



# JavaScript

ΜΑΘΗΜΑ 8.4

## ΟΟΡ: ΚΛΑΣΕΙΣ

### ΠΕΡΙΕΧΟΜΕΝΑ:

1. Κλάσεις
  1. Κατασκευαστής
  2. Properties
    1. Ιδιωτικά Properties
  3. Μέθοδοι
  4. Στατικά Μέλη
2. Ασκήσεις

Εμμανουήλ Α.

Σμαραγδένιος Χορηγός Μαθήματος

Ανδρέας Γ.

Σμαραγδένιος Χορηγός Μαθήματος

**Κλάσεις (Classes)**

- Νέος τρόπος (ES6) για να ορίσουμε πρότυπα αντικειμένων που εμπνέεται από τις παραδοσιακές αντικειμενοστρεφείς γλώσσες (C++, Java κ.λπ.)
- Δήλωση Κλάσης:

```
class ClassName {  
  ...  
}
```

- Το όνομα της κλάσης συνηθίζεται να ξεκινά με κεφαλαίο γράμμα (σύμβαση, όχι συντακτική απαίτηση)

**Κατασκευαστής (constructor)**

- Η συνάρτηση που αρχικοποιεί το αντικείμενο της κλάσης.
- Έχει το σύνηθες συντακτικό της συνάρτησης όσον αφορά τις παραμέτρους, δεν έχει επιστρεφόμενη τιμή και απαιτείται να ονομάζεται constructor (λέξη-κλειδί)
- Εντίθενται στα άγκιστρα της κλάσης:

```
class Superhero {  
  constructor(name, strength, stamina) {  
    ...  
  }  
}
```

- και είναι η συνάρτηση που θα κληθεί όταν θα κατασκευάσουμε ένα καινούργιο αντικείμενο, με την new.

**Παράδειγμα 1: class**

```
class Superhero {  
  constructor(name, strength, stamina) {  
    this.name = name;  
    this.strength = strength;  
    this.stamina = stamina;  
  }  
}  
  
let ironMan = new Superhero("Iron Man", 50, 1000);  
let batman = new Superhero("Batman", 150, 1200);  
console.log(ironMan, batman);
```

**Παρατήρηση:**

- Ο ορισμός της κλάσης είναι συντακτική ζάχαρη. Στο παρασκήνιο γίνεται ακριβώς το ίδιο με τις συναρτήσεις - κατασκευαστές:

```
▼ Superhero {name: 'Iron Man', strength: 50, stamina: 1000} ⓘ  
  name: "Iron Man"  
  stamina: 1000  
  strength: 50  
  ▼ [[Prototype]]: Object  
    ► constructor: class Superhero  
    ► [[Prototype]]: Object
```

- (Δημιουργήθηκε αντικείμενο που έχει πρωτότυπο με ιδιότητα constructor τη συνάρτηση - κατασκευαστή)

**Σημείωση:**

- Αν δεν γράψουμε constructor, δημιουργείται αυτόματα ο default constructor: constructor() {}

Properties Αντικειμένων Κλάσης

- Τα properties των παραγόμενων αντικειμένων της κλάσης μπορούν να δηλωθούν με δύο διαφορετικούς τρόπους:
  - Α' τρόπος: Μέσω του this στον κατασκευαστή, όπως είδαμε στην προηγούμενη διαφάνεια π.χ.:

```
class ClassName {  
  constructor(...) {  
    this.property = value;  
  }  
  ...  
}
```

- Β' τρόπος: Βάζοντας το όνομα ως μέλος στην κλάση (είτε με αρχικοποίηση, είτε χωρίς)

```
class ClassName {  
  property;  
  property2 = initValue;  
  ...  
}
```

- Όπως είδαμε, με τον τελεστή new παράγεται ένα σύνθητος αντικείμενο της JS, οπότε ισχύουν όλα όσα έχουμε μάθει για τα properties, π.χ.:
  - Έχουμε πρόσβαση στην τιμή του property με τον τελεστή '.'
  - Μπορούμε να διαγράψουμε ένα property με την delete, κ.λπ.

