Κεφάλαιο 7. Εισαγωγή στην JavaScript

Σύνοψη

*Στο κεφάλαιο αυτό ξεκινάμε την εισαγωγή στη γλώσσα προγραμματισμού **JavaScript**, που αποτελεί μια βασική τεχνολογία του διαδικτυακού προγραμματισμού και μια από τις γλώσσες προγραμματισμού που έχουν ευρύτατη χρήση όχι μόνο στον φυλλομετρητή αλλά και έξω από αυτόν, για προγραμματισμό του εξυπηρετητή (στο περιβάλλον Node.js, που παρουσιάζεται στο κεφάλαιο 11), καθώς και για ανάπτυξη μη διαδικτυακών διαδραστικών εφαρμογών στην επιφάνεια εργασίας του υπολογιστή μας (με το περιβάλλον electron.js).*

Η γλώσσα JavaScript είναι αντικείμενο αυτού και των επόμενων κεφαλαίων. Στο κεφάλαιο αυτό γίνεται η πρώτη γνωριμία με την JavaScript. Ξεκινάμε με μια ιστορική ανασκόπηση της εξέλιξης της γλώσσας, στη συνέχεια θα δούμε βασικά στοιχεία σύνταξης της γλώσσας, όπως τους τρόπους δήλωσης μεταβλητών και τους βασικούς τελεστές που χρησιμοποιούνται σε μαθηματικές εκφράσεις. Στη συνέχεια θα συζητήσουμε τρόπους για να ενσωματώσουμε κώδικα JavaScript σε μια ιστοσελίδα. Θα δούμε πώς η JavaScript έχει πρόσβαση στις ιδιότητες του global object (window) και στο DOM (document object model).

Στο επόμενο κεφάλαιο θα προχωρήσουμε με τους βασικούς τύπους δεδομένων και τις κύριες εντολές της JavaScript.

Προαπαιτούμενη γνώση

Για την κατανόηση αυτού του κεφαλαίου είναι επιθυμητή η γνώση της HTML (κεφάλαια 2 και 3) και της CSS (κεφ 4 και 5), καθώς συζητείται και παρουσιάζεται με παραδείγματα η χρήση της γλώσσας JavaScript ως προγραμματιστική διεπαφή με τα στοιχεία μιας ιστοσελίδας.

7.1 Εισαγωγή – Ιστορικό σημείωμα

Στην πρώτη αυτή ενότητα θα δώσουμε μια γενική εισαγωγή με κύρια την ιστορική διάσταση και τον ρόλο της JavaScript. Η JavaScript είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου, όπως η Python, η Java κ.λπ. Ακολουθεί ένα πρότυπο, το πρότυπο ECMAScript. Το επίσημο όνομά της είναι **ECMAScript** αλλά έχει καθιερωθεί να χρησιμοποιούμε το εμπορικό όνομα που αρχικά τής δόθηκε, JavaScript.

Η ιστορία της αρχίζει το 1995 από τον μηχανικό της εταιρείας Netscape Brendan Eich. Ο Netscape ήταν την εποχή εκείνη ο πιο δημοφιλής φυλλομετρητής. Στο πλαίσιο του Netscape αναπτύχθηκε η πρώιμη έκδοση της JavaScript ως γλώσσα προγραμματισμού μιας ιστοσελίδας, αφού είχε προκύψει η ανάγκη οι ιστοσελίδες να γίνουν πιο διαδραστικές.

Η JavaScript μαζί με την HTML και τη CSS, που είδαμε σε προηγούμενα κεφάλαια, είναι οι βασικές τεχνολογίες για να αναπτύσσουμε εφαρμογές στον παγκόσμιο ιστό, στο διαδίκτυο. Σύντομα θα δούμε πώς συνδέεται η JavaScript με τις προηγούμενες δύο τεχνολογίες που έχουμε ήδη δει.

Όταν κατεβάζουμε ένα αρχείο HTML (που συνοδεύεται από ένα φύλλο μορφοποίησης CSS, είτε σε ξεχωριστό αρχείο είτε στο ίδιο το αρχείο), μπορεί μέσα στο ίδιο το αρχείο HTML ή σε ένα συνοδευτικό διασυνδεδεμένο αρχείο να υπάρχει κώδικας JavaScript. Ο κώδικας αυτός, λοιπόν, πρέπει να εκτελεστεί. Αυτό γίνεται μέσα στον ίδιο τον φυλλομετρητή, ο οποίος διαθέτει μια «μηχανή» που μπορεί να κάνει συντακτική ανάλυση και μεταγλώττιση της JavaScript σε μια γλώσσα που εκτελείται από τον υπολογιστή (γλώσσα μηχανής).

Επίσης, θα πρέπει να σημειώσουμε ότι η εξαιρετική δημοφιλία αυτής της τεχνολογίας έχει οδηγήσει ώστε να αποκτήσει και άλλες χρήσεις, εκτός από γλώσσα που χρησιμοποιείται στον φυλλομετρητή, στις εφαρμογές του διαδικτύου.

Για παράδειγμα, με το runtime περιβάλλον Node.js η γλώσσα αυτή χρησιμοποιείται ως μια παραδοσιακή γλώσσα, στην ουσία για προγραμματισμό εφαρμογών στον εξυπηρετητή, να δρομολογεί αιτήματα για παροχή ιστοσελίδων, να συνδέει τις εφαρμογές αυτές με μια βάση δεδομένων, το σύστημα αρχείων κ.λπ.

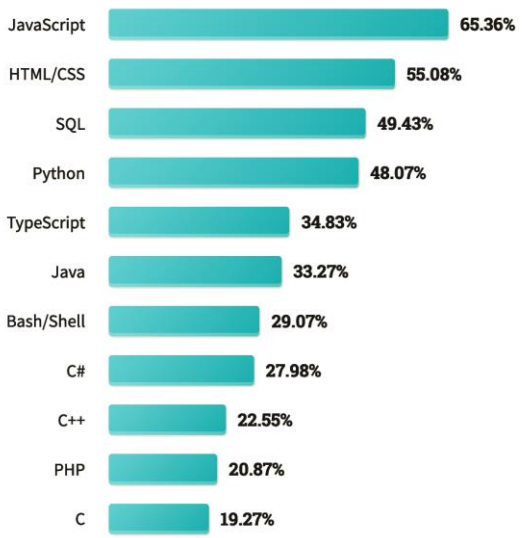
7.1.1 Κύρια χαρακτηριστικά της JavaScript

Ας δούμε τα κύρια χαρακτηριστικά της JavaScript.

- Η JavaScript χαρακτηρίζεται καταρχάς ως δυναμική γλώσσα με **ασθενή διαχείριση τύπων δεδομένων** (weakly typed language). Η JavaScript δεν είναι πολύ αυστηρή ως προς τους τύπους δεδομένων και συχνά παραβλέπει ασυμβατότητα τύπων σε εκφράσεις κάνοντας αυτόματα μετατροπές τύπων. Για παράδειγμα, η έκφραση 5 + "10" στις περισσότερες γλώσσες προγραμματισμού θα οδηγήσει σε σφάλμα, αφού δεν επιτρέπεται η πρόσθεση ασύμβατων τύπων δεδομένων (στην Python θα παραγάγει TypeError). Στην JavaScript θα δώσει το αποτέλεσμα "510", που έχει παραχθεί από τη μετατροπή του αριθμού 5 στη συμβολοσειρά “5” και στη συνέχεια στη συνένωση των δύο συμβολοσειρών.
- Συχνά αυτό το χαρακτηριστικό της JavaScript είναι αιτία κριτικής στη γλώσσα, αφού μπορεί να προκαλέσει ανεπιθύμητα σφάλματα που είναι δύσκολο να διαγνωστούν. Για αντιμετώπιση του προβλήματος αυτού έχουν δημιουργηθεί διάλεκτοι της γλώσσας, όπως η TypeScript που χειρίζονται τα δεδομένα με πιο αυστηρό τρόπο.
- Η JavaScript είναι αντικειμενοστραφής γλώσσα ως προς την αρχιτεκτονική της, τα περισσότερα δεδομένα της είναι αντικείμενα, όμως ακολουθεί ένα ιδιαίτερο μηχανισμό κληρονομικότητας που στηρίζεται σε «πρωτότυπα», όπως θα περιγραφεί σε επόμενο κεφάλαιο.
- Επίσης, η JS υποστηρίζει διαφορετικά προγραμματιστικά παραδείγματα, όπως τον προγραμματισμό με συμβάντα (**event-based programming**), συναρτησιακό προγραμματισμό (**functional programming**), διαθέτει δομές για να προγραμματίσουμε με αντικείμενα (**object-oriented programming**).
- Η JavaScript διαθέτει προγραμματιστικές διεπαφές (APIs) για χειρισμό κειμένων, πινάκων, ημερομηνιών, τυπικών εκφράσεων (regular expression), καθώς και κάτι που μας ενδιαφέρει ιδιαίτερα, διαθέτει προγραμματιστική διεπαφή προς το DOM (Document Object Model), δηλαδή προς αντικείμενα που αντιστοιχούν στα στοιχεία ενός εγγράφου HTML που προβάλλεται από τον φυλλομετρητή.
- Δεν περιλαμβάνει διεπαφή για αλληλεπίδραση απευθείας με τον χρήστη, π.χ. δεν υπάρχει αντίστοιχη εντολή print(), επίσης δεν διαθέτει διεπαφή για δικτύωση, για μόνιμη αποθήκευση στο σύστημα αρχείων ή σε βάση δεδομένων ή γραφική διεπαφή.

7.1.2 Χρήση της JavaScript

Για να έχουμε μια ιδέα του πόσο δημοφιλής είναι η JavaScript, δεν έχουμε παρά να εξετάσουμε ένα πρόσφατο ετήσιο developer survey της δημοφιλούς ιστοσελίδας StackOverflow, που έχει ευρύτατη χρήση στην κοινότητα των προγραμματιστών. Στο ερώτημα για την πιο δημοφιλή τεχνολογία οι απαντήσεις που δόθηκαν κατά την ετήσια επισκόπηση του 2022 φαίνονται στην **Εικόνα *7.1***.



Εικόνα 7.1 *Δημοφιλία της JavaScript σύμφωνα με την ετήσια επισκόπηση του StackOverflow.* *https://survey.stackoverflow.co/2022/#technology*

Μάλιστα, σύμφωνα με την ετήσια αυτή έρευνα, η γλώσσα JavaScript παραμένει στη θέση τής πιο δημοφιλούς τεχνολογίας για 10η συνεχή χρονιά, με βάση τις απαντήσεις των μελών του StackOverflow. Θα πρέπει ακόμη

να παρατηρηθεί ότι η TypeScript, που αποτελεί διάλεκτο της JavaScript, βρίσκεται επίσης στις πρώτες θέσεις των πιο δημοφιλών τεχνολογιών.

Αν εξετάσουμε ιστορικά την εξέλιξη της JavaScript, θα παρατηρήσουμε ότι αρχικά η γλώσσα αυτή δημιουργήθηκε για να υποστηρίζει και να επιτρέπει διαδραστικότητα σε έγγραφα HTML (ιστοσελίδες). Παραδείγματος χάρη, χρησιμοποιούσαμε τα σκριπτ της JavaScript για να ελέγχουμε την εγκυρότητα δεδομένων που δίνει ο χρήστης σε μια φόρμα, να υπάρχει διαδραστικότητα σε συμβάντα που προκαλεί ο χρήστης όταν χειρίζεται το ποντίκι, το πληκτρολόγιο, την οθόνη αφής κ.λπ.

