ΕΙΣΑΓΩΓΗ ΣΤΗ JavaScript

THE scripting language of the Web

ΜΕΡΟΣ Α’ - Περιεχόμενα

- Ιστορικά στοιχεία
- Τι είναι η JavaScript
- Τι μπορεί να κάνει
- Που & πως χρησιμοποιείται
- Βασικές αρχές
- Πολλά παραδείγματα

Δημιουργία Ιστοτόπων

Δομή Ιστοσελίδας

Περιεχόμενο

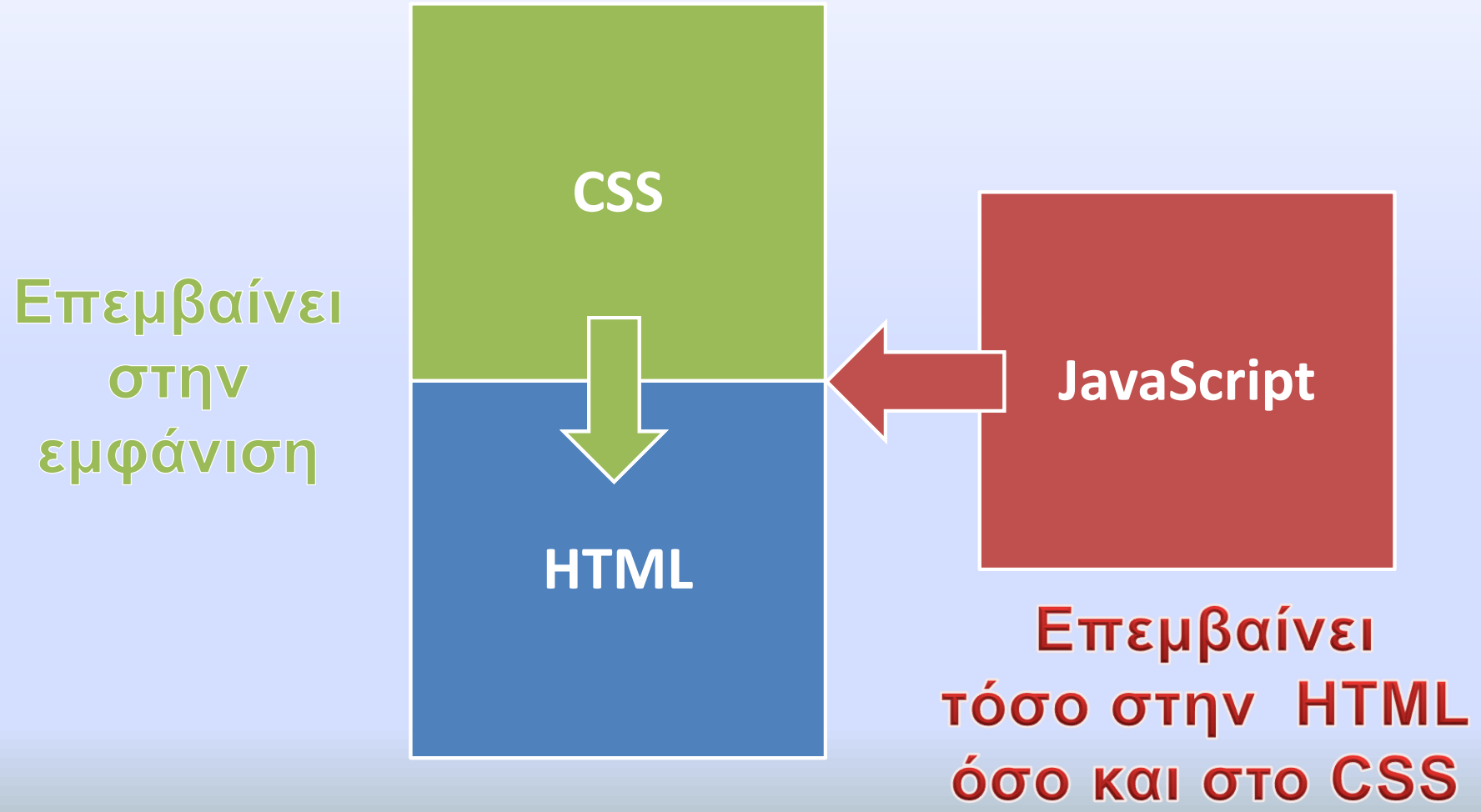
Μορφοποίηση – Στυλ

Εμφάνιση

Γλώσσα Προγραμματισμού

Συμπεριφορά

Δημιουργία Ιστοτόπων



Ιστορικά Στοιχεία

Brendan Eich

1991: Tim Burners-Lee: HTML + HTTP + Browser

1994: Håkon Wium Lie & Bert Boss: Πρότυπο CSS

1995 - 2001: Ο 1ος πόλεμος των Browsers:

- Ανταγωνιστές: *Internet Explorer* vs *Netscape Navigator*
- IE: Microsoft (επικράτησε)
- NN: Netscape. Το 1998 γίνεται open-source ο NN και ως πνευματικός διάδοχος προκύπτει ο Mozilla Firefox.

1995: Εισάγεται η JS ως ένας τρόπος εισαγωγής προγραμμάτων στον browser Netscape Navigator. Δημιουργός της γλώσσας ο **Brendan Eich.** Αρχικά ονομαζόταν Mocha, μετονομάστηκε σε LiveScript (10/1995) και τελικά ονομάστηκε JavaScript (12/1995).

- Έκτοτε υιοθετήθηκε από όλους του μεγάλους browsers
- Δεν έχει σχέση με τη Java. Το όνομα επιλέχθηκε για λόγους μάρκετινγκ. Έχουν ομοιότητες στη σύνταξη.

ECMAScript: Ο οργανισμός ECMA (European Computer Manufacturers Organization - Non Profit) έχει αναλάβει να θέτει το τι είναι στάνταρ στη γλώσσα (ECMAScript==JavaScript)

- ECMAScript 1 (**1997**): Πρώτη έκδοση
- ECMAScript 2 (**1998**): Συντακτικές αλλαγές
- ECMAScript 3 (**1999**): Επικρατούσα στα 00s
- ECMAScript 4: Δεν βγήκε ποτέ. Οι φιλόδοξες αλλαγές που προτεινόταν, αποδείχτηκαν κακές στην πράξη
- ECMAScript 5 - ES5 (**2009**): Ελάχιστες αλλαγές
- ECMAScript 6 - ES6 (**2015**): **Τεράστιες αλλαγές, πολλές βελτιώσεις στη γλώσσα**
- Έκτοτε μία νέα έκδοση κάθε χρόνο, ECMAScript 2016, ECMAScript 2017 κ.ο.κ.

Τι είναι η JavaScript

- Η JavaScript είναι μια γλώσσα προγραμματισμού (***γλώσσα συγγραφής σεναρίων - scripting language***) που έχει ως κύρια λειτουργία την **αύξηση των δυνατοτήτων των ιστοσελίδων**. Χρησιμοποιείται κυρίως για να αυξήσει τη διαδραστικότητα των ιστοσελίδων. Υποστηρίζει αντικειμενοστρεφές και συναρτησιακό στυλ προγραμματισμού.

- Χαρακτηριστικό της είναι ότι **εκτελείται από τον φυλλομετρητή (browser)** του επισκέπτη της σελίδας (**client side**) και όχι από τον web server*.

- Αυτό έχει ως αποτέλεσμα τη **δέσμευση** από τις δυνατότητες της έκδοσης και του είδους του browser που διαθέτει ο χρήστης.

* Υπάρχει βέβαια και υλοποίηση JavaScript στο διακομιστή (**server side**) π.χ. με την πλατφόρμα Node.js, ενός μοντέλου προγραμματισμού βασισμένο στα συμβάντα (events).

Τι είναι η JavaScript

- Ο κώδικας της JavaScript γράφεται σε καθαρό κείμενο και **ενσωματώνεται μέσα στον κώδικα της HTML ή σε εξωτερικό αρχείο**, μπορεί να εκτελεστεί αμέσως ή όταν λαμβάνει χώρα ένα συμβάν (**event**).

- Αν και ακούγονται ίδιες **η JavaScript δεν θα πρέπει να συγχέεται με τη Java**, που είναι διαφορετική γλώσσα προγραμματισμού και με διαφορετικές εφαρμογές.

- *Η JavaScript χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων. Τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers), οι μικρές εφαρμογές της επιφάνειας εργασίας (desktop widgets) κ.α.*

Τι μπορεί να κάνει η JavaScript

- **Παρέχει στους σχεδιαστές HTML ένα εργαλείο προγραμματισμού** – Οι συγγραφείς της HTML συνήθως δεν είναι προγραμματιστές, αλλά χρησιμοποιώντας τη JavaScript μπορεί σχεδόν ο καθένας να εισάγει μικρά «αποσπάσματα» κώδικα JavaScript στις HTML σελίδες.

- **Μπορεί να αντιδράσει σε γεγονότα (events)** – Η JavaScript μπορεί να οριστεί για να εκτελείται όταν συμβαίνει κάτι, όπως όταν έχει ολοκληρωθεί η φόρτωση της ιστοσελίδας ή όταν ο χρήστης κάνει κλικ σε ένα στοιχείο HTML (π.χ. ένα κουμπί φόρμας).

- **Μπορεί να διαβάσει και να γράψει HTML στοιχεία (ετικέτες)** – Η JavaScript μπορεί να διαβάσει και να αλλάξει το περιεχόμενο ενός στοιχείου HTML (π.χ. το περιεχόμενο μιας παραγράφου ή την πηγή (src) μιας φωτογραφίας δημιουργώντας «ειδικά εφέ» στην ιστοσελίδα π.χ. εναλλαγή φωτογραφιών).

Τι μπορεί να κάνει η JavaScript

- **Μπορεί να διαβάσει και να γράψει κανόνες CSS** μεταβάλλοντας με αυτό τον τρόπο την εμφάνιση της ιστοσελίδας.

- **Μπορεί να χρησιμοποιηθεί για την επικύρωση δεδομένων φόρμας** πριν υποβληθεί σε ένα διακομιστή (web server). Η διαδικασία αυτή γλυτώνει το διακομιστή από επιπλέον επεξεργασία.

- **Μπορεί να χρησιμοποιηθεί για την ανίχνευση του φυλλομετρητή (browser) του επισκέπτη** και ανάλογα με τον browser να φορτώσει άλλη σελίδα η οποία έχει σχεδιαστεί ειδικά για το συγκεκριμένο browser.

- **Μπορεί να χρησιμοποιηθεί για να δημιουργήσει cookies** – Η JavaScript μπορεί να χρησιμοποιηθεί για την αποθήκευση και ανάκτηση πληροφοριών από/στον υπολογιστή του επισκέπτη (αποθήκευση και ανάγνωση cookies) …***και πολλά άλλα.***

Απαιτήσεις

ΕΡΩΤΗΣΗ: Τι χρειαζόμαστε για να γράψουμε JavaScript;

ΑΠΑΝΤΗΣΗ: Ότι χρειαζόμαστε για να γράψουμε HTML. Ένα Text Editor π.χ. Notepad++

ΕΡΩΤΗΣΗ: Τι χρειαζόμαστε για να τρέξουμε τη JavaScript;

ΑΠΑΝΤΗΣΗ: Ένα φυλλομετρητή (browser)

Βασικές αρχές

Ο κώδικας μπορεί είτε να **τοποθετηθεί μέσα στη ιστοσελίδα**, είτε να βρίσκεται σε **ξεχωριστό αρχείο** και να καλείται κάθε φορά που φορτώνεται η σελίδα. Ο δεύτερος τρόπος είναι καλύτερος όταν θέλουμε να χρησιμοποιήσουμε τον ίδιο κώδικα από πολλές σελίδες.

Πριν ξεκινήσουμε το πρώτο μας πρόγραμμα πρέπει να γνωρίζουμε ότι:

- Σε αντίθεση με την HTML, η JavaScript είναι **case sensitive**, δηλαδή τα πεζά και τα κεφαλαία αποτελούν διαφορετικά γράμματα. Π.χ. οι **test**, **Test**, **TEST** αποτελούν για τη JavaScript 4 διαφορετικές λέξεις!

- Αν και μπορούμε να γράφουμε τις εντολές τη μια κάτω από την άλλη, καλό είναι να υποδηλώνουμε το τέλος της γραμμής με ένα **ελληνικό ερωτηματικό** (semicolon). Η JavaScript αντιλαμβάνεται ***το τέλος της γραμμής σαν τέλος της δήλωσης*** *(υπάρχουν ορισμένες εξαιρέσεις)*. Με χρήση του semicolon μπορούμε να γράψουμε πολλές δηλώσεις σε μία γραμμή.

- Η JavaScript είναι Γλώσσα Προγραμματισμού και σε περίπτωση λάθους εμφανίζει αντίστοιχο μήνυμα στην κονσόλα (είναι προσβάσιμη μέσω των εργαλείων προγραμματιστών του φυλλομετρητή F12).

