ΠΕΡΙΕΧΟΜΕΝΑ:

1. Ορισμός και Βασική Χρήση Πίνακα
    1. Μήκος Πίνακα
    2. Διαγραφή Στοιχείου και ο Τελεστής spread
    3. Εναλλακτικοί Τρόποι Δήλωσης Πίνακα
    4. Άλλοι Τελεστές σε Πίνακες
2. Διδιάστατοι Πίνακες
3. Επανάληψη επί Πινάκων
4. Ασκήσεις

Γεώργιος Κ.
Σμαραγδένιος Χορηγός Μαθήματος

Εμμανουήλ Α.
Σμαραγδένιος Χορηγός Μαθήματος

ΜΑΘΗΜΑ 6.1: Πίνακες

1. Ορισμός και Βασική Χρήση Πίνακα

JavaScript psounis

Πίνακες:

- Οι πίνακες είναι ένα βασικό εργαλείο για την ομαδοποίηση μεταβλητών.
- Αν και συνήθως οι πίνακες περιέχουν ομοειδή στοιχεία (π.χ. χρησιμοποιούμε έναν πίνακα αριθμών, ή έναν πίνακα συμβολοσειρών), μπορούμε να ορίσουμε και πίνακες με αντικείμενα διαφορετικών τύπων.

Κατασκευή Πίνακα:

- Δήλωση και αρχικοποίηση πίνακα:

```
let arrayName = [elem1, …, elemN];
```

  - arrayName: Το όνομα του πίνακα (δική μας επιλογή, βλ. κανόνες σύνταξης μεταβλητών)
  - Οι αγκύλες καθορίζουν ότι κατασκευάζουμε ένα αντικείμενο - πίνακας (είναι ένα literal πίνακα).

Σημείωση:

- Μπορούμε να έχουμε «κενές» θέσεις κατά τη δήλωση (βλ. και παράδειγμα 1)

Παράδειγμα 1: arrays_create

```
let array = [1, 2, 3, 4, 5];
console.log(array);

array = [1+2, "str",,5/2];
console.log(array);
```

Χρήση Μεταβλητών Πίνακα:

- Οι μεταβλητές που κατασκευάσαμε έχουν ονόματα:
  - array_name[0], array_name[1],…
  - (η αρίθμηση των στοιχείων ξεκινάει από το 0)
- και μπορούμε να τις χρησιμοποιήσουμε, όπως μάθαμε να χρησιμοποιούμε τις μεταβλητές, π.χ. για να κάνουμε ανάθεση:

```
arrayName[i] = value;
```

Παράδειγμα 2: arrays

```
let array = [1, 2, 3, 4, 5];
let length = 5;
for (let i=0; i<length; i++)
  console.log(`Position: ${i}, value: ${array[i]}`);

array[0] = 8;
array[2] = "three"
for (let i=0; i<length; i++)
  console.log(`Position: ${i}, value: ${array[i]}`);
```

Σημείωση:

- Ο πίνακας είναι αντικείμενο (ομαδοποιεί τα δεδομένα του πίνακα, με **χαρακτηριστικά** όπως το μήκος του, και **μεθόδους** όπως το να προσθέτουμε στοιχεία) και έτσι έχει πλούσια λειτουργικότητα που βλέπουμε στις επόμ. διαφάνειες.

1.1. Μήκος Πίνακα

Το μήκος του πίνακα (length)

- Είναι μια ιδιότητα του πίνακα (property) που έχει δύο (!) ορισμούς:
  - Αν ο πίνακας είναι πυκνός (dense), που σημαίνει ότι δεν περιέχει κενά/undefined στοιχεία, τότε μετράει το μέγεθος του πίνακα (δηλ. το πλήθος στοιχείων του)
  - Αν ο πίνακας είναι αραιός (sparse), δηλαδή περιέχει κενά/undefined στοιχεία, τότε είναι εγγυημένο ότι **το μήκος θα έχει τιμή που θα είναι μεγαλύτερη από τη μεγαλύτερη θέση του πίνακα που περιέχει στοιχείο**.
- Έχουμε πρόσβαση στο property με τον τελεστή τελεία (.):

```
array_name.length
```

Παράδειγμα 3: array_length

```
let array = [1, 2, 3, 4, 5];
console.log(`${array} | length=${array.length}`);

array = [1, 2, , , 5];
console.log(`${array} | length=${array.length}`);

array[100] = 1;
console.log(`${array} | length=${array.length}`);
```

Ορολογία (preview των μαθημάτων Αντικειμενοστρεφούς Προγραμματισμού - OOP):

- Ο πίνακας (array) λέμε ότι είναι η κλάση (πρότυπο) από την οποία παράγονται αντικείμενα (συγκεκριμένοι πίνακες)
- Στην κλάση ορίζονται μέθοδοι (στατικές ή μη) που ορίζουν συμπεριφορά (ενέργειες επί του πίνακα)
- και χαρακτηριστικά (attributes ή members) που είναι μεταβλητές του αντικειμένου και ιδιότητες (properties) που έχουν ίδια συμπεριφορά με τις μεταβλητές, αλλά πρόσθετα μπορούν να γίνουν ενέργειες κατά την πρόσβαση σε αυτές.