Σήμερα, εκτός από αυτή τη χρήση, που παραμένει πολύ σημαντική, υπάρχει και μια άλλη χρήση που θα τη δούμε περισσότερο στη συνέχεια. Από το τέλος της δεκαετίας του ’90 και μετά, με την έλευση της AJAX (*“Asynchronous JavaScript and XML”*), που επέτρεπε μια ιστοσελίδα να κάνει ασύγχρονες κλήσεις προς τον εξυπηρετητή, άρχισε η ανάπτυξη νέου τύπου διαδικτυακών εφαρμογών, οι εφαρμογές μοναδικής ιστοσελίδας (single page applications, SPA), οι οποίες στέλνουν και λαμβάνουν δεδομένα από τον εξυπηρετητή χωρίς να απαιτείται ξαναφόρτωμα της σελίδας. Τέτοιες εφαρμογές, που συναντάμε όλο και πιο συχνά στο διαδίκτυο, είναι εφαρμογές μέσων κοινωνικής δικτύωσης, χαρτογραφικού περιεχομένου, σχεδίασης, διαχείρισης κειμένων και φύλλων εργασίας, και μοιάζουν όλο και περισσότερο με διαδραστικές εφαρμογές που έχουμε συνηθίσει στην επιφάνεια εργασίας των υπολογιστών μας.

Αυτές δεν ακολουθούν το μοντέλο των ιστοσελίδων που φορτώνονται διαδοχικά στον φυλλομετρητή, αλλά συνήθως είναι σελίδες που κατεβαίνουν, εγκαθίστανται και, με διαδοχικές κλήσεις στον εξυπηρετητή, ανανεώνεται το περιεχόμενό τους. Αυτού του τύπου οι εφαρμογές είναι γραμμένες σε JavaScript και είναι συνήθως αρκετά σύνθετες, με πολλές γραμμές κώδικα. Το τμήμα της εφαρμογής που τρέχει στον εξυπηρετητή παίζει σε αυτή την περίπτωση λιγότερο σημαντικό ρόλο και, κυρίως, παρέχει υπηρεσίες όπως σύνδεση με βάση δεδομένων και άλλες πηγές μόνιμης αποθήκευσης κ.λπ.

7.1.3 Εκδόσεις της JavaScript

Ας εξετάσουμε ιστορικά τις πιο σημαντικές εκδόσεις της JavaScript ή ECMAScript, όπως είναι το επίσημο όνομά της:

- Η πρώτη σημαντική έκδοση ήταν η **ES3 (JavaScript 3)**, που ήταν το πρώτο πλήρες πρότυπο, στα πλαίσια του οποίου εισήχθησαν σημαντικές νέες λειτουργίες, όπως κανονικές εκφράσεις (regular expression literals), χειριστής εξαιρέσεων try/catch κ.λπ.
- Το 2009 ορίζεται το πρότυπο **ES5 (JavaScript 5)**, που υποστηρίζουν σχεδόν όλες οι εκδόσεις των φυλλομετρητών και περιλαμβάνει τις **συναρτήσεις πινάκων** map() reduce() filter() forEach, υποστήριξη JSON, ορισμό getters και setters για πρόσβαση στις ιδιότητες αντικειμένων κ.λπ.
- Η πιο σημαντική νέα έκδοση ήταν η **ES6 (ES2015)**, που θεωρείται η πιο ανεπτυγμένη και ώριμη έκδοση της γλώσσας. Η έκδοση αυτή περιλαμβάνει μεγάλες βελτιώσεις όπως τον μηχανισμό υποσχέσεων (*promises*) για ασύγχρονη εκτέλεση, εισαγωγή modules, ορισμό κλάσεων με τη λέξη *class*, ορισμό μεταβλητών με εμβέλεια στο block (*let*), συναρτήσεις βέλη (arrow functions **() => { }**), ορισμό συμβολοσειρών με ενσωμάτωση μεταβλητών (template literal), τον τελεστή spread **...*** ορισμό προκαθορισμένων τιμών σε παραμέτρους συναρτήσεων, ορισμό του τύπου symbol κ.λπ.
- Έκτοτε σε ετήσια βάση ανακοινώνονται καινούργιες εκδόσεις της γλώσσας που φέρουν το όνομα του έτους, ES2020, ES2021 κ.λπ. Όμως, μετά την έκδοση ES2015 οι αλλαγές είναι λιγότερο ρηξικέλευθες και ούτως ή άλλως η πλήρης υιοθέτηση πρόσφατων αλλαγών από τους φυλλομετρητές απαιτεί χρόνο.

Ουσιαστικά, η έκδοση ES6 είναι το τρέχον πρότυπο της JavaScript και αυτή θα χρησιμοποιήσουμε στο βιβλίο αυτό, όμως επειδή υπάρχει πολύ εγκατεστημένο λογισμικό και σε προηγούμενες εκδόσεις, κυρίως την ES5, θα πρέπει να λάβουμε υπόψη μας ποια ήταν η λειτουργία της γλώσσας σε προηγούμενες εκδόσεις. Θα πρέπει να αναφερθεί επίσης ότι υπάρχουν ειδικά εργαλεία για μεταγλώττιση μεταξύ εκδόσεων (transpilers), που επιτρέπουν να γράψουμε κώδικα στην έκδοση ES6 και να τον μεταγλωττίσουμε σε ES5 ώστε να μπορεί να εκτελεστεί σχεδόν σε όλους του φυλλομετρητές.

Η JavaScript σήμερα συνοδεύεται από ένα πλούσιο οικοσύστημα από εργαλεία, βιβλιοθήκες και πλαίσια ανάπτυξης. Η πιο σημαντική βιβλιοθήκη είναι jQuery η οποία υπάρχει από το 2006 αν και τείνει να υποχωρήσει αφού οι νέες εκδόσεις της JavaScript έχουν υποκαταστήσει πολλές από τις λειτουργίες της jQuery. Αξίζει να κάνουμε επίσης αναφορά σε κάποια από τα framework για το front-end που χρησιμοποιούνται ευρύτατα σήμερα όπως είναι το React, το Angular το Vue.js και τα οποία βασίζονται στην JavaScript.

7.1.4 Μηχανές εκτέλεσης κώδικα JavaScript

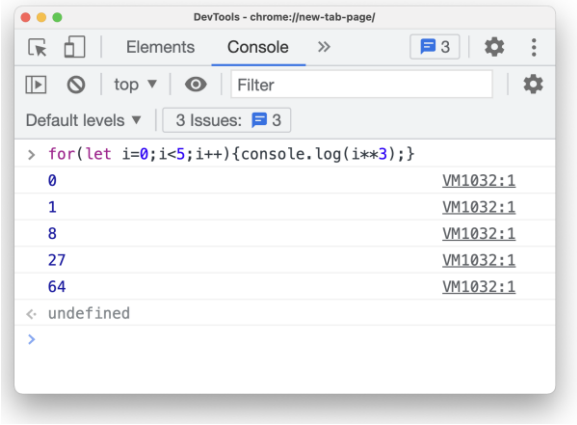
Ας δούμε τις βασικές τεχνολογίες που υπάρχουν για εκτέλεση κώδικα JavaScript. Θα πρέπει να αναφερθεί ότι κάθε φυλλομετρητής έχει διάφορες διεργασίες που εκτελούνται όταν αρχίσει το φόρτωμα μιας ιστοσελίδας. Αρχίζει μια διεργασία που κάνει συντακτική ανάλυση του κώδικα HTML και κτίζει το DOM, όπως είδαμε σε προηγούμενο κεφάλαιο. Αυτή η διεργασία, σε συνδυασμό με τον συντακτικό αναλυτή των σχετικών φύλλων CSS, τροφοδοτεί τη διεργασία που ονομάζεται *rendering engine*, που είναι η διεργασία που εμφανίζει την ιστοσελίδα στο παράθυρο του φυλλομετρητή. Επίσης, υπάρχει η *JavaScript engine* που αναλαμβάνει τη διερμηνεία και την εκτέλεση του κώδικα JavaScript που σχετίζεται με το συγκεκριμένο έγγραφο HTML. Μάλιστα, οι σύγχρονοι φυλλομετρητές δεν διερμηνεύουν την JavaScript αλλά τη μεταγλωττίζουν με τεχνολογία "just in time" για καλύτερη απόδοση.

Οι κυρίες μηχανές JavaScript που έχουμε στους φυλλομετρητές σήμερα είναι η **V8** που υπάρχει στον *Chrome* καθώς και στη **Node.js** (Ενότητα 11), που είναι το περιβάλλον εκτέλεσης εφαρμογών JavaScript στον εξυπηρετητή. Μια άλλη μηχανή JavaScript είναι η **SpiderMonkey** που είναι ενσωματωμένη στον φυλλομετρητή *FireFox*, καθώς και η **JavaScriptCore** (Nitro) που βρίσκεται στον φυλλομετρητή *Safari*.

7.2 Ανάπτυξη και αποσφαλμάτωση κώδικα JavaScript

Για την ανάπτυξη και την αποσφαλμάτωση κώδικα JavaScript μπορούμε να χρησιμοποιήσουμε οποιοδήποτε περιβάλλον ανάπτυξης, όπως το *Visual Studio Code*. Η εκτέλεση του κώδικα θα γίνει στο περιβάλλον του φυλλομετρητή αφού ενσωματώσουμε τον κώδικα σε μια άδεια σελίδα HTML. Τα μηνύματα σφάλματος εμφανίζονται στην κονσόλα στα εργαλεία προγραμματιστή του φυλλομετρητή (ενεργοποιούνται συνήθως με function-F12).

Αν επιθυμούμε να πειραματιστούμε με τμήματα κώδικα, μπορούμε να χρησιμοποιήσουμε την κονσόλα του φυλλομετρητή, την οποία μάλιστα μπορούμε να αποσπάσουμε σε ξεχωριστό παράθυρο και εκεί να γράψουμε κώδικα, ο οποίος εκτελείται αμέσως. Ένα παράδειγμα εκτέλεσης ενός βρόχου εκτύπωσης των κύβων των αριθμών 0...4 φαίνεται στην **Εικόνα *7.2*** για την κονσόλα του φυλλομετρητή Chrome.



Εικόνα 7.2 *Εκτέλεση κώδικα JavaScript στην κονσόλα του Chrome.*

Μια εναλλακτική επιλογή που επιτρέπει δοκιμές με σκριπτ της JavaScript σε περιβάλλον Node.js είναι η χρήση της εφαρμογής RunJS που διατίθεται σε δωρεάν έκδοση. Η εφαρμογή αυτή παρουσιάζει στην τυπική της μορφή δύο παράθυρα, στο ένα γράφουμε κώδικα και στο δίπλα βλέπουμε το αποτέλεσμα, όπως φαίνεται στην **Εικόνα *7.3***, για το κλασικό πρόγραμμα helloWorld.

Εικόνα 7.3 *Εκτέλεση κώδικα JavaScript στο περιβάλλον RunJS.*

Η σύνταξη της γλώσσας είναι κοινή στα δύο αυτά περιβάλλοντα (εξάλλου και τα δύο χρησιμοποιούν την ίδια μηχανή JS, τη V8), αν και στη δεύτερη περίπτωση λείπουν κάποιες διεπαφές, όπως η διεπαφή στα DOM και BOM, που παρέχονται στο περιβάλλον του φυλλομετρητή, όπως θα περιγραφεί στη συνέχεια του κεφαλαίου αυτού.

7.3 Σύνταξη ενός προγράμματος JavaScript

Όπως μπορείτε να διαπιστώσετε από τα παραδείγματα της προηγούμενης ενότητας, η σύνταξη της JavaScript μοιάζει με άλλες γλώσσες προγραμματισμού, όπως η C, Java κ.λπ. Είναι ευαίσθητη στη διαφορά κεφαλαίων και μικρών γραμμάτων και ο κώδικας του προγράμματος γράφεται με κωδικοποίηση UTF-8, κάτι που σημαίνει ότι μπορούμε να εισαγάγουμε ελληνικούς χαρακτήρες ως χαρακτήρες σε συμβολοσειρές (string) χωρίς πρόβλημα.

