ΠΕΡΙΕΧΟΜΕΝΑ:

1. Σύνολα (Sets)
    1. Μέθοδοι Συνόλων
2. Maps
    1. Μέθοδοι των Maps
3. WeakSets και WeakMaps

Σύνολο (Set)

- Είναι μία συλλογή στοιχείων (primitives ή/και αντικείμενα) χωρίς συγκεκριμένη σειρά.
- Προσοχή ότι τα στοιχεία είναι ανά δύο διαφορετικά (κάθε στοιχείο μπορεί να περιέχεται το πολύ μία φορά)

Κατασκευή Συνόλου

- Με τη συνάρτηση - Κατασκευαστή Set():

Κατασκευαστής	Επεξήγηση
Set()	Κενό Σύνολο
Set(iterable)	Αρχικοποίηση με τα στοιχεία του iterable

Παρατήρηση:

- Iterable: Είναι αντικείμενα που επιδέχονται επανάληψη με την for…of
  - Όπως είδαμε στο μάθημα 9.1 μπορούμε να ορίσουμε δικά μας αντικείμενα που επιδέχονται επανάληψη (ορίζοντας το σύμβολο Symbol.iterator)
  - Ενώ πίνακες, συμβολοσειρές, αλλά και τα ίδια τα σύνολα είναι iterable.

Παράδειγμα 1: set_create

```
let set = new Set([1,2,3,4,1]);
console.log(set);
```

```
Set(4)
  [[Entries]]
    0: 1
    1: 2
    2: 3
    3: 4
  size: 4
  [[Prototype]]: Set
```

Σημείωση:

- Ο έλεγχος της ισότητας των μελών γίνεται χρησιμοποιώντας τον αλγόριθμο “SameValueZero”
  - πρακτικά είναι εφαρμογή του τελεστή της αυστηρής ισότητας (===), (με κάποιες εκκεντρικές εξαιρέσεις: π.χ. το NaN είναι ίσο με το NaN)

Παράδειγμα 2: sets_element_equality

```
let set = new Set([NaN,1,new Number(1),
                   NaN, 1, new Number(1)]);
console.log(set);
```

```
Set(4)
  [[Entries]]
    0: NaN
    1: 1
    2: Number
      value: Number {1}
    3: Number
      value: Number {1}
  size: 4
  [[Prototype]]: Set
```

Σημείωση για την ισότητα αντικειμένων:

- Δεν χρησιμοποιείται η valueOf() για την σύγκριση αντικειμένων.
- Οπότε δύο αναφορές είναι ίσες, μόνο αν αναφέρονται στο ίδιο αντικείμενο.

Παράδειγμα 3: sets_element_objects_equality

```
let o1 = {a: 1};
let o2 = {a: 2};
let o3 = {a: 3, valueOf() { return this.a; }};
let o4 = {a: 3, valueOf() { return this.a; }};

let set = new Set([o1, o1, o2, o3, o4]);
console.log(set);
```

```
Set(4)
  [[Entries]]
    0:
      value: {a: 1}
    1:
      value: {a: 2}
    2:
      value: {a: 3, valueOf: ƒ}
    3:
      value: {a: 3, valueOf: ƒ}
  size: 4
  [[Prototype]]: Set
```

Μέθοδοι Συνόλων [Set.prototype]

- Ύπαρξη Στοιχείου:

Μέθοδος	Επεξήγηση
has(element)	true, αν το element ανήκει στο σύνολο

- Προσθήκη/Διαγραφή Στοιχείου/ων:

Μέθοδος	Επεξήγηση
add(element)	Προσθέτει το element στο σύνολο [επιστρέφει αναφορά στο ίδιο το σύνολο]
delete(element)	Αφαιρεί το element από το σύνολο [επιστρέφει true, αν το στοιχείο υπήρχε στο σύνολο]
clear()	Διαγράφει όλα τα στοιχεία του συνόλου

- Βοηθητικές Μέθοδοι (αντίστοιχες των πινάκων):

Μέθοδος	Επεξήγηση
keys()	Κάνει την ίδια ενέργεια με τη values()
values()	Iterator με τα Στοιχεία του Συνόλου (σε διάταξη που ακολουθεί τη σειρά των στοιχείων)
entries()	Πίνακας που περιέχει υποπίνακες. Κάθε υποπίνακας είναι της μορφής [value, value], όπου value το στοιχείο του συνόλου. [Η διάταξη ακολουθεί τη σειρά εισαγωγής]