Online editors

Για να εκτελέσουμε τα παραδείγματα που ακολουθούν, έχουμε τη δυνατότητα:

1. Να χρησιμοποιήσουμε έναν editor όπως το **notepad++,** να αποθηκεύουμε τοπικά τα αρχεία μας και να βλέπουμε το αποτέλεσμα εκτέλεσης του κώδικα σε κάποιον φυλλομετρητή.
2. Να χρησιμοποιήσουμε κάποιον **online editor** στον οποίο θα βλέπουμε σε πραγματικό χρόνο τα αποτελέσματα εκτέλεσης του κώδικα που γράφουμε.

   Προτεινόμενοι online editors:

   - **http://liveweave.com**
   - **http://jsbin.com**
   - **http://jsfiddle.net**
   - **http://plnkr.co**
   - **http://www.w3schools.com**

Που τοποθετούμε την JavaScript

- Ο κώδικας JavaScript μπαίνει σχεδόν σε κάθε μέρος της σελίδας.
- Υπάρχουν όμως κάποιοι "άγραφοι κανόνες":

```
<!DOCTYPE html>
<html>
  <head>
    … Εδώ τοποθετούνται οι συναρτήσεις
  </head>
  <body>

    … Συνήθως ο κώδικας JavaScript γράφεται  στο τέλος του body
  </body>
</html>
```

DOM

Document Object Model (DOM):

- Είναι η εσωτερική αναπαράσταση (στη μνήμη) μίας σελίδας.
- Αποτελεί μία δενδρική δομή δεδομένων που οργανώνει το περιεχόμενο της σελίδας.
- Οι **κόμβοι** του δένδρου χωρίζονται σε κατηγορίες ανάλογα με το περιεχόμενό τους. Ενδεικτικά αναφέρονται κάποιες κατηγορίες:
  - **Element (Στοιχείο):** Στοιχεία HTML όπως π.χ μία παράγραφος, μία επικεφαλίδα κλπ.
  - **Attribute (Ιδιότητα):** Ιδιότητα ετικέτας (π.χ. το charset="utf-8" στο <meta charset="utf-8">)
  - **Text (Κείμενο):** Κείμενο που περιέχεται σε ένα στοιχείο (π.χ. το "hello" στο hello
  - **Comment (Σχόλιο):** Κείμενο που περιέχεται σε ένα σχόλιο
  - **Document:** Ειδικός κόμβος, υπάρχει μόνο μία φορά και είναι η ρίζα του δένδρου
  - **Document Type:** Ειδικός κόμβος, που εκφράζει την οδηγία που υπάρχει στην αρχή της σελίδας (π.χ <!DOCTYPE html>)

- Ο φυλλομετρητής (browser):
  - Διαβάζει τη σελίδα και κατασκευάζει το DOM
  - Με ειδικές εντολές της JS μπορούμε να αλληλεπιδράσουμε με το DOM, π.χ. προσθαφαιρώντας κόμβους.

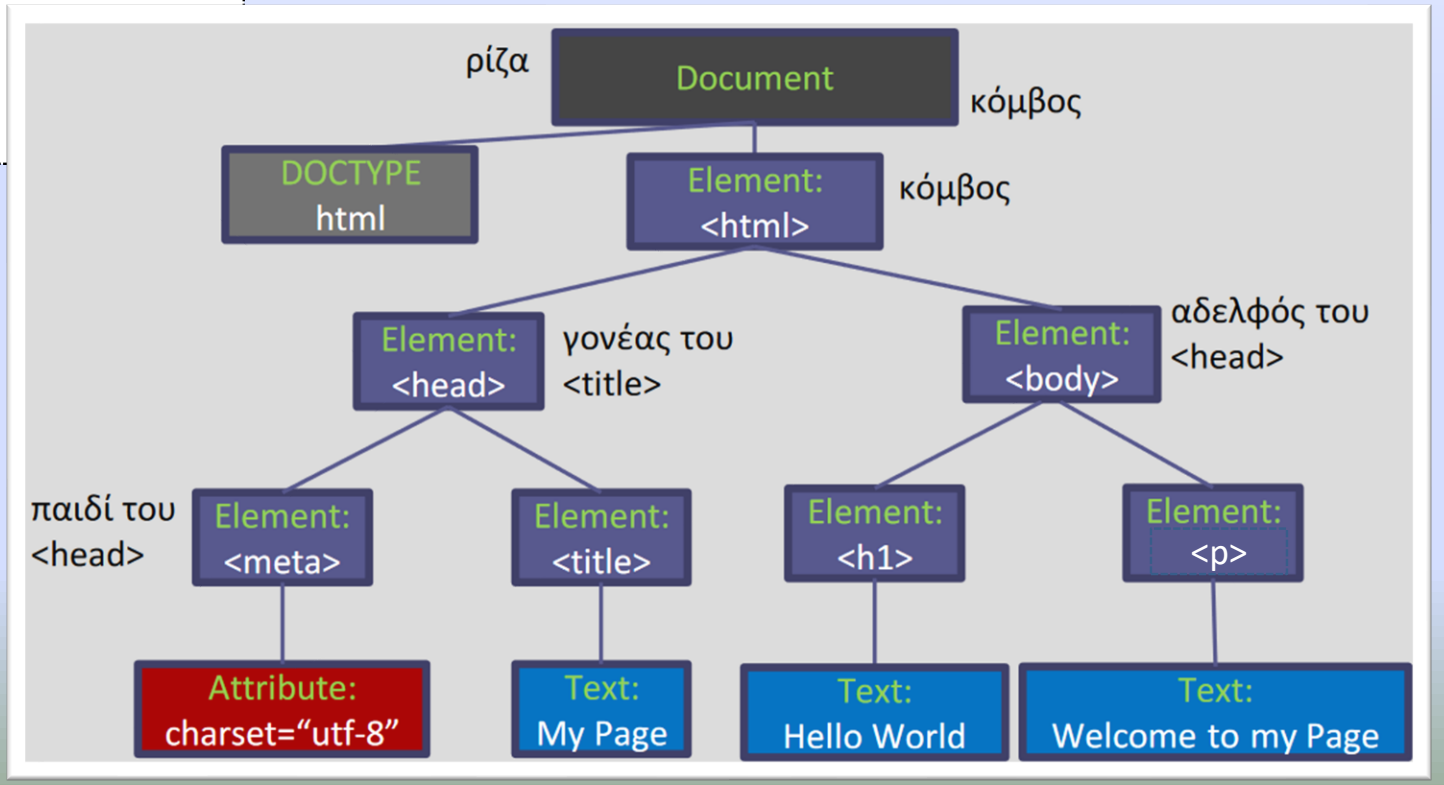
```html
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>My Page</title>
</head>

<body>
    <h1>Hello World!</h1>
    <p>Welcome to my page</p>
</body>

</html>
```

DOM



Δουλεύοντας με την κονσόλα

Στον φυλλομετρητή (browser) μεταβείτε στη διεύθυνση **about:blank**

Πατήστε το πλήκτρο **F12** για να ανοίξουν τα **εργαλεία προγραμματιστή** και επιλέξτε την καρτέλα **Κονσόλα.**

Γράψτε στον interpreter διαδοχικά τις εντολές (πατώντας ***Enter*** μετά από κάθε μία):

```
let openTag = "<h1>";
let closeTag = "</h1>";
let hello = "Hello World!";
let text = openTag + hello + closeTag;
console.log(text);
```

Η συνάρτηση console.log(…)

- Τυπώνει το όρισμά της στην κονσόλα

Εναλλακτική σύναξη:

- **console.log**(a1, a2, a3, … an)
- Θα τυπώσει όλα τα ορίσματά της στην κονσόλα, αφήνοντας ένα κενό μεταξύ των διαδοχικών ορισμάτων

Δουλεύοντας με την κονσόλα

Στον φυλλομετρητή (browser) μεταβείτε στη διεύθυνση **about:blank**

Πατήστε το πλήκτρο **F12** για να ανοίξουν τα **εργαλεία προγραμματιστή** και επιλέξτε την καρτέλα **Κονσόλα.**

Γράψτε στον interpreter διαδοχικά τις εντολές (πατώντας ***Shift-Enter*** μετά από κάθε μία):

```
document.write("<ol>");
for (let i=65; i<=90; i++)
  document.write("<li>" + String.fromCharCode(i) + "</li>");
document.write("</ol>");
```

Στο τέλος πατήστε ***Enter*** για να εκτελεστεί το script.

Τι παρατηρείτε;

Πως χρησιμοποιείται

Ο κώδικας JavaScript μπορεί να τοποθετηθεί στα τμήματα **<body>** και **<head>** μιας ιστοσελίδας.
Στο παρακάτω παράδειγμα που ήδη έχουμε δει η JavaScript τοποθετείται στο τμήμα **<body>** της ιστοσελίδας και εμφανίζει σε μια **υπάρχουσα ετικέτα <p>** τις τρέχουσες πληροφορίες ημερομηνίας και ώρας.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>JavaScript Παράδειγμα 4</title>
</head>

<body>
    <h1>Ημερομηνία / Ώρα</h1>
    <p id="demo"></p>
    <script>
        document.getElementById("demo").innerHTML = Date();
    </script>
</body>
</html>
```

ΠΡΟΣΟΧΗ:

Σημειώστε ότι η JavaScript τοποθετείται στο κάτω μέρος της σελίδας για να βεβαιωθούμε ότι δεν εκτελείται πριν από την δημιουργία του στοιχείου **<p>** .

Πως χρησιμοποιείται

- Η JavaScript σε μια σελίδα HTML εκτελείται κατά τη φόρτωση της σελίδας. Αυτό δεν είναι πάντα κάτι που θέλουμε.
- Μερικές φορές θέλουμε να εκτελεστεί ο κώδικας JavaScript όταν συμβεί κάποιο **γεγονός (event)**, όπως όταν ένας χρήστης κάνει κλικ σε ένα κουμπί. Όταν συμβαίνει αυτό μπορούμε να βάλουμε τον κώδικα (**script**) μέσα σε μια **συνάρτηση (function)**.
- Τα **γεγονότα (events)** χρησιμοποιούνται συνήθως σε συνδυασμό με τις **συναρτήσεις (functions).** Δηλαδή γίνεται κλήση μιας συνάρτησης όταν συμβεί κάποιο γεγονός (όπως παρακάτω στο Παράδειγμα 5, γίνεται κλήση της συνάρτησης changeColor() με το πάτημα ενός πλήκτρου).

Μπορούμε να τοποθετήσουμε απεριόριστο αριθμό **scripts** στην ιστοσελίδα μας, και μπορούμε να έχουμε σενάρια τόσο στο τμήμα **<body>** όσο και το τμήμα **<head>** ταυτόχρονα.

Είναι κοινή πρακτική να εισάγουμε όλες τις **συναρτήσεις** στο τμήμα **<head>** της ιστοσελίδας. Με αυτό τον τρόπο είναι όλες σε ένα μέρος και δεν επηρεάζουν το περιεχόμενο της σελίδας.

Πως χρησιμοποιείται

- Η JavaScript μπορεί επίσης να τοποθετηθεί σε **εξωτερικά αρχεία.**
- Τα εξωτερικά αρχεία JavaScript συχνά περιέχουν κώδικα που πρέπει να χρησιμοποιείται σε πολλές διαφορετικές ιστοσελίδες.
- Τα εξωτερικά αρχεία JavaScript έχουν επέκταση αρχείου **.js**
- **Προσοχή:** Τα εξωτερικά αρχεία με δηλώσεις (εντολές) JavaScript δεν μπορεί να περιέχουν τις ετικέτες **<script> </ script>** παρά μόνο δηλώσεις JavaScript.
- Για να χρησιμοποιήσετε έναν εξωτερικό αρχείο JavaScript, χρησιμοποιείστε την ετικέτα **<script>** με την παράμετρο **src** ως εξής:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Τίτλος Ιστοσελίδας</title>
    <script src="onoma_arxeiou.js"></script>
  </head>
  <body>

  </body>
</html>
```

JavaScript δηλώσεις (statements)

Κώδικας JavaScript

Ο **κώδικας JavaScript** (ή σκέτο JavaScript) είναι μια ακολουθία **JavaScript δηλώσεων** (**εντολών**).

Κάθε δήλωση εκτελείται από τον φυλλομετρητή (browser) **με τη σειρά που είναι γραμμένες**.

Το παρακάτω παράδειγμα γράφει μια επικεφαλίδα <h1> και δύο παραγράφους <p> σε μια ιστοσελίδα:

```
<script>
  document.write('<h1>Αυτή είναι μια επικεφαλίδα</h1>');
  document.write('<p>Αυτή είναι μια παράγραφος</p>');
  document.write('<p>Αυτή είναι μια 2η παράγραφος</p>');
</script>
```

JavaScript δηλώσεις (statements)

JavaScript Blocks

Οι δηλώσεις (εντολές) JavaScript μπορούν να ομαδοποιηθούν σε **μπλοκ**.

Τα **μπλοκ** αρχίζουν με μία αριστερή αγκύλη **{** και τελειώνουν με δεξιά αγκύλη **}**.