Ο χαρακτήρας **;** ορίζεται ως τερματικός χαρακτήρας εντολής, που σημαίνει ότι μπορούν να συνυπάρξουν πολλαπλές εντολές σε μια γραμμή. Όμως, σε αντίθεση με άλλες γλώσσες προγραμματισμού, η JavaScript δεν θεωρεί τον τερματικό χαρακτήρα υποχρεωτικό και μπορεί να παραληφθεί αν ο συντακτικός αναλυτής μπορεί να συνάγει σαφώς πού τελειώνει η κάθε εντολή. Γενικά είναι καλή πρακτική να χρησιμοποιείται πάντα ο τερματικός χαρακτήρας στο τέλος εντολών, αν και κάποιοι προγραμματιστές προτιμάνε να τον χρησιμοποιούν μόνο όταν είναι απαραίτητος, δηλαδή όπου η παράλειψή του μπορεί να προκαλέσει σύγχυση στον συντακτικό αναλυτή.

Τα σχόλια μιας γραμμής αρχίζουν με τους χαρακτήρες // ενώ σχόλια πολλών γραμμών αρχίζουν με την ακολουθία χαρακτήρων **/*** και τερματίζουν με την ακολουθία ***/**

```
// σχόλιο μιας γραμμής

/* αυτό είναι ένα
σχόλιο που εκτείνεται
σε πολλές γραμμές */
```

Τα ονόματα σταθερών, μεταβλητών, συναρτήσεων, κλάσεων, γνωρισμάτων αντικειμένων κ.λπ. (identifiers) στην JavaScript πρέπει να ακολουθήσουν κάποιους κανόνες. Τα ονόματα αυτά πρέπει να περιέχουν γράμματα, αριθμούς ή τα σύμβολα _ και **$**. Όμως, ένα όνομα δεν πρέπει να αρχίζει με αριθμητικό χαρακτήρα.

Συνεπώς, επιτρεπτά ονόματα μεταβλητών είναι τα **_**, **$**, **_1**, **$1**, **a1** ενώ δεν είναι επιτρεπτά τα **1a**, **1$**, **12**, **1_**.

Επίσης, υπάρχουν κάποιες δεσμευμένες λέξεις που δεν μπορούν να χρησιμοποιηθούν, όπως τα ονόματα εντολών ή δομών **if**, **else**, **const**, **null**, **continue**, **this**, **while**, **false**, **return**, **throw**, **with**, **break**, **in**, **true**, **case**, **for**, **instanceof**, **try**, **catch**, **let**, **super**, **typeof**, **class**, **function**, **new**, **switch**, **var** κ.λπ.

Θεωρητικά, θα μπορούσε να χρησιμοποιηθεί οποιοσδήποτε χαρακτήρας UTF-8 στο όνομα μιας μεταβλητής, άρα δεν είναι λάθος να γράψουμε:

```
μεταβλητή = 10;
console.log(μεταβλητή)
```

Όμως, δεν θεωρείται καλή πρακτική για τη μεταφερσιμότητα του κώδικά μας να χρησιμοποιήσουμε χαρακτήρες εκτός ASCII (λατινικό αλφάβητο) στα ονόματα μεταβλητών.

Κατά σύμβαση, τα ονόματα των μεταβλητών, των γνωρισμάτων αντικειμένων και των συναρτήσεων ακολουθούν τη σημειογραφία καμήλας camelNotation, δηλαδή αρχίζουν με μικρό γράμμα και, όταν περιέχουν πολλές λέξεις (συνήθως της αγγλικής γλώσσας), κάθε αρχικό των επόμενων λέξεων γράφεται με κεφαλαίο, γραφή που θυμίζει τις καμπούρες της καμήλας, εξού και το όνομα.

Ο τύπος των δεδομένων που θα εκχωρηθούν σε μια μεταβλητή ή μια σταθερά δεν ορίζεται κατά τη δήλωση της μεταβλητής, αλλά προκύπτει από την αρχική τιμή που η μεταβλητή θα πάρει. Η διάλεκτος TypeScript προσπαθεί να διορθώσει αυτό το πρόβλημα.

Θα πρέπει να δηλώσουμε τις μεταβλητές ή τις σταθερές πριν τις χρησιμοποιήσουμε με τις λέξεις **let**, **const**, **var**, όπως θα δούμε στη συνέχεια.

7.3.1 Δήλωση μεταβλητών let, const, var

Σε αυτή την ενότητα θα δούμε τρόπους που διαθέτουμε για δήλωση μεταβλητών ή σταθερών.

Μέχρι την έκδοση ES5 της JavaScript είχαμε μόνο έναν τρόπο να δηλώνουμε μεταβλητές, με τη λέξη-κλειδί **var** (για variable).

```
var myName = 'Nikos';
var myAge =  75;
```

Με την έκδοση ES6 προστέθηκαν δύο ακόμη λέξεις κλειδιά για δήλωση μεταβλητών, η **let** (let it be…) και η **const** (constant).

```
let myName = 'Nikos';
let myAge =  75;
const pi =  3.14;
```

Ποιος ο λόγος για εισαγωγή των δύο αυτών τρόπων ορισμού μεταβλητών, υπάρχουν διαφορές μεταξύ τους;

Η απάντηση είναι ότι υπάρχουν σημαντικές διαφορές και η πρόταση εξαρχής που έχουμε να κάνουμε είναι να ξεχάσουμε τη var και να χρησιμοποιούμε την const στις περισσότερες περιπτώσεις και τη let στις υπόλοιπες, που είναι μόνο για τις περιπτώσεις μεταβλητών που αναφέρονται σε **πρωτογενείς τύπους δεδομένων** (Number, String, Boolean κ.λπ.) των οποίων η τιμή πρόκειται να αλλάξει.

Ας εξετάσουμε στη συνέχεια τις κύριες διαφορές μεταξύ let και var. Αυτές εντοπίζονται στα εξής:

- Διαφορές στη δυνατότητα επαναδήλωσης μιας μεταβλητής (στη var επιτρέπεται, στη let όχι).
- Διαφορές στην εμβέλεια (η var έχει εμβέλεια και έξω από το μπλοκ στο οποίο δηλώνεται, ενώ η let όχι).
- Διαφορές στη δυνατότητα πρόσβασης στη μεταβλητή πριν τη δήλωσή της (στη var επιτρέπεται, η τιμή είναι αρχικά undefined, στη let αυτό δεν επιτρέπεται).

Ας δούμε μερικά σχετικά παραδείγματα:

7.3.2 Πολλαπλές δηλώσεις

Ο παρακάτω κώδικας είναι αποδεκτός:

```
var myName = 'Nikos';
var myName = 'Kostas';
```

Αντίθετα, ο παρακάτω κώδικας δίνει σφάλμα:

```
let myName = 'Nikos';
let myName = 'Kostas';
> SyntaxError: Identifier 'myname' has already been declared
```

Βεβαίως, μπορούμε να αλλάξουμε την τιμή μιας μεταβλητής που έχει δηλωθεί με let, χωρίς όμως να την ξαναδηλώσουμε:

```
let myName = 'Nikos';
myName = 'Kostas';
```

Αντίθετα, δεν μπορούμε να αλλάξουμε μια μεταβλητή που έχει δηλωθεί με τη λέξη-κλειδί const αν η τιμή της

είναι πρωτογενούς τύπου.

```
const myName = 'Nikos';
myName = 'Kostas';
> TypeError: Assignment to constant variable.
```

Θα πρέπει να προσέξουμε όμως ότι η δήλωση με τη λέξη-κλειδί const μεταβλητών που αναφέρονται σε **τύπους δεδομένων αναφοράς**, όπως τα αντικείμενα (object) ή οι πίνακες, δεν μας αποκλείει από τη δυνατότητα να τροποποιήσουμε στη συνέχεια το περιεχόμενο αυτών των αντικειμένων.

Για παράδειγμα, ο παρακάτω κώδικας είναι απόλυτα αποδεκτός:

```
const ourNames = ['Nikos', 'Kostas', 'Maria'];
ourNames[0] = 'Katerina';
console.log(ourNames);
> ["Katerina", "Kostas", "Maria"]
```

7.3.3 Εμβέλεια μεταβλητών let

Μια σημαντική διαφορά μεταξύ let και var είναι ότι η let ορίζει εμβέλεια της μεταβλητής μόνο στο μπλοκ στο οποίο έχει δηλωθεί (μπλοκ μπορεί να θεωρηθεί μια συνάρτηση αλλά και μια εντολή if, for κ.λπ.), ενώ η var έχει εμβέλεια σε ολόκληρο τον κώδικα.

Ας συγκρίνουμε το αποτέλεσμα των δύο αυτών παραδειγμάτων:

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

for(let i = 0; i<10; i++) {
}
console.log(i)
> ReferenceError: i is not defined
```

Στο δεύτερο παράδειγμα που η μεταβλητή i ορίστηκε με χρήση της λέξης κλειδί let, αυτή δεν είχε εμβέλεια εκτός του μπλοκ for μέσα στο οποίο ορίστηκε, και για αυτό πήραμε μήνυμα σφάλματος.

7.3.4 Κλήση μεταβλητής πριν τη δήλωσή της

Η let δεν μπορεί να χρησιμοποιηθεί πριν δηλωθεί, ενώ η var μπορεί να χρησιμοποιηθεί (*hoisting*) και έχει την τιμή undefined.

```
console.log('x=', typeof x, x); // x undefined NaN
console.log('y=', typeof y, y); // ReferenceError
var x=5;
let y=10;
```

7.3.5 Τελεστές και εκφράσεις

Έχουμε ήδη δώσει παραδείγματα εντολών εκχώρησης τιμής σε μεταβλητή. Αυτή γίνεται με χρήση του τελεστή =:

```
μεταβλητή = έκφραση;
```

Σε μια έκφραση μπορούν να χρησιμοποιηθούν οι δυαδικοί τελεστές αριθμητικών πράξεων που συναντώνται και σε άλλες γλώσσες προγραμματισμού: “+” για πρόσθεση, “-” για αφαίρεση, “*” για πολλαπλασιασμό, “/” για διαίρεση, “%” για υπόλοιπο διαίρεσης, ενώ από την έκδοση ES2016 προστέθηκε ο τελεστής “**” για ύψωση σε δύναμη. Επίσης, σε μια έκφραση μπορεί να χρησιμοποιηθούν οι μοναδιαίοι τελεστές:

- **a++** για απόδοση τιμής και αύξηση κατά 1
- **a--** για απόδοση τιμής και μείωση κατά 1

- **++a** για αύξηση τιμής κατά 1 και απόδοση τιμής στη συνέχεια
- **--a** για μείωση τιμής κατά 1 και απόδοση τιμής στη συνέχεια

καθώς και οι συντομογραφίες εκφράσεων:

- **a += b** Αύξηση της τιμής του a κατά b, δηλαδή a = a + b
- **a -= b** Μείωση της τιμής του a κατά b, δηλαδή a = a - b;

7.3.6 Τελεστές πράξεων δυαδικών αριθμών

Αν έχουμε δύο μεταβλητές a, b που εκφράζουν δυαδικούς αριθμούς, υπάρχουν οι εξής τελεστές που επιτρέπουν πράξεις μεταξύ τους:

- a & b Bitwise AND ανάμεσα στις μεταβλητές
- a | b Bitwise OR ανάμεσα στις μεταβλητές
- a ^ b Bitwise XOR ανάμεσα στις μεταβλητές
- ~a Bitwise NOT της μεταβλητής
- a << b Αριστερή ολίσθηση κατά b θέσεις των bit της μεταβλητής a
- a >> b Δεξιά ολίσθηση κατά b θέσεις των bit της μεταβλητής a

Θυμίζουμε ότι η αριστερή ολίσθηση στους δυαδικούς αριθμούς είναι ισοδύναμη με πολλαπλασιασμό με τη βάση 2, ενώ η δεξιά ολίσθηση είναι ισοδύναμη με διαίρεση με 2.

Στο επόμενο κεφάλαιο θα δούμε άλλους τελεστές που μπορούμε να χρησιμοποιήσουμε σε μια έκφραση, όπως λογικούς τελεστές, αλλά και τον τριαδικό τελεστή, καθώς και τελεστές σύγκρισης που χρησιμοποιούνται σε εκφράσεις που συνηθίζονται στις εντολές επιλογής.

Άσκηση
Ποιο το αποτέλεσμα;

