



## ΜΑΘΗΜΑ 3

# ΣΤΑΘΕΡΕΣ ΚΑΙ ΙΔΙΑΙΤΕΡΟΤΗΤΕΣ ΤΗΣ JS

### ΠΕΡΙΕΧΟΜΕΝΑ:

1. Σταθερές
2. Ρητές Ματατροπές Τύπων
3. Ιδιαιτερότητες της JS
  1. Ερωτηματικά
  2. Χρήση της var και hoisting
  3. Η οδηγία "use strict"
4. Τρόπος Εκτέλεσης Scripts
  1. Περιβάλλον Εκτέλεσης
  2. Το αντικείμενο window

Νικόλαος Βασιλειάδης

Σμαραγδένιος Χορηγός Μαθήματος

Σπύρος Καμινιώτης

Σμαραγδένιος Χορηγός Μαθήματος

## ΜΑΘΗΜΑ 3: Σταθερές και Ιδιαιτερότητες της JS

### Σταθερές:

- Δηλώνουμε ένα όνομα που είναι σταθερά ως εξής:

```
const constantName = value;
```

- const: Λέξη-κλειδί
- constantName: Όνομα της αρεσκείας μας (ίδιοι κανόνες ονοματοδοσίας όπως στις μεταβλητές)
- value: literal (βλ. προηγούμενο μάθημα)
- Η τιμή μίας σταθεράς δεν μπορεί να αλλάξει.**

### Παράδειγμα 1: const.html

```
const x = 1;  
console.log(x);  
x = 2; // error
```

1

```
✖ Uncaught TypeError: Assignment to constant variable.  
at const.html?_ijt=6f7t...RELOAD_ON_SAVE:15:4
```

### Παρατηρήσεις:

- Γενικά χρησιμοποιούμε σταθερές για δεδομένα που δεν πρόκειται να αλλάξουν.
- Σκεπτικό γλώσσας με δυναμικούς τύπους: Φτιάξε το αντικείμενο - τιμή και έπειτα «δέσε» το όνομα πάνω στην τιμή. Στις σταθερές το όνομα θα αναφέρεται μόνο στην συγκεκριμένη τιμή.

## 1. Σταθερές

### Άσκηση 1:

Στο ακόλουθο πρόγραμμα (exercise02\_initial.html) υπογίζεται η περίμετρος και το εμβαδόν ενός κύκλου:

```
let pi = 3.14;  
let radius = 1;  
  
let area = pi*radius*radius;  
let circumference = 2*pi*radius;  
  
console.log("Circle-1");  
console.log("area: " + area);  
console.log("circumference: " + circumference);
```

Ποια από τις παραπάνω μεταβλητές θα μπορούσε να δηλωθεί ως σταθερά;

### Οι δύο σχολές σταθερών:

- Α' σχολή: Δήλωσε ως σταθερά οποιαδήποτε τιμή δεν αλλάζει στο πρόγραμμα
- Β' σχολή: Δήλωσε ως σταθερά μόνο ότι είναι «εννοιολογικά» αμετάβλητο (όπως π.χ. στο παραπάνω πρόγραμμα το π)

### Ρητές Μετατροπές Τύπων:

- Είναι η δυνατότητα (μέσω καθολικών μεθόδων) να μετατρέψουμε μία υφιστάμενη τιμή κάποιου τύπου δεδομένων, στην αντίστοιχη ενός άλλου τύπου δεδομένων.
  - Π.χ. να πάρουμε από την τιμή-συμβολοσειρά "1" την αντίστοιχη τιμή-αριθμό 1.

### Παρατήρηση:

- Λέμε ότι κάνουμε ρητή (explicit) μετατροπή τύπου γιατί θα καλέσουμε ρητά μια μέθοδο που κάνει τη μετατροπή.
- Σε αντίθεση με τις έμμεσες (implicit) μετατροπές τύπου που κάνει μόνη της η JS (και που θα δούμε αναλυτικά στα αμέσως επόμενα μαθήματα).
  - Π.χ. η σύγκριση 1=="1" θα είναι truthy, αφού η JS αυτόματα, μετατρέπει τη συμβολοσειρά "1" στην αριθμητική τιμή 1.