Ο σκοπός του **μπλοκ** είναι να εκτελεστεί η ακολουθία των δηλώσεων (εντολών) από κοινού καθώς και να καθοριστεί η εμβέλεια μεταβλητών με χρήση της εντολής **let.**

Αυτό το παράδειγμα γράφει μια επικεφαλίδα <h1> και δύο παραγράφους <p>:

```
<script>
  {
    document.write('<h1>Αυτή είναι μια επικεφαλίδα</h1>');
    document.write('<p>Αυτή είναι μια παράγραφος</p>');
    document.write('<p>Αυτή είναι μια 2η παράγραφος</p>');
  }
</script>
```

Το παραπάνω παράδειγμα **δεν** είναι πολύ χρήσιμο. Δείχνει απλώς τη χρήση ενός **μπλοκ**. Κανονικά ένα **μπλοκ** χρησιμοποιείται για να δηλώσει μια ομάδα εντολών που εκτελούνται μαζί σε μια **συνάρτηση (function)** ή σε μια **συνθήκη** (όπου μια ομάδα εντολών, θα πρέπει να εκτελείται αν πληρείται κάποια προϋπόθεση) ή σε μία **επανάληψη** κ.λπ.

Παράδειγμα 1

Ο κώδικας που δημιουργούμε μπορεί να τοποθετηθεί μέσα στην ιστοσελίδα με την ετικέτα **<script>** ως εξής:

Το παρακάτω παράδειγμα γράφει το μήνυμα: Hello World στο έγγραφο HTML:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>JavaScript Παράδειγμα 1</title>
</head>
<body>
    <h1>Η πρώτη μου ιστοσελίδα με JavaScript</h1>
    <script type="text/javascript">
      document.write("Hello World");
    </script>
</body>
</html>
```

ΣΗΜΕΙΩΣΗ:

Η ιδιότητα **type** της ετικέτας **script** είναι προαιρετική στην HTML5.

Παράδειγμα 2

Το παράδειγμα γράφει μέσα σε μία **ετικέτα <p>** τις τρέχουσες πληροφορίες ημερομηνίας και ώρας στο έγγραφο HTML χρησιμοποιώντας την **Date()**:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>JavaScript Παράδειγμα 2</title>
</head>

<body>
    <h1>Παραδείγματα με JavaScript</h1>
    <script>
      document.write('<p><b>Ημ./Ώρα:</b> ' + Date() + '</p>');
    </script>
</body>
</html>
```

Παράδειγμα 3

Εμφάνιση μηνύματος σε πλαίσιο διαλόγου:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>JavaScript Παράδειγμα 3</title>
    <script>
      window.alert('Καλωσορίσατε στον Ιστότοπό μας');
    </script>
</head>

<body>
    <h1>Χαιρετισμός με JavaScript</h1>
  </body>
</html>
```

Παράδειγμα 4

Εμφάνιση τρέχουσας ημερομηνίας και ώρας σε μια **υπάρχουσα ετικέτα <p>**:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>JavaScript Παράδειγμα 4</title>
</head>

<body>
    <h1>Ημερομηνία / Ώρα</h1>
    <p id="demo"></p>
    <script>
        document.getElementById("demo").innerHTML = Date();
    </script>
</body>
</html>
```

Παράδειγμα 5

Αλλαγή χρώματος φόντου με τυχαία επιλογή:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Random background color</title>
  <script>
    function changeColor() {
      let r = Math.floor(Math.random() * 256);
      let g = Math.floor(Math.random() * 256);
      let b = Math.floor(Math.random() * 256);
      document.body.style.backgroundColor = `rgb(${r},${g},${b})`;
      document.getElementById("color").innerHTML = `R:${r} G:${g} B:${b}`;
    }
  </script>
</head>

<body>
  <button onclick="changeColor()">Τυχαίο χρώμα φόντου</button>
  <h1 id="color"></h1>
</body>
</html>
```

Χρήσιμος σύνδεσμος: Style Object Properties

Παράδειγμα 6

Αλλαγή χρώματος φόντου με χρόνο:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Change bg color with time</title>
    <script>
        function changeBgColor() {
            let r = Math.floor(Math.random() * 255);
            let g = Math.floor(Math.random() * 255);
            let b = Math.floor(Math.random() * 255);
            let color = `rgb(${r}, ${b}, ${g})`;
            document.body.style.backgroundColor = color;
            document.getElementById("color").innerHTML = color;
        }
    </script>
</head>

<body>
    <h1>Αλλαγή φόντου με χρόνο</h1>
    <h2 id="color"></h2>
    <script>
        setInterval(changeBgColor, 1000);
    </script>
</body>
</html>
```

Παράδειγμα 7

Ρολόι:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Javascript Clock</title>
  <style>
    h2 { text-align: center; font-size: 100px; font-family: monospace; margin: 0; }
  </style>
  <script>
    function timer() {
      let d = new Date();
      document.getElementById("clock").innerHTML = d.toLocaleTimeString();
    }
  </script>
</head>

<body>
  <h1>Ρολόι</h1>
  <h2 id="clock"></h2>
  <script>
    setInterval(timer, 1000);
  </script>
</body>
</html>
```

Παράδειγμα 8

Αλλαγή χρώματος φόντου με επιλογή χρήστη (user defined):

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>User defined background color</title>
    <script>
        function ChangeColor() {
            let xroma = document.getElementById("usercolor").value;
            document.body.style.backgroundColor = xroma;
        }
    </script>
</head>

<body>
    <p>Αλλαγή Χρώματος φόντου. Γράψε το χρώμα και πάτησε ΟΚ</p>
    <form>
        <input type="text" id="usercolor">
        <input type="button" onclick="ChangeColor()" value="OK">
    </form>
</body>
</html>
```

Παράδειγμα 9

Φόρμες και επιλογέας ολίσθησης:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Slider value</title>
  <script>
    function display(val) {
      document.getElementById("v1").innerHTML = val;
    }
  </script>
</head>

<body>
  <form name="myForm" action="#">
    <p>Εμφάνιση τιμής επιλογέα ολίσθησης</p>
    <input type="range" min="0" max="1000" name="num1" oninput="display(this.value)">
    <span id="v1"></span>
  </form>
  <script>
    document.getElementById("v1").innerHTML = document.myForm.num1.value;
  </script>
</body>
</html>
```

Παράδειγμα 10

Μετατροπή Δεκαδικού (dec) σε Δυαδικό (bin), Οκταδικό (oct), Δεκαεξαδικό (hex)

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Dec to Bin, Oct, Hex</title>
    <script>
        function showValues() {
            let dec = document.myForm.slider.value;

            document.getElementById("decValue").innerHTML = dec;
            document.getElementById("binValue").innerHTML = parseInt(dec).toString(2);
            document.getElementById("octValue").innerHTML = parseInt(dec).toString(8);
            document.getElementById("hexValue").innerHTML = parseInt(dec).toString(16);
        }
    </script>
</head>

<body>
    <form name="myForm">
        Δεκαδικό: <input type="range" name="slider" min="0" max="255" value="0" oninput="showValues()">
        <span id="decValue">0</span>
    </form>
    <p>Δυαδικό: <span id="binValue">0</span></p>
    <p>Οκταδικό: <span id="octValue">0</span></p>
    <p>16δικό: <span id="hexValue">0</span></p>
</body>
</html>
```

Παράδειγμα 11

Αλλαγή χρώματος φόντου με επιλογείς ολίσθησης

```
<!DOCTYPE html>
<html>
    <head>
    <meta charset="utf-8">
    <title>Select background color</title>
  <style>
    body { background-color: black; }
    form { background-color: white; }
  </style>
    <script>
      function display(itemID, val) {
        document.getElementById(itemID).innerHTML = val + " #" + parseInt(val).toString(16);
        changeColor();
      }
      function changeColor() {
        let R = document.myForm.r.value;
        let G = document.myForm.g.value;
        let B = document.myForm.b.value;
        document.body.style.backgroundColor = "rgb(" + [R, G , B].join(",") + ")";
      }
    </script>
  </head>
  <body>
    <form name="myForm" action="#">
      R: <input type="range" min="0" max="255" value="0" name="r" oninput="display('Rval', this.value)">
      <span id="Rval">0</span><br>
      G: <input type="range" min="0" max="255" value="0" name="g" oninput="display('Gval', this.value)">
      <span id="Gval">0</span><br>
      B: <input type="range" min="0" max="255" value="0" name="b" oninput="display('Bval', this.value)">
      <span id="Bval">0</span>
    </form>
  </body>
</html>
```

Παράδειγμα 12

Υπολογισμός με δεδομένα από στοιχεία φόρμας:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Calculate form data</title>
  <script>
    function calculate() {
      let no1 = Number(document.myForm.number1.value);
      let no2 = Number(document.myForm.number2.value);
      let no3 = no1 + no2;
      document.getElementById("result").innerHTML = no3;
    }
  </script>
</head>

<body>
  <form name="myForm" action="#">
    <input type="text" name="number1"> +
    <input type="text" name="number2"> =
    <span id="result"></span>
    <p><input type="button" value="Υπολόγισε" onClick="calculate()"></p>
  </form>
</body>
</html>
```

Παράδειγμα 13

Εμφάνιση κωδικού σε πλαίσιο κειμένου:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Show / Hide password</title>
    <script>
        let visible = false;
        function showPass() {
            if (visible) {
                document.myform.pass.setAttribute("type", "password");
                document.myform.btn.setAttribute("value", "Δείξε με");
                visible = false;
            } else {
                document.myform.pass.setAttribute("type", "text");
                document.myform.btn.setAttribute("value", "Κρύψε με");
                visible = true;
            }
        }
    </script>
</head>

<body>
    <form name="myForm" action="#">
        <input type="password" name="pass">
        <input type="button" name="btn" value="Δείξε με" onclick="showPass()">
    </form>
</body>
</html>
```

Παράδειγμα 14

Εμφάνιση κωδικού σε πλαίσιο κειμένου με χρήση εικόνας

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Show / Hide pass with eye
image</title>
    <style>
        input, img {
            vertical-align: middle;
        }
        input {
            height: 32px;
            width: 250px;
        }
        img {
            margin: -40px;
            cursor: pointer;
        }
    </style>
    <script>
        let visible = false;
        function showPass() {
            let passfield = document.myForm.pass;
            let eye = document.getElementById("eye");
            if (visible) {
                passfield.setAttribute("type", "password");
                passfield.style.backgroundColor = "#ffffff";
                eye.setAttribute("src", "images/eye-open.png");
                visible = false;
            } else {
                passfield.setAttribute("type", "text");
                passfield.style.backgroundColor = "#aaff99";
                eye.setAttribute("src", "images/eye-close.png");
                visible = true;
            }
        }
    </script>
</head>

<body>
    <form name="myForm" action="#">
        <input type="password" name="pass" maxlength="18">
        <img src="images/eye-open.png" id="eye"
onclick="showPass()">
    </form>
</body>
</html>
```

Εικόνες: 

Παράδειγμα 15

Δουλεύοντας με εικόνες:

```html
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>Working with Images</title>
    <style>
        div {
            width: 60vw;
            height: 80vh;
            margin: 0 auto;
            padding: 5px;
            border: 10px solid maroon;
            background-color: #fff9d6);
        }
        img {
            width: 60vw;
            height: 80vh;
            object-fit: contain;
        }
        form {
            margin-top: 10px;
            text-align: center;
        }
        input {
            width: 12vw;
            height: 5vh;
        }
        .sel {
            background-color: #ffcc00;
            font-weight: bold;
        }
    </style>
```

```html
    <script>
        let number = 1; // current bird number

        function imageClicked() {
            deselectButton(number);
            number++;
            if (number>5) number=1;
            document.getElementById("image").setAttribute("src", "images/bird"+number+".jpg");
            selectButton(number);
        }

        function buttonPressed(birdNumber) {
            deselectButton(number);
            number = birdNumber;
            document.getElementById("image").setAttribute("src", "images/bird"+number+".jpg");
            selectButton(number);
        }

        function deselectButton(btnNumber) {
            document.getElementById("bird" + btnNumber).removeAttribute("class");
        }

        function selectButton(btnNumber) {
           document.getElementById("bird" + btnNumber).setAttribute("class", "sel");
        }
    </script>
</head>
<body>
    <div>
        <img src="images/bird1.jpg" id="image" onclick="imageClicked()">
    </div>
    <form name="myForm" action="#">
        <input type="button" name="bird1" id="bird1" value="Βασιλαετός" onclick="buttonPressed(1)"
class="sel">
        <input type="button" name="bird2" id="bird2" value="Κραυγαετός" onclick="buttonPressed(2)">
        <input type="button" name="bird3" id="bird3" value="Μαυρόγυπας" onclick="buttonPressed(3)">
        <input type="button" name="bird4" id="bird4" value="Ασπροπάρης" onclick="buttonPressed(4)">
        <input type="button" name="bird5" id="bird5" value="Όρνιο" onclick="buttonPressed(5)">
    </form>
</body>
</html>
```

Παράδειγμα 16...

Επικύρωση και χειρισμός φόρμας

Κώδικας HTML (1 / 3)

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Form Validation and Handle</title>
    <!-- CSS -->
    <link rel="stylesheet" href="js-16_form.css">
    <!-- JavaScript -->
    <script src="js-16_form.js"></script>
</head>

<body>
    <form name="myForm">
        <p>
            <input type="text" name="fullname" placeholder="Δώσε όνομα (>3 χαρ)">
            <span id="s1"></span>
        </p>
        <p>
            <input type="text" name="factor" placeholder="Δώσε x (0 &le; x &lt; 1)">
            <span id="s2"></span>
        </p>
        <p>
            <input type="checkbox" name="lista" value="A">A
            <input type="checkbox" name="lista" value="B">B
            <input type="checkbox" name="lista" value="Γ">Γ
            <input type="checkbox" name="lista" value="Δ">Δ
            <input type="checkbox" name="lista" value="E">E
            <input type="button" value="Toggle" onclick="toggle()">
            <span id="s3"></span>
        </p>
        <input type="button" value="Έλεγχος Φόρμας" onclick="validateForm()">
        <span id="s4"></span>
    </form>
</body>
</html>
```

Παράδειγμα ...16...

