



JavaScript

ΜΑΘΗΜΑ 8.3

PROPERTIES

ΠΕΡΙΕΧΟΜΕΝΑ:

1. Ιδιότητες των Properties
2. Έλεγχος Properties και Ιδιοτήτων τους
3. constructor και αντικείμενο - κατασκευαστής
 1. constructor και αλλαγή πρωτοτύπου
4. Accessors
5. Κληρονομικότητα
6. Ασκήσεις

Εμμανουήλ Α.

Σμαραγδένιος Χορηγός Μαθήματος

Βασιλική Κ.

Χρυσός Χορηγός Μαθήματος

Properties

- Είδαμε ότι κάθε αντικείμενο αποτελείται από properties.
- Μαθαίνουμε τώρα, ότι κάθε property έχει τρεις ιδιότητες:
 - **writable**: Καθορίζει αν μπορούμε να (επανα)γράψουμε την τιμή του property
 - **enumerable**: Αν το property θα εμφανίζεται σε for...in επαναλήψεις
 - **configurable**: Αν το property μπορεί να διαγραφεί και οι ιδιότητές του μπορούν να τροποποιηθούν.
- By default, τα νέα properties που ορίζουμε έχουν και τις τρεις παραπάνω ιδιότητες.

Ορισμός property που δεν έχει την default συμπεριφορά:

- Χρησιμοποιούμε τη μέθοδο του αντικειμένου Object:

| Μέθοδος | Επεξήγηση |
|---|--|
| defineProperty (object, property, descriptor) | <p>Ορίζουμε property (με όνομα: τη συμβολοσειρά που θέτουμε στο 2^ο όρισμα) στο αντικείμενο object, με τις ρυθμίσεις που περιγράφονται στο αντικείμενο descriptor.</p> <p>Το descriptor είναι αντικείμενο με properties:</p> <ul style="list-style-type: none"> • value: Η τιμή του property • writable, enumerable, configurable, με τιμές true/false (default: true) |

Παράδειγμα 1: property non enumerable

```
let object = {
  x: 1
};

Object.defineProperty(object, "y", {
  value: 2,
  enumerable: false
});

for (let property in object)
  console.log(property, object[property]);
```

Παράδειγμα 2: property non writable-configurable

```
...
Object.defineProperty(object, "y", {
  value: 2,
  writable: false,
  configurable: false
});
object.y = 3; // error -- non writable
delete object.y; // error -- non configurable
```

Σημείωση:

- Το Object έχει και τη μέθοδο defineProperties, με παραμέτρους (1) το αντικείμενο, (2) ένα αντικείμενο με μέλη key:value (key=όνομα property, value=descriptor)

ΜΑΘΗΜΑ 8.3: Properties

2. Έλεγχος Properties και Ιδιοτήτων τους

Έλεγχος Ιδιοτήτων Properties:

- Μέσω του built-in αντικειμένου Object:

| Μέθοδος | Επεξήγηση |
|---|---|
| getOwnPropertyNames(obj) | Πίνακας με τις ιδιότητες του ίδιου του αντικειμένου (και όχι της σειράς πρωτοτύπων του) |
| getOwnPropertyDescriptor(obj, propertyName) | Αντικείμενο-descriptor του property propertyName |
| getOwnPropertyDescriptors(obj) | Πίνακας με ζεύγη property name - descriptor του obj |

- Μέσω του πρωτοτύπου του αντικείμενου Object:

| Μέθοδος | Επεξήγηση |
|------------------------------------|--|
| hasOwnProperty(propertyName) | Αν το propertyName είναι property του αντικειμένου (και όχι της σειράς πρωτοτύπων του) |
| propertyIsEnumerable(propertyName) | Αν το propertyName έχει την ιδιότητα enumerable |

- Ενώ ελέγχουμε ότι όλα τα properties προέρχονται από κάποιο πρωτότυπο (ελέγχοντας ότι το αντικείμενο έχει ένα συγκεκριμένο πρωτότυπο) με τον τελεστή instanceof:
 - Σύνταξη: **object instanceof Prototype**