### Καθολικές Μέθοδοι Μετατροπής:

Μέθοδος	Επεξήγηση
Number(value)	Μετατρέπει την τιμή σε αριθμό
String(value)	Μετατρέπει την τιμή σε συμβολοσειρά
Boolean(value)	Μετατρέπει την τιμή σε λογική μεταβλητή
BigInt(value)	Μετατρέπει την τιμή σε μεγάλο ακέραιο

### Παράδειγμα 2: conversions.html

```
console.log(Number("1"), Number("3.14"));
console.log(Number(true), Number(false));
console.log(Number(123n));
console.log(Number(null), Number(undefined), Number("str"));
console.log("-----");
console.log(String(1), String(3.14));
console.log(String(true), String(123n), String(null),
               String(undefined));
console.log("-----");
console.log(Boolean(1), Boolean(0), Boolean(3.14));
console.log(Boolean("1"), Boolean(""));
console.log(Boolean(0n), Boolean(1n), Boolean(null), Boolean(undefined));
console.log("-----");
console.log(BigInt(1), BigInt(0));
console.log(BigInt("1"), BigInt(null), BigInt(undefined));
```

### Υπενθύμιση:

- Falsy Τιμές: 0, null, undefined, "", NaN

### Παρατήρηση:

- Σπάνια κάποια μετατροπή θα προκαλέσει σφάλμα. Σχεδόν όλες οι μετατροπές επιστρέφουν μία τιμή.
- Όπως π.χ. η ανισόρροπη μετατροπή της συμβολοσειράς "str" σε αριθμό επιστρέφει NaN

#### Χωρισμός Εντολών με Ερωτηματικά:

- Σε αντίθεση με άλλες γλώσσες προγραμματισμού:
  - Τα ερωτηματικά είναι προαιρετικά για το χωρισμό των εντολών.
  - Εφόσον δεν υπάρχει ερωτηματικό, τότε η αλλαγή γραμμής συνήθως ερμηνεύεται ως ερωτηματικό.
    - Συγκεκριμένα ερμηνεύεται ως ερωτηματικό, εφόσον ότι προηγήθηκε δημιουργεί μια συντακτικά έγκυρη εντολή της JS.

#### Παρατήρηση:

- Θυμόμαστε ότι η JS είναι ανεκτική σε λάθη, ώστε να τρέχει ακόμη κι αν υπάρχουν ιδιαιτερότητες.
- Καλό πάντως θα είναι, να βάζουμε ερωτηματικά στις εντολές μας.

#### Παράδειγμα 2: semicolons.html

```
let x = 1  
console.log(x)
```

Το script τρέχει κανονικά (οι αλλαγές γραμμής ερμηνεύονται ως ερωτηματικά)

#### Παράδειγμα 3: semicolons2.html

Το ακόλουθο script εκτελείται κανονικά (μόνο η 3<sup>η</sup> και η 7<sup>η</sup> αλλαγή γραμμής γίνονται ερωτηματικά):

```
let x  
=  
1  
console.  
log(  
  x  
)
```

#### Σημείωση:

- Αν και πρέπει να είναι κάποιος ανισόρροπος για να γράψει έτσι τον κώδικά του, ας γνωρίζουμε ότι υπάρχουν και κάποιες εξαιρέσεις στον κανόνα των προαιρετικών ερωτηματικών:
  - Ο προθεματικός τελεστής ++ (και ο --)
  - Επιστροφές τιμών με τη return
  - Η σύνταξη των arrow functions.

## ΜΑΘΗΜΑ 3: Σταθερές και Ιδιαιτερότητες της JS

### 3.2. Χρήση της var και hoisting

#### Ορισμός μεταβλητής με τη var:

- Παλιος (<ES6) τρόπος να ορίζουμε μεταβλητές.
- Συντάσσεται όπως η let.
- Ιδιαιτερότητα 1: Οι μεταβλητές μπορούν να χρησιμοποιηθούν προτού δηλωθούν (!!)

#### Παράδειγμα 4: var.html

```
x = 1;
console.log(x);
var x;
```



#### Σημείωση:

- Το παραπάνω πρόγραμμα δουλεύει λόγω του **hoisting** (μτφ~= ανύψωση)
  - που σημαίνει ότι οι δηλώσεις μεταβλητών με τη var, μεταφέρονται στην αρχή του script και αρχικοποιούνται με undefined.
  - Έτσι το παραπάνω πρόγραμμα εσωτερικά μετατρέπεται στο:

```
var x = undefined;
x = 1;
console.log(x);
```

- Hoisting (χωρίς αρχικ/ση) γίνεται και στις δηλώσεις let και const. Ωστόσο:
  - Το αντίστοιχο πρόγραμμα με let θα προκαλούσε ReferenceError
  - Το αντίστοιχο πρόγραμμα με const θα προκαλούσε συντ. λάθος.