Επικύρωση και χειρισμός φόρμας

Κώδικας CSS (2 / 3) `js-16_form.css`

```css
body {
    font-size: 14pt;
}
input[type="text"] {
    height: 30px;
    width: 350px;
}

input[type="checkbox"] {
    height: 25px;
    width: 30px;
}
input[type="button"], input[type="submit"] {
    height: 30px;
    width: 120px;
}
span {
    vertical-align: middle;
}
.error {
    color: red;
}
.ok {
    color: green;
    font-weight: bold;
}
```

Παράδειγμα ...16

Επικύρωση και χειρισμός φόρμας

Κώδικας JS (3 / 3) `js-16_form.js`

```
function toggle() {
    for (let i = 0; i < myForm.lista.length; i++) {
        let item = myForm.lista[i];
        if (item.checked)
            item.checked = false;
        else
            item.checked = true;
    }
}

function validateForm() {
    let readyToSubmit = true;

    // Έλεγχος fullname
    let s1 = document.getElementById("s1");
    if (document.myForm.fullname.value.length <= 3) {
        s1.setAttribute("class", "error");
        s1.innerHTML = "Εισάγετε πάνω από 3 χαρακτήρες!";
        readyToSubmit = false;
    } else {
        s1.setAttribute("class", "ok");
        document.getElementById("s1").innerHTML = "&check;";
    }

    // Έλεγχος παράγοντα x
    let s2 = document.getElementById("s2");
    let factor = parseFloat(document.myForm.factor.value);
    if (factor < 0 || factor >= 1 || isNaN(factor) ) {
        s2.setAttribute("class", "error");
        s2.innerHTML = "0 &le; x &lt; 1";
        readyToSubmit = false;
    } else {
        s2.setAttribute("class", "ok");
        document.getElementById("s2").innerHTML = "&check;";
    }

    // Έλεγχος checkbox. Απαιτείται τουλάχιστον 1 επιλεγμένο
    let s3 = document.getElementById("s3");
    let oneChecked = false;
    for (let i = 0; i < document.myForm.lista.length; i++)
        if (document.myForm.lista[i].checked)
            oneChecked = true;
    if (!oneChecked) {
        s3.setAttribute("class", "error");
        s3.innerHTML = "Παρακαλώ επιλέξτε τουλάχιστον ένα!";
        readyToSubmit = false;
    } else {
        s3.setAttribute("class", "ok");
        document.getElementById("s3").innerHTML = "&check;";
    }

    // Αποτέλεσμα ελέγχου
    let s4 = document.getElementById("s4");
    if (!readyToSubmit) {
        s4.setAttribute("class", "error");
        s4.innerHTML = "Λάθος στη φόρμα! Παρακαλώ διορθώστε!";
        return false;
    } else {
        s4.setAttribute("class", "ok");
        document.getElementById("s4").innerHTML = "Ο έλεγχος
ολοκληρώθηκε &check;";
        return true;
    }
}
```

Παράδειγμα 17

Εμφάνιση πληροφοριών σελίδας:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Page Info</title>
</head>
<body>
    Πληροφορίες σελίδας:<br>
    <script>
    document.write("<br>Title: ", document.title);
    document.write("<br>Referrer: ", document.referrer);
    document.write("<br>Domain: ", document.domain);
    document.write("<br>URL: ", document.URL);
    </script>
</body>
</html>
```

Παράδειγμα 18

Προπαίδεια

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Πολλαπλασιασμός</title>
    <script>
        function multiply() {
            let number = document.myForm.number.value;
            let result = document.getElementById("result")
            result.innerHTML = null;
            for (let i = 1; i <= 10; i++) {
                result.innerHTML += i + ' * ' + number + ' = ' + i * number + '<br>';
            }
        }
    </script>
</head>
<body>
    <form name="myForm">
        <input type="number" name="number">
        <input type="button" onclick="multiply()" value="Προπαίδεια">
        <p id="result"></p>
    </form>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>JavaScript Παράδειγμα 6</title>
    <script type="text/javascript">
      function calculate() {
        var no1 = parseFloat(document.myform.number1.value);
        var no2 = parseFloat(document.myform.number2.value);
        var no3 = no1 + no2;
        document.getElementById('result').innerHTML = no3;
      }
    </script>
  </head>
  <body>
    <form name="myform" action="#">
      <input type="text" name="number1"> +
      <input type="text" name="number2"> =
      <span id="result"></span>
      <p><input type="button" value="Υπολόγισε"
    </form>
```

ΤΕΛΟΣ Α' ΜΕΡΟΥΣ

Ευχαριστώ για την προσοχή σας

ΜΕΡΟΣ Β’ - Περιεχόμενα

- Δομικά στοιχεία της γλώσσας:
  - Σχόλια
  - Μεταβλητές
  - Τύποι Δεδομένων
  - Τελεστές
- Απλά παραδείγματα

Σχόλια στη JavaScript

Τα σχόλια χρησιμοποιούνται στη JavaScript (όπως σε όλες τις γλώσσες προγραμματισμού) για να εξηγούν τα τμήματα του κώδικα και να τον κάνουν περισσότερο αναγνώσιμο και κατανοητό.

Τα σχόλια μονής γραμμής ξεκινούν με *//*. **Παράδειγμα:**

```
<script>
    // Επικεφαλίδα
    document.write('<h1>Αυτή είναι μια επικεφαλίδα</h1>');
    // Ακολουθούν δύο παράγραφοι
    document.write('<p>Αυτή είναι μια παράγραφος</p>');
    document.write('<p>Αυτή είναι μια 2η παράγραφος</p>');
</script>
```

Σχόλια στο τέλος της γραμμής. **Παράδειγμα:**

```
<script>
    document.write('<h1>Αυτή είναι μια επικεφαλίδα</h1>'); // Επικεφαλίδα
    document.write('<p>Αυτή είναι μια παράγραφος</p>');    // 1η παράγραφος
    document.write('<p>Αυτή είναι μια 2η παράγραφος</p>'); // 2η παράγραφος
</script>
```

Σχόλια στη JavaScript

Σχόλια πολλαπλών γραμμών /* */. **Παράδειγμα:**

```
<script>
    /*
       Ακολουθούν μια επικεφαλίδα
       και δύο παράγραφοι
    */
    document.write('<h1>Αυτή είναι μια επικεφαλίδα</h1>');
    document.write('<p>Αυτή είναι μια παράγραφος</p>');
    document.write('<p>Αυτή είναι μια 2η παράγραφος</p>');
</script>
```

Τα σχόλια χρησιμοποιούνται επίσης κατά την εκσφαλμάτωση του κώδικα για την αποτροπή εκτέλεσης δηλώσεων (εντολών). **Παράδειγμα:**

```
<script>
    // document.write('<h1>Αυτή είναι μια επικεφαλίδα</h1>');
    document.write('<p>Αυτή είναι μια παράγραφος</p>');
    document.write('<p>Αυτή είναι μια 2η παράγραφος</p>');
</script>
```

Μεταβλητές στη JavaScript

Οι μεταβλητές είναι θέσεις μνήμης (containers) για την αποθήκευση πληροφοριών.

Όλοι θυμόμαστε την Άλγεβρα που διδασκόμαστε στο σχολείο (**x=5, y=6, z=x+y**).

Ένα γράμμα λοιπόν (π.χ. το **x**) μπορεί να χρησιμοποιηθεί για να κρατήσει μια τιμή (όπως το 5), αντίστοιχα στο γράμμα **y** αποθηκεύεται άλλη τιμή. Στη συνέχεια χρησιμοποιήσουμε τις παραπάνω πληροφορίες για τον υπολογισμό της αξίας του **z** που προφανώς είναι 11.

Τα γράμματα αυτά ονομάζονται **μεταβλητές** και οι μεταβλητές μπορούν να χρησιμοποιηθούν για να αποθηκεύουν τιμές (**x = 5**) ή αποτελέσματα εκφράσεων (**z = x + y**).

JavaScript μεταβλητές

Όπως στην άλγεβρα, οι μεταβλητές στη JavaScript χρησιμοποιούνται για να συγκρατήσουν τιμές ή εκφράσεις.

Μια μεταβλητή μπορεί να έχει ένα σύντομο όνομα, όπως το **x**, το **y**, το **z** ή ένα πιο περιγραφικό όνομα, όπως το **carName**.

Είναι καλή πρακτική να ονομάζουμε μια μεταβλητή σύμφωνα με το περιεχόμενο που επιθυμούμε να αποθηκεύσουμε σε αυτή. (Π.χ. firstName, lastName, telephone κλπ).

Υπάρχουν διάφορες μέθοδοι (συμβάσεις) ονοματολογίας όπως: **camelCase**, **Snake case**, **Hungarian notation** κλπ**. (Προτεινόμενο: camelCase)**

Μεταβλητές στη JavaScript

Κανόνες για σωστή χρήση ονομασίας μεταβλητών (identifiers) στη JavaScript:

- Τα ονόματα των μεταβλητών μπορεί να περιέχουν **γράμματα (a-z, A-Z)**, **αριθμούς (0-9)**, **το χαρακτήρα υπογράμμισης (_** underscore**)** και **το χαρακτήρα του δολαρίου ($).**
- Τα ονόματα των μεταβλητών **πρέπει να ξεκινούν με ένα γράμμα.**
- Μπορεί επίσης να ξεκινούν με ή το χαρακτήρα υπογράμμισης (_) ή το χαρακτήρα του δολαρίου ($) *- Δεν θα το χρησιμοποιήσουμε στα παραδείγματά μας -*
- Τα ονόματα των μεταβλητών είναι **case sensitive** (y και Y είναι δύο διαφορετικές μεταβλητές).
- Δεσμευμένες λέξεις (λέξεις κλειδιά της JavaScript π.χ. for, while κλπ.) δε μπορούν να χρησιμοποιηθούν ως ονόματα μεταβλητών.
- Εργαλείο επικύρωσης ονομασίας μεταβλητών: https://mothereff.in/js-variables

Μεταβλητές στη JavaScript

Δήλωση (δημιουργία) μεταβλητών – Η λέξη-κλειδί var

Η δημιουργία μεταβλητών αναφέρεται ως "**δήλωση**" μεταβλητής.
Δηλώνουμε τις μεταβλητές στη JavaScript με τη λέξη-κλειδί **var**.

```
var firstName;
```

Μετά τη δήλωση, οι μεταβλητές είναι κενές (δεν έχουν καμία τιμή - *undefined*).

```
firstName = 'Τάσος';
```

Για εκχώρηση τιμής σε μια μεταβλητή, χρησιμοποιούμε το σύμβολο του ίσον (=).

Μπορούμε επίσης να εκχωρήσουμε τιμές για τις μεταβλητές, κατά τη δήλωση:

```
var x = 5;
var firstName = 'Τάσος';
```

Μετά την εκτέλεση των δηλώσεων, η μεταβλητή **x** θα έχει την τιμή **5**, και η **firstName** θα έχει την τιμή **Τάσος**.

Σημείωση: **(α)** Για εκχώρηση αλφαριθμητικής τιμής σε μεταβλητή πρέπει να περικλείουμε την τιμή με μονά ή διπλά εισαγωγικά.

(β) Αν δηλώσουμε ξανά μια μεταβλητή με **var**, δεν χάνει την τιμή της.

(γ) Είναι καλή προγραμματιστική τακτική να δηλώνουμε τις μεταβλητές στην αρχή του script.

Μεταβλητές στη JavaScript

Η JavaScript είναι μια **δυναμική γλώσσα** (dynamically typed language). Ανήκει στις **loosely typed** γλώσσες. Αυτό σημαίνει ότι **δεν χρειάζεται να καθορίζουμε τους τύπους δεδομένων των μεταβλητών κατά τη δήλωσή τους**.

Ο τύπος δεδομένων (data type) καθορίζεται αυτόματα κατά την εκτέλεση του προγράμματος.