- και η συναρτησιακή:

Μέθοδος	Επεξήγηση
forEach((elem[, key[, set]])=>… [, this])	Αντίστοιχη με την forEach() των πινάκων

Παράδειγμα 4: sets_add_delete

```
let set = new Set();
set.add("a");
set.add("b");
console.log(new Set(set));
set.delete("b");
console.log(new Set(set));
set.clear();
console.log(new Set(set));
```

```
▸ Set(3) {'a', 'b', 'c'}
▸ Set(2) {'a', 'c'}
▸ Set(0) {size: 0}
```

Properties Συνόλων:

- Μοναδικό property είναι το size: Πλήθος Στοιχείων Συνόλου

Παράδειγμα 5: sets_values_foreach

```
let set = new Set([5,2,1,4,3]);
console.log(set.values());

set.forEach((element, key, thisSet)=>{
  console.log(element,key,thisSet);
});
```

```
▾ SetIterator
  ▾ [[Entries]]
    ▸ 0: 5
    ▸ 1: 2
    ▸ 2: 1
    ▸ 3: 4
    ▸ 4: 3
  ▸ [[Prototype]]: Set Iterator
    [[IteratorHasMore]]: true
    [[IteratorIndex]]: 0
    [[IteratorKind]]: "values"
5 5 ▸ Set(5)
2 2 ▸ Set(5)
1 1 ▸ Set(5)
4 4 ▸ Set(5)
3 3 ▸ Set(5)
```

Map (Απεικόνιση, Αντιστοιχία, Λεξικό)

- Σύνολο ζευγών κλειδιού - τιμής.
- Τα κλειδιά είναι μοναδικά.
- Σε αντίθεση με τα αντικείμενα, τα κλειδιά μπορούν να είναι οποιουδήποτε τύπου δεδομένων

Κατασκευή Map

- Με τη συνάρτηση - Κατασκευαστή Map():

Κατασκευαστής	Επεξήγηση
Map()	Κενό Map
Map(iterable)	Αρχικοποίηση με τα στοιχεία του iterable (βλ. παρατ.2)

Παρατήρηση 1:

- Το Map είναι και αυτό iterable
  - Άρα μπορούμε να κάνουμε διαπέραση στα στοιχεία του με την for…of

Βασικές Μέθοδοι του Map:

- Οι πιο χρήσιμες μέθοδοι του Map είναι οι μέθοδοι:

Κατασκευαστής	Επεξήγηση
set(key, value)	Θέτει την τιμή του κλειδιού key ίση με value [Αν το κλειδί υπάρχει ήδη, ενημερώνει την τιμή του] Επιστρέφει αναφορά στο map
get(key)	Επιστρέφει την τιμή του κλειδιού key [undefined, αν δεν έχει οριστεί το κλειδί]

Παρατήρηση 2:

- Η 2η εκδοχή του κατασκευαστή, απαιτεί το iterable να έχει στοιχεία που να είναι πίνακες με 2 στοιχεία (κλειδί, τιμή)

Παράδειγμα 6: maps

```
// example 1: an empty map filled with set()
let map = new Map();
map.set("1", "value1");
map.set(1, "value2");
map.set(null, "value3");
console.log(map.get("1"));
console.log(map.get(1));
console.log(map);

// example 2: initializing with another map
let map2 = new Map(map);
console.log(map2);

// example 3: initializing with an array
let map3 = new Map([
  ["key1", "value1"],
  [2, "value2"],
  [undefined, "value3"]
]);
console.log(map3);

// example 4: initializing with a set
let map4 = new Map(new Set([["k1", "v1"], ["k2", "v2"]]));
console.log(map4);
```

```
value1
value2
▸ Map(3) {'1' => 'value1', 1 => 'value2', null => 'value3'}
▸ Map(3) {'1' => 'value1', 1 => 'value2', null => 'value3'}
▸ Map(3) {'key1' => 'value1', 2 => 'value2', undefined => 'value3'}
▸ Map(2) {'k1' => 'v1', 'k2' => 'v2'}
```

Μέθοδοι Maps [Map.prototype]

- Ύπαρξη Κλειδιού:

Μέθοδος	Επεξήγηση
has(key)	true, αν υπάρχει το κλειδί

- Διαγραφή Στοιχείου/ων:

Μέθοδος	Επεξήγηση
delete(key)	Αφαιρεί το ζεύγος με κλειδί key [επιστρέφει true, αν το στοιχείο υπήρχε στο map]
clear()	Διαγράφει όλα τα στοιχεία του map

