Κεφάλαιο 8. Πρωτογενείς τύποι δεδομένων και εντολές της JavaScript

Σύνοψη

Στο δεύτερο αυτό κεφάλαιο με αντικείμενο την εισαγωγή στη γλώσσα προγραμματισμού JavaScript θα ξεκινήσουμε την επισκόπηση των διαφορετικών πρωτογενών τύπων δεδομένων που υποστηρίζει η γλώσσα. Συγκεκριμένα, θα δούμε τους πρωτογενείς τύπους δεδομένων (primitive data types), τους τύπους δεδομένων Number, String, Boolean, null, undefined.

Επίσης, στο κεφάλαιο αυτό θα γίνει η εισαγωγή στον πυρήνα των εντολών της JavaScript που υλοποιούν δομές ελέγχου ροής και επανάληψης που διαθέτει η γλώσσα. Θα προχωρήσουμε σε ανασκόπηση των βασικών προγραμματιστικών δομών της JavaScript: των εντολών ελέγχου της ροής του προγράμματος, όπως οι if και switch, των εντολών που επιτρέπουν την επαναληπτική εκτέλεση ενός κώδικα, όπως οι for, while, do/while και, τέλος, εντολών που επιτρέπουν τη μετάβαση σε άλλο τμήμα του κώδικα, όπως οι break, continue και throw.

Προαπαιτούμενη γνώση

Είναι απαραίτητη η εξοικείωση με το κεφάλαιο 7.

8.1 Τύποι δεδομένων

Υπάρχουν οι εξής πρωτογενείς (primitive) τύποι δεδομένων:

Οι τύποι **number**, **string**, **boolean** που συναντιούνται στις περισσότερες γλώσσες προγραμματισμού.

Ακόμη, ορίζονται οι ειδικοί τύποι **null** και **undefined** που θα περιγραφούν στη συνέχεια.

Επίσης, σε τελευταίες εκδόσεις της γλώσσας έχουν προστεθεί οι τύποι **bigInt** για αναπαράσταση μεγάλων ακέραιων αριθμών μεγαλύτερων από 2^{53}, καθώς και ο τύπος **symbol**.

Εκτός των πρωτογενών τύπων δεδομένων, υπάρχουν οι τύποι **δεδομένων αναφοράς**. Στην κατηγορία αυτή είναι ο τύπος δεδομένων που αφορά όλα τα δεδομένα πλην των ανωτέρω, το **αντικείμενο (object)**. Ένα αντικείμενο της JavaScript είναι μια μη ταξινομημένη συλλογή από ζευγάρια *κλειδιών-τιμών*. Η γλώσσα ορίζει επίσης ένα ειδικό είδος αντικειμένου, τον **πίνακα (array)**, που αντιπροσωπεύει μια διαταγμένη συλλογή αριθμημένων τιμών.

Οι πρωτογενείς τύποι δεδομένων θα συζητηθούν στο παρόν κεφάλαιο, ενώ οι πίνακες και τα αντικείμενα θα συζητηθούν μαζί με τις συναρτήσεις στο επόμενο κεφάλαιο.

8.2 Πρωτογενείς τύποι δεδομένων: Number

Ο τύπος δεδομένων **Number** αφορά τιμές ακεραίων με τιμές $\pm 2^{53}$ και κλασματικών αριθμών μεγέθους μέχρι $\pm 1.7976931348623157 \times 10^{308}$ και μικρών τιμών μέχρι $\pm 5 \times 10^{-324}$. Οι κλασματικοί αριθμοί αναπαρίστανται με κωδικοποίηση 64-bit IEEE 754, όπως στις περισσότερες σύγχρονες γλώσσες προγραμματισμού (αντίστοιχος τύπου double της Java, C++, float της Python κ.λπ.).

Για αναπαράσταση πολύ μεγάλων ακέραιων αριθμών πέρα από το παραπάνω όριο έχει εισαχθεί από την έκδοση ES2020 ο τύπος **BigInt**, ο οποίος αναπαριστά τον αριθμό ως ακολουθία ψηφίων και δεν έχει άνω όριο. Όμως, η υλοποίησή του δεν έχει ολοκληρωθεί στους φυλλομετρητές ακόμη.

Οι τιμές μεταβλητών τύπου Number μπορεί να είναι είτε ακέραιοι είτε κλασματικοί αριθμοί είτε αριθμοί σε δεκαεξαδική μορφή, σε δυαδική μορφή, σε μορφή ύψωσης σε δύναμη κ.λπ. Παραδείγματα ακολουθούν:

```
let x = 12345 // ακέραιος
let y = 123.456 // δεκαδικός
let z = 1.473E+32 // δύναμη 1.473 × 10³²
let k = 0xBADCAFE // δεκαεξαδικός => 195939070
```

Υπάρχει ακόμη η δυνατότητα από την έκδοση ES6 και μετά αναπαράστασης οκταδικών αριθμών (αρχίζουν με **0o**) και δυαδικών αριθμών (αρχίζουν με **0b**).

Με χρήση του τελεστή typeof μπορούμε να ελέγξουμε τον τύπο μιας μεταβλητής. Σε όλες τις παραπάνω περιπτώσεις η έκφραση, για παράδειγμα, typeof x επιστρέφει τη συμβολοσειρά 'number'.

Όπως έχει ήδη αναφερθεί, στους αριθμούς μπορούμε να εφαρμόσουμε τελεστές αριθμητικών πράξεων, +, -, *, /, %, **, καθώς και τους τελεστές ++, --, +=, -= κ.λπ.

Το global object διαθέτει το αντικείμενο Math το οποίο έχει μεθόδους για πιο σύνθετες μαθηματικές συναρτήσεις.

Οι κυριότερες από αυτές είναι:

- **Math.abs(-20)**, 20 απόλυτη τιμή
- **Math.ceil(10.5)**, 11 ο επόμενος ακέραιος
- **Math.E**, 2.718281828459045 η σταθερά *e*
- **Math.exp(5)**, e**5 = 148.413, δύναμη του *e*
- **Math.floor(8.9)**, 8 το ακέραιο μέρος
- **Math.log(3), // 1.0986122886681096
- **Math.max(10,20)**, 20 μέγιστη τιμή
- **Math.min(10,20)**, 10 ελάχιστη τιμή,
- **Math.pow(3,4)**, 81 ύψωση του x στη δύναμη y, x^y
- **Math.round(5.6)**, 6 στρογγύλεμα στον πλησιέστερο ακέραιο
- **Math.sqrt(4)**, 2 τετραγωνική ρίζα
- **Math.PI**, π = 3,14
- **Math.sin(θ)**, ημίτονο
- **Math.cos(θ)**, συνημίτονο
- **Math.tan(θ)**, εφαπτομένη

Θα πρέπει να σημειωθεί ότι η γωνία θ στις τριγωνομετρικές συναρτήσεις εκφράζεται σε rad, δηλαδή 90° = Math.PI/2, συνεπώς: Math.cos(Math.PI/2) = 0.

Επίσης, το αντικείμενο Math διαθέτει αρκετές ακόμη χρήσιμες συναρτήσεις, μεταξύ των οποίων η Math.random(), η οποία όταν κληθεί επιστρέφει έναν ψευδο-τυχαίο δεκαδικό αριθμό στο διάστημα από 0 μέχρι 1, χωρίς να περιλαμβάνεται το 1.