```
let c = 5;
console.log(c++);
console.log(++c);
```

Απάντηση:

```
> 5
> 7
```

Αυτό γιατί η πρώτη εντολή εκτύπωσης στην κονσόλα τυπώνει την τιμή της μεταβλητής c πριν την αύξησή της, ενώ η δεύτερη μετά τη νέα αύξησή της.

7.3.7 Διεπαφή με τον χρήστη

Όπως ήδη αναφέρθηκε, η JavaScript δεν έχει εγγενώς διεπαφή για επικοινωνία με τον χρήστη, όπως άλλες γλώσσες προγραμματισμού. Εξάλλου, είναι προορισμένη να συνεργάζεται στενά με την HTML/CSS για τον σκοπό αυτό. Όμως, συχνά χρειαζόμαστε είτε στο περιβάλλον του εξυπηρετητή είτε στο περιβάλλον του φυλλομετρητή το πρόγραμμά μας να τυπώσει την κατάστασή του, ή να ζητήσει από τον χρήστη κάποια δεδομένα. Στη συνέχεια κάνουμε μια σύντομη αναφορά σε τρόπους που μπορεί να χρησιμοποιήσουμε για τον σκοπό αυτό.

console.log(μήνυμα)

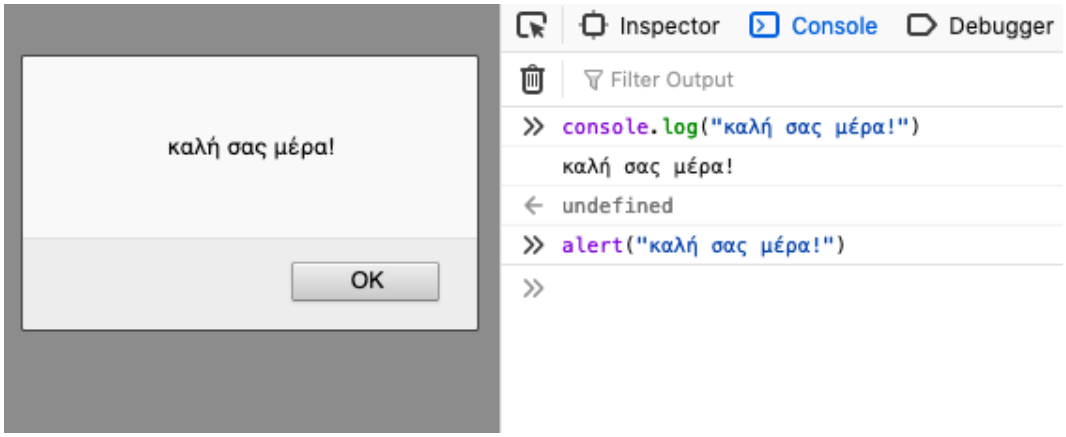
Ο πιο συνηθισμένος τρόπος είναι να χρησιμοποιήσουμε το αντικείμενο console που ανήκει στο global object και να καλέσουμε τη μέθοδο console.log(). Το αντικείμενο αυτό έχει οριστεί, όπως θα δούμε στη συνέχεια, και στο καθολικό επίπεδο της Node.js με παρόμοια συμπεριφορά με αυτή του φυλλομετρητή.

Στα ορίσματα της συνάρτησης αυτής μπορούμε να περάσουμε εκφράσεις ή συμβολοσειρές και να πάρουμε τα αποτελέσματα στην κονσόλα. Η κονσόλα βρίσκεται στα εργαλεία ανάπτυξης όλων των φυλλομετρητών (Chrome, Firefox, Safari κ.λπ.).

```
console.log("Καλημέρα");
> "Καλημέρα"
```

alert(μήνυμα)

Ένας εναλλακτικός τρόπος, μόνο για το περιβάλλον του φυλλομετρητή όμως, είναι να χρησιμοποιήσουμε τη μέθοδο alert() του window object.

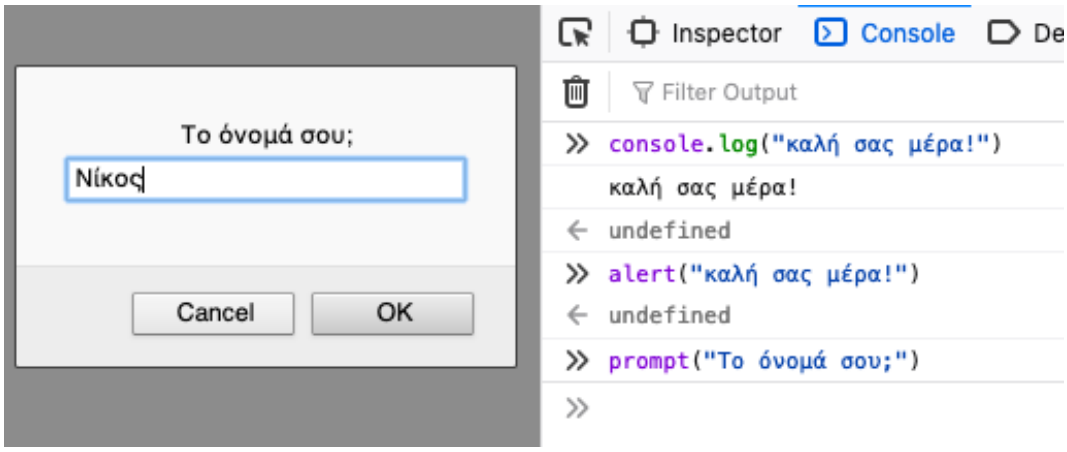


Εικόνα 7.4 *Η 'alert()' όπως εκτελείται στον φυλλομετρητή Firefox.*

Αυτή παράγει ένα αναδυόμενο μονοτροπικό παράθυρο (modal) με το μήνυμα, όπως βλέπουμε στην **Εικόνα *7.4***, από την κονσόλα του Firefox.

prompt()

Παρόμοια είναι η χρήση της μεθόδου prompt() του global object, η οποία μέσω ενός αναδυόμενου μονοτροπικού παραθύρου ζητάει μια τιμή από τον χρήστη. Παράδειγμα στην **Εικόνα *7.5***.



Εικόνα 7.5 *Η prompt() όπως εκτελείται στον φυλλομετρητή Firefox.*

Επικοινωνία μέσω της HTML

Εκτός από τους παραπάνω, υπάρχουν πολλοί τρόποι να χρησιμοποιηθεί η ίδια η ιστοσελίδα για επικοινωνία με τον χρήστη, όπως για παράδειγμα με τροποποίηση ενός στοιχείου με την ιδιότητα element.textContent ή element.innerHTML, αλλά και να πάρουμε στοιχεία από τον χρήστη μέσω στοιχείων <input> μιας φόρμας, όπως έχουμε δει στο κεφάλαιο της HTML.

Αυτές είναι οι βασικές τεχνικές που διαθέτει η JavaScript για επικοινωνία με τον χρήστη. Στη συνέχεια του κεφαλαίου θα επιχειρήσουμε μια επισκόπηση της αρχιτεκτονικής του φυλλομετρητή και του τρόπου που εκτελείται ο κώδικας JavaScript στο πλαίσιό του, ενώ θα εξετάσουμε τη διεπαφή της JavaScript με το περιβάλλον της ιστοσελίδας και του φυλλομετρητή.

7.4 Η JavaScript στον φυλλομετρητή

Ο κώδικας JavaScript μιας ιστοσελίδας σχετίζεται με το αρχείο .html. Μια ιστοσελίδα μπορεί να συνοδεύεται από πολλά τμήματα κώδικα JavaScript που μπορεί να βρίσκονται είτε ενσωματωμένα στο ίδιο το αρχείο ή σε εξωτερικά αρχεία που φορτώνονται από το αρχείο .html.

Όλα αυτά τα τμήματα του κώδικα συνυπάρχουν στον ίδιο κοινό χώρο, συναποτελώντας το *«πρόγραμμα JavaScript»* της σελίδας. Μάλιστα, όλα αυτά τα τμήματα κώδικα μοιράζονται τον κοινό χώρο ονομάτων μεταβλητών που ονομάζεται **global object**.

Ο κώδικας JavaScript που προέρχεται από το ίδιο το αρχείο .html εμφανίζεται ως περιεχόμενο στοιχείων <script>.

```
<script>
    // κώδικας JavaScript
</script>
```

Το στοιχείο αυτό μπορεί να βρίσκεται είτε στο τμήμα <head> του εγγράφου HTML ή οπουδήποτε στο τμήμα <body>.

Ένας δεύτερος τρόπος να φορτώσουμε στη σελίδα κώδικα JavaScript είναι από εξωτερικό αρχείο, στο οποίο γίνεται αναφορά μέσω του γνωρίσματος src του στοιχείου <script>.

Για παράδειγμα, ένα εξωτερικό αρχείο “scriptfile.js” φορτώνεται ως εξής:

```
<script src="scriptfile.js"></script>
```

Σε αυτή την περίπτωση το στοιχείο μπορεί να έχει μια από τις ιδιότητες async ή defer. Οι ιδιότητες αυτές δεν παίρνουν τιμή, είναι λογικές ιδιότητες και καθορίζουν τη σειρά φορτώματος και εκτέλεσης του αντίστοιχου κώδικα JavaScript.

- Η ιδιότητα **async** ορίζει ασύγχρονο φόρτωμα και εκτέλεση του κώδικα JavaScript, που μπορεί να γίνει ακόμη και πριν την ολοκλήρωση φορτώματος της HTML.
- Η ιδιότητα **defer** ορίζει ότι το φόρτωμα του κώδικα θα γίνει ασύγχρονα αλλά η εκτέλεση του κώδικα JavaScript θα γίνει μετά τη φόρτωση HTML ή άλλου κώδικα JS που επίσης έχει την ίδια ιδιότητα αλλά προηγείται.

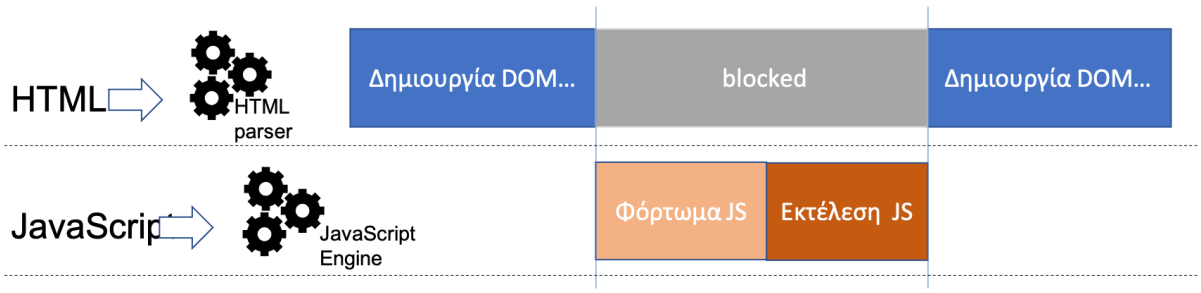
Για παράδειγμα, αν στη σελίδα μας περιέχεται ο παρακάτω κώδικας:

```
<script src="js/script2.js" defer ></script>
<script src="js/script3.js" defer ></script>
```

το script3.js θα εκτελεστεί μετά το script2.js.

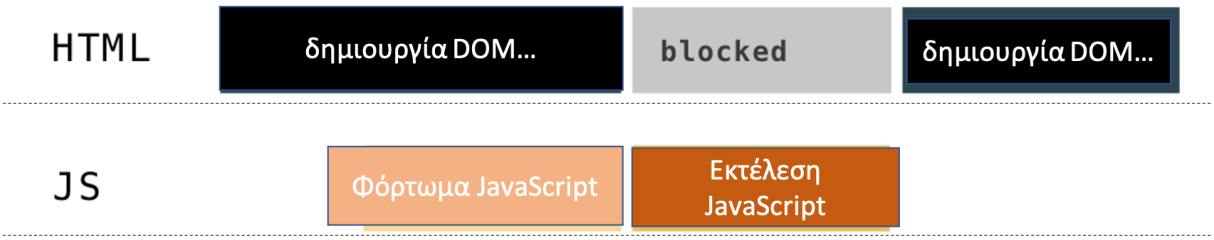
Αν δεν χρησιμοποιηθεί καμιά από αυτές τις ιδιότητες, ο κώδικας JavaScript μπλοκάρει το φόρτωμα της HTML και άρα το χτίσιμο του DOM. Αυτό γίνεται γιατί μέσα στον κώδικα JS μπορεί να κρύβονται εντολές που συμβάλλουν στο χτίσιμο της HTML, π.χ. με την εντολή document.write() που επιτρέπει εισαγωγή κώδικα HTML στο σημείο της συγκεκριμένης εντολής. Η χρήση αυτής της δυνατότητας δεν προτείνεται, όμως οι φυλλομετρητές πρέπει να φροντίσουν και για αυτό το ενδεχόμενο.

Η παρακάτω ***Εικόνα 7.6*** περιγράφει αυτό το σενάριο.



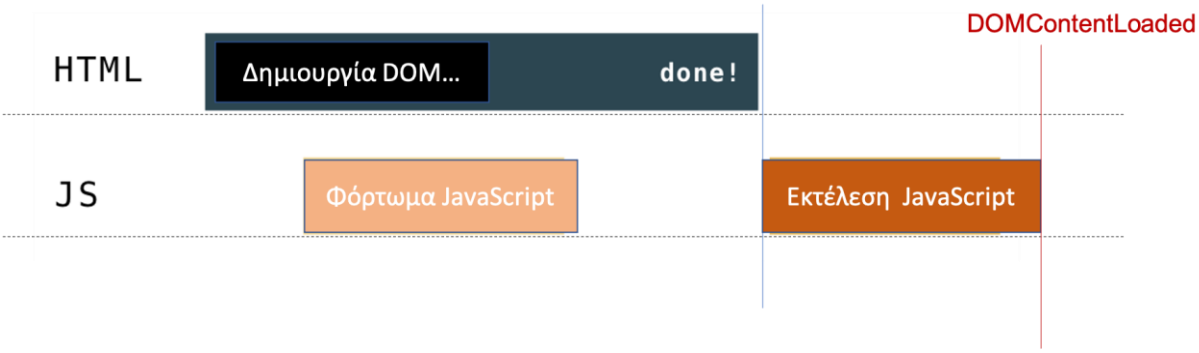
Εικόνα 7.6 *Φόρτωμα κώδικα JavaScript με μπλοκάρισμα φορτώματος της HTML/DOM.*

Η περίπτωση φορτώματος κώδικα JavaScript με την ιδιότητα async φαίνεται στην **Εικόνα *7.7***.



Εικόνα 7.7 *Φόρτωμα κώδικα JavaScript με την ιδιότητα async, ασύγχρονα με την HTML.*

Τέλος, η περίπτωση φορτώματος κώδικα JavaScript με την ιδιότητα defer φαίνεται στην ***Εικόνα 7.8***.



Εικόνα 7.8 *Φόρτωμα κώδικα JavaScript με την ιδιότητα defer: η εκτέλεση του κώδικα μεταφέρεται μετά την ολοκλήρωση του DOM.*

Ποια είναι η καλύτερη πολιτική για φόρτωμα κώδικα JS; Γενικά, τοποθετούμε το <script> μέσα στο <head> και χρησιμοποιούμε τα γνωρίσματα async και defer ώστε η JS να μην μπλοκάρει το κατέβασμα της σελίδας.

Χρησιμοποιούμε <script async ...> αν το περιεχόμενο της σελίδας είναι ανεξάρτητο από τον κώδικα JavaScript.

Χρησιμοποιούμε <script defer ...> αν επιθυμούμε ο κώδικας να τρέξει μετά την ολοκλήρωση φορτώματος της HTML, αν για παράδειγμα ο κώδικάς μας κάνει αναφορά σε στοιχεία του DOM.

Επίσης, μπορούμε να θέσουμε το <script> στο τέλος του <body> ώστε το φόρτωμα και η εκτέλεση του κώδικα να μην καθυστερεί το φόρτωμα της σελίδας, ιδιαίτερα αν ο κώδικας JavaScript είναι εκτενής.

Εναλλακτικά, μπορούμε να βάλουμε τμήμα του κώδικα μέσα σε μια συνάρτηση η οποία να κληθεί μετά από ένα συμβάν, π.χ. το συμβάν DOMContentLoaded που σηματοδοτεί την ολοκλήρωση φόρτωσης του DOM.

Στη συνέχεια θα κάνουμε ιδιαίτερη αναφορά στα συμβάντα που σχετίζονται με το φόρτωμα μιας σελίδας.

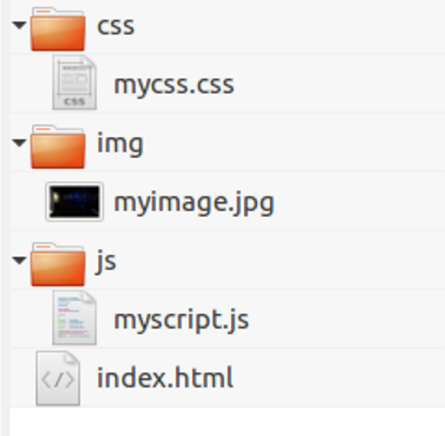
7.5 Διαχωρισμός κώδικα HMTL, CSS, JavaScript

Σε μια σύγχρονη διαδικτυακή εφαρμογή μια ιστοσελίδα περιλαμβάνει εκτός από το κυρίως αρχείο HTML και ένα σύνολο από αρχεία (π.χ. index.html). Αυτά τυπικά περιλαμβάνουν αρχείο ή αρχεία CSS, αρχεία JavaScript, καθώς και αρχεία εικόνων και άλλων πολυμέσων. Ο τρόπος που συνδέονται αυτά τα αρχεία, όπως έχει ήδη αναφερθεί, είναι αυτός που φαίνεται στο παράδειγμα.

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="css/mycss.css">
  </head>
  <body>
    <h1 id="first">Καλή σας μέρα</h1>
    <img src="img/myimage.jpg"></img>
    <script src="js/myscript.js"></script>
```

```
  </body>
</html>
```

Η οργάνωση των αρχείων του πρότζεκτ ανάπτυξης της ιστοσελίδας συνήθως έχει την παρακάτω δομή (**Εικόνα 7.9**):



Εικόνα 7.9 *Συνήθης οργάνωση των αρχείων ενός πρότζεκτ ανάπτυξης ιστοσελίδας.*

7.6 Η ακολουθία συμβάντων κατά το φόρτωμα μιας σελίδας

Η εκτέλεση της JavaScript στον φυλλομετρητή ακολουθεί τη λογική των γλωσσών προγραμματισμού που στηρίζονται στο **μοντέλο χειρισμού συμβάντων (event-based model)**. Τέτοιες είναι οι γλώσσες που λειτουργούν σε διαδραστικές συνθήκες, όπως σε γραφικά περιβάλλοντα αλληλεπίδρασης με τον χρήστη. Το μοντέλο αυτό περιλαμβάνει φόρτωμα του κώδικα και αρχικοποίηση των μεταβλητών και αντικειμένων, ορισμό των χειριστών συμβάντων, δηλαδή των τμημάτων του κώδικα (συναρτήσεων) που θα εκτελεστούν όταν συμβούν καθορισμένα συμβάντα (events). Στη συνέχεια το πρόγραμμα βρίσκεται σε αναμονή των συμβάντων αυτών, τα οποία προκύπτουν συνήθως από ενέργειες του χρήστη, αλλά και συμβάντων του συστήματος, όπως ανάκτηση δεδομένων από άλλες πηγές, ολοκλήρωση φόρτωσης τμημάτων του κώδικα κ.λπ. Όταν συμβεί ένα αναμενόμενο συμβάν, ενεργοποιείται ο αντίστοιχος χειριστής.

Ας ακολουθήσουμε στη συνέχεια την ακολουθία φάσεων και αντίστοιχων συμβάντων κατά το φόρτωμα της ιστοσελίδας.

Φάση 1: Φόρτωμα DOM. Ο φυλλομετρητής δημιουργεί ένα αντικείμενο **document** και αρχίζει τη συντακτική ανάλυση της ιστοσελίδας με προσθήκη κόμβων στο DOM καθώς αναλύει ένα προς ένα τα στοιχεία HTML και το περιεχόμενό τους. Η ιδιότητα document.readyState έχει την τιμή *loading* σε αυτή τη φάση. Αν κατά τη διάρκεια της συντακτικής ανάλυσης βρεθεί ετικέτα <script> χωρίς async, defer, φορτώνεται και εκτελείται ο κώδικας JavaScript, ενώ διακόπτεται το φόρτωμα της HTML. Αν κατά τη φάση αυτή ο αναλυτής συναντήσει ετικέτα <script async ...> με ασύγχρονο τρόπο, προχωράει στο φόρτωμα και την εκτέλεση του κώδικα JavaScript.

Φάση 2: Η ανάλυση της HTML ολοκληρώνεται, η ιδιότητα document.readyState παίρνει την τιμή *interactive*. Σε αυτή τη φάση τα τμήματα κώδικα JavaScript που ήταν σε κατάσταση defer εκτελούνται διαδοχικά το ένα μετά το άλλο. Μετά την ολοκλήρωση εκτέλεσης των τμημάτων αυτών του κώδικα, ενεργοποιείται το συμβάν DOMContentLoaded, που σηματοδοτεί τη μετάβαση της JavaScript από κατάσταση φορτώματος και εκτέλεσης κώδικα στη φάση της διαδραστικής εκτέλεσης, σε αναμονή συμβάντων.

Φάση 3: Η JavaScript έχει μπει σε κατάσταση αναμονής συμβάντων. Η σελίδα έχει φορτωθεί πλήρως, αν και τμήματα των πρόσθετων (εικόνες κ.λπ.) ίσως ακόμη φορτώνονται. Επίσης, πιθανόν τμήματα κώδικα JavaScript σε κατάσταση async να είναι ακόμη σε κατάσταση φορτώματος/εκτέλεσης. Η φάση αυτή ολοκληρώνεται όταν όλα τα πρόσθετα φορτωθούν και ολοκληρωθεί η εκτέλεση του κώδικα async. Τότε εμφανίζεται το συμβάν window.load, ενώ η ιδιότητα document.readyState = "complete". Με την ολοκλήρωση αυτής της φάσης ξεκινάει η ασύγχρονη λειτουργία της JavaScript.

7.6.1 Ένα παράδειγμα

Στο παράδειγμα αυτό θα φορτώσουμε μια ιστοσελίδα η οποία μας γνωστοποιεί τη χρονική διάρκεια κάθε φάσης φορτώματος όπως προκύπτει από τα σχετικά συμβάντα. Η σελίδα φορτώνει μια σειρά από φωτογραφίες διαφορετικής ανάλυσης από την ιστοσελίδα διάθεσης φωτογραφιών https://picsum.photos. Ο κώδικας καταγράφει τον χρόνο που απαιτείται για τα εξής συμβάντα: (α) Αλλαγή της ιδιότητας readyState σε interactive (ολοκλήρωση φορτώματος του DOM), (β) ενεργοποίηση του συμβάντος DOMContentLoaded που σηματοδοτεί την ολοκλήρωση εκτέλεσης του κώδικα JavaScript, (γ) ενεργοποίηση του συμβάντος window.load που σηματοδοτεί την ολοκλήρωση φορτώματος των πρόσθετων (εικόνων).