Μπορούμε επίσης στην ίδια μεταβλητή να αναθέσουμε διαφορετικούς τύπους δεδομένων κατά την διάρκεια εκτέλεσης του προγράμματος. π.χ.:

```
var x = 42;       // Η x είναι τώρα αριθμός (number)
var x = 'Τάσος'; // Η x είναι τώρα αλφαριθμητικό (string)
var x = true;     // Η x είναι τώρα λογική (boolean)
```

Χρησιμοποιήστε τον τελεστή **typeof** για να βρείτε τον τύπο μιας μεταβλητής:

```
typeof 42;       // Επιστρέφει: number
typeof 'Τάσος'; // Επιστρέφει: string
typeof true;     // Επιστρέφει: boolean
```

Μεταβλητές στη JavaScript

Δήλωση (δημιουργία) μεταβλητών – Η λέξη-κλειδί let

```
let firstName;
firstName = 'Τάσος';

let x = 5;
```

Άλλος τρόπος δήλωσης μεταβλητών είναι με τη λέξη-κλειδί **let** (χρησιμοποιείται συχνότερα).

Μπορούμε να δηλώσουμε τις μεταβλητές ή και να εκχωρήσουμε τιμές σε αυτές κατά τη δήλωση.

Διαφορές var vs let

var	let
Οι μεταβλητές που ορίζονται με την **var** έχουν εμβέλεια (scope) **συνάρτησης**	Οι μεταβλητές που ορίζονται με την **let** έχουν εμβέλεια (scope) **μπλοκ**
Μπορούμε να δηλώσουμε ξανά μια μεταβλητή ακόμα κι αν έχει οριστεί προηγουμένως στο ίδιο πεδίο εμβέλειας.	**Δεν μπορούμε** να δηλώσουμε μια μεταβλητή πάνω από μία φορές αν ήδη την έχουμε ορίσει στο ίδιο πεδίο εμβέλειας
Επιτρέπεται το Hoisting (δήλωση μεταβλητής μετά τη χρήση της)	**Δεν επιτρέπεται** το Hoisting
Υποστηρίζεται από **ECMAScript 1** standard	Υποστηρίζεται από ECMAScript 2015 (**ES6**)

Μεταβλητές στη JavaScript

Η **τιμή μιας μεταβλητής** μπορεί να **αλλάξει** κατά την **εκτέλεση του προγράμματος**. Μπορούμε να ανατρέξουμε σε μια μεταβλητή με το όνομά της για να την **εμφανίσουμε** να την **χρησιμοποιήσουμε σε μια έκφραση** (πράξη) ή να **αλλάξουμε την αξία της**.

Παράδειγμα:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"><title>Μεταβλητές</title>
  </head>
  <body>
    <script>
      let firstName;
      firstName = 'Τάσος';
      document.write(firstName);
      document.write('<br>');
      firstName = 'Νίκος';
      document.write(firstName);
    </script>
    <p>Το παραπάνω script δηλώνει μια μεταβλητή, αναθέτει μια τιμή σε αυτή, εμφανίζει την τιμή, αλλάζει την τιμή και την εμφανίζει ξανά.</p>
  </body>
</html>
```

Σταθερές στη JavaScript

Δήλωση (δημιουργία) σταθερών

Δηλώνουμε τις σταθερές με τη λέξη-κλειδί **const**.

Μαζί με τη δήλωση πραγματοποιούμε και ανάθεση τιμής.

```
const PI = 3.14159;

const NAME = 'Tassos';

const BIRTHDATE = '12-11-1985';
```

* Υποστηρίζεται από την έκδοση ECMAScript 2015 (ES6)

Μετά τη δήλωσή τους, οι σταθερές δεν μπορούν να αλλάξουν τιμή!

Αν αναθέσουμε ξανά μια νέα τιμή σε σταθερά τότε προκαλείται μήνυμα λάθους.

Όταν ένας προγραμματιστής είναι σίγουρος ότι μια μεταβλητή δεν πρόκειται να αλλάξει ποτέ τιμή, μπορεί να τη δηλώσει με **const** για να εγγυηθεί και να κοινοποιήσει με σαφήνεια το γεγονός αυτό σε όλους όσους εμπλέκονται στη συγγραφή του προγράμματος.

Συνήθως γράφονται με κεφαλαία γράμματα.

Τύποι δεδομένων (data types)

Η JavaScript διαθέτει οκτώ τύπους δεδομένων.

Οι επτά από αυτούς ονομάζονται **primitive** (θεμελιώδεις) και είναι οι εξής:

- **number** Αριθμός (Integer & Floating Point)
- **bigint** Μεγάλος ακέραιος [$< -(2^{53}-1)$ ή $> 2^{53}-1$] - ECMAScript 2020
- **string** Αλφαριθμητικό
- **boolean** Λογικός τύπος – true / false
- **null** Άδεια τιμή, Τίποτα, Ανυπαρξία τιμής.
- **undefined** Μία μεταβλητή έχει δηλωθεί αλλά δεν της έχει ανατεθεί ακόμη τιμή. Επιστροφή κλήσης συνάρτησης που δεν επιστρέφει τιμή.
- **symbol** Μοναδικό αναγνωριστικό - ECMAScript 2015 (ES6)

Ο όγδοος τύπος είναι το:

- **object** Αντικείμενο. Χρησιμοποιείται για την αποθήκευση συλλογών δεδομένων και πολύπλοκων οντοτήτων (π.χ. Πίνακας, Ημερομηνία κλπ.)

Αριθμός (number)

Οι αριθμοί στη JavaScript αποθηκεύονται σε 64-bit μορφή (πρότυπο IEEE-754), γνωστοί ως “αριθμοί κινητής υποδιαστολής διπλής ακρίβειας”. Γράφονται:

- **Με ή χωρίς υποδιαστολή**:

```
let x = 42;
let y = 12.536;
```

- Σε διαφορετικά **συστήματα αρίθμησης**:

```
let x = 0xFF;   // 16δικό
let y = 0o77;   // 8δικό
let z = 0b111;  // 2δικό
```

- **Με επιστημονική μορφή**:

```
let megalosArithmos = 234e8;      // 23400000000
let mikrosArithmos = 234e-8;      // 0.00000234
```

Εκτός από τους κανονικούς αριθμούς, υπάρχουν οι λεγόμενες «**ειδικές αριθμητικές τιμές**» που ανήκουν σε αυτόν τον τύπο δεδομένων: **Infinity**, **-Infinity** και **NaN**.

Σημείωση: **NaN** ** 0 = 1 **Infinity** ** 0 = 1 1 / **Infinity** = 0 1 / **-Infinity** = -0

Αλφαριθμητικό (string)

Ο αλφαριθμητικός τύπος δεδομένων (ή συμβολοσειρά) σημαίνει κείμενο (text)

- Μπορούμε να χρησιμοποιήσουμε **μονά ή διπλά εισαγωγικά:**

```
let school = "4ο ΕΠΑΛ Αλεξανδρούπολης";
let address = 'Δήμητρας 3, 68100, Αλεξανδρούπολη';
```

- Χρησιμοποιούμε **μονά εισαγωγικά μέσα σε διπλά** και **το αντίστροφο:**

```
let myText1 = "Κατ' εξαίρεση";
let myText2 = 'Θέμα: "Εισαγωγή στη JavaScript"';
```

- Χρησιμοποιούμε τον **χαρακτήρα διαφυγής (\)**:

```
let myText = "Το θέμα:\"Data Type\" μπήκε κατ' εξαίρεση";
```

Αλφαριθμητικό (string)

- Χρησιμοποιούμε μονά ή/και διπλά εισαγωγικά **μέσα σε πλάγια εισαγωγικά:**

```
let myText = `Το θέμα:"Data Type" μπήκε κατ' εξαίρεση`;
```

- **Τα πλάγια εισαγωγικά** επιτρέπουν σε μια συμβολοσειρά:
  - να εκτείνεται σε πολλές γραμμές
  - και να ενσωματώνει εκφράσεις της μορφής **${…}**

(Ονομάζονται **Template Strings** – Εισάχθηκαν στην ES6)

```
let name = 'Τάσος';
let age = 35;

let str = `Καλημέρα.
Ονομάζομαι ${name}
και είμαι ${age} ετών.`;
```

Αλφαριθμητικό (string)

Μήκος αλφαριθμητικού

```
let myName = 'Τάσος';
console.log(myName.length);
```

Θα εμφανίσει: 5

Bracket Notation

```
let myName = 'Τάσος';
console.log(myName[0]);
console.log(myName[2]);
console.log(myName[myName.length - 1]);
console.log(myName[5]);
```

Θα εμφανίσει: Τ
Θα εμφανίσει: σ
Θα εμφανίσει: ς
Θα εμφανίσει: undefined

Τα strings είναι αμετάβλητα (immutable)

```
let myStr = 'Jello World';
myStr[0] = 'H';
console.log(myStr);
```

Δεν θα αλλάξει το J σε H
Θα εμφανίσει: Jello World

Λογικός τύπος (boolean)

- Ο λογικός τύπος δέχεται τιμές μόνο **true** ή **false**. Χρησιμοποιείται κυρίως σε δομές επιλογής (συνθήκες).

```
let var1 = true;
let var2 = false;
```

- Δεν πρέπει να συγχέουμε τον λογικό τύπο (boolean) με τον αλφαριθμητικό τύπο (string):

```
let var1 = true;          // Λογική τιμή (boolean)
let var2 = "true";        // Αλφαριθμητική τιμή (string)
```

Σημείωση: Σε λογικούς ελέγχους (π.χ. στη συνθήκη της if, while κλπ.) οι τιμές:

- **0**, **undefined**, **null**, **NaN** και η **κενή συμβολοσειρά “”** ερμηνεύονται ως **falsy**
- Κάθε άλλη τιμή ερμηνεύεται ως **truthy**

Μετατροπή Τύπων Δεδομένων

Ρητή μετατροπή τύπου δεδομένων (explicit conversion)

Η JavaScript διαθέτει διάφορες μεθόδους μετατροπής τύπων δεδομένων.

Μέθοδος:	**Σημασία:**
`Number(value)`	Μετατροπή σε αριθμό. Εάν η τιμή δεν μπορεί να μετατραπεί, επιστρέφεται το NaN.
`String(value)`	Μετατροπή σε συμβολοσειρά.
`Boolean(value)`	Μετατροπή σε λογική τιμή: true / false.
`BigInt(value)`	Μετατροπή σε μεγάλο ακέραιο.
`parseInt(value, radix)`	Μετατροπή σε ακέραιο αριθμό (integer) ή NaN, radix: σύστημα αρίθμησης (προαιρετικό).
`parseFloat(value, radix)`	Μετατροπή σε αριθμό κινητής υποδιαστολής ή NaN.

Ερώτηση: Τι τιμή θα έχουν οι μεταβλητές a και b μετά την εκτέλεση των εντολών;

```
let x = 10;
let y = 20;
let a = x + y;
let b = String(x) + String(y);
```

Μεταβλητές στη JavaScript

Όπως στην άλγεβρα, έτσι και στη JavaScript μπορούμε να εκτελέσουμε αριθμητικές πράξεις χρησιμοποιώντας μεταβλητές και τους αντίστοιχους τελεστές (στη συνέχεια θα αναφερθούμε εκτενέστερα στους τελεστές).

```
let a, b, c, x, y, z;

a = 20;
b = 50;
c = (a + b) * 2;

x = 10;
y = x + 5;
z = y - 10;
```

Αριθμητικοί Τελεστές

Οι αριθμητικοί τελεστές χρησιμοποιούνται για την εκτέλεση αριθμητικών πράξεων μεταξύ μεταβλητών ή/και τιμών.

Δεδομένου ότι **y=5**, ο παρακάτω πίνακας εξηγεί τους αριθμητικούς τελεστές:

Τελεστής	Περιγραφή	Παράδειγμα	Αποτέλεσμα	
+	Πρόσθεση	x = y + 2	x=7	y=5
-	Αφαίρεση	x = y - 2	x=3	y=5
*	Πολλαπλασιασμός	x = y * 2	x=10	y=5
/	Διαίρεση	x = y / 2	x=2.5	y=5
%	Υπόλοιπο ακέραιης διαίρεσης	x = y % 2	x=1	y=5
**	Ύψωση σε δύναμη	x = y ** 2	x=25	y=5
++	Αύξηση (πρόθεμα)	x = ++y	x=6	y=6
	Αύξηση (επίθεμα)	x = y++	x=5	y=6
--	Μείωση (πρόθεμα)	x = --y	x=4	y=4
	Μείωση (επίθεμα)	x = y--	x=5	y=4

θα πρέπει να δοθεί προσοχή στο εάν ο τελεστής αύξησης ή μείωσης χρησιμοποιείται ως πρόθεμα **(++y**) ή ως επίθεμα (**y++**). Εάν χρησιμοποιείται ως πρόθεμα πρώτα αυξάνεται ή μειώνεται κατά ένα η τιμή της και μετά πραγματοποιείται η ανάθεση τιμής. Ενώ όταν χρησιμοποιείται ως επίθεμα πρώτα πραγματοποιείται η ανάθεση τιμής και στο τέλος αλλάζει κατά μία μονάδα η τιμή της μεταβλητής

Πρόθεμα (Prefix Increment) vs Επίθεμα (Postfix Increment)

Παράδειγμα:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>JavaScript Παράδειγμα 9</title>
  </head>
  <body>
    <script>
      let x, y;
      x = 1;
      y = 1;
      document.write('<p>Pre Increment ++x </p>');
      document.write('x=', ++x, '<br>');
      document.write('x=', x);
      document.write('<p>Post Increment y++ </p>');
      document.write('y=', y++, '<br>');
      document.write('y=', y);
    </script>
  </body>
</html>
```

Τελεστές καταχώρησης (απόδοσης τιμής)

Οι τελεστές αυτοί χρησιμοποιούνται για την εκχώρηση τιμών σε μεταβλητές.