Αντίθετα με άλλες γλώσσες δεδομένων, στην JavaScript αν μια μαθηματική έκφραση προκαλέσει υπερχείλιση (overflow), όπως για παράδειγμα η διαίρεση με το μηδέν, δεν προκύπτει σφάλμα, αλλά το αποτέλεσμα της έκφρασης μπορεί να πάρει μια από τις παρακάτω ειδικές τιμές:

- **Infinity** (θετικό άπειρο)
- **-Infinity** (αρνητικό άπειρο)
- **NaN** μη αριθμητική τιμή (Not a Number)

Μάλιστα, το global object διαθέτει συναρτήσεις για έλεγχο αυτών των τιμών, όπως isNaN(x).

Για παράδειγμα:

```
console.log(-10/0);
> -Infinity
```

Επίσης:

```
isNaN("αβγ");
> true
```

8.3 Πρωτογενείς τύποι δεδομένων: Συμβολοσειρές

Οι συμβολοσειρές (string) είναι ο τύπος δεδομένων που αναπαριστά ακολουθία χαρακτήρων. Οι συμβολοσειρές στην JavaScript περιέχουν χαρακτήρες Unicode, 16bit ο καθένας. Οριοθετούμε συμβολοσειρές με απλά ' ή διπλά " εισαγωγικά ή με τον χαρακτήρα *backquote* `. Χρησιμοποιούμε το ένα για να οριοθετήσουμε συμβολοσειρά που περιέχει το άλλο.

Ο χαρακτήρας backquote χρησιμοποιείται για συμβολοσειρές που μπορούν να περιέχουν εκφράσεις JavaScript, όπως θα εξηγηθεί στη συνέχεια.

Ο χαρακτήρας διαφυγής \ μπορεί να χρησιμοποιηθεί για να ακυρώσει τον ειδικό ρόλο κάποιου χαρακτήρα, όπως του χαρακτήρα οριοθέτησης της συμβολοσειράς.

Για παράδειγμα:

```
console.log("του είπαν: \"τι χαμπάρια μας φέρνεις;\"")

>του είπαν: "τι χαμπάρια μας φέρνεις;"
```

Ο τελεστής “+” επιτρέπει τη συνένωση συμβολοσειρών.

```
let one = 'Καλή σας μέρα ';
let two = 'άρχοντες! ';
console.log(one + two);
>Καλή σας μέρα άρχοντες!
```

Στο επόμενο παράδειγμα, όταν ο χρήστης πατήσει το πλήκτρο, προκύπτει παράθυρο με ερώτημα για το όνομά του και ακολουθεί χαιρετισμός με σύνθεση συμβολοσειρών.

```
<button> ok </button>
let button = document.querySelector('button');
button.onclick = function() {
    let name = prompt(Πώς σε λένε;');
    alert('Καλωσήρθες ' + name); }
```

Ένα αξιοπρόσεκτο χαρακτηριστικό της γλώσσας, καθώς έχει *δυναμικό* σύστημα τύπων, είναι πως μπορούμε να κάνουμε πράξεις με τιμές διαφορετικών τύπων, π.χ., μεταξύ συμβολοσειρών και αριθμών:

```
10 + '20'
> "1020"
10+20
> 30
'abc'+10
> "abc10"
```

8.3.1 Δείκτης συμβολοσειράς

Μπορούμε να αναφερθούμε σε επιμέρους στοιχεία μιας συμβολοσειράς με χρήστη δεικτών που αρχίζουν από 0 μέχρι length-1, όπου length το πλήθος χαρακτήρων της συμβολοσειράς.

Για παράδειγμα:

```
const hero = "Αθανάσιος Διάκος";
console.log(hero[1]);
> 'θ'
```

Το μήκος της συμβολοσειράς μπορούμε να το πάρουμε με χρήση της ιδιότητας length

```
const hero = "Αθανάσιος Διάκος";
console.log(hero.length);
> 16
```

επειδή η συμβολοσειρά hero περιέχει 16 χαρακτήρες.

Μια λεπτομέρεια που θα πρέπει να προσέξουμε είναι ότι, αν η συμβολοσειρά περιέχει σύμβολα που συντίθενται από ακολουθίες Unicode 16bit, η ιδιότητα length θα μετρήσει το πλήθος αυτών των χαρακτήρων. Για παράδειγμα, emoji σημαιών χωρών:

```
const greekFlag = "GR";
greekFlag.length;
> 4
```

Θα πρέπει όμως να σημειωθεί ότι κείμενα στις περισσότερες γλώσσες, συμπεριλαμβανομένων των ευρωπαϊκών γλωσσών αλλά και των ιδεογραμμάτων των ασιατικών γλωσσών, περιλαμβάνονται στην κωδικοποίηση Unicode 16bit, συνεπώς το μήκος της συμβολοσειράς μπορεί να μετρηθεί.

```
const hello = "你好" // Nǐ hǎo (γεια σας στα κινεζικά)
hello.length;
> 2
```

8.3.2 Μετατροπές από συμβολοσειρές σε αριθμούς

Για τη μετατροπή μιας συμβολοσειράς με αριθμητικούς χαρακτήρες σε αριθμό μπορούμε να χρησιμοποιήσουμε τον δημιουργό του αντικείμενου Number().

```
// μετατροπή από string σε αριθμό
let myString = '123';
let myNum = Number(myString);
typeof myNum;
>"number"
```

Αν το όρισμα του δημιουργού Number() δεν εκφράζει έναν αριθμό, τότε η συνάρτηση επιστρέφει ένα αντικείμενο τύπου number με την τιμή NaN.

Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε τις συναρτήσεις parseInt() και parseFloat(), που ανήκουν στο global object, για μετατροπή συμβολοσειράς σε ακέραιο ή κλασματικό αριθμό αντίστοιχα.

Θα πρέπει να προσέξουμε μια ιδιαιτερότητα των συναρτήσεων αυτών. Αν η συμβολοσειρά αρχίζει με αριθμητικούς χαρακτήρες και στη συνέχεια περιέχει μη αριθμητικούς χαρακτήρες, αγνοεί τους μη αριθμητικούς και επιστρέφει τον αριθμό που προκύπτει, και όχι NaN που επιστρέφει η δημιουργός Number().

```
Number('123abc'); \\ NaN
parseInt('123abc'); \\ 123
```

Η αντίστροφη μετατροπή από αριθμό στην αντίστοιχη συμβολοσειρά γίνεται με κλήση της μεθόδου toString() του αντικείμενου Number.

```
// μετατροπή από αριθμό σε string
let myNum = 123;
let myString = myNum.toString();
```

8.3.3 Κύριες μέθοδοι και τελεστές του String

Έχουμε ήδη δει πώς αναφερόμαστε σε επιμέρους χαρακτήρες μιας συμβολοσειράς και πώς βρίσκουμε το μήκος μια συμβολοσειράς.

```
let myName = 'Nikos';
myName.length;    // 5

//πρώτος χαρακτήρας
myName[0];     // "N"

//τελευταίος χαρακτήρας
myName[myName.length-1];  // "s"
```

Στη συνέχεια θα δούμε δύο ενδιαφέρουσες μεθόδους του αντικείμενου String, τη slice() και την indexOf():

```
//τμήμα, από 0 έως 3
myName.slice(0,3); // returns 'Nik'

//βρες τη θέση της υπο-συμβολοσειράς  -1 αν δεν βρεθεί
myName.indexOf('kos');     // 2

//τεμάχιο συμβολοσειράς από θέση i1, έως i2
myName.slice(3); // επιστρέφει 'os'
myName.slice(1,4); // επιστρέφει 'iko'
```

Όπως φαίνεται από τα παραδείγματα, μπορούμε να αναφερθούμε σε τμήμα της συμβολοσειράς από τη θέση index1 μέχρι τη θέση index2, χωρίς να περιλαμβάνεται η τελευταία με χρήση της s.slice(index1, index2).