Παράδειγμα 3: check properties

```
function Superhero(name, strength, stamina) {  
  this.name = name;  
  this.strength = strength;  
  this.stamina = stamina;  
}  
Superhero.prototype.isVillain = false;  
  
let ironMan = new Superhero("Iron Man", 50, 1000);  
  
// object Object methods for testing properties  
console.log(Object.getOwnPropertyNames(ironMan));  
console.log(Object.getOwnPropertyDescriptors(ironMan));  
console.log(Object.getOwnPropertyDescriptor(ironMan, "name"));  
  
console.log(Object.getOwnPropertyNames(  
  Object.getPrototypeOf(ironMan)));  
  
// object ironMan methods for testing properties  
console.log(ironMan.hasOwnProperty("isVillain"),  
  ironMan.hasOwnProperty("name"));  
console.log(ironMan.propertyIsEnumerable("name"));  
  
// the instanceof operator  
console.log(ironMan instanceof Superhero);
```

Η ιδιότητα constructor:

- Κάθε αντικείμενο έχει την ιδιότητα constructor στο πρωτότυπό του.
 - Αν το αντικείμενο κατασκευάστηκε με συνάρτηση - κατασκευαστή: Η ιδιότητα είναι μια αναφορά στο αντικείμενο της συνάρτησης - κατασκευαστή.
 - Αν το αντικείμενο κατασκευάστηκε με literal, τότε η συνάρτηση - κατασκευαστής είναι το αντικείμενο Object.

Παράδειγμα 1: constructor

```
function Superhero(name, strength, stamina) { ... }  
Superhero.prototype.isVillain = false;  
let ironMan = new Superhero("Iron Man", 50, 1000);  
let voidMan = {};  
console.log(ironMan);  
console.log(voidMan);
```

```
▼ Superhero ⓘ  
  name: "Iron Man"  
  stamina: 1000  
  strength: 50  
  ▼ [[Prototype]]: Object  
    isVillain: false  
    ▶ constructor: f Superhero(name, strength, stamina)
```

```
▼ Object ⓘ  
  ▼ [[Prototype]]: Object  
    ▶ constructor: f Object()
```

Ενδιαφέρουσα χρήση αυτής της ιδιότητας:

- Μπορούμε να ελέγξουμε αν ένα αντικείμενο προέρχεται από κάποιον κατασκευαστή.

Παράδειγμα 4: constructor check type

```
console.log(ironMan instanceof Superhero,  
            ironMan instanceof Object);  
console.log(ironMan.constructor.name == "Superhero",  
            ironMan.constructor.name == "Object");
```

```
▶ | 🔍 | top ▼ | 👁 | Filter  
true true  
true false  
>
```

Βλέπουμε ότι:

- Ο τελεστής instanceof επιστρέφει true για οποιοδήποτε αντικείμενο στην αλυσίδα πρωτοτύπων.
- Ο constructor αναφέρεται απ' ευθείας (και μόνο) στο πρωτότυπο του αντικειμένου.

ΜΑΘΗΜΑ 8.3: Properties

3.1. Constructor και αλλαγή πρωτοτύπου

- Βλέπουμε ένα πρόβλημα που προκύπτει με την ιδιότητα constructor, όταν ορίζουμε το πρωτότυπο σε μια συνάρτηση - κατασκευαστή.
- Βλέπουμε το παρακάτω πρόγραμμα:

Παράδειγμα 5: constructor problem1

```
function Superhero(name, strength, stamina) {  
  this.name = name;  
  this.strength = strength;  
  this.stamina = stamina;  
}  
Superhero.prototype.isVillain = false;  
Superhero.prototype.memberOfShield = true;  
let ironMan = new Superhero("Iron Man", 50, 1000);
```