Δεδομένου ότι **x=10** και **y=5**, ο παρακάτω πίνακας εξηγεί τους τελεστές καταχώρησης:

Τελεστής	Παράδειγμα	Όμοιο με	Αποτέλεσμα
=	x = y		x=5
+=	x += y	x = x + y	x=15
-=	x -= y	x = x - y	x=5
*=	x *= y	x = x * y	x=50
/=	x /= y	x = x / y	x=2
%=	x %= y	x = x % y	x=0

Τελεστές επαυξημένης καταχώρησης

Τελεστές (operators)

Χρήση του τελεστή + σε αλφαριθμητικά μεγέθη (strings)

Ο τελεστής **+** χρησιμοποιείται επίσης για **συνένωση αλφαριθμητικών τιμών και μεταβλητών**.

Για να προσθέσουμε **δύο ή περισσότερες** αλφαριθμητικές μεταβλητές μαζί:

```
txt1 = 'Σήμερα είναι μια';
txt2 = 'ωραία μέρα';
txt3 = txt1 + txt2;
```

Μετά την εκτέλεση των δηλώσεων η μεταβλητή txt3 θα περιέχει την τιμή:
"**Σήμερα είναι μιαωραία μέρα**".

Για να προσθέσουμε ένα κενό χαρακτήρα μεταξύ των δύο αλφαριθμητικών:

```
txt1 = 'Σήμερα είναι μια ';
txt2 = 'ωραία μέρα';
txt3 = txt1 + txt2;
```

```
txt1 = 'Σήμερα είναι μια';
txt2 = 'ωραία μέρα';
txt3 = txt1 + ' ' + txt2;
```

Μετά την εκτέλεση των δηλώσεων η μεταβλητή txt3 θα περιέχει την τιμή:
"**Σήμερα είναι μια ωραία μέρα**".

* *Στα string μπορούμε επίσης να κάνουμε χρήση του τελεστή* **+=**

Κανόνας: Εάν προσθέσουμε αλφαριθμητικό με αριθμό, το αποτέλεσμα είναι αλφαριθμητικό.

Εμφανίστε το x όταν:

```
x = "Tassos" + 16 + 4;
```

```
x = 16 + 4 + "Tassos";
```

Τελεστές (operators)

Παραδείγματα χρήσης Τελεστών

Ο κώδικας:	Θα μπορούσε να γραφτεί:
`counter = counter + 1`	`counter++`
`counter = counter - 1`	`counter--`
`counter = counter + 20`	`counter += 20`
`message = message + "!"`	`message += "!"`
`counter = counter - 20`	`counter -= 20`
`mutli = mutli * 5`	`mutli *= 5`
`amount = amount / sum`	`amount /= sum`

Quiz

```
let cats = 5;

cats = "black";

cats = parseInt("12.5");

document.write(cats + " είναι τύπου: " + typeof cats);
```

Μετά την εκτέλεση του παραπάνω κώδικα ποιος είναι ο τύπος δεδομένων που θα εμφανίσει ο τελεστής **typeof**;

- **boolean**
- **string**
- **number**
- **integer**
- **real**

Σχεσιακοί τελεστές

Οι **σχεσιακοί τελεστές** χρησιμοποιούνται σε εκφράσεις για να καθορίσουν την ισότητα ή την ανισότητα μεταξύ μεταβλητών ή τιμών. Επιστρέφουν **true** ή **false**.

Τελεστής	Περιγραφή	Παραδείγματα			
==	ισότητα	5==5	true	5=='5'	true
===	αυστηρή ισότητα (τιμή και τύπος)	5===5	true	5==='5'	false
!=	ανισότητα (διάφορο ≠)	8!=8	false	8!='8'	false
!==	αυστηρή ανισότητα (τιμή και τύπος)	8!==8	false	8!=='8'	true
>	μεγαλύτερο	5 > 8	false	8 > 5	true
<	μικρότερο	5 < 8	true	5 < 5	false
>=	μεγαλύτερο ή ίσο	5 >= 8	false	5 >= 5	true
<=	μικρότερο ή ίσο	5 <= 8	true	8 <= 8	true

Ειδικές περιπτώσεις

undefined === undefined **true**
null === null **true**
Infinity === Infinity **true**
Infinity === -Infinity **false**

NaN == NaN **false**
NaN === NaN **false**
undefined == null **true**
undefined === null **false**

Λογικοί τελεστές

Οι **λογικοί τελεστές** χρησιμοποιούνται συνήθως σε συνδυασμό με τους σχεσιακούς τελεστές σε λογικές εκφράσεις (συνθήκες) για να καθορίσουν αν κάποια λογική πρόταση είναι **αληθής** (**true**) ή **ψευδής** (**false**).

Ο παρακάτω πίνακας εξηγεί τους λογικούς τελεστές:

Τελεστής	Περιγραφή	Παράδειγμα
&&	Λογικό ΚΑΙ (**AND**)	x=6 , y=3 (x < 10 && y > 1) -> **true**
||	Λογικό Ή (**OR**)	x=6, y=3 (x==5 || y==5) -> **false**
!	Άρνηση – Λογικό ΟΧΙ (**NOT**)	x=6, y=3 !(x==y) -> **true**

Λογικοί τελεστές

Λογικός τελεστής OR ||

```
A || B
```

- Αν το Α έχει **truthy** τιμή τότε επιστρέφεται το ίδιο το Α.
- Αν το Α έχει **falsy** τιμή, τότε επιστρέφεται το Β.

Παράδειγμα:

```
let firstName = null;
let message = firstName || "Άκυρο";
// Θα εμφανίσει Άκυρο
console.log(message);

firstName = "Τάσος";
message = firstName || "Άκυρο";
// Θα εμφανίσει Τάσος
console.log(message);
```

Λογικός τελεστής AND &&

```
A && B
```

- Αν το Α έχει **falsy** τιμή τότε επιστρέφεται το ίδιο το Α.
- Αν το Α έχει **truthy** τιμή τότε επιστρέφεται το Β.

Παράδειγμα:

```
let firstName;
let message = firstName && "Άκυρο";
// Θα εμφανίσει undefined
console.log(message);

firstName = "Τάσος";
message = firstName && "Άκυρο";
// Θα εμφανίσει Άκυρο
console.log(message);
```

Conditional operator (?)

Τελεστής υπό συνθήκη - Conditional operator (?)

Η JavaScript περιέχει τον τελεστή υπό συνθήκη ο οποίος αναθέτει τιμή σε μια μεταβλητή ανάλογα με το αποτέλεσμα κάποιας συνθήκης. Αποκαλείται και τριαδικός τελεστής (ternary operator).

Σύνταξη:

```
ΟνομαΜεταβλητής = (Συνθήκη) ? Τιμή1 : Τιμή2
```

Παράδειγμα:

```
greet = (visitor=="ΑΝΔΡΑΣ") ? 'Αγαπητέ κύριε' : 'Αγαπητή κυρία';
```

Αν η μεταβλητή **visitor** έχει τιμή "ΑΝΔΡΑΣ", τότε η μεταβλητή **greeting** θα πάρει τιμή "Αγαπητέ κύριε" αλλιώς θα πάρει τιμή "Αγαπητή κυρία".

Σημείωση: Μπορούμε να χρησιμοποιήσουμε εφωλευμένους τελεστές υπό συνθήκη (**?**) για να επιστραφεί μια τιμή που εξαρτάται από περισσότερες από μία συνθήκες. Π.χ.:

```
msg = (age < 3) ? 'Baby' : (age < 18) ? 'Young' : (age < 100) ? 'Adult' : 'Elder!';
```

Conditional operator (? :)

Παράδειγμα:

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Conditional Operator</title>
    <script>
        function greet() {
            let greeting, gender = document.myForm.gender;
            greeting = (gender.value == 'male') ? 'Αγαπητέ κύριε' : 'Αγαπητή κυρία';
            document.getElementById('message').innerHTML = greeting;
        }
    </script>
</head>

<body>
    <form name="myForm" onChange="greet()">
        <input type="radio" name="gender" value="male">Άνδρας
        <input type="radio" name="gender" value="female">Γυναίκα
    </form>
    Χαιρετισμός: <span id="message"></span>
</body>
</html>
```

Προτεραιότητα τελεστών

Προτεραιότητα τελεστών είναι οι κανόνες αποσαφήνισης της σειράς με την οποία υπολογίζονται οι εκφράσεις και προκύπτουν οι τιμές τους.

Για παράδειγμα, στα μαθηματικά και στις περισσότερες γλώσσες προγραμματισμού ο πολλαπλασιασμός υπολογίζεται πριν την πρόσθεση, αφού ο πολλαπλασιασμός και η διαίρεση έχουν υψηλότερη προτεραιότητα σε σχέση με την πρόσθεση και την αφαίρεση.

Οι περισσότερες γλώσσες προγραμματισμού χρησιμοποιούν την πρότυπη προτεραιότητα των μαθηματικών για τους μαθηματικούς τελεστές, και προσθέτουν σε αυτήν τις δικές τους συμβάσεις για τους επιπλέον τελεστές που προσφέρει η κάθε γλώσσα. Συνήθως ισχύει:

- **Αριθμητικοί Τελεστές**: Συνήθης μαθηματική προτεραιότητα.
- **Σχεσιακοί Τελεστές**: Μικρότερη προτεραιότητα από τους αριθμητικούς τελεστές.
- **Λογικοί Τελεστές**: Μικρότερη προτεραιότητα από τους σχεσιακούς τελεστές [εξαιρείται το not (!)].

Μπορούμε να αλλάξουμε την προτεραιότητα **χρησιμοποιώντας παρενθέσεις**.

Πλήρης πίνακας με την προτεραιότητα τελεστών της JavaScript:

developer.mozilla.org

Προτεραιότητα τελεστών

Προτεραιότητα	Τύπος τελεστή	Κατεύθυνση	Τελεστής
(Υψηλότερη) 12	Grouping	n/a	(…)
11	Postfix Increment Postfix Decrement	n/a n/a	… ++ … --
10	Logical NOT Prefix Increment Prefix Decrement typeof	n/a n/a n/a n/a	! … ++ … -- … typeof …
9	Exponentiation (**)	right-to-left	… ** …
8	Multiplication (*) Division (/) Remainder (%)	left-to-right left-to-right left-to-right	… * … … / … … % …
7	Addition (+) Subtraction	left-to-right left-to-right	… + … … - …
6	Less Than Less Than Or Equal Greater Than Greater Than Or Equal	left-to-right left-to-right left-to-right left-to-right	… < … … <= … … > … … >= …
5	Equality Inequality Strict Equality	left-to-right left-to-right left-to-right	… == … … != … … === …
4	Logical AND	left-to-right	… && …
3	Logical OR	left-to-right	… || …
2	Assignment	right-to-left	… = …, … += …, … -= …, … *= …, … /= …, … %= …
	Conditional	right-to-left	… ? … : …
(Χαμηλότερη) 1	Comma	left-to-right	… , …

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>JavaScript Παράδειγμα 6</title>
    <script type="text/javascript">
      function calculate() {
        var no1 = parseFloat(document.myform.number1.value);
        var no2 = parseFloat(document.myform.number2.value);
        var no3 = no1 + no2;
        document.getElementById('result').innerHTML = no3;
      }
    </script>
  </head>
  <body>
    <form name="myform" action="#">
      <input type="text" name="number1"> +
      <input type="text" name="number2"> =
      <span id="result"></span>
      <p><input type="button" value="Υπολόγισε"
    </form>
```

ΤΕΛΟΣ Β' ΜΕΡΟΥΣ

Ευχαριστώ για την προσοχή σας

ΜΕΡΟΣ Γ’ - Περιεχόμενα

- Συνάρτηση (function)
- Εμβέλεια μεταβλητών (scope)
- Γεγονότα ή συμβάντα (events)
- JavaScript popup boxes (αναδυόμενα πλαίσια διαλόγου)
- Δομή επιλογής – Συνθήκη (condition)
- Δομή επανάληψης (loop)

με πολλά απλά παραδείγματα

Συνάρτηση (function)

Για να εμποδίσουμε την εκτέλεση ενός script κατά την φόρτωση της σελίδας, μπορούμε να βάλουμε το script σε μια **συνάρτηση** (function).