Επίσης, με τη μέθοδο s.indexOf (υπο-συμβολοσειρά) βρίσκουμε τη θέση που για πρώτη φορά συναντάμε την υπο-συμβολοσειρά στην s. Αν αυτή δεν υπάρχει, τότε μας επιστρέφει την τιμή -1.

Μπορούμε να διαχωρίσουμε μια συμβολοσειρά σε έναν πίνακα των επιμέρους χαρακτήρων της με χρήση

της μεθόδου split(). Η μέθοδος μοιάζει με την αντίστοιχη της Python με μια διαφορά, ότι πρέπει να δώσουμε οπωσδήποτε όρισμα. Για παράδειγμα:

```
//διαχωρισμός συμβολοσειράς σε πίνακα στοιχείων
//split(delimiter_string)
'αβγ'.split('');
> [ 'α', 'β', 'γ' ]
'αβγ'.split('β');
> [ 'α', 'γ' ]
```

Άλλες χρήσιμες μέθοδοι του αντικείμενου String είναι:

```
myName.toLowerCase(); // "nikos"
myName.toUpperCase(); // "NIKOS"

myName.replace('kos','na') //"Nina"

myName.charAt(2); // "k"

myName.charCodeAt(2); // 107
```

Επίσης, η μέθοδος trim() αποκόπτει τα κενά στην αρχή και στο τέλος μιας συμβολοσειράς με αντίστοιχο τρόπο όπως η μέθοδος strip() στην Python:

```
"       καλή σας μέρα           ".trim();
> 'καλή σας μέρα'
```

8.3.4 Συμβολοσειρές-πρότυπα (template literals)

Ένας ειδικός τύπος συμβολοσειρών εισήχθη στην JavaScript στην έκδοση ES6. Είναι οι **συμβολοσειρές πρότυπα**, που έχουν το χαρακτηριστικό ότι μπορούν να περιλάβουν εκφράσεις JavaScript ${έκφραση}, οι οποίες αντικαθίστανται από την τιμή τους.

Οι συμβολοσειρές αυτές οριοθετούνται από το σύμβολο backquote (`).

Ας δούμε ένα παράδειγμα.

```
let a = 5;
let b = 10;
console.log(`a+b=${a+b} ενώ 2a+b=${2*a + b}.`);
>'a+b=15 ενώ 2a+b=20.'
```

Ασκήσεις

Άσκηση 1

Έστω

```
let input = "ΘεσΑΛΛΟνίκη";
```

Να γράψετε εντολές JavaScript που παράγουν σωστή συμβολοσειρά για το όνομα της πόλης.

Απάντηση

```
let temp = input.toLowerCase();
let startCharacter = temp[0].toUpperCase();
let result = startCharacter + temp.slice(1);
```

Άσκηση 2

Έστω

```
let input = "ATH675847583748sjt567654;Athens El.Venizelos";
```

Ζητείται να γράψετε κώδικα που εξάγει από την παραπάνω συμβολοσειρά την εξής (τους τρεις πρώτους χαρακτήρες και την υπόλοιπη συμβολοσειρά μετά το ;): ATH:Athens El.Venizelos

Απάντηση

```
console.log(input.slice(0,3)+ ":" + input.slice(input.indexOf(";")+1))
> 'ATH:Athens El.Venizelos'
```

8.4 Πρωτογενείς τύποι δεδομένων: null και undefined

Στην ενότητα αυτή θα δούμε δύο ακόμη πρωτογενείς τύπους δεδομένων, ειδικού σκοπού.

Ο πρώτος τύπος είναι το null, μια λέξη-κλειδί που υποδεικνύει την απουσία τιμής. Η χρήση του τελεστή typeof στο null επιστρέφει τη συμβολοσειρά “object”, υποδεικνύοντας ότι το null μπορεί να θεωρηθεί ως μια ειδική τιμή αντικείμενου που υποδηλώνει την απουσία αντικείμενου. Σύμφωνα με τον ορισμό της γλώσσας, το null δεν θεωρείται αντικείμενο αλλά το μοναδικό μέλος ενός ξεχωριστού τύπου δεδομένων που μπορεί να χρησιμοποιηθεί για να δείξει «μη τιμή» για αριθμούς και συμβολοσειρές, καθώς επίσης για αντικείμενα.

Έστω σε μια ιστοσελίδα χωρίς υπερσυνδέσμους (χωρίς στοιχεία <a>) ας δούμε τι θα συμβεί αν αναζητήσουμε στοιχεία αυτού του τύπου στο DOM:

```
let x = document.querySelector("a");
console.log(x);
> null
typeof x;
> "object"
```

Υπάρχει όμως ένας ακόμη τύπος δεδομένων που υποδηλώνει μη τιμή: ο τύπος undefined.

Ο τύπος αυτός αποδίδεται σε μεταβλητές που έχουν οριστεί χωρίς να λάβουν τιμή, σε συναρτήσεις που δεν επιστρέφουν δεδομένα (δεν έχουν εντολή return), σε ιδιότητες αντικειμένων που δεν υπάρχουν, σε τιμές ορισμάτων συναρτήσεων που δεν τους έχει δοθεί τιμή. Η τιμή αυτή επιστρέφει ως τύπος δεδομένων από τον τελεστή typeof. Ας δούμε μερικά παραδείγματα:

```
let x; // μεταβλητή χωρίς αρχική τιμή
typeof x;
> 'undefined'

const ob = {name: "Nikos"} // αντικείμενο
typeof ob.age; // ιδιότητα που δεν υπάρχει
> 'undefined'

function f(){} // συνάρτηση που δεν επιστρέφει τιμή
typeof f();
> 'undefined'
```

8.5 Πρωτογενείς τύποι δεδομένων: Boolean

Ο τύπος δεδομένων Boolean αντιπροσωπεύει λογικές μεταβλητές οι οποίες παίρνουν τιμή αληθές/ψευδές. Οι δεσμευμένες λέξεις true και false αντιστοιχούν στις τιμές αυτές αντίστοιχα.

Η τιμή αληθές/ψευδές είναι το αποτέλεσμα μιας σύγκρισης, η οποία εισάγεται με τον τελεστή σύγκρισης === (ισότητα τιμής και τύπου δεδομένων) ή == (ισότητα μόνο τιμής, όχι απαραίτητα και τύπου δεδομένων) ή και άλλων τελεστών σύγκρισης >, <, >=, <=, !== (έλεγχος ανισότητας τιμής και τύπου δεδομένων), != (έλεγχος ανισότητας τιμής).

Εκφράσεις που περιέχουν σύγκριση συνήθως χρησιμοποιούνται σε εντολές που απαιτούν τιμή αληθές/ψευδές, όπως η εντολή if ή η εντολή while.

