ΠΕΡΙΕΧΟΜΕΝΑ:

Τριαντάφυλλος Μυλωνάς
Χρυσός Χορηγός Μαθήματος

Πάνος Γ.
Χρυσός Χορηγός Μαθήματος

Πρωτότυπα Αντικειμένων (Object Prototypes)

- (Σχεδόν) κάθε αντικείμενο έχει πρόσβαση στις ιδιότητες ενός άλλου αντικειμένου που καλείται **το πρωτότυπό του**
  - Αυτό γίνεται μέσω ενός property που περιέχεται στο αντικείμενο (όχι με απ' ευθείας πρόσβαση).
    - Ωστόσο μπορούμε να πάρουμε το αντικείμενο - πρωτότυπο με τη μέθοδο του **Object: getPrototypeOf(object)**
- Ενώ μπορούμε να θέσουμε το πρωτότυπο ενός αντικειμένου με τη μέθοδο του Object: **setPrototypeOf(object, prototypeObject)**

Παράδειγμα 1: object_prototype

```
let fly = {
   canFly: true
}

let ironMan = {
   heroName: "Stark",
};

Object.setPrototypeOf(ironMan, fly);

console.log(ironMan);
console.log(ironMan.canFly);
console.log(Object.getPrototypeOf(ironMan));
```

```
▼ {heroName: 'Stark'}
    heroName: "Stark"
  ▼ [[Prototype]]: Object
      canFly: true
    ▶ [[Prototype]]: Object
```

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

- Το πρωτότυπο φέρνει στο νου την κληρονομικότητα σε τυπικές OOP γλώσσες. Λέμε ότι τα πρωτότυπα είναι μηχανισμός κληρονομικότητας, αλλά δουλεύει με διαφορετικό τρόπο:
  - Με το πρωτότυπο έχουμε πρόσβαση ανάγνωσης σε ένα άλλο αντικείμενο.
    - (και όχι πρόσβαση ενημέρωσης των ιδιοτήτων του πρωτοτύπου)
  - Και υπάρχει η συντακτική διευκόλυνση: Όταν π.χ. γράφουμε ironMan.canFly
    - Πρώτα αναζητείται η ιδιότητα canFly στις ιδιότητες του αντικειμένου
    - Αν δεν το βρεί, αναζητά την ιδιότητα στο πρωτότυπο του, κ.ο.κ.

Άσκηση 1:

- Επεκτείνετε το παράδειγμα 1 (για να επαληθεύσουμε ότι δεν τροποποιούμε τις ιδιότητες του πρωτοτύπου):
  - Ορίστε ένα αντικείμενο με όνομα batman με τιμή στο heroName: "Wayne"
  - Ορίστε στο αντικείμενο batman: Το πρωτότυπό του να είναι fly
  - Αλλάξτε την τιμή fly.canFly σε false
  - Τυπώστε τα αντικείμενα ironMan, batman

Συνάρτηση - Κατασκευαστής Αντικειμένων (Constructor Function):

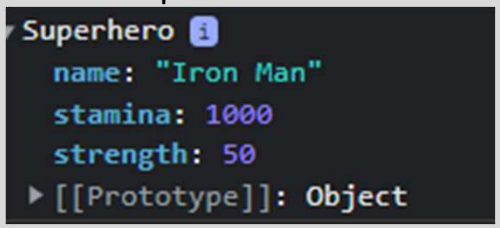
- Είναι μία συνάρτηση που κατασκευάζει αντικείμενα.
- Έχει ιδιωματική σύνταξη και χρήση που θα την δούμε με ένα παράδειγμα

Παράδειγμα 2: constructor

```
function Superhero(name, strength, stamina) {
  this.name = name;
  this.strength = strength;
  this.stamina = stamina;
}

let ironMan = new Superhero("Iron Man", 50, 1000);
console.log(ironMan);
```

κατασκευάζει το αντικείμενο:



Σημειώσεις:

- Σύμβαση (αλλά όχι συντακτική απαίτηση) είναι οι συναρτήσεις - κατασκευαστές ξεκινούν με κεφαλαίο γράμμα.
- Είναι το «ισοδύναμο» της κλάσης σε παραδοσιακές OOP γλώσσες

Παρατηρήσεις:

- Η συνάρτηση - κατασκέυαστής:
  - Ορίζει στο σώμα της properties του αντικειμένου.
- Κατασκευάζουμε το νέο αντικείμενο:
  - Χρησιμοποιώντας τον **τελεστή new** πριν από την κλήση της συνάρτησης - κατασκευαστή
  - (Χωρίς τον τελεστή new, δεν κατασκευάζεται αντικείμενο, αλλά προκαλείται συντακτικό λάθος)
- Το νέο αντικείμενο:
  - Έχει ως μέλη τα properties που ορίστηκαν στο σώμα της συνάρτησης - κατασκευαστή (με τον τελεστή this).

Άσκηση 2:

Ένας σκύλος χαρακτηρίζεται από:

- Το όνομά του
- Το είδος του
- Το βάρος του (default τιμή παραμέτρου: 10)

Κατασκευάστε τον κατασκευαστή σκύλων και ορίστε δύο στιγμιότυπα:

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

Κατασκευαστής και Πρωτότυπο Παραγόμενων Αντικειμένων:

- Η συνάρτηση - κατασκευαστής μπορεί να ορίσει και ένα αντικείμενο - πρωτότυπο.
  - Αυτό το αντικείμενο, θα είναι το πρωτότυπο οποιουδήποτε αντικειμένου δημιουργηθεί από τον κατασκευαστή.
- Το όνομα του αντικειμένου - πρωτοτύπου είναι:
  - **constructorName.prototype**
  - (Στο οποίο μπορούμε να προσθέσουμε properties, αφού πρώτα δηλώσουμε τον κατασκευαστή).

Παρατήρηση:

- Βλέπουμε στο παράδειγμα, ότι το πρωτότυπο του αντικειμένου που κατασκευάσαμε με τη συνάρτηση - κατασκευαστή, περιέχει μία αναφορά (**constructor**) στο αντικείμενο της συνάρτησης - κατασκευαστή.

Παρατήρηση 2:

- (Πολύ σημαντικό) Ο τελεστής this σε συναρτήσεις - μέλη του πρωτοτύπου, θα αναφέρεται στο νέο αντικείμενο.

Παράδειγμα 3: constructor-prototype

```
function Superhero(name, strength, stamina) {
  this.name = name;
  this.strength = strength;
  this.stamina = stamina;
}
Superhero.prototype.fly = function () {
  return "I am flying...!"
}
let ironMan = new Superhero("Iron Man", 50, 1000);
console.log(ironMan);
console.log(ironMan.fly());
```

παράγει το αντικείμενο:

```
▼ Superhero {
    name: "Iron Man"
    stamina: 1000
    strength: 50
  ▼ [[Prototype]]: Object
    ▶ fly: ƒ ()
    ▶ constructor: ƒ Superhero(name, strength, stamina)
    ▶ [[Prototype]]: Object
I am flying...!
```

Άσκηση 3:

Επεκτείνετε τον κατασκευαστή του σκύλου της προηγούμενης άσκησης με τα εξής:

- Το μέλος mood (προσομοιώνει τη διάθεση του σκύλου: ελάχιστο: 5, μέγιστο: 10). Να αρχικοποιείται σε 5.
- Τη μέθοδο eat (αυξάνει την διάθεση κατά 1)
- Τη μέθοδο bark:
  - Αν η διάθεση είναι πάνω από 5, να τυπώνεται “Woof Woof Woof”
  - Αλλιώς να τυπώνεται μόνο “Woof”
- Τη μέθοδο walk:
  - Αυξάνει τη διάθεση κατά 1