#### Παράδειγμα 5: hoisting.html

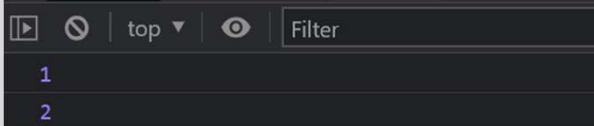
```
console.log(x);
x = 1; console.log(x);
var x = 2; console.log(x);
```



- Ιδιαιτερότητα 2: Δεν προκαλείται λάθος, αν ξαναδηλώσουμε την ίδια μεταβλητή.

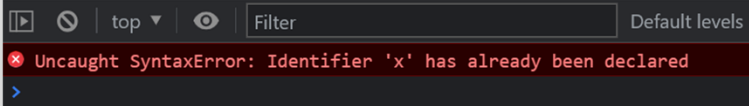
#### Παράδειγμα 5: var2.html

```
var x = 1;
console.log(x);
var x = 2;
console.log(x);
```



#### Παράδειγμα 6: let.html

```
let x = 1;
let x = 2;
```



- Ιδιαιτερότητα 3: Η var δεν έχει εμβέλεια μπλοκ (επομ. μάθημα)
- Γενικά η var είναι παρωχημένη:
  - Ξέρουμε πως δουλεύει αν τύχει να διαβάσουμε παλιά (προ <ES6) scripts.
  - Δεν θα χρησιμοποιούμε ποτέ τη var στα προγράμματά μας!

#### Use strict:

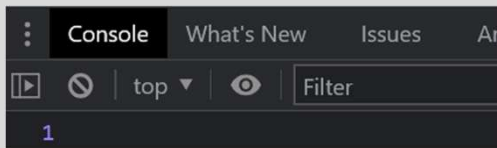
- (Σχεδόν) σε όλα τα scripts που είδαμε, η 1<sup>η</sup> δήλωση είναι η:

```
<script>
"use strict"
...
</script>
```

- Αυτή η λειτουργία προστέθηκε στην ES5, ώστε να μην επιτρέπεται μία σειρά από αρχικές σχεδιαστικές αποφάσεις της γλώσσας.
- Παραδείγματα που απαγορεύονται με την "use strict" (τα περισσότερα θα τα δούμε στα επόμενα μαθήματα)
  - Χρήση μεταβλητής/συνάρτησης/αντικειμένου που δεν έχει δηλωθεί νωρίτερα (βλ. παραδείγματα "strict" και "strict2")
  - Διαγραφή με την delete (βλ. παράδειγμα delete)
  - Περιορισμός κάποιων λέξεων-κλειδιών (with, this κ.α)

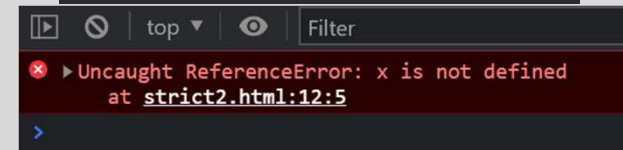
#### Παράδειγμα 5: strict.html

```
x = 1;
console.log(x);
```



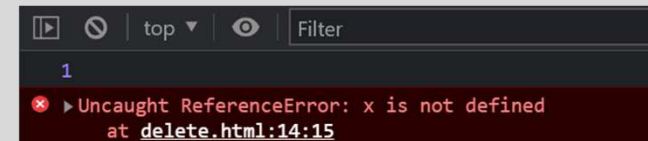
#### Παράδειγμα 6: strict2.html

```
"use strict";
x = 1;
console.log(x);
```



#### Παράδειγμα 7: delete.html

```
x = 1;
console.log(x);
delete x;
console.log(x);
```



#### Σημείωση:

- Καλό θα είναι σε όλα τα script μας η πρώτη γραμμή να είναι η "use strict"
  - και σημειώνουμε ότι δουλεύει μόνο εφόσον είναι η 1<sup>η</sup> γραμμή (και όχι π.χ. να τη γράψουμε από ένα σημείο του script μας κι έπειτα)
  - (Θα δούμε πάντως ότι μπορεί να αφορά μόνο μια συνάρτηση - γράφεται στην αρχή της συνάρτησης)



#### Περισσότερες Λεπτομέρειες για το πως τρέχει μία «σελίδα»:

- Θεωρούμε τη σελίδα: execution\_context.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Psounis JS tutorial</title>
</head>
<body>
  <h1>A page</h1>