Όμως, θα πρέπει να σημειωθεί ότι οποιαδήποτε έκφραση της JavaScript, ανάλογα με την τιμή που παίρνει, αντιστοιχεί σε αληθές/ψευδές. Ο κανόνας που ισχύει είναι ο εξής:

- Αν η έκφραση έχει την τιμή undefined, null, 0, -0, NaN, "" (κενή συμβολοσειρά), τότε είναι ισοδύναμη με false.
- Σε οποιαδήποτε άλλη περίπτωση είναι true.

Οι λογικοί τελεστές της JavaScript είναι:

- **&&** τελεστής AND (λογικό «και»)
- || τελεστής OR (λογικό «ή»)
- **!** τελεστής NOT (λογικό «όχι»)

8.5.1 Τιμή λογικών εκφράσεων

Ας δούμε τι επιστρέφουν οι λογικές εκφράσεις, κάτι που συχνά εκμεταλλεύονται οι προγραμματιστές JavaScript για να ελέγξουν τιμή σε μεταβλητές ή να δώσουν μια προκαθορισμένη τιμή σε μια μεταβλητή αν αυτή δεν έχει ήδη οριστεί.

Μια έκφραση λογικής OR θα λάβει την τιμή της πρώτης μεταβλητής που είναι true, επειδή η λογική OR ικανοποιείται σε αληθές αρκεί ένας όρος να είναι αληθής, ή την τελευταία τιμή αν κανένας όρος δεν είναι αληθής.

Παραδείγματα εκφράσεων OR:

- 0 || undefined || 5 : θα πάρει την τιμή 5 (true)
- 0 || 5 || 10 : θα πάρει την τιμή 5 (true)
- 0 || undefined || null : θα πάρει την τιμή null (false)

Αντίστοιχα, σε μια έκφραση με λογική AND θα επιστρέψει τον τελευταίο όρο αν όλοι είναι αληθείς και τον πρώτο ψευδή όρο αν υπάρχει ψευδής. Αυτό γιατί στη λογική AND αρκεί ένας όρος να είναι ψευδής για να πάρει η έκφραση την τιμή *ψευδής*.

Παραδείγματα εκφράσεων AND:

- 0 && 10 && 5 : θα πάρει την τιμή 0 (false)
- 8 && 10 && 5 : θα πάρει την τιμή 5 (true)
- 5 && 10 && null : θα πάρει την τιμή null (false)

8.6 Προγραμματιστικές δομές της JavaScript

Στη συνέχεια θα δούμε τις βασικές προγραμματιστικές δομές που υποστηρίζει η JavaScript και υλοποιούνται ως εντολές για έλεγχο της ροής του προγράμματος, όπως οι if και switch, εντολές που επιτρέπουν την επαναληπτική εκτέλεση ενός κώδικα, όπως οι for και while, και τέλος εντολές που επιτρέπουν τη μετάβαση σε άλλο τμήμα του κώδικα, όπως οι break, continue και throw.

- Θα πρέπει να σημειώσουμε ότι η JavaScript συντακτικά ακολουθεί το παράδειγμα πολλών άλλων γλωσσών προγραμματισμού, όπως η C, C++, Java, ως προς τη σύνταξη των εντολών αυτών, τη χρήση του ";" ως τερματικού εντολής και τη χρήση των άγκιστρων { ... } για τον ορισμό ενός μπλοκ εντολών.
- Θα πρέπει να σχολιάσουμε, όμως, ότι οι ομοιότητες μεταξύ JavaScript και Java σταματούν εδώ, αφού πρόκειται για δύο πολύ διαφορετικές τεχνολογίες και το όνομα JavaScript είναι ατυχές αφού παραπέμπει στην Java, με την οποία η JavaScript δεν έχει ιδιαίτερη σχέση.
- Αν ένα μπλοκ εντολών περιέχει μία μόνο εντολή, τότε τα άγκιστρα μπορούν να παραληφθούν. Ωστόσο, πολλοί προγραμματιστές βάζουν πάντα τα άγκιστρα, ακόμη και στην περίπτωση του μπλοκ με μια εντολή, για να γίνει πιο σαφής η εμβέλεια κάθε εντολής.
- Επίσης, μπορούμε να ορίσουμε ως κενή εντολή την εντολή ";".
- Ως προς τη στοίχιση των εντολών, αυτή δεν είναι υποχρεωτική, όπως γίνεται για παράδειγμα στην Python, όμως είναι πολύ καλή πρακτική ένα μπλοκ εντολών να στοιχίζεται μέσα στα άγκιστρα που το ορίζουν, κάτι που επεξεργαστές κώδικα όπως ο *VS Code* κάνουν αυτόματα.

8.6.1 Εντολή if-else

Η πρώτη δομή που θα δούμε ότι επιτρέπει την υπό συνθήκη εκτέλεση ενός τμήματος του κώδικα είναι η εντολή if που συναντάται σε όλες τις γλώσσες προγραμματισμού.

Στη γενική περίπτωση αυτή η δομή συντάσσεται ως εξής:

```
if (έκφραση){
    μπλοκ-εντολών-1
}
else {
    μπλοκ-εντολών-2
}
```

Το μπλοκ-εντολών-1 θα εκτελεστεί αν η έκφραση είναι αληθής, ενώ το μπλοκ-εντολών-2 αν είναι ψευδής.

Θα πρέπει να παρατηρήσουμε εδώ ότι η έκφραση θα πρέπει να παίρνει μια τιμή η οποία να αντιστοιχεί είτε σε αληθή είτε σε ψευδή τιμή, σύμφωνα με τους κανόνες αληθότητας/ψευδότητας που έχουν ήδη αναφερθεί. Υπενθυμίζουμε ότι οι τιμές undefined, null, 0, -0, NaN, "" (κενή συμβολοσειρά), false αντιστοιχούν στην τιμή **«ψευδής»**, ενώ οποιαδήποτε άλλη τιμή σε **«αληθής»**. Για παράδειγμα, ένας κενός πίνακας [ ] αντιστοιχεί στην τιμή «αληθής», το ίδιο και ένα κενό αντικείμενο { }, κάτι που διαφοροποιεί την JavaScript από την Python που η κενή λίστα [ ] αντιστοιχεί στην τιμή «ψευδής».

Επίσης, θα πρέπει να παρατηρήσουμε ότι το τμήμα else της παραπάνω δομής μπορεί να παραληφθεί:

```
if (έκφραση){
    μπλοκ-εντολών-1
}
```

Ακόμη, μπορούμε να κάνουμε διαδοχικούς ελέγχους εισάγοντας στο τμήμα else έναν νέο έλεγχο if:

```
if (συνθήκη1){
    μπλοκ-εντολών-1 // αν συνθήκη1 αληθής
} else if (συνθήκη2) {
    μπλοκ-εντολών-2 // αν όχι συνθήκη1 και συνθήκη2 αληθής
} else if (συνθήκη3) {
    μπλοκ-εντολών-3 // αν όχι συνθήκη1, ούτε συνθήκη 2 και συνθήκη3 αληθής
} else {
    μπλοκ-εντολών-4 // αν όχι συνθήκη1, ούτε συνθήκη2, ούτε συνθήκη3
}
```

Ας δούμε ένα παράδειγμα:

```
let forecast = 'snowing';

if (forecast === 'sunny') {
  console.log('Πάμε για μπάνιο.');
} else if (forecast === 'rainy') {
  console.log('Να πάρουμε ομπρέλα');
} else if (forecast === 'snowing') {
  console.log('Να πάμε για σκι');
} else {
  console.log('Μένουμε σπίτι');
}
> 'Να πάμε για σκι'
```

Στο παράδειγμα η μεταβλητή forecast ελέγχεται για την τιμή της και ανάλογα τυπώνεται ένα μήνυμα στην κονσόλα. Αν η τιμή της μεταβλητής είναι διαφορετική από τις τιμές που ελέγχονται διαδοχικά στη δομή αυτή, π.χ. forecast="cloudy", θα πάρουμε το μήνυμα που τυπώνεται από το σκέλος else που ακολουθεί τους ελέγχους.

