – **push()**

`arr.push ()` - Dodaje jeden lub więcej elementów na koniec tablicy i zwraca jej nową długość. Metoda ta zmienia długość tablicy.

```
var heroes = ['Batman', 'Robin', 'Gordon'];  
var newHeroes = heroes.push('Rob', 'Gor');
```

```
heroes      == ['Robin', 'Gordon', 'Batman', 'Rob', 'Gor'];  
newHeroes   == 5
```

- Stwórz tablicę związaną z Twoim projektem. Niech się składa z 5-7 elementów
- Przy każdej operacji wyświetl zawartość i długość tablicy.
- Usuń ostatni element tablicy
- Dodaj dwa dodatkowe elementy na początek tablicy
- Dodaj element na koniec tablicy
- Usuń element z początku tablicy

array.forEach()

```
var heroes = ['Batman', 'Robin', 'Gordon'];
```

```
heroes.forEach(function(hero, index) {  
    console.log(hero, index);  
})
```

- wywołuje funkcję dla każdego elementu tablicy
- za każdym wywołaniem przekazuje do funkcji dwa parametry:
 - pierwszy - aktualny element tablicy
 - index - index aktualnego elementu tablicy

array.map()

```
var heroes = ['Batman', 'Robin', 'Gordon'];
```

```
var modifiedHeroes = heroes.map(function(hero) {  
    return 'Hero - ' + hero;  
})
```

- tworzy nową tablicę
- nowa tablica zawiera nowe elementy, stworzone w oparciu o elementy ze starej tablicy
- stara tablica pozostaje niezmienną

- `var names = ['Arek', 'Darek', 'Czarek', 'Mariusz', 'Marek', 'Wiesław', 'Aneta', 'Stanisława'];`
- Korzystając z `map` dodaj do elementu w tablicy nazwisko Kowalski lub Kowalska w zależności od imienia.
- Korzystając z `forEach` wyświetl dla tablicy z nazwiskami tylko Imię i Nazwisko osób które w imieniu mają ciąg 'rek'.

array.filter()

```
var numbers = [1,2,3,4,5,6,7,8,9,10];
```

```
numbers.filter(function(num) {  
    return num > 9  
});
```

➡ [10]

```
numbers.filter(function(num) {  
    return num > 5 && num < 9  
});
```

➡ [6,7,8]

```
numbers.filter(function(num) {  
    return num < 3 || num > 8  
});
```

➡ [1,2,9,10]

array.reduce()

Metoda **reduce()** wywołuje funkcję względem wartości przyrostowej z każdego wywołania i kolejnego elementu tablicy

```
var numbers = [1,2,3,4,5,6,7,8,9,10];
```

```
numbers.reduce(function(result, number) {  
    return result += number; },  
0);
```

 55

```
numbers.reduce(function(result, number) {  
    return result -= number; },  
100);
```

 45

array.find()

Metoda **find()** zwraca pierwszy element tablicy, który spełnia warunek podanej funkcji testującej.

```
var numbers = [12, 5, 8, 130, 44];
```

```
function isBigEnough(element) {  
    return element >= 15;  
}
```

```
numbers.find(isBigEnough);
```



130

Array - Kolejowanie funkcji - method chaining


```
['Batman', 'Robin', 'Gordon']  
  .map(function (hero) {  
    return 'Hero - ' + hero;  
  })  
  .forEach(function(hero) {  
    console.log(hero);  
  })
```

- Dla wynikowej tablicy z nazwiskami zwróć sumę długości wszystkich imion krótszych niż 6 znaków.

OBIEKTY



Obiekty

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

Zapis i odczyt wartości

```
var myObject = {  
  prop1 : ... ,  
  prop2 : ... ,  
  method1 : ... ,  
  method2 : ...  
};
```



Zapis i odczyt wartości

```
var car = {  
    name : 'Fiat',  
    model : 500,  
    color : 'white'  
};
```

```
car.name    // 'Fiat'  
car.model   // 500  
car.color   // 'white'
```

Metody obiektu

```
var car = {  
  name : 'Fiat' ,  
  model : 500 ,  
  color : 'white' ,  
  start : function() { console.log('bruum') } ,  
  fullModel : function() {  
    return this.name + ' ' + this.model;  
  }  
};
```

```
car.start();           // undefined  
car.fullModel()        // 'Fiat 500'
```

Obiekty – alternatywna metoda adresacji pól

`car.type == car ['type']`

`car.model == car ['model']`

`car.color == car ['color']`

`car.start() == car ['start'] ()`

`car.fullModel() == car ['fullModel'] ()`

- Każdy obiekt zawiera funkcję `hasOwnProperty(propertyName)`, która zwraca czy obiekt ma właściwość o podanej nazwie
- Stwórz funkcję, której przekazując obiekt wypisze informacje o jego właściwościach
- Stwórz funkcję, której przekazując dwa obiekty, dopisze właściwości drugiego obiektu do pierwszego

- Korzystając z przygotowanego wcześniej kodu przygotuj grę w oczko wykorzystując do tej pory poznane techniki przetwarzania danych