- Βοηθητικές Μέθοδοι:

Μέθοδος	Επεξήγηση
keys()	Iterator επί των κλειδιών (σε διάταξη που ακολουθεί τη σειρά των στοιχείων)
values()	Iterator με τις τιμές των κλειδιών(σε διάταξη που ακολουθεί τη σειρά των στοιχείων, αν μία τιμή αντιστοιχεί σε πολλά κλειδιά, θα τυπωθεί μία φορά για κάθε κλειδί)
entries()	Πίνακας που περιέχει υποπίνακες. Κάθε υποπίνακας είναι της μορφής [key, value], όπου key το κλειδί και value η τιμή του. [Η διάταξη ακολουθεί τη σειρά εισαγωγής]

- και η συναρτησιακή:

Μέθοδος	Επεξήγηση
forEach((value[, key[, map]])=>… [, this])	Αντίστοιχη με την forEach() των πινάκων

Παράδειγμα 7: map_methods

```
let map = new Map([
   ["key1", "value1"],
   [2, "value2"],
   [undefined, "value3"]
]);
console.log([[...map.keys()], [...map.values()], [...map.entries()]]);
map.delete(2);
console.log(new Map(map));

map.clear();
console.log(new Map(map));
```

Properties Συνόλων:

- Μοναδικό property είναι το size: Πλήθος Στοιχείων του Map

Παράδειγμα 8: map_values_foreach

```
let map = new Map([
   ["key1", "value1"],
   [2, "value2"],
   [undefined, "value3"]
]);

map.forEach((value, key, thisMap)=>{
   console.log(value,key,thisMap);
});
```

```
value1 key1
▶ Map(3) {'key1' => 'value1', 2 => '
value2 2
▶ Map(3) {'key1' => 'value1', 2 => '
value3 undefined
▶ Map(3) {'key1' => 'value1', 2 => '
```

3. WeakSets και WeakMaps

WeakSets

- Ειδική Περίπτωση Συνόλου με την εξής λειτουργικότητα:
  - Αν για ένα στοιχείο του συνόλου δεν υπάρχει αναφορά εκτός του ίδιου του WeakSet, τότε αυτό διαγράφεται αυτόματα από το WeakSet.
- Λόγω της λειτουργίας αυτής:
  - Δεν έχει το property size και δεν υποστηρίζει τις μεθόδους clear(), keys(), values(), entries() και forEach()
  - Τα στοιχεία είναι αποκλειστικά αντικείμενα (όχι primitives)

Παράδειγμα 9: weak_sets

```
let array = [{ "key1": 1 },
             { "key2": 2 }];
let weakSet = new WeakSet(array);
array.shift();
console.log(array, weakSet);
```

Σημείωση:

- Ο Garbage Collector είναι υπεύθυνος για να μαζεύει τα «σκουπίδια» (= αντικείμενα στα οποία δεν αναφέρεται κανένα όνομα).
- Οι αναφορές των WeakSets δεν «μετράνε» στον GC (βλ. βίντεο)

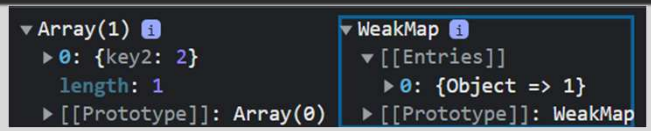
WeakMaps

- Αντίστοιχα με τα WeakMaps:
  - Αν για ένα κλειδί του συνόλου δεν υπάρχει αναφορά εκτός του ίδιου του WeakMap, τότε αυτό διαγράφεται αυτόματα από το WeakMap.
- Τα κλειδιά πρέπει να είναι αντικείμενα και δεν υποστηρίζει τις αντίστοιχες μεθόδους με το WeakSet.

Παράδειγμα 10: weak_maps

```
let array = [{ "key1": 1 },
             { "key2": 2 }];

let weakMap = new WeakMap([[array[0], 0],[array[1], 1]]);
array.shift();
console.log(array, weakMap);
```



Παρατήρηση:

- Τόσο το WeakMap όσο και το WeakSet είναι προχωρημένες δομές δεδομένων που βρίσκουν χρήση σε εξειδικευμένες εφαρμογές.
- Π.χ. βλέπε MDN (google “MDN WeakSets”) για τον εντοπισμό κυκλικών αναφορών σε μία δομή δεδομένων)