8.6.2 Εντολή switch

Αν πρέπει να κάνουμε μια σειρά από ελέγχους οι οποίοι αφορούν την ίδια μεταβλητή, όπως στο προηγούμενο παράδειγμα, μια καλύτερη επιλογή από διαδοχικά if - else if είναι να χρησιμοποιηθεί η δομή switch.

Η εντολή αυτή συντάσσεται ως εξής:

```
switch(n) {
case 1:  // if n === 1
    block-εντολών-1
```

```
        break;
    case 2: // if n === 2
        block-εντολών-2
        break;
    case 3: // if n === 3
        block-εντολών-3
        break;
    default: // αν δεν ισχύει καμιά από τις περιπτώσεις
        block-εντολών-4
        break;
    }
```

Σύμφωνα με το πρότυπο αυτό, το προηγούμενο παράδειγμα θα διαμορφωθεί ως ακολούθως:

```
switch (forecast) {
  case 'sunny':
    console.log('Πάμε για μπάνιο.');
    break;
  case 'rainy':
    console.log('Να πάρουμε ομπρέλα.');
    break;
  case 'snowing':
    console.log('Να πάμε για σκι.');
    break;
  default:
    console.log('Μένουμε σπίτι');
}
```

Να σημειωθεί εδώ ότι οι έλεγχοι που γίνονται σε καθεμία περίπτωση (case) είναι τύπου “strict equality” "===".

8.6.3 Υπό συνθήκη εκχώρηση τιμής (τριαδικός τελεστής)

Ένας εναλλακτικός τρόπος να εκχωρηθεί διαφορετική τιμή σε μια μεταβλητή ανάλογα με την τιμή μιας έκφρασης είναι με χρήση του τριαδικού τελεστή a ? b : c.

Ο τελεστής αυτός, που λέγεται τριαδικός επειδή παίρνει τρεις παραμέτρους, είναι συντομογραφία μας εντολής if-else, όπου a είναι μια έκφραση που παίρνει τιμή αληθής/ψευδής, b η τιμή αν η έκφραση είναι αληθής, και c η τιμή αν η έκφραση είναι ψευδής.

(έκφραση) ? τιμή-αν-αληθής : τιμή-αν-ψευδής

Ας δούμε ένα παράδειγμα:

Έστω ότι η μεταβλητή result εκφράζει αν ένας φοιτητής πήρε προβιβάσιμο βαθμό ή όχι. Παίρνει την τιμή «επιτυχία» αν ο βαθμός είναι μεγαλύτερος ή ίσος του 5 και την τιμή «αποτυχία» αν ο βαθμός είναι μικρότερος του 5.

Η παρακάτω εντολή if-else αποδίδει τιμή στη μεταβλητή result, ανάλογα με την τιμή της vathmos:

```
let result;
if (vathmos < 5){
    result = "αποτυχία";
}
else {
    result = "επιτυχία";
}
```

Η παραπάνω λογική μπορεί να απλοποιηθεί όμως με χρήση του τριαδικού τελεστή ως εξής:

```
let result = (vathmos<5) ? "αποτυχία" : "επιτυχία";
```

8.6.4 Διαχείριση σφαλμάτων με try-catch

Η JavaScript διαθέτει μηχανισμό για διαχείριση σφαλμάτων ή *εξαιρέσεων (exceptions)* όπως λέγονται. Ο

μηχανισμός αυτός έχει πολλά κοινά με την εντολή if-else. Πρόκειται για την εντολή try-catch-finally.

Η σύνταξη της εντολής αυτής είναι:

```
try {
    // κώδικας στον οποίο γίνεται έλεγχος εξαίρεσης
}
catch (e) {
    // κώδικας που θα τρέξει αν συμβεί εξαίρεση
}
finally {
    // κώδικας που θα τρέξει σε κάθε περίπτωση
}
```

Η εντολή catch ακολουθείται από ένα μπλοκ κώδικα στον οποίο γίνεται έλεγχος για κάποια εξαίρεση (απρόβλεπτη κατάσταση σφάλματος). Αν αυτή συμβεί, τότε θα εκτελεστεί ο κώδικας που ακολουθεί την εντολή catch(e). Η παράμετρος e στο catch περιέχει στοιχεία για το σφάλμα, είτε κάποιο μήνυμα είτε κωδικό σφάλματος. Το τμήμα finally είναι προαιρετικό.

Ένα παράδειγμα:

```
try {
    function factorial(num){
        if (num === 0) return 1;
        else return num * factorial( num - 1 );
    }
    let n = Number(prompt("Δώστε θετικό αριθμό", ""));
    let f = factorial(n);
    alert(n + "! = " + f);
}
catch(ex) {
    alert(ex);
}
```

Στο παράδειγμα αυτό μέσα στο τμήμα try ορίζουμε συνάρτηση αναδρομικού υπολογισμού του παραγοντικού n! = n*(n-1)*(n-2) ... * 1

Στο τμήμα αυτό του κώδικα ζητείται από τον χρήστη να δώσει θετικό αριθμό και το πρόγραμμα του επιστρέφει μέσω alert την τιμή του παραγοντικού. Αν όμως ο χρήστης δεν δώσει θετικό αριθμό, ενεργοποιείται το τμήμα catch που στέλνει στον χρήστη μέσω alert το μήνυμα σφάλματος. Για παράδειγμα, αν ο χρήστης δώσει αρνητικό αριθμό, το μήνυμα είναι: "RangeError: Maximum call stack size exceeded".

Μια εντολή που συνήθως εμφανίζεται στη δομή try-catch είναι η εντολή throw.

Η εντολή αυτή, όταν εμφανίζεται σε ένα μπλοκ try, είναι εντολή εξόδου από το μπλοκ και μετάβασης στο μπλοκ catch. Μοιάζει δηλαδή με την εντολή break που θα δούμε στους βρόχους επανάληψης στη συνέχεια.

Η σύνταξη της throw είναι:

```
throw έκφραση-σφάλματος;
```

Η έκφραση σφάλματος είναι είτε ένα αντικείμενο τύπου Error ή μια συμβολοσειρά που σχετίζεται με τη συγκεκριμένη εξαίρεση, για παράδειγμα:

```
if (x < 0) throw new Error("το x πρέπει να είναι θετικό");
```

Στο παράδειγμα αυτό η throw επιστρέφει ένα αντικείμενο τύπου Error στο οποίο περνάμε μια συμβολοσειρά σφάλματος.

8.7 Δομές επανάληψης

Στην ενότητα αυτή θα δούμε τις προγραμματιστικές δομές που διαθέτει η JavaScript για επαναληπτική εκτέλεση ενός τμήματος του κώδικα. Οι εντολές που επιτρέπουν την επαναληπτική εκτέλεση ενός μπλοκ κώδικα είναι οι while, do/while, for, for/in, for/of. Η πιο τυπική χρήση μιας δομής επανάληψης είναι όταν ζητείται να διαπεράσουμε τα στοιχεία ενός πίνακα.