Η ιστοσελίδα φαίνεται στη συνέχεια. Θα πρέπει να σημειωθεί ότι το παράδειγμα περιέχει κώδικα JavaScript με συναρτήσεις και δομές που δεν έχουμε αναφέρει ακόμη, όμως παρουσιάζεται κυρίως για το αποτέλεσμα που παράγει, το οποίο φαίνεται στην εικόνα που ακολουθεί.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ex0</title>
    <script>
        const round = (v) => (v/1000).toFixed(2);
        report = "";
        function checkReady(e){
            report += `readyState=${document.readyState} --
${round(e.timeStamp)}sec <br>`;
        }
        document.addEventListener('readystatechange', checkReady);
        document.addEventListener('DOMContentLoaded', (e) => {
            report += `DOMContentLoaded -- ${round(e.timeStamp)}sec
:φόρτωμα DOM<br>`;
        });
        window.addEventListener('load', (e) => {
            report += `window.load -- ${round(e.timeStamp)}sec :φόρτωμα
σελίδας`;
            document.querySelector("#report").innerHTML = report;
            console.log(report);
        });
    </script>
</head>
<body>
    <div style="display: flex"></div>
    <div id="report" style="margin:10px;font-family:sans-serif;
    font-size:2rem;"></div>
    <script>
        const container = document.querySelector("div");
        for (let i=0;i<5;i++){
            const d = document.createElement("div");
            d.style.padding = "5px";
            const im = document.createElement("img");
            im.src = `https://picsum.photos/${i+2}00`
            d.appendChild(im);
            container.append(d);
        }
    </script>

</body>
</html>
```

Το αποτέλεσμα από μια τυπική φόρτωση της σελίδας φαίνεται στη συνέχεια (**Εικόνα *7.10***):

Εικόνα 7.10 *Τυπική φόρτωση της σελίδας.*

Προκύπτει ότι το φόρτωμα της σελίδας που περιλαμβάνει τις 5 εικόνες που φορτώνονται από το https://picsum.photos απαίτησε στη συγκεκριμένη περίπτωση συνολικά 0.42 sec.

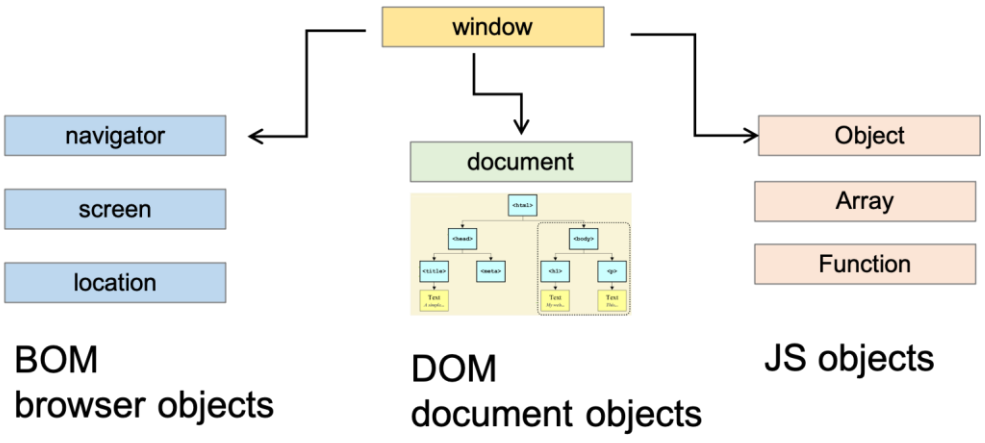
Συγκεκριμένα, το συμβάν ολοκλήρωσης της συντακτικής ανάλυσης της HTML ολοκληρώθηκε σε 0,18 δευτερόλεπτα, ενώ αμέσως ολοκληρώθηκε και το φόρτωμα του JavaScript κώδικα με κατάσταση defer (συμβάν DOMContentLoaded), αφού δεν υπήρχαν τέτοια τμήματα κώδικα JS. Τέλος, η ολοκλήρωση φορτώματος των εικόνων που προκαλεί την αλλαγή της document.readyState = "complete" καθώς και του συμβάντος window.load, απαιτεί, όπως προκύπτει, χρόνο μέχρι 0,42 δευτερόλεπτα.

Μελετήστε τον κώδικα του παραδείγματος και ιδιαίτερα την ιδιότητα του συμβάντος που επιτρέπει την καταγραφή χρόνων, που χρησιμοποιούν οι *χειριστές συμβάντων* για να μας παρουσιάσουν τη σχετική πληροφορία.

7.7 Το περιβάλλον εκτέλεσης της JavaScript

Στο περιβάλλον του φυλλομετρητή το πρόγραμμα JavaScript που φορτώθηκε με τη διαδικασία, και περιγράφηκε στην προηγούμενη παράγραφο, έχει πρόσβαση στον κοινό χώρο ονομάτων (global scope) που λέγεται *global object*. Το αντικείμενο αυτό είναι το αντικείμενο window, στο οποίο μάλιστα μπορούμε να αναφερθούμε με χρήση της λέξης-κλειδί this. Ιδιότητες του αντικειμένου αυτού είναι τα βασικά στοιχεία της γλώσσας (built-in objects), τα οποία μπορεί να χρησιμοποιηθούν είτε με σημειογραφία τελείας window.αντικείμενο είτε παραλείποντας την αναφορά στο αντικείμενο window.

Οι κατηγορίες αντικειμένων που έχει πρόσβαση η JavaScript μέσω του global object φαίνονται στο διάγραμμα στην **Εικόνα *7.11***.



Εικόνα 7.11 *Κατηγορίες αντικειμένων που έχει πρόσβαση η JavaScript.*

- **Αντικείμενα της γλώσσας JavaScript**. Μια πρώτη κατηγορία είναι τα αντικείμενα της γλώσσας, είτε τα συστατικά της στοιχεία είτε τα αντικείμενα που δημιουργεί ο κώδικας του χρήστη.
- **DOM**. Μια δεύτερη κατηγορία είναι τα αντικείμενα του *Document Object Model* που ανήκουν στο αντικείμενο document. Σε επόμενη ενότητα θα κάνουμε ιδιαίτερη αναφορά στη **διεπαφή DOM**, δηλαδή το σύνολο των μεθόδων που διαθέτει η JavaScript για αλληλεπίδραση με τα στοιχεία του DOM.
- **Browser objects**. Συχνά τα στοιχεία αυτά αναφέρονται ως BOM (browser object model), περιλαμβάνουν, μεταξύ άλλων, το αντικείμενο navigator, που περιέχει στοιχεία για το περιβάλλον του φυλλομετρητή και του λειτουργικού συστήματος του χρήστη, το αντικείμενο screen που αφορά την οθόνη του υπολογιστή του χρήστη, το αντικείμενο location που αφορά το URL της σελίδας κ.λπ.

7.7.1 Δραστηριότητα

Μεταβείτε στην κονσόλα του φυλλομετρητή σας και πληκτρολογήσετε window ή this. Επιθεωρήστε τις ιδιότητες του αντικειμένου αυτού. Ιδιαίτερα, εξετάστε τα αντικείμενα του φυλλομετρητή που αναφέρθηκαν, καθώς και το περιεχόμενο του DOM.

7.8 Βασικά στοιχεία του αντικείμενου window

Αν επιθεωρήσουμε τις ιδιότητες του αντικείμενου window, θα βρούμε, μεταξύ άλλων, αντικείμενα με ιδιότητες και μεθόδους που χρησιμοποιούνται ευρύτατα στην JavaScript.

Θα κάνουμε σύντομη αναφορά σε αυτά, ενώ σε επόμενες ενότητες θα γίνει ιδιαίτερη μνεία στις κυριότερες από αυτές.

7.8.1 Συναρτήσεις

- **Eval (έκφραση)**: Υπολογισμός του αποτελέσματος της έκφρασης. Θεωρείται κακή πρακτική η χρήση της για λόγους ασφάλειας.
- **isFinite (εκφραση)**: Επιστρέφει false αν το αποτέλεσμα της έκφρασης είναι +Infinity, -Infinity, NaN ή undefined, αλλιώς true
- **isNaN (τιμή)**: Επιστρέφει true αν η *τιμή* επιστρέφει NaN, αλλιώς false.
- **parseFloat (συμβολοσειρά)**: Επιστρέφει την αριθμητική τιμή ως δεκαδικό αριθμό. Αν όμως η συμβολοσειρά περιέχει μη αριθμητικούς χαρακτήρες από έναν χαρακτήρα και μετά, λαμβάνει υπόψη του το ως τότε τμήμα της συμβολοσειράς. Αν όμως η συμβολοσειρά αρχίζει με μη αριθμητικό χαρακτήρα, επιστρέφει NaN
- **parseInt (συμβολοσειρά, βάση)**: Παρόμοια με την parseFloat() για ακέραιο αριθμό, παίρνει ως δεύτερο προαιρετικό όρισμα τη βάση του αριθμητικού συστήματος (π.χ. 16 για το δεκαεξαδικό κ.λπ.)
- **encodeURI(URI)**: Κωδικοποίηση δεδομένων με την τεχνική URL Encoding, βάσει της κωδικοσελίδας UTF-8, λαμβάνοντας υπόψη ειδικούς χαρακτήρες, όπως ;,/?:@&=+$#, για παράδειγμα, κωδικοποιεί τα κενά ως %20.
- **encodeURIComponent()**: Όπως η προηγούμενη, μόνο που αφορά δεδομένα που θα τοποθετηθούν σε μεταβλητές PUT GET.
- **decodeURI()**: Αποκωδικοποίηση της encodeURI()
- **decodeURIComponent()**: Αποκωδικοποίηση της encodeURIComponent()

7.8.2 Βασικά αντικείμενα

- **Object**: Το αρχέτυπο αντικείμενο. Τα περισσότερα αντικείμενα της JavaScript κληρονομούν από αυτό.
- **Function**: Η κλάση των συναρτήσεων της JavaScript που είναι αντικείμενα πρώτης τάξης.
- **Boolean**: Αντικείμενο που αντιστοιχεί σε λογικές τιμές.
- **Symbol**: Αντικείμενο που αντιστοιχεί στα δεδομένα τύπου Symbol.

7.8.3 Αντικείμενα αριθμών

- **Number**: Αντικείμενο που αντιστοιχεί στον πρωτογενή τύπο δεδομένων Number.
- **BigInt**: Αντικείμενο που αντιστοιχεί στον πρωτογενή τύπο δεδομένων BigInt για αριθμούς

μεγαλύτερους από $2^{53} - 1$.

- **Math**: Αντικείμενο με ιδιότητες και μεθόδους για μαθηματικές συναρτήσεις και σταθερές.
- **Date**: Αντικείμενο που εκφράζει χρονική στιγμή σε χιλιοστά του δευτερολέπτου με αφετηρία μέτρησης 1 Ιανουαρίου 1970 UTC (τύπου Number).

7.8.4 Κείμενο

- **String**: Αντικείμενο που αφορά την αναπαράσταση και μεθόδους που αφορούν συμβολοσειρές.
- **RegExp**: Αντικείμενο που επιτρέπει την αντιστοίχιση προτύπων χαρακτήρων (κανονικές εκφράσεις) με συμβολοσειρές.

7.8.5 Συλλογές αντικειμένων

- **Array**: Αντικείμενο που αφορά την κλάση αντικειμένων τύπου Array (πίνακες). Να σημειωθεί ότι υπάρχουν ακόμη αντικείμενα για "typed" Arrays, πίνακες που δέχονται ορισμένου τύπου δεδομένα, όπως Float32Array κ.λπ.
- **Map**: Αντικείμενο που αφορά την κλάση Map, ακολουθία ζευγών «κλειδί: τιμή» που θυμάται τη σειρά δημιουργίας της.
- **Set**: Αντικείμενο που αφορά την κλάση Set, που επιτρέπει την αποθήκευση συλλογής μοναδικών στοιχείων.

7.8.6 Σφάλματα

- **Error** : αντικείμενα τα οποία δημιουργούνται όταν συμβούν σφάλματα κατά τη διάρκεια εκτέλεσης του κώδικα. Υπάρχουν ειδικοί τύποι σφαλμάτων, π.χ. ReferenceError κ.λπ.

Υπάρχουν πολλά ακόμη αντικείμενα ως ιδιότητες του αντικείμενου window.

Ακόμη, το αντικείμενο window, πέραν του να ορίζει τον καθολικό χώρο ονομάτων και να έχει ως ιδιότητες τα βασικά στοιχεία της γλώσσας (τα λεγόμενα "built-ins"), έχει έναν ακόμη ρόλο να παίξει: περιέχει πληροφορίες για το παράθυρο του φυλλομετρητή στο οποίο εμφανίζεται η ιστοσελίδα, όπως οι ιδιότητες innerHeight, innerWidth που περιέχουν τις διαστάσεις του παράθυρου.

Για πλήρη περιγραφή των αντικειμένων και ιδιοτήτων αυτών μπορείτε να συμβουλευτείτε τη MDN σχετικά με τα global objects.

7.9 Διεπαφή με το Document Object Model (DOM)

Σε αυτή την ενότητα θα εξετάσουμε τη διεπαφή της JavaScript με τα στοιχεία της ιστοσελίδας μέσω της ιδιότητας document του global object, η οποία έχει ως τιμή το αντικείμενο Document.

Το αντικείμενο αυτό αναπαριστά τα στοιχεία του εγγράφου HTML, που εμφανίζεται στο παράθυρο ή στην καρτέλα του φυλλομετρητή. Έχει οριστεί μια προγραμματιστική διεπαφή με τα στοιχεία αυτά, που ονομάζεται Document Object Model (DOM). Έχει γίνει ήδη αναφορά στην αναπαράσταση των στοιχείων ενός εγγράφου HTML ως ιεραρχία κόμβων με ρίζα το στοιχείο <html>. Το DOM API ορίζει στο περιβάλλον της JavaScript ένα σύνολο από αντικείμενα, που αντανακλούν την ιεραρχία στοιχείων της HTML. Για κάθε στοιχείο HTML υπάρχει ένα αντίστοιχο αντικείμενο (Element) JavaScript και για κάθε κείμενο του αρχείου HTML υπάρχει ένα αντικείμενο κειμένου (Text). Τα αντικείμενα αυτά είναι εξειδικεύσεις του αντικείμενου Node.

Συνεπώς, στην ιεραρχία του DOM εμπεριέχονται σαν αντικείμενα όλα τα στοιχεία της ιστοσελίδας. Να σημειωθεί ότι *αντικείμενο (object)* είναι μια βασική δομή δεδομένων της JavaScript που θα δούμε στη συνέχεια (Ενότητα 9.5). Στα αντικείμενα αυτά έχει πρόσβαση ένα πρόγραμμα JavaScript το οποίο μπορεί να τα τροποποιήσει. Συνεπώς, το document object model είναι η προγραμματιστική διεπαφή του εγγράφου HTML και είναι ο μηχανισμός που παρέχει στην JavaScript δυνατότητα πρόσβασης στο περιεχόμενο της ιστοσελίδας.

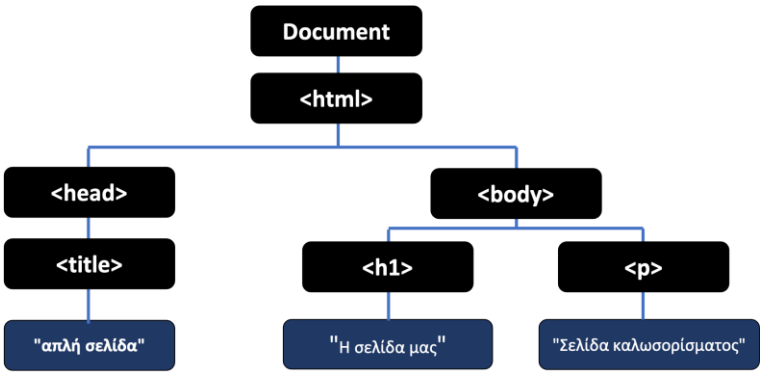
Ας πάρουμε καταρχάς ένα παράδειγμα απλού εγγράφου HTML.