Μια **συνάρτηση** περιέχει κώδικα που θα εκτελεστεί από ένα **γεγονός** (event) ή από την **κλήση** της συνάρτησης.

Μπορούμε να καλέσουμε μια **συνάρτηση** από οπουδήποτε μέσα σε μια ιστοσελίδα. Η **συνάρτηση** μπορεί να βρίσκεται και σε **εξωτερικό .js αρχείο** το οποίο έχουμε ενσωματώσει στη σελίδα μας.

Οι **συναρτήσεις** μπορούν να οριστούν τόσο στην ενότητα <head> όσο και στην ενότητα <body> μιας ιστοσελίδας. Ωστόσο, είναι μια καλή προγραμματιστική τακτική οι συναρτήσεις να ορίζονται στην ενότητα <head>.

Οι **συναρτήσεις** μπορούν να δέχονται παραμέτρους (**είσοδος**) και να επιστρέφουν τιμή (**έξοδος**).

Συνάρτηση (function)

Ορισμός Συνάρτησης

Σύνταξη:

```
function functionName(param1, param2, ..., paramX) {
    Εντολές προς εκτέλεση
}
```

Το ***functionName*** είναι το όνομα της συνάρτησης. Οι παράμετροι ***param1***, ***param2***, κλπ. είναι μεταβλητές ή τιμές που θέλουμε να περάσουμε στη συνάρτηση.

Τα άγκιστρα **{** και **}** καθορίζουν το σώμα (την αρχή και το τέλος) της συνάρτησης.

Σημείωση 1: Μια συνάρτηση χωρίς παραμέτρους **πρέπει να περιλαμβάνει τις παρενθέσεις ()** μετά το όνομα της συνάρτησης.

Σημείωση 2: Μην ξεχνάτε την ευαισθησία στα πεζά / κεφαλαία της JavaScript. Η λέξη ***function*** πρέπει να γράφεται με πεζά γράμματα. Επίσης πρέπει να καλούμε μια συνάρτηση με το ίδιο ακριβώς όνομα που χρησιμοποιήσαμε κατά τον ορισμό της (ακριβής χρήση πεζών / κεφαλαίων γραμμάτων).

Παράμετροι και Ορίσματα

- Μία συνάρτηση μπορεί να έχει καμία ή περισσότερες παραμέτρους. Κατά την κλήση, μπορούμε να διοχετεύσουμε λιγότερα η περισσότερα ορίσματα από τις παραμέτρους, **χωρίς να προκαλείται συντακτικό λάθος**.

- **Προεπιλεγμένες** (**default**) τιμές παραμέτρων (param=value):
  - Τίθεται η τιμή value στην param, αν δεν έχει καθοριστεί τιμή, ή έχει διοχετευτεί ως όρισμα: undefined.
  - Η default τιμή κάθε παραμέτρου εάν αυτή δεν έχει καθοριστεί είναι undefined.

- **rest operator** (...name) (τρεις τελείες ακολουθούμενες από ένα όνομα):
  - Μπαίνει το πολύ μία φορά, ως τελευταία παράμετρος και δημιουργεί ένα πίνακα με τα υπόλοιπα ορίσματα.

- Η έμμεση παράμετρος **arguments**:
  - Είναι ένα αντικείμενο που περιέχει όλα τα ορίσματα.
  - Υποστηρίζει τις arguments.length, arguments[i] κ.α.

```
function demo1(x, y, z = 5) {
    console.log(x, y, z)
}

demo1();  // undefined undefined 5
demo1(3);  // 3 undefined 5
demo1(1, 3);   // 1 3 5
demo1(1, 3, 8);  // 1 3 8
```

```
function demo2(first, ...rest) {
    console.log(first, rest);
}

demo2(); // undefined []
demo2(1); // 1 []
demo2(1, 2, 3); // 1 [2, 3]
demo2(1, 2, 3, 4); // 1 [2, 3, 4]
```

```
function demo3() {
    let len = arguments.length;
    for (let i = 0; i < len; i++)
        console.log(arguments[i]);
}

demo3("Nick", "Mary", "Joe");
demo3(1, 5, 10);
```

Συνάρτηση (function)

Χρήση ενός **γεγονότος** (event) για την **κλήση** μιας συνάρτησης.

Παράδειγμα:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>JavaScript function example</title>
    <script>
      function message() {
        alert('Το πάτησες τελικά. Είσαι τολμηρός!');
      }
    </script>
</head>

<body>
  <form>
    <input type="button" value="Πάτα με αν τολμάς" onclick="message()">
  </form>
</body>
</html>
```

Συνάρτηση (function)

Επιστροφή τιμής από συνάρτηση (Η εντολή: **return**)

Η εντολή **return** χρησιμοποιείται για να καθορίσει την τιμή που θα **επιστρέψει** η συνάρτηση.

Συνεπώς οι συναρτήσεις που επιστρέφουν τιμή πρέπει να χρησιμοποιούν την εντολή **return**.

Όταν δεν υπάρχει εντολή **return**, η συνάρτηση επιστρέφει την τιμή **undefined**.

Παράδειγμα:
Επιστρέφει το γινόμενο δύο αριθμών (x και y)

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>JavaScript function example</title>
    <script>
      function ginomeno(x,y) {
        return x*y; }
    </script>
</head>
<body>
    <script>
     document.write('Γινόμενο: ' + ginomeno(4,3));
    </script>
</body>
</html>
```

Όταν καλούμε μια συνάρτηση, η τιμή που μας επιστρέφει με την εντολή **return**, είναι απλά το αποτέλεσμα εκτέλεσης της συνάρτησης.

Η τιμή αυτή μπορεί να χρησιμοποιηθεί όπως και κάθε άλλη τιμή στη JavaScript!

Π.χ. να εκχωρηθεί σε μεταβλητή, να πάρει μέρος σε μια παράσταση κλπ.

Συνάρτηση (function)

Don't Repeat Yourself (D.R.Y) - Μην επαναλαμβάνεσαι

Ο κανόνας **D.R.Y.** είναι πολύ σημαντικός στον προγραμματισμό.

Κάθε φορά που θα βρεθείτε να πληκτρολογείτε το ίδιο πράγμα, τροποποιώντας μόνο ένα "μικρό μέρος" του, μπορείτε πιθανώς να χρησιμοποιήσετε **συνάρτηση**.

- Το "μικρό μέρος" που θα τροποποιείτε είναι η παράμετρος (-οι).
- Το τμήμα που επαναλαμβάνεται είναι ο κώδικας στο κυρίως σώμα της συνάρτησης που γράφεται μεταξύ { }.

Short-Circuit Evaluation

```
function f1() {
  ...;
  return false;
}

function f2() {
  ...;
  return true;
}
let x = f1() && f2();
```

Σημαντικό: Σε μία λογική πρόταση π.χ.: **`x = f1() && f2()`** αν το πρώτο μέρος **`f1()`** είναι **false**, η JavaScript δεν ασχολείται με την **`f2()`** και δίνει κατ' ευθείαν αποτέλεσμα **false**.

Αυτό σημαίνει ότι η συνάρτηση f2(), δε θα εκτελεστεί!

```
function f1() {
  ...;
  return true;
}

function f2() {
  ...;
  return false;
}
let x = f1() || f2();
```

Σημαντικό: Σε μία λογική πρόταση π.χ.: **`x = f1() || f2()`** αν το πρώτο μέρος **`f1()`** είναι **true**, η JavaScript δεν ασχολείται με την **`f2()`** και δίνει κατ' ευθείαν αποτέλεσμα **true**.

Αυτό σημαίνει ότι η συνάρτηση f2(), δε θα εκτελεστεί!

Εμβέλεια μεταβλητών (scope)

Τοπικές μεταβλητές (Local variables)

Οι μεταβλητές που δηλώνονται **μέσα σε μια συνάρτηση** (**function**) γίνονται **τοπικές** και μπορεί κανείς να έχει πρόσβαση σε αυτές μόνο εντός της συνάρτησης. Οι μεταβλητές αυτές έχουν **τοπική εμβέλεια – local scope**.

Μπορούμε να έχουμε **τοπικές μεταβλητές** με το ίδιο όνομα σε διαφορετικές **συναρτήσεις**, επειδή αυτές αναγνωρίζονται μόνο από τη **συνάρτηση** στην οποία έχουν δηλωθεί.

Οι **τοπικές μεταβλητές** καταστρέφονται όταν βγούμε από τη συνάρτηση.

Καθολικές μεταβλητές (Global variables)

Οι μεταβλητές που δηλώνονται **έξω από μια συνάρτηση** γίνονται **καθολικές**, και όλα τα scripts και οι συναρτήσεις της ιστοσελίδας, έχουν πρόσβαση σε αυτές. Οι μεταβλητές αυτές έχουν **καθολική εμβέλεια – global scope**.

Οι **καθολικές μεταβλητές** καταστρέφονται όταν κλείσουμε την ιστοσελίδα.

Εμβέλεια μεταβλητών (scope)

Παράδειγμα 1. Τοπικές & Καθολικές μεταβλητές με **var**

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <script>
      function showX() {
        var x = 10;
        document.write("Στη συνάρτηση: " + x);
      }
      var x = 200;
      document.write("Στο κύριο μέρος: " + x);
      showX();
      document.write("Στο κύριο μέρος: " + x);
    </script>
  </body>
</html>
```

Μπορούμε να έχουμε **Τοπικές (local)** και **Καθολικές (global)** μεταβλητές **με το ίδιο όνομα**.

Η JavaScript μπορεί να τις ξεχωρίσει αφού οι τοπικές μεταβλητές έχουν προτεραιότητα έναντι των καθολικών μεταβλητών.

Για αποφυγή σύγχυσης και προγραμματιστικών λαθών, η πρακτική αυτή δεν προτείνεται.

Εμβέλεια μεταβλητών (scope)

Παράδειγμα 2. Καθολική μεταβλητή χωρίς **var**

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
   <script>
      function showX() {
        document.write("Στη συνάρτηση: " + x);
      }
      x = 200; // Χωρίς χρήση var: Καθολική
      document.write("Στο κύριο μέρος: " + x);
      showX();
      document.write("Στο κύριο μέρος: " + x);
   </script>
  </body>
</html>
```

Παρατήρηση:

Αν δηλώσουμε μια μεταβλητή, χωρίς τη λέξη-κλειδί "**var**", η JavaScript ψάχνει «προς τα πάνω» στα επίπεδα δράσης μεταβλητών (variable scope) και αν δε βρεθεί τότε η μεταβλητή γίνεται **καθολική (global)**.

Εμβέλεια μεταβλητών (scope)

Παράδειγμα 3. Καθολική μεταβλητή χωρίς **var**

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
   <script>
    function showX() {
        x = 200; // Χωρίς χρήση var Καθολική
        document.write("Στη συνάρτηση: " + x);
      }
      showX();
      document.write("Στο κύριο μέρος: " + x);
   </script>
  </body>
</html>
```

Παρατήρηση:

Χωρίς τη λέξη-κλειδί **var**, η μεταβλητή γίνεται **καθολική *ακόμη και αν πάρει τιμή εντός της συνάρτησης***

Ερώτηση:

Εντός της συνάρτησης αντικαθιστούμε το:

`x = 200;` με

`var x = 200;`

Ποιες οι συνέπειες;

Απάντηση: Η **x** γίνεται τοπική μεταβλητή και δεν υπάρχει εκτός της συνάρτησης. Η συνάρτηση εκτελείται κανονικά, αλλά στο κύριο μέρος η **x** χρησιμοποιείται χωρίς να έχει δηλωθεί. Άρα παράγει σφάλμα.

Εμβέλεια μεταβλητών (scope)

Παράδειγμα 4. Με χρήση της **let**