8.7.1 Εντολή while

Η πιο απλή εντολή επαναληπτικής εκτέλεσης κώδικα είναι η while, που συναντάται σε πολλές γλώσσες προγραμματισμού. Η while συντάσσεται ως εξής:

```
while (συνθήκη) {
    μπλοκ-εντολών
}
```

Ο διερμηνευτής της JS μόλις φτάσει στην εντολή αυτή υπολογίζει την τιμή της *συνθήκης*. Αν είναι ψευδής, παραλείπει το μπλοκ εντολών και προχωράει στην επόμενη εντολή του προγράμματος. Αν η συνθήκη είναι αληθής, εκτελεί το μπλοκ εντολών και επανέρχεται στην κορυφή στον εκ νέου έλεγχο της συνθήκης.

Είναι φανερό ότι αν κάτι δεν αλλάξει στο μπλοκ εντολών σε σχέση με τη συνθήκη και η συνθήκη παραμείνει αληθής, προκαλείται ατέρμων βρόχος επανάληψης.

Ας δούμε ένα παράδειγμα μιας εντολής while που επιτρέπει να τυπωθούν οι αριθμοί από 0 μέχρι 4.

```
let count = 0;
while(count < 5) {
    console.log(count++);
}
```

Το πρόγραμμα τυπώνει τους αριθμούς αρχίζοντας από το 0 και αυξάνοντας τον μετρητή count σε κάθε επανάληψη. Όταν τυπωθεί και ο αριθμός 4 και αυξηθεί ο μετρητής count σε 5, τότε η συνθήκη count < 5 είναι ψευδής και τερματίζει η επαναληπτική εκτέλεση του μπλοκ εντολών.

Μια παραλλαγή της δομής αυτής είναι η while(true)/break. Στην περίπτωση αυτή θέτουμε ως συνθήκη μια πάντα αληθή έκφραση και ελέγχουμε τη συνθήκη εξόδου μέσα στο μπλοκ εντολών, όταν δε η συνθήκη εξόδου ικανοποιηθεί, τότε εκτελούμε την εντολή break που μας στέλνει εκτός του βρόχου. Θυμίζουμε ότι έχουμε δει την εντολή αυτή στην περίπτωση της εντολής case. Παράδειγμα του προηγούμενου προγράμματος με αυτή τη δομή είναι:

```
let count = 0;
while(true) {
    console.log(count++);
    if (count >= 5) break;
}
```

Η break χρησιμοποιείται γενικότερα για να μεταφέρουμε τον έλεγχο εκτέλεσης του προγράμματος σε άλλο σημείο, όπως εκτός ενός βρόχου επανάληψης. Μάλιστα, μπορούμε να ορίσουμε το όνομα της εντολής στην οποία μεταφέρεται ο έλεγχος του προγράμματος προς μια εντολή που φέρει όνομα: όνομα: εντολή; και να συνοδεύσουμε την εντολή break με το όνομα του στόχου: break όνομα; (δομή αντίστοιχη της goto που δεν συναντάται πλέον συχνά στον προγραμματισμό). Γενικά όμως θεωρείται όχι καλή πρακτική η χρήση της break, πλην της περίπτωσης εξόδου από έναν βρόχο επανάληψης.

Επίσης, μια άλλη εντολή που σχετίζεται με επαναληπτικές δομές είναι η εντολή continue. Η εντολή αυτή όταν βρεθεί μέσα σε ένα μπλοκ εντολών ενός βρόχου κάνει τον διερμηνευτή του προγράμματος να παραλείψει τις υπόλοιπες εντολές του βρόχου και να μεταβεί στην επόμενη επανάληψη.

Ένα παράδειγμα χρήσης της break για έξοδο από έναν βρόχο είναι η αναζήτηση μιας τιμής target σε έναν πίνακα:

```
let i = 0;
while (i < ar.length){
    if (ar[i++] === target) break;
}
```

Θα πρέπει να σημειωθεί ότι οι εντολές break και continue μπορούν να χρησιμοποιηθούν σε όλες τις δομές επανάληψης που περιγράφονται στη συνέχεια.

8.7.2 Εντολή do/while

Η do/while είναι μια παραλλαγή της while που επίσης συναντάται σε πολλές γλώσσες προγραμματισμού

(εξαίρεση η Python που δεν διαθέτει δομή do/while). Η διαφορά από τη while είναι ότι ο έλεγχος της συνθήκης γίνεται στο τέλος του βρόχου επανάληψης, και συνεπώς το μπλοκ εντολών εκτελείται τουλάχιστον μια φορά, ακόμη και αν η συνθήκη είναι ψευδής.

Η σύνταξη της do/while είναι:

```
do {
    μπλοκ-εντολών
} while (συνθήκη);
```

Η χρήση της δομής αυτής είναι πιο σπάνια από της while. Βεβαίως, και εδώ ισχύει ο κίνδυνος του ατέρμονος βρόχου σε περίπτωση που η συνθήκη παραμείνει αληθής χωρίς να επηρεάζεται από το μπλοκ εντολών.

Το προηγούμενο παράδειγμα εκτύπωσης των αριθμών 0 μέχρι 4 με αυτή τη δομή είναι:

```
let count = 0;
do {
  console.log(count++);
} while (count < 5);
```

8.7.3 Εντολή for

Η δομή for, η οποία όπως θα δούμε στη συνέχεια εμφανίζεται σε διάφορες παραλλαγές, είναι λίγο πιο σύνθετη από τη while αλλά χρησιμοποιείται πολύ πιο συχνά στον προγραμματισμό. Είναι μια δομή που συναντάται σε όλες τις γλώσσες προγραμματισμού.

Η πιο συνήθης έκδοση της for απαιτεί την ύπαρξη μιας βοηθητικής μεταβλητής, του μετρητή. Η μεταβλητή αυτή αρχικοποιείται πριν αρχίσει η εκτέλεση των επαναλήψεων, ελέγχεται σύμφωνα με κάποια συνθήκη στην αρχή κάθε επανάληψης και στο τέλος κάθε επανάληψης μεταβάλλεται (συνήθως αυξάνεται). Αυτές οι τρεις ενέργειες κωδικοποιούνται σε μια εντολή for ως εξής:

```
for (αρχικοποίηση; έλεγχος; τροποποίηση) {
    μπλοκ-εντολών
}
```

Αν προσπαθήσουμε να επαναλάβουμε το παράδειγμα των προηγούμενων ενοτήτων, να τυπώσουμε δηλαδή τους αριθμούς από 0 .. 4, ακολουθεί η υλοποίησή του με χρήση της δομής for:

```
for (let count = 0; count < 5; count++) {
    console.log(count);
}
```

Η πιο συνήθης χρήση της δομής αυτής είναι στη διαπέραση ενός πίνακα, όπως στο παράδειγμα:

```
const fruits = ['Μπανάνα', 'Αχλάδι', 'Μήλο', 'Μάνγκο'];
for (let i = 0; i < fruits.length; i++) {
  console.log(`${i + 1}. ${fruits[i]} `);
}
```

Ο κώδικας αυτός θα διαπεράσει τον πίνακα των φρούτων και θα τα τυπώσει:

```
1. Μπανάνα
2. Αχλάδι
3. Μήλο
4. Μάνγκο
```

Με την ευκαιρία, να υπενθυμίσουμε ότι αν η μεταβλητή μετρητής οριστεί με τη λέξη let, έχει εμβέλεια μόνο μέσα στο μπλοκ for.