```
<!DOCTYPE html>
<html lang="el">
    <head>
```

```
        <meta charset="utf-8">
        <title>απλή σελίδα</title>
    </head>
    <body>
        <h1>Η σελίδα μας</h1>
        <p>Σελίδα καλωσορίσματος</p>
    </body>
</html>
```

Σε αυτό το έγγραφο βλέπουμε ότι η ρίζα είναι το <html>, υπάρχει το στοιχείο <head> που είναι παιδί του <html>, το <body> που είναι επίσης ένα άλλο παιδί του <html> και αυτά με τη σειρά τους έχουν άλλα στοιχεία ως παιδιά τους.

Το document object model θα μπορούσαμε να το δούμε ως μια ιεραρχία από κόμβους ως εξής:



Εικόνα 7.12 *Το Document Object Model (DOM) είναι μια ιεραρχία από κόμβους.*

Στη συνέχεια θα δούμε την προγραμματιστική διεπαφή DOM API, που περιλαμβάνει τις συναρτήσεις που θα χρησιμοποιήσουμε σε ένα πρόγραμμα JavaScript για να επικοινωνήσει με αυτά τα στοιχεία.

7.9.1 Ανάκτηση στοιχείων του DOM

Η διεπαφή περιλαμβάνει διάφορες μεθόδους οι οποίες μας επιτρέπουν την ανάκτηση στοιχείων του δένδρου.

Η πιο σημαντική συνάρτηση είναι η **querySelector()**. Όπως και όλες οι συναρτήσεις του DOM API, είναι μέθοδος του αντικείμενου document, γι' αυτό την καλούμε ως document.querySelector(). Ως όρισμα περνάμε μια προδιαγραφή για το στοιχείο που αναζητούμε, χρησιμοποιώντας τη σύνταξη των επιλογέων της CSS (Ενότητα 4.4). Για παράδειγμα, αν δώσουμε όρισμα το "a", η συνάρτηση θα μας επιστρέψει το πρώτο στοιχείο τύπου <a>, δηλαδή το πρώτο anchor του εγγράφου. Η κλήση της document.querySelector(".myClass") θα επιστρέψει το πρώτο στοιχείο που έχει γνώρισμα class = "myClass", ενώ η document.querySelector("#myId") θα επιστρέψει το στοιχείο με γνώρισμα id = "myId" κ.λπ.

Μια παραλλαγή του querySelector είναι η document.querySelectorAll(). Η μέθοδος αυτή επιστρέφει μια συλλογή με στοιχεία, σε αντίθεση με την querySelector που επιστρέφει ένα μόνο στοιχείο. Θα μας επιστρέψει όλα τα στοιχεία που ικανοποιούν τον επιλογέα, για παράδειγμα η document.querySelectorAll("a") θα επιστρέψει όλα τα στοιχεία τύπου anchor <a> που υπάρχουν στην ιστοσελίδα.

Υπάρχουν και άλλες μέθοδοι για αναζήτηση στοιχείων του DOM, αυτές είναι:

- **document.getElementById("myid")** Επιστρέφει το στοιχείο που έχει id="myid".
- **document.getElementsByClassName("myClass")** Επιστρέφει τη συλλογή στοιχείων με όνομα κλάσης class = "myClass".
- **document.getElementsByTagName("tag")** Επιστρέφει συλλογή στοιχείων με όνομα ετικέτας tag.

Κατά κάποιον τρόπο, οι μέθοδοι αυτές έχουν υπερκαλυφθεί από τη διεπαφή querySelector που είδαμε, η οποία παίρνει ως όρισμα έναν επιλογέα CSS που μπορεί να αντιστοιχεί σε στοιχείο HTML, σε κλάση CSS, σε γνώρισμα (attribute), id κ.λπ.

Να σημειωθεί πως όλες οι παραπάνω μέθοδοι, εκτός από την getElementById, μπορούν να εκτελεστούν όχι μόνο στο document, αλλά και σε κάποιο στοιχείο.

Π.χ. η κλήση document.querySelector('#someId').querySelectorAll('.someClass') θα επιστρέψει μια

λίστα με τα στοιχεία που έχουν κλάση “someClass” και είναι απόγονοι του στοιχείου #someId.

7.9.2 Διαχείριση των γνωρισμάτων στοιχείων DOM

Στη συνέχεια θα δούμε τις μεθόδους της διεπαφής DOM που αφορούν τα γνωρίσματα των στοιχείων. Έστω πως ένα στοιχείο element είναι ένα αντικείμενο που μας επέστρεψε η μέθοδος querySelector. Μπορούμε με τη μέθοδο element.setAttribute(atr, val) να δώσουμε τιμή σε ένα γνώρισμά του element.getAttribute(attr), να πάρουμε την τιμή ενός γνωρίσματος, ενώ με την element.removeAttribute(attr) να καταργήσουμε ένα γνώρισμα.

7.9.3 Διαχείριση περιεχομένου στοιχείων DOM

Για τη διαχείριση του περιεχομένου των στοιχείων του DOM η διεπαφή διαθέτει δύο ιδιότητες, την innerHTML και την textContent.

Η ιδιότητα innerHTML έχει δύο χρήσεις:

(α) Η element.innerHTML επιστρέφει το περιεχόμενο του στοιχείου element, περιλαμβανομένου του κειμένου που αυτό περιέχει, καθώς και του κώδικα HTML.

(β) Επιπροσθέτως, μπορεί να χρησιμοποιηθεί για να δώσουμε τιμή στο περιεχόμενο του στοιχείου element, element.innerHTML = text.

Αντίστοιχη είναι η ιδιότητα element.textContent, η οποία μας επιστρέφει μόνο το κείμενο που περιέχεται στο στοιχείο.

Συνεπώς, θα πρέπει να τονιστεί πως η διαφορά της ιδιότητας textContent από την ιδιότητα innerHTML ενός στοιχείου είναι ότι η πρώτη περιέχει μόνο το κείμενο που περιέχεται στο στοιχείο, ενώ η δεύτερη περιέχει όλο τον κώδικα HTML που περιέχεται στο στοιχείο, τα στοιχεία παιδιά του κ.λπ.

7.9.4 Διαχείριση του στιλ των στοιχείων DOM

Η διεπαφή DOM μάς επιτρέπει να έχουμε πρόσβαση και να διαχειριστούμε το στιλ των στοιχείων.

Συγκεκριμένα, η ιδιότητα element.style μας επιστρέφει την προδιαγραφή του στιλ του συγκεκριμένου στοιχείου όπως καθορίζεται από τα φύλλα στιλ που σχετίζονται με το στοιχείο αυτό, ενώ επίσης μας επιτρέπει να τροποποιήσουμε αυτή την προδιαγραφή, αφού μπορούμε να ορίσουμε παραμέτρους στιλ αλλάζοντας την τιμή τους.

```
element.style.ιδιότητα = τιμή
```

Ένα σημείο που πρέπει να προσέξουμε, όμως, είναι η έκφραση των ιδιοτήτων στιλ της CSS ως μεταβλητών JavaScript. Επειδή στη CSS έχουμε συχνά μεταβλητές που περιέχουν τον χαρακτήρα παύλα-ενωτικό, π.χ. font-size, background-color κ.λπ., αυτές οι μεταβλητές δεν επιτρέπονται στην JavaScript, αφού ο χαρακτήρας "-" δεν επιτρέπεται στο όνομα μεταβλητής. Μια σύμβαση που ισχύει είναι να αναφερόμαστε στην ιδιότητα font-size της CSS ως fontSize στην JavaScript.

