JavaScript

psounis

ΠΕΡΙΕΧΟΜΕΝΑ:

1. Αντικείμενα
    1. Επεξεργασία properties
    2. Συναρτήσεις ως Μέλη και η λέξη-κλειδί this
    3. Ο Τελεστής in και Ανάθεση μέσω Αποδόμησης
    4. Επανάληψη επί Αντικειμένου
    5. Αντιγραφή Αντικειμένων και Ιδιοτήτων
    6. JSON: Σειριοποίηση/Αποσειρίοποίηση
2. Ασκήσεις

Γεώργιος Κ.

Σμαραγδένιος Χορηγός Μαθήματος

Μιχάλης Μπ.

Σμαραγδένιος Χορηγός Μαθήματος

ΜΑΘΗΜΑ 8.1: Αντικείμενα

1. Αντικείμενα

JavaScript psounis

Αντικείμενο (Object):

- **Είναι μία μη διατεταγμένη συλλογή ζευγών κλειδιού - τιμής:**
  - Τα ζεύγη κλειδιού-τιμής ονομάζονται και **properties.**
  - Το **κλειδί (key)** είναι όνομα (το πιο σύνηθες) ή συμβολοσειρά ή σύμβολο
  - Η **τιμή(value)** είναι οποιαδήποτε τιμή:
    - μπορεί να είναι primitive (βλ. μάθ.2)
    - μπορεί να είναι αντικείμενο.
- **Σύνταξη Object Literal:**
  - Μέσα σε άγκιστρα, τα ζεύγη κλειδιού - τιμής είναι χωρισμένα με κόμματα.
    - Μεταξύ κλειδιού και τιμής, βάζουμε άνω κάτω τελεία.

Παράδειγμα 1: object_literal

```
let object1 = {
   x: 1,
   y: 2
};
```

```
let object2 = {
   attribute: "value",
   "complex attribute": {},
};
```

Παρατηρήσεις στη σύνταξη:

- Στο τελευταίο property μπορεί να υπάρχει κόμμα στο τέλος
- {}: είναι το κενό αντικείμενο
- Ο κενός χώρος (whitespace) δεν έχει σημασία, αλλά συνηθίζεται τα αντικείμενα να έχουν την παραπάνω στοίχιση.

Σημείωση (Εφόσον υπάρχει εμπειρία σε άλλες γλώσσες προγ/μου):

- Το αντικείμενο είναι το αντίστοιχο της δομής δεδομένων: λεξικό (στην Python) και του Map (στη Java ή τη C++). Εκφράζει την «αντιστοιχία»: κάθε (μοναδικό) κλειδί έχει ακριβώς μία τιμή.

Object Literal (>=ES2016)

- Στο συντακτικό που έχουμε δει ως τώρα, επιτρέπεται να έχουμε έκφραση, μόνο στην τιμή του property, η οποία υπολογίζεται σε τιμή (βλ. object1, παράδειγμα 2).
- Το συντακτικό των literals επεκτάθηκε ώστε να υποστηρίζει:
  - Ορισμός του property, βάζοντας μόνο ένα όνομα: Το κλειδί θα έχει το ίδιο όνομα, ενώ η τιμή θα είναι η τιμή του ονόματος (βλ. object2, παράδειγμα 2)
  - Ορισμός του κλειδιού, βάζοντας υπολογιζόμενη έκφραση σε συμβολοσειρά (ή σύμβολο): Απαιτείται η έκφραση να είναι σε αγκύλες (βλ. object3, παράδειγμα 2)

Παράδειγμα 2: object_literal2

```
let a = 1;
let b = 2;
let object1 = {
   x: a+1,
   y: b+2
};
console.log(object1);
```

```
let object2 = {
   a,
   b
};
console.log(object2);
```

```
let object3 = {
   ["attribute"+1]: 1,
   ["attribute"+2]: 2
};
console.log(object3);
```

Διάβασμα τιμής κλειδιού:

- Παίρνουμε την τιμή ενός κλειδιού του αντικειμένου (ή undefined, αν το κλειδί δεν υπάρχει στο αντ/νο), ως εξής:
  - **object.key**
  - **object["key"]**

Παράδειγμα 3: property_read

```
let object = {
   x: 1,
   "a property": 2
};
console.log(object.x, object["x"]);
console.log(object["a property"]);
```

Δημιουργία property:

- Μπορούμε να δημιουργήσουμε property, γράφοντας:
  - **object.key = value**
  - **object["key"] = value**

Παράδειγμα 4: property_create

```
let object = {
   x: 1,
   y: 2
};
object.z = 3;
console.log(object);
```

Σημείωση:

- Αν το κλειδί υπάρχει ήδη, με το ίδιο συντακτικό, γίνεται ενημέρωση της τιμής του κλειδιού

Παράδειγμα 5: property_update

```
let object = {
   x: 1,
   y: 2
};
object.x = 3;
console.log(object);
```

Διαγραφή property:

- Χρησιμοποιούμε την delete (δεν προκαλεί σφάλμα αν το κλειδί δεν υπάρχει):
  - delete object.key
  - delete object["key"]

Παράδειγμα 6: property_delete

```
let object = {
   x: 1,
   y: 2
};
delete object.x;
console.log(object);
```

- (Μαθ. 7) Οι συναρτήσεις είναι και αυτές αντικείμενα, συνεπώς μπορούν να είναι τιμές σε properties (τις λέμε **μεθόδους**)

Παράδειγμα 7: this

```
let john = {
  firstname: "John",
  lastname: "Dutton",
  fullname: function() {
    return `${this.firstname} ${this.lastname}`;
  },
};

console.log(john.fullname());
```

Η λέξη-κλειδί this:

- Είναι αναφορά στο αντικείμενο του οποίου μέλος είναι η συνάρτηση. Διοχετεύεται έμμεσα στις συναρτήσεις-μέλη.
  - **Δηλαδή εδώ this===john**
  - Και απαιτείται για να έχουμε πρόσβαση στα κλειδιά των properties του αντικειμένου, σε συναρτήσεις - μέλη.

Σημείωση για το συντακτικό:

- Παρατηρήστε ότι χρησιμοποιούμε συναρτησιακή εκφραση κατά τον ορισμό της συνάρτησης.
- Δεν μπορούμε να χρησιμοποιήσουμε arrow function (για προχωρημένους λόγους(βλ. κλειστότητα (επόμενο μάθημα))

Σημείωση 2 για το συντακτικό:

- (>=ES6) Επιτρέπεται και η συντομογραφία του συντακτικού της συνάρτησης-μέλους, όπου παραλείπεται η άνω κάτω τελεία και η λέξη κλειδί function, π.χ. στο παράδειγμα 7:

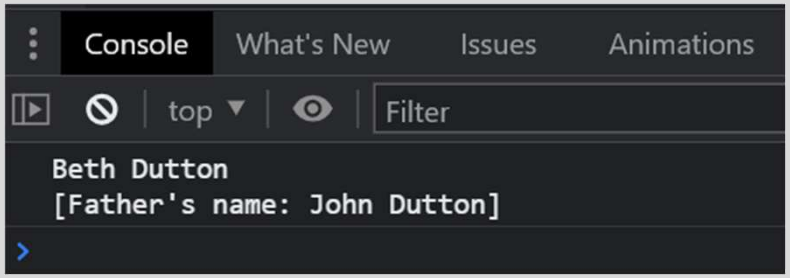
```
  fullname() {
    return `${this.firstname} ${this.lastname}`;
  }
```

Άσκηση 1:

- Ορίστε την “Beth Dutton” με τον ίδιο τρόπο όπως στο παραδειγμα 7.
- Επεκτείνετε το αντικείμενό της με ένα ακόμη property:
  - Όνομα property: father
  - Τιμή property: Το αντικείμενο john του παραδείγματος 7.

Άσκηση 2:

- Επεκτείνετε το αντικείμενο της άσκησης 1, με μία ακόμη μέθοδο, με όνομα fullInfo() που θα τυπώνει τις πληροφορίες του αντικειμένου ως εξής:



ΜΑΘΗΜΑ 8.1: Αντικείμενα — 1.3. Ο Τελεστής in και Ανάθεση μέσω Αποδόμησης

JavaScript psounis

Ο τελεστής in:

- Σύνταξη: **propertyName in object**
  - Ελέγχει αν το propertyName είναι κλειδί του αντικειμένου.
  - [Το propertyName πρέπει να είναι συμβολοσειρά όταν χρησιμοποιούμε τον τελεστή]

Παράδειγμα 8: in

```
let john = {
   firstname: "John",
   lastname: "Dutton",
   fullname: function() {
      return `${this.firstname} ${this.lastname}`;
   },
};
console.log("firstname" in john);
console.log("fullname" in john);
```

Ανάθεση μέσω Αποδόμησης:

- Αντίστοιχα με τους πίνακες, μπορούμε να γράψουμε συντομογραφικά την ανάθεση σε μεταβλητές (βλ. παράδειγμα 9)

Παράδειγμα 9: destructuring

```
let object = {
   x: 1,
   y: 2
};
let {x, y} = object;
```

Παρατήρηση:

- Στο παράδειγμα 9 βλέπουμε ότι οι μεταβλητές που ορίζονται, έχουν ίδιο όνομα με τα κλειδιά.
- Ωστόσο, μπορούμε να δηλώσουμε νέα όνόματα, με το συντακτικό του παραδείγματος 10

Παράδειγμα 10: destructuring2

```
let object = {
   x: 1,
   y: 2
};
let {x: a, y: b} = object;
```

- Και αν και κάπως περίπλοκο συντακτικά, επιτρέπεται η αποδόμηση και σε σύνθετα αντικείμενα (αντικείμενα που περιέχουν αντικείμενα)

Παράδειγμα 11: destructuring2

```
let object = {
   first: {
      a: 1,
      b: 2
   },
   second: {
      c: 3
   }
};
let {first: {a: x, b: y}} = object;
```

Επανάληψη με την for…in:

- Σύνταξη:

```
for (let propertyName in object) { …}
```

- Η μεταβλητή propertyName (όνομα της αρεσκείας μας), θα διαπεράσει τις συμβολοσειρές με τα κλειδιά των properties του αντικειμένου object

Παράδειγμα 12: for_in

```
let object = {
  x: 1,
  y: 2,
  sum() { return x+y; }
};
for (let propertyName in object)
  console.log(property, object[propertyName]);
```

Μέθοδοι του built-in αντικειμένου Object:

- Το αντικείμενο Object (βλ. επόμενο μάθημα) έχει χρήσιμες μεθόδους για την διαπέραση ενός αντικειμένου:
  - **Object.keys(object):** Πίνακας με τα ονόματα των μελών του αντικειμένου
  - **Object.values(object):** Πίνακας με τις τιμές των μελών του αντικειμένου
  - **Object.entries(object):** Πίνακας με ζεύγη κλειδιού-τιμής (ως πίνακες)

Παράδειγμα 13: object_methods

```
let object = {
  x: 1,
  y: 2,
  z: 1,
  sum() { return x+y+z; }
};

console.log(Object.keys(object));
console.log(Object.values(object));
console.log(Object.entries(object));
```

```
(4) ['x', 'y', 'z', 'sum']
  0: "x"
  1: "y"
  2: "z"
  3: "sum"
  length: 4
  [[Prototype]]: Array(0)
(4) [1, 2, 1, ƒ]
  0: 1
  1: 2
  2: 1
  3: ƒ sum()
  length: 4
  [[Prototype]]: Array(0)
(4) [Array(2), Array(2), Array(2), Array(2)]
  0: (2) ['x', 1]
  1: (2) ['y', 2]
  2: (2) ['z', 1]
  3: (2) ['sum', ƒ]
  length: 4
  [[Prototype]]: Array(0)
```

Σημείωση:

- Προσοχή ότι από αυτές τις μεθόδους επιστρέφεται πίνακας, οπότε η διαπέραση πρέπει να γίνεται με την for(… of …)

Παράδειγμα 14: for_of_entries

```
let object = {
  x: 1,
  y: 2,
  z: 1,
  sum() { return x+y+z; }
};

for (let [key, value] of Object.entries(object)) {
  console.log(key, value);
}
```

Αντιγραφή Αντικειμένου:

- Ο προφανής τρόπος (ob1 = ob2) θέτει το ob1 να αναφέρεται στο ίδιο αντικείμενο με το ob2 (δεν γίνεται δηλαδή αντιγραφή αντικειμένου)
- Μπορούμε όμως να χρησιμοποιήσουμε τον **spread operator** (αντίστοιχα με την αντιγραφή πίνακα):

```
{...object}
```

Παράδειγμα 15: copy_spread

```
let object = {
   x: 1,
   y: 2
};
let object2 = {...object};
```

Σημείωση:

- Μπορούμε να «απλώσουμε» τις ιδιότητες περισσότερων αντικειμένων, χωρίζοντας τις εφαρμογές του spread operator με κόμμα:

Παράδειγμα 16: copy_spread2

```
let object1 = { x: 1 };
let object2 = { y: 1 };
let object3 = {...object1, ...object2};
```

Η μέθοδος assign():

- Αντιγράφει τις ιδιότητες αντικειμένων σε ένα υφιστάμενο αντικείμενο (διατηρώντας τις ιδιότητες του υφιστάμενου αντικειμένου)
- Είναι στατική μέθοδος της κλάσης Object. Σύνταξη:
  - **Object.assign(target, source1, source2, …)**

Παράδειγμα 17: copy_assign

```
let object1 = {
   x: 1,
   y: 2
};
let object2={ z: 3};
Object.assign(object2, object1);
```

Σημείωση:

- Υπάρχουν προχωρημένες διαφορές των δύο τρόπων (getters/setters: επόμενο μάθημα)