Παράδειγμα 2: properties

```
class Superhero {  
  stamina;  
  strength = 0;  
  constructor(name) {  
    this.name = name;  
  }  
}
```

```
let ironMan = new Superhero("Iron Man");  
ironMan.stamina = 1000;  
console.log(ironMan);
```

```
▼ Superhero {stamina: 1000, strength: 0, name: 'Iron Man'} ⓘ  
  name: "Iron Man"  
  stamina: 1000  
  strength: 0  
  ▼ [[Prototype]]: Object  
    ► constructor: class Superhero  
    ► [[Prototype]]: Object
```

Σημείωση:

- Μπορούμε να ορίσουμε properties μέσω accessors όπως είδαμε στο μάθημα 8.3.

Παράδειγμα 3: properties2

```
class Superhero {  
  ...  
  _stamina;  
  get stamina() {  
    return this._stamina;  
  }  
  set stamina(stamina) {  
    this._stamina = stamina  
  }  
}
```

Ιδιωτικά Properties

- Όλα τα properties που κατασκευάσαμε ως τώρα είναι δημόσια (public)
  - που σημαίνει ότι είναι προσβάσιμα έξω από την κλάση με τον τελεστή '.'
- Σε μία κλάση μπορούμε να έχουμε και ιδιωτικά properties (private)
  - Είναι properties που είναι ορατά μόνο μέσα στην κλάση (τυχόν πρόσβαση έξω από την κλάση θα προκαλέσει λάθος)
  - Ορίζουμε ένα ιδιωτικό property βάζοντας # μπροστά από το όνομά του.

Προσοχή:

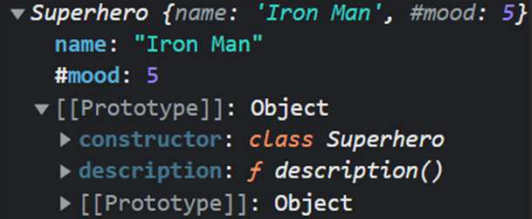
- Τα private properties πρέπει να έχουν δηλωθεί στην κλάση (δεν γίνεται να δηλωθούν απ' ευθείας στον constructor)

Παράδειγμα 5: properties2

```
class Superhero {  
  // #mood; // without this it doesn't work  
  
  constructor(name, mood) {  
    this.name = name;  
    this.#mood = mood;  
  }  
  ...  
}
```

Παράδειγμα 4: private

```
class Superhero {  
  #mood = 5;  
  constructor(name) {  
    this.name = name;  
  }  
  description() {  
    return `${this.name} (mood: ${this.#mood})`;  
  }  
}  
  
let ironMan = new Superhero("Iron Man");  
console.log(ironMan, ironMan.description());  
// console.log(ironMan.#mood); // error
```



```
▼ Superhero {name: 'Iron Man', #mood: 5}  
  name: "Iron Man"  
  #mood: 5  
  [[Prototype]]: Object  
    ► constructor: class Superhero  
    ► description: f description()  
    ► [[Prototype]]: Object
```

Άσκηση 1:

- Υλοποιήστε εκ νέου την Άσκηση 2/Μάθημα 8.3, αλλά αυτήν τη φορά, τα Point/Line να είναι κλάσεις
  - Οι μεταβλητές - μέλη να είναι ιδιωτικές.
  - Επί αυτών να ορίζονται getters/setters
- Ελέγξτε ότι το πρόγραμμα έχει την ίδια λειτουργικότητα, με την νέα υλοποίηση

### Μέθοδοι:

- Ορίζουμε μεθόδους στην κλάση μας, παραθέτοντας τις στο σώμα της.
  - Οι συναρτήσεις είναι μέρος του πρωτοτύπου
  - Με την λέξη-κλειδί `this` έχουμε πρόσβαση στα `properties` του αντικειμένου.