Θέτοντας τιμή στο length:

- Αν είναι μικρότερη ή ίση από θέσεις στοιχείων του πίνακα, τότε τα στοιχεία αυτά αφαιρούνται από τον πίνακα.
- Αν είναι μεγαλύτερη από θέσεις στοιχείων του πίνακα, οι επιπλέον θέσεις θα επιστρέφουν undefined σε ενδεχόμενη πρόσβαση.

Παράδειγμα 4: array_length_attribute

```
let array = [1, 2, 3, 4, 5];
console.log(`${array} | length=${array.length}`);
array.length = 3;
console.log(`${array} | length=${array.length}`);
array.length = 10;
console.log(`${array} | length=${array.length}`);
console.log(array)
```

ΜΑΘΗΜΑ 6.1: Πίνακες — 1.2. Πίνακας ως αντικείμενο και ο τελεστής spread (…)

JavaScript psounis

Από τα προηγούμενα είναι εμφανές ότι:

- Ένα καλύτερο μοντέλο για να σκεφτόμαστε τον πίνακα της JavaScript είναι ως:
  - Έχει ένα χαρακτηριστικό, το μήκος (length) που είναι μεγαλύτερο από κάθε δείκτη θέσης.
  - Αποθηκεύονται μόνο οι δείκτες που έχουν κάποια τιμή (οι υπόλοιποι δεν αποθηκεύονται και επιστρέφουν undefined)
  - Έτσι είναι καλύτερα να έχουμε στο μυαλό μας ότι η τριάδα:
    - θέση 1: τιμή 5
    - θέση 4: τιμή 8
    - length: 6

    απεικονίζει τον πίνακα [,5,,,8,]

Με βάση αυτά δεν πρέπει να μας ξενίζει ότι:

- Αν κάνουμε **ανάθεση εκτός των ορίων του πίνακα**, θα αποθηκευτεί η τιμή και θα διορθωθεί αυτόματα το length.

Παράδειγμα 5: assignment

```
let array = [1, 2, 3, 4, 5];
array[10] = 4;
console.log(array.length)
console.log(array)
```

Διαγραφές Στοιχείων:

- Ο τελεστής delete (π.χ. delete array[pos]) εξαφανίζει τη θέση από το αντικείμενο.
- Λεπτή διαφορά με το array[pos] = undefined, όπου η θέση pos έχει πλέον τιμή undefined.

Παράδειγμα 6: delete

```
let array = [1, 2, 3, 4, 5];
delete array[2];
array[0]=undefined;
console.log(array);
```

Ο τελεστής spread (…):

- **Απλώνει τα στοιχεία του πίνακα** ως διαδοχικά στοιχεία μέσα σε έναν άλλο πίνακα:
  - Έτσι π.χ. αν έχουμε έναν πίνακα array
  - μέσα στις αγκύλες ενός άλλου πίνακα μπορούμε να απλώσουμε τα στοιχεία του array, γράφοντας **…array**
- (απλοποιεί το συντακτικό για να αντιγράψουμε τα στοιχεία του πίνακα μέσα σε έναν άλλο πίνακα)

Παράδειγμα 7: spread

```
let array = [1, 2, 3, 4, 5];
let array2 = [0, 0, ...array, 1, 1]
console.log(array2);
```

1.3. Εναλλακτικοί Τρόποι Κατασκευής Πίνακα

Εναλλακτικά μπορούμε να κατασκευάσουμε έναν πίνακα με τους εξής τρόπους:

- **Με τον κατασκευαστή Array([n])**
  - παίρνει προαιρετικό όρισμα το πλήθος των στοιχείων του πίνακα, π.χ. η ακόλουθη δήλωση κατασκευάζει έναν πίνακα με τα στοιχεία [1,2,3,4,5]:

```
let my_array = new Array(1, 2, 3, 4, 5);
```

- **Με τη στατική μέθοδο Array.of([n])**
  - παίρνει προαιρετικό όρισμα το πλήθος των στοιχείων του πίνακα, π.χ. η ακόλουθη δήλωση κατασκευάζει έναν πίνακα με 5 στοιχεία:

```
let my_array = Array.of(1, 2, 3, 4, 5);
```

- Λεπτή Διαφορά των δύο μεθόδων:
  - Ο κατασκευαστής Array (<ES2016) με όρισμα ακριβώς έναν αριθμό, κατασκευάζει έναν πίνακα του οποίου το μήκος είναι ίσο με τον αριθμό (χωρίς στοιχεία).
  - Αντίθετα, η Array.of (ES2016) διορθώνει το παραπάνω «πρόβλημα» και κατασκευάζει πίνακα μίας θέσης που περιέχει τον αριθμό.