Άσκηση 3 (βλ. και βίντεο περί shallow copy vs deep copy):

- Αντιγράψτε το αντικείμενο “beth” της άσκησης 2, σε ένα νέο αντικείμενο (π.χ. με όνομα jamie) χρησιμοποιώντας τον τελεστή spread.
- Τροποποιήστε το όνομα του πατέρα του δευτερου αντικειμένου.
- Τυπώστε τα αντικείμενα beth, jamie

Σειριοποίηση Αντικειμένου:

- Είναι η μετατροπή ενός αντικειμένου σε συμβολοσειρά.
- Built-in αντικείμενο: JSON
  - Περιέχει τη μέθοδο:

Μέθοδος	Επεξήγηση
stringify(object)	Μετατρέπει το αντικείμενο σε συμβολοσειρά

Κυριότερος Σκοπός:

- Τα δεδομένα του αντικειμένου γίνονται μία συμβολοσειρά, που μπορεί να αποσταλλεί σε έναν server [Ο πιο συνηθισμένος τρόπος επικοινωνίας client-server]

Λεπτομέρειες Σειριοποίησης Αντικειμένου:

- Το συντακτικό του **JSON (JavaScript Object Notation)** είναι υποσύνολο του συντακτικού του αντικειμένου.
  - Κλειδία: Είναι συμβολοσειρές
  - Τιμές: Επιτρέπονται αριθμοί, συμβολοσειρές, null, true και false (όχι undefined=>το μέλος θα παραλειφθεί). Οι τιμές επίσης μπορεί να είναι πίνακες και ανικείμενα.
  - Δεν ενσωματώνονται συναρτήσεις - μέλη

Σημείωση:

- Η stringify() επιδέχεται παραμετροποίησης (θα το δούμε πιο αναλυτικά σε επόμενο μάθημα)

Παράδειγμα 18: stringify

```
let beth = {
    firstname: "Beth",
    lastname: "Dutton",
    spouse: undefined,
    children: [],
    mood: null,
    fullname: function() {
        return `${this.firstname} ${this.lastname}`;
    },
    father: john
};
let json = JSON.stringify(beth);
console.log(json);
```

```
{
    "firstname": "Beth",
    "lastname": "Dutton",
    "children": [],
    "mood": null,
    "father": {
        "firstname": "John",
        "lastname": "Dutton"
    }
}
```

Αποσειριοποίηση Αντικειμένου:

- Η μετατροπή μίας JSON συμβολοσειράς σε αντικείμενο:
- Μέσω της μεθόδου του built-in αντικειμένου JSON:

Μέθοδος	Επεξήγηση
parse(jsonString)	Μετατρέπει το jsonString σε αντικείμενο

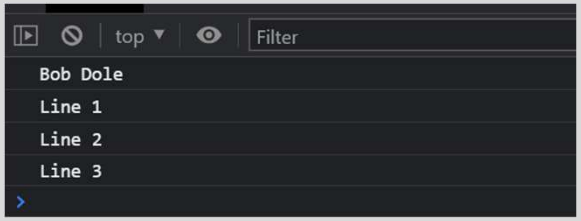
Παράδειγμα 18: stringify

```
... // συνέχεια του παραδείγματος 17
let object = JSON.parse(json);
console.log(object);
```

Άσκηση 4: Σκεφθείτε πως θα χρησιμοποιήσετε τις parse/stringify για να κατασκευάσετε βαθύ αντίγραφο ενός αντικειμένου.

Άσκηση 5:

- Ορίστε ένα αντικείμενο με όνομα billingDetails
- Περιέχει:
  - Το ονοματεπώνυμο (fullname)
  - Τρία properties με ονόματα addressLine1, addressLine2, addressLine3
  - Μία μέθοδο με όνομα print, που τυπώνει σε μία γραμμή το ονοματεπώνυμο και έπειτα τις τρεις γραμμές της διεύθυνσης, όπως ακολούθως:



[βλ. και βίντεο για μια ενδιαφέρουσα παραλλαγή της πρόσβασης στα μέλη addressLineXX]

Άσκηση 6:

- Μελετήστε το exercise12.html
- Βγάλετε συμπέρασμα για τον τύπο των δεδομένων των κλειδιών [βλ. και βίντεο]

```
let array = {};
array[1] = 11;
array[2] = 12;
array.length = 3;

for (let value of Object.values(array)) {
  console.log(value);
}

array["1"] = 4;
// array.1 = 10; // error
```

Συμπέρασμα:

- Όλα τα κλειδιά ερμηνεύονται ως συμβολοσειρες (με αλλαγές τύπου) [ή σύμβολα - βλ. επόμενο μάθημα]