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

ΜΑΘΗΜΑ 8.2: Πρωτότυπα και Κατασκευαστές — 3. Κατασκευή με την Object.create()

JavaScript psounis

Δημιουργία Αντικειμένων με την Object.create()

- Η μέθοδος create() του αντικειμένου Object, μπορεί να χρησιμοποιηθεί για να κατασκευάσουμε νέα αντικείμενα (εναλλακτικός τρόπος σε σχέση με το object literal και τις συναρτήσεις - κατασκευαστές):

Μέθοδος	Επεξήγηση
create(proto [, propertiesObject])	Δημιουργεί νέο αντικείμενο με πρωτότυπο proto και μέλη αυτά που περιέχονται στο propertiesObject

Σημείωση:

- Η προαιρετική παράμετρος propertiesObject απαιτεί τα properties να έχουν δηλωθεί ως αντικείμενα (το βλέπουμε στο επόμενο μάθημα)

Παράδειγμα 5: object_create

```
let fly = {
   canFly: true
}

let ironMan = Object.create(fly)
console.log(ironMan);
```

Διαφορά Object.create από new

- Η new κάνει πρακτικά τρία βήματα:
  - Κατασκευάζει το καινούργιο αντικείμενο (κενό)
  - Θέτει το πρωτότυπο του αντικειμένου, αυτό του κατασκευαστή
  - Τρέχει τη συνάρτηση - κατασκευαστή
- Αντίθετα η Object.create()
  - Κατασκευάζει το καινούργιο αντικείμενο
  - Θέτει το πρωτότυπο του αντικειμένου, ίσο με το όρισμα
- (Άρα πρακτικά η Object.create() δεν τρέχει συνάρτηση - κατασκευαστή)

Παράδειγμα 6: object_create_vs_new

```
function Superhero(strength, stamina) {
   this.strength = strength;
   this.stamina = stamina;
}
Superhero.prototype = {
   strength: 50,
   canFly() { return true; }
}

let ironMan = new Superhero();
let batman = Object.create(Superhero.prototype);
let voidman = Object.create(null);
```

Σημείωση:

- Θέτοντας null στην παράμετρο proto, παίρνουμε αντικείμενο που δεν έχει πρωτότυπο (μοναδικός τρόπος για να το πετύχουμε)

Το αντικείμενο Object (built-in):

- Περιέχει πολλές χρήσιμες μεθόδους γενικής χρήσης
  - Ήδη έχουμε δει τις μεθόδους:
    - getPrototypeOf(object)
    - setPrototypeOf(object, prototypeObject)
    - assign(target, source1, …) (Μάθημα 8.1)
    - keys(ob), values(ob), entries(ob) (Μάθημα 8.1)
  - και έχει πολλές ακόμη που θα δούμε στα επόμενα μαθήματα.
- Η πιο σημαντική του ιδιότητα όμως είναι η εξής:
  - Είναι συνάρτηση - κατασκευαστής αντικειμένων
    - Ως κατασκευαστής (δεν απαιτείται η new)

Κλήση	Επεξήγηση
new Object() / Object()	Επιστρέφει {}
new Object(x) / Object(x)	Αν x==null ή x==undefined επιστρέφει {} Αν x==primitive επιστρέφει το αντίστοιχο αντικείμενο - primitive. Αν x==Αντικείμενο επιστρέφει το ίδιο το αντικείμενο - όρισμα.

  - **και ισχύει για το πρωτότυπό του (Object.prototype):**
    - **Τίθεται αυτόματα ως πρωτότυπο σε κάθε αντικείμενο του οποίου δεν έχει καθοριστεί ρητά άλλο πρωτότυπο.**