Σημειώσεις:

- Με τον τελεστή new, κατασκευάζουμε νέο αντικείμενο μίας κλάσης (εδώ κλάση είναι η Array)
- «Στατική Μέθοδος» είναι συνάρτηση που προσδιορίζεται από την κλάση (και έχουμε πρόσβαση με τον τελεστή τελεία (.). Συντακτικό: **ονομα_κλάσης.όνομα_στατικής_μεθόδου(ορίσματα)**)

- **Με τη στατική μέθοδο: from**
  - παίρνει όρισμα άλλον πίνακα και τα στοιχεία «αντιγράφονται».

```
let my_array = Array.from(my_old_array);
```

Παρατηρήσεις:

- Παρατηρήστε (παράδειγμα) ότι με απλή καταχώριση (my_new_array = my_array, ΔΕΝ γίνεται αντιγραφή πίνακα)
- Τυπικά, η from() παίρνει όρισμα iterable (θα το μελετήσουμε σε επόμενο μάθημα => αντικείμενο που επιδέχεται επανάληψη)

Παράδειγμα 8: arrays_construct

```
console.log(new Array(1,2,3,4,5));
console.log(Array.of(1,3,5,7));

console.log(new Array(5));
console.log(Array.of(5));

let array1 = [1,2,3,4,5];
let array2 = array1;
array1[0]=6;
console.log(array2);
array2 = Array.from(array1);
array1[1]=7;
console.log(array2);
```

Ο τελεστής in:

- Σύνταξη **item in object**
- Εξετάζει (επιστρέφοντας true/false) αν ένα στοιχείο ανήκει σε ένα αντικείμενο.
- Μπορεί να χρησιμοποιηθεί στους πίνακες, για να εξετάσουμε αν μια θέση ανήκει σε έναν πίνακα.

Παράδειγμα 9: in

```
let array = [11, 12, 13, 14, 15];
console.log(1 in array);
console.log(!(2 in array));
```

Καταχώριση μέσω αποδόμησης πίνακα (destructuring assignment):

- Συντομογραφικός τρόπος για να αντικαταστήσουμε το συνηθισμένο τμήμα κώδικα (βλ. destructuring.html):

```
let array = [1, 2];
let x = array[0];
let y = array[1];
console.log(x,y);
```

- Με το ισοδύναμο (και πιο συντομογραφικό):

```
let array = [1, 2];
let [x, y] = array;
console.log(x, y);
```

Επεξήγηση destructuring assignment:

- Πρακτικά σημαίνει ότι μπορούμε να έχουμε πίνακα αριστερά από τον τελεστή ανάθεσης, όπου οι μεταβλητές-στοιχεία του πίνακα, αρχικοποιούνται με τις αντιστοιχες τιμές του πίνακα δεξιά του τελεστή ανάθεσης.

Destructuring Assigment και ο τελεστής rest:

- Το συντακτικό υποστηρίζει και τον τελεστή rest, ο οποίος μπορεί να μπει στην τελευταία θέση του πίνακα στα αριστερά του τελεστή ανάθεσης.

Παράδειγμα 10: destructuring_rest

```
let array = [1, 2, 3, 4, 5];
let [x, y, ...rest] = array;
console.log(x, y, rest);
```

Παράδειγμα 11: destructuring_special_cases

```
[x,y]=array; // first two
console.log(x,y);
[x,,y]=array; // first and third
console.log(x,y);
array=[1];
[x,y]=array;
console.log(x,y);
```

2. Διδιάστατοι πίνακες

Διδιάστατοι Πίνακες:

- Είναι και πάλι μια συλλογή αντικειμένων με δύο διαστάσεις, μία για τις γραμμές και μία για τις στήλες
- Η πρώτη διάσταση δηλώνει το πλήθος γραμμών και η δεύτερη το πλήθος στηλων.

Ουσιαστικά:

- Δεν υπάρχουν 2Δ πίνακες, αλλά ορίζουμε έναν πίνακα του οποίου κάθε στοιχείο είναι ένας 1Δ πίνακας

Δημιουργία 2Δ Πίνακα:

- Η δημιουργία μπορεί να γίνει με οποιοδήποτε συνδυασμό των κατασκευαστών που είδαμε σε προηγ. διαφάνειες:
  - πρώτα θα κατασκευάσουμε τον πίνακα γραμμών, π.χ.:

```
let my_array = new Array(n);
```

  - έπειτα για κάθε γραμμή i, κατασκευάζουμε έναν πίνακα με τα στοιχεία που περιέχει η γραμμή.

```
array[i] = new Array(m);
```

  - rows, cols: Πλήθος γραμμών και στηλών αντίστοιχα.

