MAP ORAZ SET

CO TO JEST MAP?

- **Mapy** służą do tworzenia zbiorów z parami [klucz - wartość]. Przypominają one klasyczne obiekty, natomiast główną różnicą odróżniającą je od klasycznych obiektów, jest to, że kluczami może być tutaj dowolny typ danych.
- Aby stworzyć mapę możemy skorzystać z jednej z 2 konstrukcji:

```
const map = new Map();
map.set("kolor1", "red");
map.set("kolor2", "blue");

//lub

const map = new Map([
    ["kolor1", "red"],
    ["kolor2", "blue"],
]);
```

- Dla każdej mapy mamy dostęp do kilku metod:

set(key, value)	Ustawia nowy klucz z daną wartością
get(key)	Zwraca wartość danego klucza
has(key)	Sprawdza czy mapa ma dany klucz
delete(key)	Usuwa dany klucz i zwraca true/false jeżeli operacja się udała
clear()	Usuwa wszystkie elementy z mapy
entries()	Zwraca iterator zawierający tablicę par [klucz-wartość]
keys()	Zwraca iterator zawierający listę kluczy z danej mapy
values()	Zwraca iterator zawierający listę wartości z danej mapy
forEach	robi pętlę po elementach mapy
prototype[@@iterator]()	Zwraca iterator zawierający tablicę par [klucz-wartość]

MAPY W PRZECIWIEŃSTWIE DO OBIEKTÓW MOGĄ MIEĆ KLUCZE DOWOLNEGO TYPU, GDZIE W PRZYPADKU OBIEKTÓW (W TYM TABLIC) SĄ ONE KONWERTOWANE NA TEKST:

```
const map = new Map();

map.set("1", "Kot");
map.set(1, "Pies");

console.log(map); //{"1" => "Kot", 1 => "Pies"}
```

```
const ob = {}

ob["1"] = "Kot";
ob[1] = "Pies";

console.log(ob); //{"1" : "Pies"}
```

```
const map = new Map();

const ob1 = { name : "test1" }
const ob2 = { name : "test2" }

map.set(ob1, "koty");
map.set(ob2, "psy");
map.set("[object Object]", "świnki");

console.log(map); //{{…} => "koty", {…} => "psy", "[object Object]" =>
"świnki"}
```

WEAKMAP TO ODMIANA MAPY, KTÓRĄ OD MAP ROZRÓŻNIAJĄ TRZY RZECZY:
NIE MOŻNA PO NIEJ ITEROWAĆ (W PRZYSZŁOŚCI BĘDZIE MOŻNA, BO JUŻ ZAPOWIEDZIANO ODPOWIEDNIE ZMIANY)
KLUCZAMI MOGĄ BYĆ TYLKO OBIEKTY
JEJ ELEMENTY SĄ AUTOMATYCZNIE USUWANE GDY DO DANEGO OBIEKTU (KLUCZA) NIE BĘDZIE REFERENCJI

ABY STWORZYĆ NOWĄ WEAKMAP, SKORZYSTAMY Z INSTRUKCJI:

```
const map = new Map();

map.set("1", "Kot");
map.set(1, "Pies");

console.log(map); //{"1" => "Kot", 1 => "Pies"}
```

```
const ob = {}

ob["1"] = "Kot";
ob[1] = "Pies";

console.log(ob); //{"1" : "Pies"}
```

```
const map = new Map();

const ob1 = { name : "test1" }
const ob2 = { name : "test2" }

map.set(ob1, "koty");
map.set(ob2, "psy");
map.set("[object Object]", "świnki");

console.log(map); //{{…} => "koty", {…} => "psy", "[object Object]" =>
"świnki"}
```

KAŻDA MAPA DAJE NAM KILKA METOD:

set(key, value)	Ustawia wartość dla klucza
get(key)	Pobiera wartość klucza
has(key)	Zwraca true/false w zależności czy dana WeakMap posiada klucz o danej nazwie
delete(key)	Usuwa wartość przypisaną do klucza

CO TO JEST SET

- Obiekt Set jest kolekcją składającą się z **unikalnych wartości**, gdzie każda wartość może być zarówno typu prostego jak i złożonego. W przeciwieństwie do mapy jest to zbiór pojedynczych wartości.
- Żeby stworzyć Set możemy użyć jednej z 2 składni:
- `const set = new Set();`
- `set.add(1);`
- `set.add("text");`
- `set.add({name: "kot"});`
- `console.log(set); //{1, "text", {name : "kot"}}`
-
- `//lub`
- `//const set = new Set(elementIterowalny);`
- `const set = new Set([1, 1, 2, 2, 3, 4]); //{1, 2, 3, 4}`
- `const set = new Set("kajak"); //{"k", "a", "j"}`
- Obiekty Set mają podobne właściwości i metody co obiekty typu Map, z małymi różnicami:

add(value)	Dodaje nową unikatową wartość. Zwraca Set
clear()	Czyści całą mapę
delete(key)	Usuwa dany klucz i zwraca true/false jeżeli operacja się udała
entries()	Zwraca iterator zawierający tablicę par [klucz-wartość]
has(key)	Sprawdza czy mapa ma dany klucz
keys()	Zwraca iterator zawierający listę kluczy z danej mapy
values()	Zwraca iterator zawierający listę wartości z danej mapy
forEach	robi pętlę po elementach mapy
prototype[@@iterator]()	Zwraca iterator zawierający tablicę par [klucz-wartość]

DZIĘKI TEMU, ŻE SET ZAWIERA NIEPOWTARZAJĄCE SIĘ WARTOŚCI, MOŻEMY TO WYKORZYSTAĆ DO ODSIEWANIA DUPLIKATÓW W PRAKTYCZNIE DOWOLNYM ELEMENCIE ITERACYJNYM - NP. W TABLICY:

```
const tab = [1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 5, 5];

const set = new Set(tab);
console.log(set); //{1, 2, 3, 4, 5}

const uniqueTab = [...set];
console.log(uniqueTab); //[1, 2, 3, 4, 5]
const tab = [
    "ala",
    "bala",
    "cala",
    "ala",
    "ala"
]

const tabUnique = [... new Set(...tab)];
console.log(tabUnique); //["ala", "bala", "cala"]
```

PODOBNIE JAK DLA MAP ISTNIEJĄ WEAKMAP, TAK DLA SETÓW ISTNIEJĄ **WEAKSET**. SĄ TO KOLEKCJE SKŁADAJĄCE SIĘ Z UNIKALNYCH OBIEKTÓW. PODOBNIE DO WEAKMAP OBIEKTY TAKIE BĘDĄ

AUTOMATYCZNIE USUWANE Z WEAKSET, JEŻELI DO DANEGO OBIEKTU ZOSTANĄ USUNIĘTE WSZYSTKIE REFERENCJE.

```
const set = new WeakSet();
const a = {};
const b = {};

set.add(a);
set.add(b);
set.add(b);
console.log(set); //{a, b}
```

KAŻDY WEAKSET UDOSTĘPNIA NAM METODY:

add(ob)	Dodaje dany obiekt do kolekcji
delete(ob)	Usuwa dany obiekt z kolekcji
has(ob)	Zwraca true/false w zależności, czy dana kolekcja zawiera dany obiekt