```
element.style.fontSize = "2rem";
```

7.9.5 Διαπέραση δένδρου DOM

Μια άλλη δυνατότητα που μας παρέχει η διεπαφή DOM είναι να διαπεράσουμε τα στοιχεία του δένδρου.

Ας δούμε καταρχήν τους όρους στην προγραμματιστική διεπαφή του DOM. Είναι όροι που συναντώνται γενικότερα στις δομές δεδομένων δένδρου.

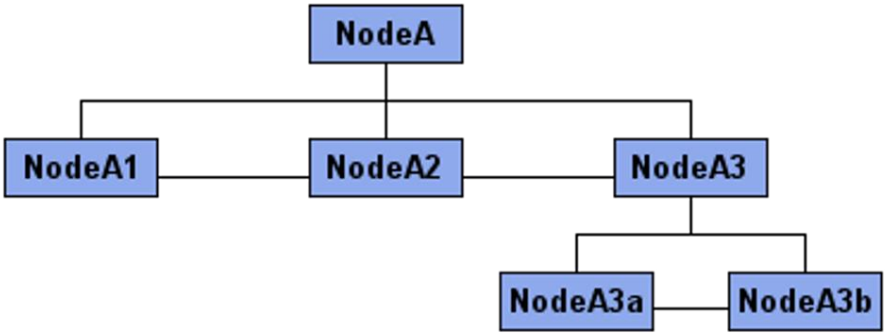
- Το **Element** αντιστοιχεί σε ένα στοιχείο.
- Το **Root** αντιστοιχεί στη ρίζα του δέντρου, το <html>.
- **Child** είναι το παιδί ενός κόμβου.
- **Descendant** είναι οποιοδήποτε κόμβος είτε παιδί είτε παιδί παιδιού, δηλαδή οποιοσδήποτε κόμβος του υποδένδρου κάτω από έναν κόμβο.
- **Parent** είναι ο πατέρας, το γονικό στοιχείο όπως ονομάζεται.
- **Sibling** είναι τα αδέλφια, είναι άλλα στοιχεία που ανήκουν στο ίδιο επίπεδο.
- **Text** node είναι ένας κόμβος που περιέχει κείμενο.

Ακολουθούν μερικά παραδείγματα διαπέρασης του δένδρου:

- element.firstChild επιστρέφει το πρώτο παιδί του στοιχείου element.
- element.lastChild επιστρέφει το τελευταίο παιδί του στοιχείου element.
- element.childNodes επιστρέφει ως συλλογή τα παιδιά του element.
- element.childNodes.length επιστρέφει το πλήθος των παιδιών του element.
- element.childNodes[0] επιστρέφει το πρώτο παιδί του element, ισοδύναμο με το element.firstChild.
- element.nextSibling είναι ο επόμενος αδελφός και element.prevSibling ο προηγούμενος αδελφός του element.
- element.parentNode είναι ο πατέρας του στοιχείου element.

Άσκηση

Έστω το παρακάτω απόσπασμα από ένα δένδρο DOM.



Εικόνα 7.13 *Απόσπασμα ενός δέντρου DOM.*

Ζητείται να βρείτε τις απαντήσεις στα παρακάτω ερωτήματα:

1. NodeA.firstChild
2. NodeA.lastChild
3. NodeA.childNodes.length
4. NodeA.childNodes[0]
5. NodeA.childNodes[1]
6. NodeA.childNodes[2]
7. NodeA1.parentNode
8. NodeA1.nextSibling
9. NodeA3.prevSibling
10. NodeA3.nextSibling
11. NodeA.lastChild.firstChild
12. NodeA3b.parentNode.parentNode

Απάντηση

(1) NodeA1, (2) NodeA3, (3) 3, (4) NodeA1, (5) NodeA2, (6) NodeA3, (7) NodeA, (8) NodeA2, (9) NodeA2, (10) null, (11) NodeA3a, (12) NodeA.

7.9.6 Εισαγωγή και διαγραφή στοιχείων του DOM

Μια ακόμη χρήσιμη δυνατότητα που μας παρέχει η διεπαφή του DOM είναι αυτή της δημιουργίας και της διαγραφής στοιχείων του δένδρου.

Αυτή η δυνατότητα υποστηρίζεται από τη μέθοδο createElement() που δημιουργεί ένα νέο στοιχείο, και στη συνέχεια από τη μέθοδο element.appendChild() για προσθήκη του στοιχείου στην ιεραρχία, ή τη μέθοδο element.removeChild() για διαγραφή στοιχείου από την ιεραρχία.

Ας δούμε ένα παράδειγμα. Έστω ότι θέλουμε να δημιουργήσουμε έναν κόμβο <p> κάτω από τον κόμβο <section> ενός εγγράφου HTML.

```
const sect = document.querySelector('section');
const para = document.createElement('p');
para.textContent = 'Ευχαριστούμε για την επίσκεψη!';
sect.appendChild(para);
```

Παρατηρούμε στο παράδειγμα αυτό ότι αρχικά στη μεταβλητή sect εκχωρήσαμε το στοιχείο <section> του εγγράφου (αν υπήρχαν πολλά στοιχεία αυτού του τύπου, μας επιστρέφει το πρώτο από αυτά), στη συνέχεια δημιουργούμε ένα νέο στοιχείο (μεταβλητή para) που του δίνουμε ένα περιεχόμενο κείμενο και, τέλος, το επισυνάπτουμε στο στοιχείο sect με τη μέθοδο appendChild().

Ας δούμε ένα ακόμη παράδειγμα, της διαγραφής του πρώτου παιδιού ενός κόμβου του δένδρου.

```
const sect = document.querySelector('section');
sect.removeChild(sect.childNodes[0]);
```

Να σημειωθεί ότι η μέθοδος removeChild() επιστρέφει ένα αντικείμενο τύπου Node μετά την επιτυχή διαγραφή του στοιχείου, ή null αν το στοιχείο δεν βρέθηκε στο δένδρο.

Άσκηση

Να εισάγετε ως πρώτο παιδί του στοιχείου με ταυτότητα id = "myID" έναν υπερσύνδεσμο με κείμενο *Πανεπιστήμιο Πατρών* προς την ιστοσελίδα του Πανεπιστημίου Πατρών.

Απάντηση

Θα χρησιμοποιήσουμε την appendChild():

```
const el = document.querySelector('#myID');
const link = document.createElement('a');
link.textContent = 'Πανεπιστήμιο Πατρών';
link.href = 'https://www.upatras.gr';
el.appendChild(link)
```

Αυτά ως μια σύντομη εισαγωγή στη διεπαφή προς το DOM και μια ιστοσελίδα. Οι μέθοδοι που είδαμε σε αυτή την ενότητα και ιδιαίτερα η querySelector θα είναι το βασικό μας εργαλείο για την επικοινωνία ανάμεσα σε ένα πρόγραμμα JavaScript και την ιστοσελίδα με την οποία θέλει να επικοινωνήσει.

7.10 Ερωτήσεις αυτοαξιολόγησης

1. Η JavaScript είναι μια παραλλαγή της Java για την κατασκευή ιστοσελίδων
   – Σωστό/Λάθος
2. Η μηχανή της JavaScript που τρέχει στο περιβάλλον του φυλλομετρητή Chrome είναι η
   Απάντηση _____
3. Το επίσημο όνομα της JavaScript είναι:
   1. JS
   2. ECMAScript
   3. node.js
   4. jQuery
   5. Vue.js
4. Δεν επιτρέπεται η ενσωμάτωση κώδικα JavaScript σε ένα αρχείο HTML.
   – Σωστό/Λάθος
5. Ποια είναι η ετικέτα μέσω της οποίας εισάγουμε κώδικα JavaScript σε ένα αρχείο HTML;
   1. <javascript>
   2. <script>
   3. <code>
   4. <link>
6. Ποια είναι η ιδιότητα που πρέπει να προσθέσουμε στο στοιχείο <script> ώστε ο κώδικας να τρέξει μετά την ολοκλήρωση φορτώματος της HTML;
   Απάντηση: _____
7. Αν θέλουμε να μάθουμε το πλάτος του παράθυρου της ιστοσελίδας, από ποιο αντικείμενο θα το πάρουμε;
   1. navigator.innerWidth
   2. window.innerWidth
   3. document.innerWidth
   4. canvas.innerWidth
8. Η μέθοδος document.querySelectorAll("p") θα επιστρέψει:
   1. όλα τα στοιχεία <p> του εγγράφου.
   2. το πρώτο στοιχείο <p> του εγγράφου.
   3. όλα τα στοιχεία του εγγράφου που αρχίζουν από "p".
   4. όλα τα στοιχεία του εγγράφου που έχουν id = "p".
   5. όλα τα στοιχεία του εγράφου που έχουν class = "p".
9. Έστω το παρακάτω έγγραφο HTML:

```
<h1>κατοικίδια</h1>
<h2>γάτες</h2>
<h2>σκύλοι</h2>
```

   Ποιο το αποτέλεσμα του document.querySelector('h2').textContent;
   Απάντηση: _____
10. Αν δώσουμε την εντολή console.log("hi") σε ένα script JS…
    1. η συμβολοσειρά ‘hi’ θα γραφτεί σε ένα αρχείο log file, στον ίδιο φάκελο με το αρχείο javaScript.
    2. η συμβολοσειρά ‘hi’ θα γραφτεί στο τέλος της ιστοσελίδας.
    3. η συμβολοσειρά ‘hi’ θα παρουσιαστεί σε ένα αναδυόμενο παράθυρο (alert window).
    4. η συμβολοσειρά ‘hi’ θα γραφτεί στην κονσόλα της JS.
11. Έστω η παρακάτω εντολή JS: document.querySelector('#a').textContent = 'info'. Ποιο το αποτέλεσμα;
    1. Ο υπερσύνδεσμος #a θα εμφανίζεται στον χρήστη ως η λέξη info.
    2. Το στοιχείο με id="a" θα περιέχει τη λέξη info που θα παρουσιάζεται στον χρήστη.
    3. Το στοιχείο <a> θα περιέχει τη λέξη info που θα παρουσιάζεται στον χρήστη.
    4. Το στοιχείο με ιδιότητα #a θα περιέχει τη λέξη info που θα παρουσιάζεται στον χρήστη.
    5. Το στοιχείο με class = "a" θα περιέχει τη λέξη info που θα παρουσιάζεται στον χρήστη.
12. Έστω ο παρακάτω κώδικας JS: let x = window.prompt('x='). Αν ο χρήστης δώσει την τιμή 33 στο αναδυόμενο παράθυρο, ποια θα είναι η τιμή που θα λάβει η μεταβλητή x;

1. Ο αριθμός 33.
2. Η συμβολοσειρά ‘33’.
3. Undefined.
4. Null.

13. Έστω οι παρακάτω εντολές JS:

```
let year = 1821;
let year = 1940;
```

Ποια η τελική τιμή της μεταβλητής year;

1. 1940.
2. 1821 γιατί η δεύτερη εντολή αγνοείται.
3. Θα πάρουμε σφάλμα, δεν επιτρέπεται ο επανορισμός της x X.

14. Έστω η παρακάτω εντολή JS : let x = 12.5; Τι τύπου είναι η μεταβλητή x;

1. real
2. integer
3. number
4. string
5. boolean

15. Ποια από τα παρακάτω ονόματα μεταβλητών είναι επιτρεπτά; (σημειώστε όλα όσα ταιριάζουν)

1. a
2. _a
3. $a
4. 12a
5. a12

16. Η JavaScript έχει τις παρακάτω χρήσεις (σημειώστε όλα όσα ισχύουν):

1. Εκτελείται στο περιβάλλον του φυλλομετρητή.
2. Ως γλώσσα εξυπηρετητή εφαρμογών.
3. Ως γλώσσα για ανάπτυξη εφαρμογών σε κινητά τηλέφωνα.
4. Ως γλώσσα προγραμματισμού εφαρμογών desktop.

17. Η JavaScript έκδοση ES6 έχει διάδοση σε πάνω από το 95% των φυλλομετρητών που χρησιμοποιούνται σήμερα σύμφωνα με την πηγή caniuse.com

– Σωστό/Λάθος

18. Ποιο το αποτέλεσμα;

```
let a=10;
let b = a++;
console.log(b);
```

Απάντηση: _____

7.11 Βιβλιογραφία και Αναφορές

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 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 η 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 των 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 έκδοση). 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.