Χρήση Μεταβλητών 2Δ Πίνακα:

- Η αρίθμηση τόσο των γραμμών όσο και των στηλών ξεκινά από το 0. Τα ονόματα των μεταβλητών πχ. για ένα 2x3 πίνακα είναι:
  - arrayName[0][0], arrayName[0][1], arrayName[0][2]
  - arrayName[1][0], arrayName[1][1], arrayName[1][2]

Παράδειγμα 12: array2d.java

```
// init
let array = new Array(5);
for (let i=0; i<array.length; i++) {
  array[i] = new Array(6);
  for (let j=0; j<array[i].length; j++)
    array[i][j] = 0;
}

// do something
for (let i=0; i<array.length; i++)
  for (let j=0; j<array[i].length; j++)
    array[i][j] = i+j;

// print them
for (let i=0; i<array.length; i++) {
  let s = "";
  for (let j=0; j<array[i].length; j++)
    s += array[i][j] + " ";
  console.log(s);
}
```

Παρατηρήσεις:

- Μπορούμε να δεσμεύουμε χώρο για διαφορετικό πλήθος στοιχείων για κάθε γραμμή.
- Αντίστοιχα μπορούμε να δηλώσουμε 3Δ, 4Δ πίνακες κ.λπ.

3. Επανάληψη επί Πινάκων

- Εκτός από το «κλασσικό» συντακτικό της for (init; cond; step), προσφέρονται και δύο ακόμη παραλλαγές: Οι for..of και for..in

for..of: Συντάσσεται ως:

```
for (element of array)
  // do something with element
```

- Το element θα πάρει διαδοχικά τις τιμές του πίνακα (ακόμη κι αν είναι undefined - σε αραιούς πίνακες)
- Ίδιοι κανόνες για τη χρήση άγκιστρων σε πολλές εντολές
- Συνηθισμένο η μεταβλητή να δηλώνεται επί τόπου:

```
for (let element of array)
  // do something with element
```

Επίσης με το συντακτικό:

```
for (let index, element of array.entries())
  // do something with index and element
```

- Έχουμε ταυτόχρονα και το δείκτη και την τιμή κάθε στοιχείου

Παράδειγμα 13: forOf

```
let array = [1, 2, , 4, 5];
for (let elem of array)
  console.log(elem);

for (let [ind, elem] of array.entries())
  console.log(ind, elem);
```

for..in: Συντάσσεται ως:

```
for (index in array)
  // do something with index
```

- Το index θα πάρει διαδοχικά τους δείκτες (των θέσεων του πίνακα)
  - Ωστόσο δεν περιλαμβάνονται οι δείκτες που έχουν τιμή undefined
- Ίδιοι κανόνες για τη χρήση άγκιστρων σε πολλές εντολές
- Συνηθισμένο η μεταβλητή να δηλώνεται επί τόπου:

```
for (let index in array)
  // do something with index
```

Παράδειγμα 14: forIn

```
let array = [1, 2, , 4, 5];
for (let elem of array)
  console.log(elem);
```

Παρατηρήσεις:

- Η for..of εισήχθηκε στην ES6, ώστε να δουλεύει με τους πίνακες έτσι όπως θα έπρεπε να δουλευεί η for..in (η οποία χρησιμοποιείται κυρίως για επανάληψη επί αντικειμένων - επόμενο μάθημα)
- Συνεπώς, αν θέλουμε πιο εύκολα από το τυπικό for να κάνουμε επανάληψη επί των τιμών του πίνακα για να «δούμε» τα περιεχόμενα και όχι να τα τροποποιήσουμε, τότε το for..of είναι η συνιστώμενη επιλογή.

4. Ασκήσεις στους Πίνακες

Άσκηση 1:

Δηλώστε έναν πίνακα 10 θέσεων και αρχικοποιήστε τον με ακεραίους της αρεσκείας σας. Έπειτα εξετάστε αν υπάρχει ο αριθμός 12 σε αυτούς.

- Εαν υπάρχει, να βγάζει κατάλληλο μήνυμα και να τυπώνεται η θέση στην οποία βρίσκεται η πρώτη εμφάνισή του.
- Εαν δεν υπάρχει, να βγάζει κατάλληλο μήνυμα.

Άσκηση 2:

Δηλώστε έναν πίνακα 4x3 και αρχικοποιήστε τον με τιμές της αρεσκείας σας. Στην συνέχεια:

- Να τυπώνεται το άθροισμα των στοιχείων κάθε γραμμής.
- Να τυπώνεται το άθροισμα των στοιχείων κάθε στήλης.