### Παράδειγμα 6: methods

```
class Superhero {
  constructor(name, strength, stamina) {
    this.name = name;
    this.strength = strength;
    this.stamina = stamina;
  }
  fly() {
    return `${this.name} is flying`;
  }
}

let ironMan = new Superhero("Iron Man", 50, 1000);
let batman = new Superhero("Batman", 150, 1200);
console.log(ironMan, batman);
console.log(ironMan.fly(), batman.fly());
```

<p>▼ Superhero ⓘ</p> <p>name: "Iron Man"</p> <p>stamina: 1000</p> <p>strength: 50</p> <p>▼ [[Prototype]]: Object</p> <p>  ▶ constructor: class Superhero</p> <p>  ▶ fly: f fly()</p> <p>  ▶ [[Prototype]]: Object</p>	<p>▼ Superhero ⓘ</p> <p>name: "Batman"</p> <p>stamina: 1200</p> <p>strength: 150</p> <p>▼ [[Prototype]]: Object</p> <p>  ▶ constructor: class Superhero</p> <p>  ▶ fly: f fly()</p> <p>  ▶ [[Prototype]]: Object</p>
---	--

Iron Man is flying Batman is flying

### Άσκηση 2:

Ένας σκύλος χαρακτηρίζεται από:

- Το όνομά του
- Το είδος του
- Το βάρος του (default τιμή παραμέτρου: 10)
- Το μέλος mood (προσομοιώνει τη διάθεση του σκύλου: ελάχιστο: 5, μέγιστο: 10). Να αρχικοποιείται σε 5.

Αφού ορίσετε την κλάση με τα παραπάνω `properties`, ορίστε και την εξής συμπεριφορά (μέθοδοι της κλάσης):

- Μέθοδος `eat` (αυξάνει την διάθεση κατά 1)
- Μέθοδος `bark`:
  - Αν η διάθεση είναι πάνω από 5, να τυπώνεται "Woof Woof Woof"
  - Αλλιώς να τυπώνεται μόνο "Woof"
- Μέθοδος `walk`:
  - Αυξάνει τη διάθεση κατά 1

Ορίστε την κλάση `Dog` και ορίστε δύο στιγμιότυπα:

- Piko (10 κιλά – Terrier)
- Lassie (30 κιλά – Colley)

Έπειτα υλοποιήστε ένα σενάριο με τον σκύλο Πικο που χρησιμοποιεί τις παραπάνω μεθόδους

### Παρατήρηση:

- Η άσκηση αυτή είναι σύμπτυξη των ασκήσεων 1-2 του μαθήματος 8.2
- Παρατηρήστε ότι τα παραγόμενα αντικείμενα (με κατασκευαστή αντικειμένων (μάθημα 8.2) και κλάση (μάθημα 8.4) διαφέρουν μόνο στον `constructor` του πρωτοτύπου των αντικειμένων.

### Στατικά Μέλη:

- Properties που ορίζονται στο επίπεδο της κλάσης (κοινή τιμή για όλα τα αντικείμενα). Συχνές Χρήσεις:
  - Στατικές μεταβλητές που μετράνε μία κοινή ιδιότητα για όλα τα αντικείμενα
  - Στατικές μέθοδοι που επενεργούν πάνω σε αντικείμενα της κλάσης, είναι υποκατάστατο κατασκευαστών, κ.λπ.

### Παράδειγμα 6: static

```
class Superhero {
  static count = 0;
  constructor(name, strength, stamina) {
    this.name = name;
    this.strength = strength;
    this.stamina = stamina;
    Superhero.count++;
  }
  fly() {
    return `${this.name} is flying`;
  }
  static compare(hero1, hero2) {
    return hero1.strength - hero2.strength;
  }
}

let ironMan = new Superhero("Iron Man", 50, 1000);
let batman = new Superhero("Batman", 150, 1200);
if (Superhero.compare(batman, ironMan)) {
  console.log(`${batman.name} is stronger than ${ironMan.name}`);
}
console.log(`Superheroes in program: ${Superhero.count}`);
```

### Σημείωση:

- Τα στατικά μέλη ορίζονται ως properties της συνάρτησης κατασκευαστή

```
> batman
< ▼ Superhero {name: 'Batman', strength: 150, stamina: 1200} ⓘ
  name: "Batman"
  stamina: 1200
  strength: 150
  ▼ [[Prototype]]: Object
    ▼ constructor: class Superhero
      count: 2
      ► compare: f compare(hero1, hero2)
```

### Άσκηση 3:

Ένας κύκλος με κέντρο την αρχή των αξόνων περιγράφεται από την εξίσωση:  $x^2 + y^2 = c^2$ . Ορίστε την κλάση Circle στην οποία να ορίζονται:

- constructor: να αρχικοποιεί το c (αριθμός), metric (συμβολοσειρά «in» ή «cm»)
- Getter/Setter: Για το c
- inToCm: Μετατρέπει τις ίντσες (όρισμα) σε εκατοστά. 1in = 2,54cm)
- CmToIn: Μετατρέπει τα εκατοστά (όρισμα) σε ίντσες.
- perimeterCm: Property μόνο-ανάγνωσης που υπολογίζει και να επιστρέφει την περίμετρο του κύκλου σε εκατοστά (2πc, το π είναι σταθερά στο Math: Math.PI)
- perimeterIn: Property μόνο-ανάγνωσης που υπολογίζει και να επιστρέφει την περίμετρο του κύκλου σε ίντσες (2πc)

Σκεφθείτε ποια από τα παραπάνω μέλη μπορούν να είναι στατικά και έπειτα κατασκευάστε πρόγραμμα για τον έλεγχο της παραπάνω συμπεριφοράς.



### Σύνοψη:

- Δεδομένης μίας κλάσης και κατασκευής αντικειμένου της, π.χ. (βλέπε template.html)

```
class MyClass {
  #private_value;
  public_value;
  _accessed_value;
  constructor(value){
    this.#private_value = value;
  }
  get accessed_value() {
    return this._accessed_value;
  }
  set accessed_value(val) {
    this._accessed_value = val;
  }
  method() {
    return 0;
  }
  static static_value = 0;
  static static_method() {
    return 1;
  }
}
```

```
let object = new MyClass(0);
console.log(object);
```

```
▼ MyClass
  public_value: undefined
  _accessed_value: undefined
  #private_value: 0
  accessed_value: (...)
  ▼ [[Prototype]]: Object
    accessed_value: (...)
    ▼ constructor: class MyClass
      static_value: 0
      length: 1
      name: "MyClass"
      ► prototype: {constructor: f, method: f}
      ► static_method: f static_method()
      arguments: (...)
      caller: (...)
      [[FunctionLocation]]: template.html:18
      ► [[Prototype]]: f ()
      ► [[Scopes]]: Scopes[2]
      ► method: f method()
      ► get accessed_value: f accessed_value()
      ► set accessed_value: f accessed_value(val)
      ► [[Prototype]]: Object
```

- Properties -Τιμές (όχι συναρτήσεις):
  - Μη στατικά: Αποθηκεύονται στο αντικείμενο
  - Στατικά: Στον constructor του πρωτοτύπου
- Properties - Συναρτήσεις:
  - Μη στατικά: Αποθηκεύονται στο πρωτότυπο
  - Στατικά: Στον constructor του πρωτοτύπου

### Άσκηση 4:

Με την άσκηση αυτή εξερευνούμε το πόσο δυναμική είναι η JS στην τροποποίηση των αντικειμένων που έχουν παραχθεί από μία κλάση. Για τον σκοπό αυτό, συνεχίζουμε την άσκηση 2 (προσοχή, γράφουμε κώδικα αμέσως μετά από όσα έχουμε ήδη γράψει, τροποποιούμε τη συμπεριφορά ήδη υφιστάμενων αντικειμένων, δεν πειράζουμε καθόλου τον κώδικα που έχουμε ήδη γράψει):

- Ορίστε στον σκύλο Πίκο μία μέθοδο huntCats() που εμφανίζει ένα πληροφοριακό μήνυμα ότι ο συγκεκριμένος σκύλος κηνυγά γάτες.
- Ορίστε μία μέθοδο με όνομα sleep() η οποία να εμφανίζει ένα πληροφοριακό μήνυμα ότι ο συγκεκριμένος σκύλος κοιμάται.
  - Επιθυμούμε αυτή η μέθοδος να εφαρμοστεί και στους δύο σκύλους (να μην οριστεί ξεχωριστά σε κάθε σκύλο)
- Ορίστε μία στατική μέθοδο με όνομα newTerrier με όρισμα μια συμβολοσειρά (όνομα του νέου σκύλου).
  - Θα επιστρέφει ένα νέο αντικείμενο - Σκύλος όπου η ράτσα του θα είναι «Terrier» και το όνομα του θα είναι το όρισμα της μεθόδου.

Έπειτα ελέγξτε τις παραπάνω μεθόδους με κατάλληλες κλήσεις των παραπάνω μεθόδων.