Τέλος, να αναφέρουμε ότι μπορούμε να παραλείψουμε κάποια από τις εκφράσεις της for ή και όλες τις εκφράσεις, όμως θα πρέπει να διατηρήσουμε τα σύμβολα “;”. Για παράδειγμα, μια δομή που είναι ισοδύναμη με while (true) θα μπορούσε να γραφτεί ως δομή for ως εξής:

```
for (;;){
    μπλοκ-εντολών
}
```

8.7.4 Εντολή for/of

Η δομή for/of εισήχθη στην JavaScript με την έκδοση ES6. Είναι δομή που θυμίζει την αντίστοιχη δομή for της Python.

Η δομή αυτή επιτρέπει να διαπεράσουμε μια ακολουθία στοιχείων. Μια ακολουθία είναι, για παράδειγμα, ένας πίνακας ή μια συμβολοσειρά.

Η σύνταξη της εντολής αυτής είναι:

```
for (let στοιχείο of ακολουθία) {
    μπλοκ εντολών
}
```

Ο βρόχος επαναλαμβάνεται για κάθε στοιχείο της ακολουθίας.

Ένα παράδειγμα:

```
const fruits = ['Μπανάνα', 'Αχλάδι', 'Μήλο', 'Μάνγκο'];
for (let fruit of fruits) {
  console.log(fruit);
}
```

Εδώ θα πάρουμε ένα προς ένα τα ονόματα των φρούτων.

Ας δούμε ένα ακόμη παράδειγμα διαπέρασης των χαρακτήρων μιας συμβολοσειράς στο πρόβλημα καταγραφής συχνότητας εμφάνισης χαρακτήρων.

```
let freq = {}; // αντικείμενο JavaScript
for (let letter of 'Ελλάδα') {
  freq[letter] = (freq[letter] || 0) + 1;
}
freq;
> { 'Ε': 1, 'λ': 2, 'ά': 1, 'δ': 1, 'α': 1 }
```

Μερικές παρατηρήσεις για τη λύση αυτή: Χρησιμοποιούμε ένα αντικείμενο, το freq (θα γίνουν οι επίσημες συστάσεις αντικειμένων σε επόμενο κεφάλαιο, για την ώρα θεωρήστε ότι είναι παρόμοιο με ένα λεξικό της Python). Για κάθε γράμμα, αν το γράμμα υπάρχει ήδη στο αντικείμενο freq, προστίθεται στην τιμή της ιδιότητας με τιμή τον χαρακτήρα +1, αν δεν υπάρχει, δημιουργείται η ιδιότητα και αρχικοποιείται σε 1. Αυτό γιατί η έκφραση: a || b παίρνει την τιμή του πρώτου στοιχείου της έκφρασης το οποίο είναι αληθές, αλλιώς του τελευταίου στοιχείου.

8.7.5 Εντολή for/in

Η δομή for/in μοιάζει με τη for/of που είδαμε στην προηγούμενη ενότητα, υπάρχει στην JavaScript από παλαιότερες εκδόσεις και έχει ευρύτερη χρήση από την προηγούμενη, αφού στην περίπτωση αυτή το στοιχείο μπορεί να είναι οι ιδιότητες ενός αντικείμενου, όχι απαραίτητα μιας ακολουθίας.

Η σύνταξη είναι:

```
for (let στοιχείο in αντικείμενο) {
    μπλοκ εντολών
}
```

Για τους πίνακες και τις συμβολοσειρές το *στοιχείο* παίρνει τις τιμές του δείκτη που, ως γνωστόν, παίρνει τιμές από 0 μέχρι length-1.

Συνεπώς, το παράδειγμα της προηγούμενης ενότητας παρουσίασης των αγαπημένων μας φρούτων τροποποιείται ως ακολούθως:

```
const fruits = ['Μπανάνα', 'Αχλάδι', 'Μήλο', 'Μάνγκο'];
for (let indx in fruits) {
  console.log(fruits[indx]);
}
```

Είναι αντιληπτό από το παράδειγμα ότι η δομή for/in δεν εκφράζει με την ίδια απλότητα τη διαπέραση των στοιχείων ενός πίνακα. Για τον λόγο αυτό, για την περίπτωση πινάκων ή συμβολοσειρών, που είναι και η πιο συνήθης περίπτωση χρήσης του βρόχου for, η νεότερη δομή for/of είναι προτιμότερη.

8.8 Ερωτήσεις αυτοαξιολόγησης

1. Τι θα επιστρέψει ο παρακάτω κώδικας:

```
let a;
console.log(a==null);
```

    1. true
    2. false
    3. “true”
    4. “error”
    5. Syntax Error

2. Έστω σε αρχείο HTML το οποίο δεν περιέχει υπερσυνδέσμους, ποια η τιμή της μεταβλητής a;
let a = document.querySelector("a");
Απάντηση: ……

3. Τι θα επιστρέψει ο παρακάτω κώδικας:

```
let a;
console.log(typeof a);
```

    1. 'null'
    2. 'undefined'
    3. null
    4. undefined
    5. Syntax Error

4. Τι θα επιστρέψει ο κώδικας:

```
let b = 5;
typeof(b>1);
```

    1. ‘boolean’
    2. true
    3. false
    4. ‘true’

5. Ποιο το αποτέλεσμα;

```
let a = 10;
let b = 9;
console.log(a>10 || b>=9 )
```

Απάντηση: _____

6. Ποιο το αποτέλεσμα;

```
let a = 8
let b = 2
console.log(a | b)
```

Απάντηση: _____

7. Ποιο το αποτέλεσμα;

```
let a = 8
console.log(a << 2)
```

Απάντηση: _____

8. Ποιο το αποτέλεσμα;

```
let a = -10/0;
console.log(a)
```

    1. NaN

2. Infinity
3. -Infinity
4. ZeroDivisionError

9. Ποιο το αποτέλεσμα: 10 == '10'
   Απάντηση: _____
10. Ποιο το αποτέλεσμα;

```
for(let i = 0; i<10; i++) {}
console.log(i);
```

1. 10
2. 11
3. ReferenceError

11. Ποιο το αποτέλεσμα;

```
console.log('x=', typeof x, x);
var x=1;
```

1. ReferenceError: x is not defined
2. undefined undefined
3. ‘undefined’ undefined

12. Ποιο το αποτέλεσμα;

```
let x =  10;
let x = 5;
console.log(x);
```

1. Duplicate declaration
2. 10
3. 5
4. 15

13. Ποια η τιμή του y;

```
const x=1;
let y = x++;
```

1. 1
2. 2
3. TypeError
4. -1

14. Ποιο το αποτέλεσμα; typeof (10+'10')
    1. TypeError
    2. ‘number’
    3. ‘string’
    4. integer
15. Ποιο το αποτέλεσμα;
    console.log(3 > Number(1+'1'))
    Απάντηση: _____
16. Ποιο το αποτέλεσμα;
    console.log(Number("καλημέρα".length+ "5"));
    Απάντηση: _____
17. Ποιο το αποτέλεσμα;

```
let myCountry="Greece"
console.log(myCountry.slice(4))
```