- Ορίσαμε την συνάρτηση - κατασκευαστή και βλέπουμε ότι στο αντικείμενο η ιδιότητα constructor του πρωτοτύπου είναι: το αντικείμενο-συνάρτηση Superhero.

```
▼ Superhero ⓘ  
  name: "Iron Man"  
  stamina: 1000  
  strength: 50  
  ▼ [[Prototype]]: Object  
    isVillain: false  
    memberOfShield: true  
    ► constructor: f Superhero(name, strength, stamina)
```

- Ωστόσο, θα μπορούσαμε να έχουμε πιο «μαζεμένο» τον ορισμό του πρωτοτύπου όπως φαίνεται στο ακόλουθο παράδειγμα:

Παράδειγμα 6: constructor problem2

```
Superhero.prototype = {  
  isVillain: false,  
  memberOfShield: true  
}
```

Το πρόβλημα:

- Επαναορίζοντας την αναφορά στο πρωτότυπο, χάνεται η τιμή της ιδιότητας constructor:

```
▼ Superhero ⓘ  
  name: "Iron Man"  
  stamina: 1000  
  strength: 50  
  ▼ [[Prototype]]: Object  
    isVillain: false  
    memberOfShield: true  
    ► [[Prototype]]: Object
```

Η λύση:

- Όταν ορίζουμε το πρωτότυπο ως αντικείμενο, πάντα θα θέτουμε την ιδιότητα constructor ίση με τη συνάρτηση - κατασκευαστή.

Παράδειγμα 7: constructor problem solution

```
Superhero.prototype = {  
  constructor: Superhero,  
  isVillain: false,  
  memberOfShield: true  
}
```

Accessors:

- Ειδική μορφή properties για προσθήκη λειτουργικότητας όταν θέτουμε/ανασύρουμε μία τιμή. Ορίζονται με ειδικό τρόπο μέσω:
 - **του getter:** Μία συνάρτηση μέλος που επιστρέφει την τιμή
 - Σύνταξη: **get propertyName() {...}**
 - **του setter:** Συνάρτηση - μέλος που θέτει την τιμή
 - Σύνταξη: **set propertyName(value) {...}**

Παράδειγμα 8: constructor

```
let object = {
  _data: 0,
  get data() {
    return this._data;
  },
  set data(value) {
    this._data = value;
  }
}
console.log(object.data);
object.data=1;
console.log(object.data);
```

Παρατήρηση 1:

- Properties που το όνομα τους ξεκινά με underscore: Συμβολίζουν ιδιωτικές μεταβλητές (στην πραγματικότητα δεν είναι ιδιωτικές, αλλά απλά μια σύσταση, να μην χρησιμοποιούνται εκτός του αντικειμένου)

Παρατήρηση 2:

- Property που έχει μόνο getter είναι read-only.
- Στο ακόλουθο παράδειγμα βλέπουμε και το πλήρες συντακτικό της defineProperty με accessors.
 - Σημειώστε ότι μπορούμε στο description object να χρησιμοποιήσουμε τους accessors και τα κλειδιά writable, enumerable, configurable (καθώς και το value) με κάποιον συνδυασμό που να είναι λογικός.
 - Π.χ. δεν είναι λογικό να έχουμε setter με writable=false

Παράδειγμα 9: constructor check type

```
let object = {}
Object.defineProperty(object, "data", {
  get() {
    return this._data;
  },
  set(value) {
    this._data = value;
  },
  enumerable: false,
});
console.log(object.data);
object.data=1;
console.log(object.data);
```


Κληρονομικότητα:

- Ένα αντικείμενο κληρονομεί τις ιδιότητες ενός άλλου αντικειμένου.
 - Έτσι, το παραγόμενο αντικείμενο (derived) δεν χρειάζεται να επαναορίσει τη λειτουργικότητα που έχουμε ορίσει στο βασικό αντικείμενο.
- Ένας τρόπος να επιτύχουμε κληρονομικότητα στη JavaScript:
 - Το βασικό αντικείμενο ορίζεται από τον κατασκευαστή και το πρωτότυπό του.
 - Το παραγόμενο αντικείμενο έχει ως πρωτότυπο ένα νέο βασικό αντικείμενο.
 - Προσοχή στο «νέο». Αλλιώς όλα τα παραγόμενα αντικείμενα, θα έχουν κοινό βασικό αντικείμενο.