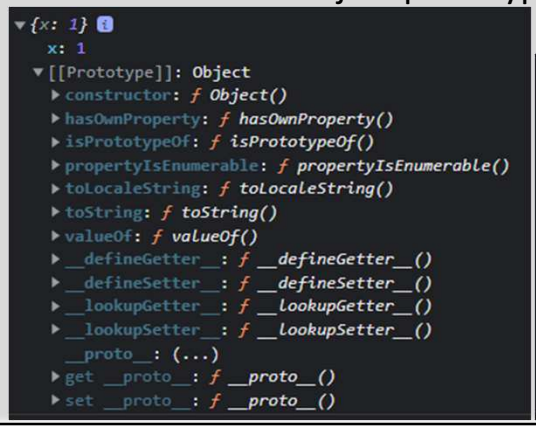
Παράδειγμα 4: constructor_object

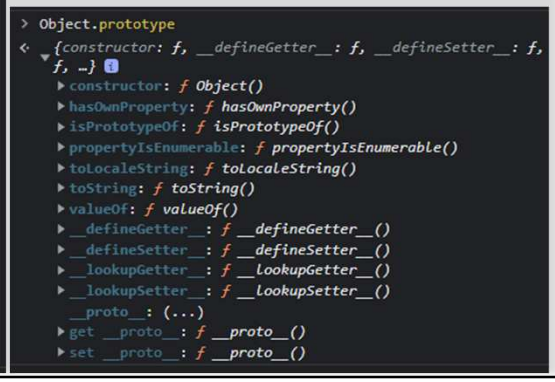
```
let a = new Object();
let b = new Object(function(){});
let c = new Object(null);
let d = new Object(1);
let e = new Object({x:1});
console.log(a,b,c,d,e);
console.dir(b);

let f = Object();
console.log(f);
```

Άσκηση (στην Κονσόλα):

- Δείτε ότι κάθε αντικείμενο που έχουμε κατασκευάσει και στο οποίο δεν έχουμε καθορίσει έμμεσα (π.χ. με τη new) ή άμεσα (π.χ. με την setPrototypeOf) έχει ως πρωτότυπο το Object.prototype:
  - Π.χ το αντικείμενο e του παραπάνω παραδείγματος και δίπλα το Object.prototype





Μέθοδοι του Object.prototype:

- Οι ακόλουθες μέθοδοι είναι μέλη του Object.prototype και έχουν ειδικές χρήσεις:

Μέθοδος	Επεξήγηση
toString()	Υπερφορτώνεται ώστε να επιστρέφει μία συμβολοσειρά που περιγράφει το αντικείμενο. Καλείται αυτόματα, οποτεδήποτε απαιτείται η μετατροπή του αντικειμένου σε συμβολοσειρά (π.χ. όταν κάνουμε μετατροπή σε συμβολοσειρά)
toLocaleString()	(Προχωρημένο) Χρησιμοποιείται σε κάποια αντικείμενα όπως το Date
valueOf()	Μετατρέπει ένα αντικείμενο σε μία τιμή - primitive. Καλείται αυτόματα όταν απαιτείται τιμή - primitive, π.χ. στον έλεγχο ισότητας, ανισότητας
toJSON()	Μετατρέπει το αντικείμενο σε συμβολοσειρά JSON (Η JSON.stringify() καλεί αυτήν τη μέθοδο, όποτε πρέπει να σειριοποιήσει το αντικείμενο)

Σημείωση:

- Υπάρχουν αρκετές ακόμη μέθοδοι του Object που θα δούμε στο επόμενο μάθημα.

Παράδειγμα 7: toString

```
let cow = {
   name: "molly",
   weight: 400,
   toString() {
      return `${this.name} (${this.weight}kgr)`;
   }
}
console.log(cow, `${cow}`);
```

Παράδειγμα 8: valueOf

```
...
let cow2 = {
   name: "helga",
   weight: 500,
   valueOf() {
      return this.weight;
   }
}
console.log(cow<cow2);
```

Παράδειγμα 9: toJSON

```
let cow = {
   toJSON() {
      return {weight: this.weight};
   }
}
console.log(JSON.stringify(cow));
```

Άσκηση 4:

Κατασκευάστε την συνάρτηση-κατασκευαστή: Student

- Μέλη: Όνομα, επώνυμο και Βαθμός
- Να επαναορίζει (στο πρωτότυπο) την toString ώστε ένας φοιτητής να εμφανίζεται ως:

```
Hector Owston (grade: 7)
```

Κατασκευάστε έναν πίνακα 10 φοιτητών:

- Αρχικοποιήστε το ονοματεπώνυμο με κάποιον τυχαίο τρόπο (π.χ. αναζητήστε στο google για τυχαία ονόματα και επώνυμα και κατασκευάστε λίστες π.χ. με 20 ονόματα και 20 επώνυμα. Έπειτα κάνετε τυχαία επιλογή από αυτά τα δεδομένα)

Κατασκευάστε μια συνάρτηση με όνομα gradeStudent:

- Θα παίρνει ως όρισμα έναν φοιτητή και θα τον βαθμολογεί με έναν τυχαίο αριθμό από το 0 έως 10.
- Χρησιμοποιήστε τη συνάρτηση ώστε να βαθμολογηθούν οι φοιτητές του πίνακα.

Τέλος, κατασκευάστε μία συνάρτηση με όνομα average.

- Να παίρνει ως όρισμα έναν πίνακα φοιτητών και να υπολογίζει το μέσο όρο των βαθμολογιών τους και να τον τυπώνει.
- Εφαρμόστε τη συνάρτηση στον πίνακα φοιτητών.

Παράδειγμα τελικής εκτύπωσης:

```
Warren Franklyn (grade: 1)
Warren Waldroup (grade: 4)
Lonnie Dickens (grade: 6)
Ferdie Dobbs (grade: 2)
Chance Harvey (grade: 6)
Ferdie Dobbs (grade: 7)
Tom Ashley (grade: 1)
Hector Waldroup (grade: 10)
Tom Poole (grade: 5)
Kayden Franklyn (grade: 1)
The average is 4.3
```

Άσκηση 5:

- Επιστρέψτε στο μάθημα 6.2 και μελετήστε ξανά τη συνάρτηση sort()
- Με βάση την «πράσινη» παρατήρηση, ταξινομήστε τον πίνακα φοιτητών της άσκησης 4, σε φθίνουσα σειρά βαθμολογίας.

Άσκηση 6:

(Α) Ένας τραπεζικός λογαριασμός ορίζεται από:

- τον αριθμό του (15-ψήφια συμβολοσειρά)
- τον κάτοχό του (ονοματεπώνυμο)
- το ποσό που περιέχει (πραγματικός)

Ορίστε κατασκευαστή τραπεζικών λογιαριασμών και έπειτα κατασκευάστε δύο τραπεζικούς λογαριασμούς (με στοιχεία της αρεσκείας σας).

Επεκτείνετε το λογαριασμό με μία μέθοδο (transfer) η οποία θα παίρνει σαν όρισμα έναν άλλο τραπεζικό λογαριασμό και ένα ποσό και θα μεταφέρει το ποσό σε αυτόν (εφόσον υπάρχει επαρκές υπόλοιπο) εκτυπώνοντας ένα περιεκτικό μήνυμα που να περιέχει τα ονόματα των κατόχων των λογαριασμών.

Επαληθεύστε τη σωστή λειτουργία της μεθόδου transfer.

(Β) Κατασκευάστε μία κλάση με όνομα Person η οποία περιέχει το ονοματεπώνυμο, την ηλικία και τον αριθμό ταυτότητας του ατόμου. Ορίστε δύο άτομα στο πρόγραμμα.

Τροποποιήστε τον ορισμό του λογαριασμού, ώστε να περιέχει ένα αντικείμενο τύπου Person, αντί να περιέχει απλά το ονοματεπώνυμο του κατόχου.

Σημείωση:

- Σε αυτήν την άσκηση, βλέπουμε την εφαρμογή του “has-a” (aggregation/composition) στις παραδοσιακές αντικειμενοστρέφεις γλώσσες