  <script>
    console.log(x);
    var x = 1;
  </script>

  <p>A paragraph</p>

  <script>
    console.log(x);
    let y = 2;
    console.log(y);
  </script>
</body>
</html>
```

#### Βήματα που εκτελούνται στο παρασκήνιο:

- Ο browser ανοίγει τη σελίδα και ξεκινάει το parsing (διάβασμα γραμμή - γραμμή)
- Ο V8 αρχικοποιεί το καθολικό περιβάλλον εκτέλεσης (global execution context) της σελίδας:
  - Μπορούμε να το θεωρήσουμε σαν έναν χώρο μνήμης, αποκλειστικά για τη συγκεκριμένη σελίδα.
- Όταν φτάσει στο 1<sup>ο</sup> script, διαβάζεται το περιεχόμενό του και τρέχει αμέσως.
  - (Γίνεται hoisting και τυπώνεται η τιμή undefined)
- Συνεχίζεται το parsing της σελίδας.
- Όταν φτάσει στο 2<sup>ο</sup> script, διαβάζεται το περιεχόμενό του και τρέχει αμέσως.
  - Το περιβάλλον εκτέλεσης είναι το ίδιο (από την εκτέλεση του προηγούμενου script)
  - (Γίνεται και πάλι hoisting και έπειτα τυπώνονται οι τιμές 2 και 3)

#### Άσκηση 2:

- Μπορούμε να αλλάξουμε δυναμικά την τιμή μιας μεταβλητής στο περιβάλλον εκτέλεσης, π.χ.:

```
> y=3
< 3
> y
< 3
```

- Ανοίξτε δύο φορές της σελίδα execution\_context.html στον browser και αλλάξτε την τιμή του y στην 1<sup>η</sup>, και διαπιστώστε ότι η αλλαγή δεν μεταφέρεται στη 2<sup>η</sup> σελίδα.

Στο καθολικό περιβάλλον εκτέλεσης, όταν ανοίγουμε μια σελίδα:

- Κατασκευάζεται αυτόματα το **αντικείμενο window** (αναφέρεται και ως global object)
- Βλέπουμε το αντικείμενο γράφοντας window στην κονσόλα:

```
> window
< Window {window: Window, self: Window, document: document, ...}
  ▶ alert: f alert()
  ▶ atob: f atob()
  ▶ blur: f blur()
  ▶ btoa: f btoa()
  ▶ caches: CacheStorage {}
  ▶ cancelAnimationFrame: f cancelAnimationFrame()
  ▶ cancelIdleCallback: f cancelIdleCallback()
  ▶ captureEvents: f captureEvents()
  ▶ chrome: {loadTimes: f, csi: f}
  ▶ clearInterval: f clearInterval()
  ▶ clearTimeout: f clearTimeout()
  ▶ clientInformation: Navigator {vendorSub: '', produc
```

- Περιέχει πολλά χρήσιμα μέλη (που θα μελετήσουμε στο 2<sup>ο</sup> μέρος της σειράς), μεταξύ των οποίων:
  - document: Περιέχει, μεταξύ άλλων, το DOM
  - console: Αντικείμενο - διαχειριστής της κονσόλας
  - history: Ιστορικό επίσκεψης σελίδων
  - location: Πληροφορίες για το τρέχον URL
  - navigator: Πληροφορίες browser
  - screen: Πληροφορίες οθόνης
- Σε όλα τα μέλη έχουμε πρόσβαση στα script, είτε με window και τελεία, είτε απ' ευθείας (βλ. παράδειγμα window)

### Παράδειγμα 8: window.html

```
let x = 1;
console.log(x);
window.console.log(x);
window.document.write(screen.width + "-" + innerWidth);
```

### Διαφορά let/const και var:

- Οι let/const τιμές αποθηκεύονται στο περιβάλλον εκτέλεσης
- Οι var τιμές αποθηκεύονται ως μέλη στο καθολικό αντικείμενο (που αποθηκεύεται στο περιβάλλον εκτέλεσης).

### Παράδειγμα 9: window\_variables

```
let a_let = 1;
var a_var = 2;
const a_const = 3;
```

- Βλέπουμε την ενσωμάτωση της a\_var στο αντικείμενο window:

```
> window
< Window {window: Window, self: Window, document: document, ...}
  ▶ a_var: 2
  ▶ alert: f alert()
  ▶ atob: f atob()
  ▶ blur: f blur()
  ▶ btoa: f btoa()
```