Παράδειγμα xx: inheritance

```
function Base() {
  this.baseProperty = 1;
}

function Derived() {
  this.derivedProperty = 2;
}
Derived.prototype = new Base();

let d = new Derived();
```

Παρατήρηση:

- Ρητή κλήση κατασκευαστή:
 - Ο κατασκευαστής της παραγόμενης κλάσης καλό είναι να καλεί τον κατασκευαστή της βασικής κλάσης, για να μην επαναλαμβάνουμε τον ίδιο κώδικα.
 - Αυτό μπορεί να επιτευχθεί όπως στο ακόλουθο παράδειγμα:

Παράδειγμα xx: inheritance2

```
function Base(baseProperty) {
  this.baseProperty = baseProperty;
}

function Derived(baseProperty, derivedProperty) {
  this.derivedProperty = derivedProperty;
  Base.call(this, baseProperty);
}
Derived.prototype = new Base();
Derived.prototype.constructor = Derived;

let d = new Derived(1, 2);
console.log(d);
```

• Η συνάρτηση call(object, arguments)

- Είναι μέρος κάθε αντικειμένου - συνάρτησης
- Προκαλεί την κλήση της συνάρτησης όπου το this αναφέρεται στο object (άρα στο παράδειγμά μας, στην παραγόμενη κλάση)

ΜΑΘΗΜΑ 8.3: Properties

Άσκηση 1:

Ορίζουμε μια οντολογία του ζωϊκού βασιλείου:

- Ζώο: Έχει χαρακτηριστικά βάρος και ύψος
- Άλογο: Κληρονομεί το Ζώο και το επεκτείνει με:
 - το χρώμα του
 - την ούρά του (μήκος)
- Σκύλος: Κληρονομεί το Ζώο και το επεκτείνει με:
 - την ένταση γαβγίσματος (σε dB)
 - τη μέθοδο bark() που βγάζει έναν κατάλληλο ήχο
- Doberman: Κληρονομεί το Σκύλο και το επεκτείνει με:
 - τη μέθοδο run() που βγάζει κατάλληλο μήνυμα
- Bulldog: Κληρονομεί το Σκύλο και το επεκτείνει με:
 - το μέγεθος των αυτιών του
 - τη μέθοδο sleep() που βγάζει κατάλληλο μήνυμα

Έπειτα ορίστε ένα συγκεκριμένο άλογο, ένα Doberman και ένα bulldog:

- Τυπώστε το χρώμα του αλόγου
- Το Doberman γαβγίζει, τρέχει και μετά γαβγίζει
- Το Bulldog γαβγίζει, κοιμάται και μετά κοιμάται ξανα.

Ασκήσεις

Άσκηση 2:

Ορίστε συνάρτηση κατασκευαστή για την έννοια «Σημείο» του 2Δ χώρου:

- Ορίζεται μέσω των συντεταγμένων του x,y

Ορίστε συνάρτηση-κατασκευαστή για την έννοια «Ευθεία» του 2Δ χώρου:

- Αποτελείται από δύο σημεία
- Έχει ένα ακόμη property με όνομα length:
 - Θέλουμε να είναι μόνο-ανάγνωσης και να υπολογίζεται αυτόματα (π.χ. αν αλλάξει κάποιο σημείο) ως το μήκος της ευθείας από τον τύπο:
$$\sqrt{(A.x - B.x)^2 + (A.y - B.y)^2}.$$
- Σημείωση: Η τετραγωνική ρίζα υπολογίζεται με χρήση της συνάρτησης Math.sqrt(x)

Στο κυρίως πρόγραμμα, πρώτα ορίστε μια γραμμή με άκρα τα σημεία (1,1) και (4,5) και έπειτα τυπώστε το μήκος του ευθυγράμμου τμήματος