    Απάντηση: _____
18. Ποιο το αποτέλεσμα;

```
let myCountry="Greece"
console.log(myCountry.indexOf('rec'))
```

Απάντηση: _____

19. Ποιο το αποτέλεσμα;

```
console.log(`${3+2}${3*2}`);
```

Απάντηση: _____

20. Ποιο το αποτέλεσμα;

```
let x = 10;
if (0<x && x<5){console.log('a')}
else if (5<=x && x<=10){console.log('b')}
else console.log('c')
```

Απάντηση: _____

21. Ποιο το αποτέλεσμα;

```
let x = 1;
console.log((typeof x === 'string')?1:2);
```

Απάντηση: _____

22. Αν έχουμε να επιλέξουμε μεταξύ 5 διαφορετικών τιμών μιας μεταβλητής x, ποια η δομή που θα ήταν καλύτερο να χρησιμοποιήσουμε;
    1. if (x==v1) {} else if (x==v2){} ...
    2. try {x==v1} catch { ... }
    3. (x==v1) ? v2 : v3;
    4. switch(x){case v1: ...}
23. Ποιο το αποτέλεσμα;

```
for (var _i=0; _i<10; _i+=3){}
console.log(_i)
```

Απάντηση: _____

24. Ποιο το αποτέλεσμα;

```
let s = "";
for (let x=10; x>5 ;x--){s+=x}
console.log(s);
```

    1. ‘10 9 8 7 6 5’
    2. ‘10,9,8,7,6,5’
    3. ‘1098765’
    4. ‘10 9 8 7 6’
    5. ‘109876’

25. Συμπληρώστε τον κώδικα ώστε το αποτέλεσμα να είναι 123.

```
let s=[120,121,122]
for (let i of s){
    if (s.indexOf(i) == s.length-1)console.log( ......)}
```

Απάντηση: _____

26. Ποιο το αποτέλεσμα;

```
let ar=[5,10,15]
let s=''
for (let i in ar){s+=i}
console.log(s)
```

1. '51015'
2. '5 10 15'
3. '012'

27. Συμπληρώστε τον κώδικα ώστε το αποτέλεσμα να είναι “123”.

```
let s='';
let i=1;
while (true){
    s+=i
    ......
    i++}
console.log(s);
```

1. if(i>2)break;
2. if(i>3)break;
3. if(i>4)break;

28. Έστω ο παρακάτω κώδικας

```
let i=1;
while (i<10){
    i++;
    if (i%2 ===0 || i%3===0 || i%5===0)continue;
    console.log(i);
}
```

Ποιο το αποτέλεσμα;

1. Θα τυπώσει τις τιμές 1 έως 9.
2. Θα τυπώσει τις τιμές 1 7.
3. Θα τυπώσει την τιμή 7.
4. Θα τυπώσει τις τιμές 1 3 7.

29. Από τις δομές for, while και do while, ποια είναι αυτή που θα εκτελεστεί τουλάχιστον μια φορά;
    1. Καμιά από τις 3.
    2. Η δομή for.
    3. Η δομή while.
    4. Η δομή do while.

8.9 Βιβλιογραφία και Αναφορές

Και στο κεφάλαιο αυτό ισχύει η βιβλιογραφία του προηγούμενου κεφαλαίου. Το πρότυπο EcmaScript (ECMA-262) συντηρείται από τον οργανισμό ecma.

Στο διαδίκτυο υπάρχουν πολλές πηγές για εκμάθηση της JavaScript καθώς και για αναφορά στα στοιχεία της γλώσσας. Η MDN είναι μια καλή πηγή για εισαγωγικά και προχωρημένα μαθήματα, όπως και για την HTML και τη CSS. Επίσης, η w3schools περιέχει μαθήματα, ενώ μια πλήρη σειρά μαθημάτων για την JavaScript με παραδείγματα περιλαμβάνει η JavaScript.info, ξεκινώντας από τη γλώσσα και προχωρώντας στη διεπαφή της με τον φυλλομετρητή.

Η JavaScript περιλαμβάνεται σε βιβλία που ήδη αναφέρθηκαν που αφορούν τις τεχνολογίες του προγραμματισμού στον ιστό όπως η HTML και CSS. Αυτή την προσέγγιση στην ελληνική βιβλιογραφία ακολουθεί το βιβλίο των Δουληγέρη κ.ά. (2021), ενώ έχουν μεταφραστεί κάποια συγγράμματα και διατίθενται από τον Εύδοξο, όπως το βιβλίο των Kyrnin και Morrison (2021) και αυτό των Lemay, Coburn και Kyrnin (2016).

Μια άλλη προσέγγιση είναι αυτή εγχειριδίων που ξεκινούν με τη σύνταξη της γλώσσας JavaScript και περιλαμβάνουν τη διεπαφή με το DOM σε μεταγενέστερο κεφάλαιο. Στις πηγές αυτές συχνά περιλαμβάνονται και κεφάλαια που αφορούν τη λειτουργία της γλώσσας στο περιβάλλον node.js. Αυτή την προσέγγιση ακολουθεί το βιβλίο του Λιακέα (2021). Από τη διεθνή βιβλιογραφία παρόμοια προσέγγιση ακολουθεί το βιβλίο του Flanagan (2020), ενώ υπάρχουν και πολλά άλλα, όπως Frisbie (2020) κ.λπ.

Επιπλέον, οι συγγραφείς αυτού του βιβλίου έχουν δημιουργήσει *ανοιχτά διαδικτυακά μαθήματα* στην πλατφόρμα mathesis, υλικό από τα οποία έχει χρησιμοποιηθεί και σε αυτό το σύγγραμμα. Για αυτό το κεφάλαιο το σχετικό μάθημα είναι οι εισαγωγικές διαλέξεις του μαθήματος «Εισαγωγή στην ανάπτυξη ιστοσελίδων με HTML5, CSS3, Javascript».

A. Ξενόγλωσσες

Flanagan, D. (2020). *JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language* (7th ed.). O'Reilly Media, Inc.

Frisbie, M. (2020). *Professional JavaScript for Web Developers*. Wiley.

McFedries, P. (2019). *Web Design Playground - HTML & CSS The Interactive Way*. Manning.

B. Ελληνόγλωσσες

Αβούρης, Ν. (2018). Εισαγωγή στην ανάπτυξη ιστοσελίδων με HTML5, CSS3, JavaScript. Ανοικτό διαδικτυακό μάθημα, https://mathesis.cup.gr

Δουληγέρης, Χ., Μαυροπόδη, Ρ., Κοπανάκη, Ε., & Καραλής, Α. (2021). *Τεχνολογίες και Προγραμματισμός στον Παγκόσμιο Ιστό* (2η έκδ.). Εκδόσεις Νέων Τεχνολογιών. Κωδικός βιβλίου στον Εύδοξο: 102125023.

Kyrnin, J., & Meloni, J. (2021). *Sams Teach Yourself HTML5, CSS and Javascript* (3rd ed.). Εκδόσεις Γκιούρδας.

Lemay, L., Coburn, R., & Kyrnin, J. (2016). *Sams Teach Yourself HTML, CSS & JavaScript* (7th ed). Εκδόσεις Γκιούρδας. Κωδικός βιβλίου στον Εύδοξο: 59357307.

Λιακέας, Γ. (2021). *Η γλώσσα JavaScript* (3η έκδ.). Κλειδάριθμος. Κωδικός βιβλίου στον Εύδοξο: 102070465.