```
let a = 100;
if (true) {
    let b = 200;
    console.log(a);  // Εμφανίζει 100
    console.log(b);  // Εμφανίζει200
}
console.log(a);  // Εμφανίζει 100
console.log(b);  // Error: b is not
defined
```

Δηλώνοντας μεταβλητή με την λέξη-κλειδί **let** η μεταβλητή έχει πλέον **εμβέλεια μπλοκ (block scope)** και είναι προσβάσιμη μόνο σε αυτό το μπλοκ (και σε ενδεχόμενα μπλοκ που αυτό περιέχει) από το σημείο της δήλωσης της και μετά.

- Αποτελεί καλή πρακτική η δήλωση μεταβλητών να γίνεται με τη λέξη-κλειδί **let**
- Η χρήση της **var** καλό είναι να αποφεύγεται αφού στη **var** δεν υπάρχει η έννοια του **block scope** αλλά μόνο του **function scope**

Γεγονότα (events)

Με τη χρήση της JavaScript, έχουμε τη δυνατότητα να δημιουργήσουμε δυναμικές ιστοσελίδες. Τα **γεγονότα ή συμβάντα** (events), είναι ενέργειες που μπορούν να ανιχνευθούν από τη JavaScript. Το γεγονός λοιπόν είναι όταν κάτι συμβαίνει.

Κάθε στοιχείο σε μια ιστοσελίδα έχει ορισμένα γεγονότα τα οποία μπορεί θέσει σε ενέργεια και να ανιχνευτούν από τη JavaScript. Για παράδειγμα, μπορούμε να χρησιμοποιήσουμε το **onClick** event ενός πλήκτρου (button) για να κληθεί μια **συνάρτηση** όταν ένας χρήστης κάνει κλικ στο πλήκτρο.

Τα γεγονότα ορίζονται στις ετικέτες HTML.

Παραδείγματα γεγονότων:

- Κλικ του ποντικιού (**onClick**).
- Φόρτωση μιας ιστοσελίδας ή μιας εικόνας (**onLoad**).
- Πέρασμα του ποντικιού πάνω από ένα συγκεκριμένο στοιχείο (**onMouseOver**).
- Επιλογή ενός πεδίου σε μια φόρμα HTML (**onFocus**).
- Υποβολή μιας φόρμας HTML (**onSubmit**).
- Πάτημα ενός πλήκτρου στο πληκτρολόγιο (**onKeyDown**).

Σημείωση: Τα **γεγονότα** συνήθως χρησιμοποιούνται σε συνδυασμό με τις **συναρτήσεις**, καθώς η **συνάρτηση** δεν πρόκειται να εκτελεστεί πριν από το **γεγονός**.

Γεγονότα (events)

Για μια πλήρης αναφορά των γεγονότων (events) που αναγνωρίζονται από τη JavaScript με αναλυτικά παραδείγματα, επισκεφθείτε τον παρακάτω σύνδεσμο:

http://www.w3schools.com/jsref/dom_obj_event.asp

JavaScript popup boxes

Η JavaScript έχει τριών ειδών **αναδυόμενα πλαίσια διαλόγου** (popup boxes): **Alert box** (προειδοποίηση), **Confirm box** (επιβεβαίωση), και **Prompt box** (προτροπή).

Alert box (προειδοποίηση)

Χρησιμοποιείται συνήθως για να δώσουμε μια πληροφορία στο χρήστη και να είμαστε σίγουροι ότι την διάβασε.

Όταν εμφανίζεται το **alert box**, ο χρήστης πρέπει να πατήσει OK για να συνεχίσει.

Παράδειγμα:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>JavaScript Alert Box</title>
    <script>
     function ShowAlert()
      { alert('Αυτό είναι το alert box') }
    </script>
  </head>
  <body>
   <form>
    <input type="button" value="Πάτα το"
onclick="ShowAlert()">
   </form>
  </body>
</html>
```

JavaScript popup boxes

Confirm box (επιβεβαίωση)

Χρησιμοποιείται συνήθως για να επιβεβαιώσει ή να αποδεχθεί κάτι ο χρήστης.

Όταν εμφανίζεται το **confirm box**, ο χρήστης πρέπει να πατήσει "OK" ή "ΑΚΥΡΟ" για να συνεχίσει.

Πατώντας "OK" το confirm box επιστρέφει true (αληθής), διαφορετικά επιστρέφει false (ψευδής).

Παράδειγμα:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>JavaScript Confirm Box</title>
    <script>
     function show_confirm()
      {
       let r=confirm('Είσαι σίγουρος;');
       if (r==true)
        { alert('Πάτησες OK!');  }
       else
        { alert('Πάτησες Άκυρο!');  }
       }
    </script>
  </head>
  <body>
   <form>
    <input type="button" value="Εμφάνισε
Επιβεβαίωση" onclick="show_confirm()">
   </form>
  </body>
</html>
```

JavaScript popup boxes

Prompt box (προτροπή)

Χρησιμοποιείται συνήθως για να δώσει ο χρήστης μια τιμή.

Όταν εμφανίζεται το **prompt box**, ο χρήστης πρέπει να δώσει μια τιμή και να πατήσει "OK" ή "ΑΚΥΡΟ" για να συνεχίσει.

Πατώντας "OK" το prompt box επιστρέφει την τιμή που έδωσε ο χρήστης, διαφορετικά επιστρέφει μηδενική τιμή (null).

Παράδειγμα:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>JavaScript Prompt Box</title>
    <script>
     function show_prompt()
      {
       let name=prompt('Γράψε το όνομά σου:');
       if (name!=null && name!='')
        {
        document.write('Γεια σου. Ονομάζεσαι ' + name);
        }
      }
    </script>
  </head>
  <body>
   <form>
    <input type="button" value="Εμφάνισε προτροπή"
onclick="show_prompt()">
   </form>
  </body>
</html>
```

Δομή επιλογής (if)

Οι εντολές επιλογής χρησιμοποιούνται για την εκτέλεση διαφόρων ενεργειών που βασίζονται σε διαφορετικές συνθήκες.

Η δομή επιλογής επιτρέπει στο προγραμματιστή να καθορίσει μια ενέργεια που θα εκτελεστεί όταν η συνθήκη είναι αληθής και μια διαφορετική ενέργεια που θα εκτελεστεί όταν η συνθήκη θα είναι ψευδής.

Η εντολή if

Χρησιμοποιείται για να εκτελεστεί μία ή περισσότερες εντολές, μόνο εάν μια συγκεκριμένη συνθήκη είναι αληθής.

Σύνταξη:

```
if (Συνθήκη) {
    Εντολές που εκτελούνται
    εάν η συνθήκη είναι αληθής
}
```

Σημείωση: Τα άγκιστρα είναι **απαραίτητα** όταν υπάρχουν περισσότερες από μία εντολές προς εκτέλεση. Εάν πρόκειται για μία μόνο εντολή τότε είναι **προαιρετικά**.

Παράδειγμα:

```
<script>
  // Γράψε Καλησπέρα, εάν η ώρα
  // είναι μεγαλύτερη του 12

  let d=new Date();
  let time=d.getHours();

  if (time > 12) {
    document.write('<b>Καλησπέρα</b>');
  }
</script>
```

Δομή επιλογής (if...else)

Η εντολή if … else ...

Χρησιμοποιείται για να εκτελεστεί ένα τμήμα εντολών, εάν μια συγκεκριμένη συνθήκη είναι αληθής ή ένα διαφορετικό τμήμα εντολών, εάν η συνθήκη είναι ψευδής.

Σύνταξη:

```
if (Συνθήκη)  {
    Εντολές που εκτελούνται
    εάν η συνθήκη είναι αληθής
} else {
    Εντολές που εκτελούνται
    εάν η συνθήκη είναι ψευδής
}
```

Παράδειγμα:

```
<script>
  // Εάν η ώρα είναι μικρότερη του 12
  // γράψε Καλημέρα αλλιώς γράψε Καλησπέρα

  var d=new Date();
  var time=d.getHours();

  if (time < 12) {
    document.write('<b>Καλημέρα</b>');
  } else {
    document.write('<b>Καλησπέρα</b>');
  }
</script>
```

Δομή επιλογής (if...else if...else)

Η εντολή if … else if ... else …

Για επιλογή με βάση περισσότερες (χωρίς περιορισμό) συνθήκες.

Σύνταξη:

```
if (Συνθήκη1)
  {
   Εντολές που εκτελούνται
   εάν η συνθήκη1 είναι αληθής
  }
else if (Συνθήκη2)
  {
   Εντολές που εκτελούνται
   εάν η συνθήκη2 είναι αληθής
  }
else
  {
   Εντολές που εκτελούνται
   εάν η συνθήκη1 και η
   συνθήκη2 είναι ψευδής
  }
```

Παράδειγμα:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>JavaScript if...else if...else</title>

    <script>
      function FindMax() {
        let no1 = parseFloat(document.myForm.number1.value);
        let no2 = parseFloat(document.myForm.number2.value);
        let result;
        if (no1 > no2) {
          result = 'Μεγαλύτερος αριθμός είναι ο ' + no1; }
        else if (no1 < no2) {
          result = 'Μεγαλύτερος αριθμός είναι ο ' + no2; }
        else {
          result = 'Οι αριθμοί είναι ίσοι'; }
        document.getElementById('max').innerHTML = result;
      }
    </script>

  </head>
  <body>
    <form name="myForm" action="#">
      <input type="text" name="number1">
      <input type="text" name="number2"> <br>
      <input type="button" value="Εύρεση" onClick="FindMax()">
      <p id="max"></p>
    </form>
  </body>
</html>
```

Quiz

```
let userName = "Τάσος";
if (userName == "Νίκος")
    document.write("Ωραίο όνομα!");
else if (userName != "Γιώργος")
    document.write("Υπάρχουν και καλύτερα ονόματα!");
else if (userName == "Τάσος")
    document.write("Αυτό μάλιστα!");
else
    document.write("Σιγά το όνομα!");
```

Μετά την εκτέλεση του παραπάνω κώδικα ποιο μήνυμα θα εμφανιστεί;

- **Ωραίο όνομα!**
- **Υπάρχουν και καλύτερα ονόματα!**
- **Αυτό μάλιστα!**
- **Σιγά το όνομα!**

Δομή επιλογής (switch)

Η εντολή **switch** μας παρέχει δυνατότητα πολλαπλής επιλογής. Επιλέγει ένα από τα πολλά μπλοκ κώδικα προς εκτέλεση, ανάλογα με την τιμή της μεταβλητής ελέγχου (n).

Σημείωση: Ο έλεγχος πραγματοποιείται με αυστηρή ισότητα (===).

Σύνταξη:

```
switch(n)
{
case 1:
  μπλοκ εντολών 1 (n===1)
  break;
case 2:
  μπλοκ εντολών 2 (n===2)
  break;
case 3:
  μπλοκ εντολών 3 (n===3)
  break;
default:
  κώδικας που εκτελείται εάν
  το n είναι διαφορετικό από
  1, 2 ή 3
}
```

Παράδειγμα 1:

```
<script>
//Διαφορετικός χαιρετισμός με βάση την ημέρα της
εβδομάδας. Κυριακή=0, Δευτέρα=1,… Σάββατο=6.
let d=new Date();
let theDay=d.getDay();

switch (theDay) {
 case 5:
   document.write('Επιτέλους Παρασκευή');
   break;
 case 6:
   document.write('Σούπερ Σάββατο!!!');
   break;
 case 0:
   document.write('Αύριο Δευτέρα :(');
   break;
 default:
   document.write('Πότε θα έρθει το Σ/Κ;'); }
</script>
```

Δομή επιλογής (switch)

Η εντολή switch

```
<script>
  let flavor = prompt("Ποια γεύση σου αρέσει?");
  switch(flavor) {
    case "Βανίλια":
    case "Σοκολάτα":
      document.write("Κλασσική γεύση!");
      break;
    case "Μόκα":
    case "Καραμέλα":
      document.write("Δεν τρελαίνομαι!");
      break;
    case "Φράουλα":
      document.write ("Το αγαπημένο μου!");
      break;
    default:
      document.write("Δε μου αρέσει καθόλου!");
  }
</script>
```

Παράδειγμα 2:
Χρήση της break.

Κάθε case δεν είναι απαραίτητο να τελειώνει με break.

Εξαρτάται από τη λογική που θέλουμε να υλοποιήσουμε στον κώδικα.

Δομή επιλογής (switch)

Η εντολή switch

```
<script>
  let num = prompt("Δώσε αριθμό (1-5)");
  switch(parseInt(num)) {
    case 5:
      document.write("Πέντε ");
    case 4:
      document.write("Τέσσερα ");
    case 3:
      document.write("Τρία ");
    case 2:
      document.write("Δύο ");
    case 1:
      document.write("Ένα");
      break;
    default:
      document.write("Λάθος αριθμός");
  }
</script>
```

Παράδειγμα 3:
Χρήση της break.

Κάθε case δεν είναι απαραίτητο να τελειώνει με break.

Εξαρτάται από τη λογική που θέλουμε να υλοποιήσουμε στον κώδικα.

Δομή επανάληψης (loop)

Η **δομή επανάληψης** (βρόγχος – loop) χρησιμοποιείται για την εκτέλεση ενός μπλοκ εντολών για συγκεκριμένο αριθμό επαναλήψεων ή όσο μια συγκεκριμένη συνθήκη είναι αληθής.

Συχνά, όταν γράφουμε ένα πρόγραμμα, θέλουμε το ίδιο μπλοκ εντολών για να εκτελεστεί ξανά και ξανά. Για την εκτέλεση μια τέτοιας εργασίας, αντί να γράφουμε τον ίδιο κώδικα πολλές φορές, χρησιμοποιούμε βρόχους.

Στη JavaScript, υπάρχουν δύο διαφορετικά είδη βρόχων:

- **for** - Για επανάληψη του κώδικα με συγκεκριμένο αριθμό επαναλήψεων.
- **while** - Για επανάληψη του κώδικα, όσο μια συγκεκριμένη συνθήκη είναι αληθής.

Η εντολή for

Η εντολή **for** χρησιμοποιείται όταν γνωρίζουμε εκ των προτέρων πόσες φορές θα πρέπει να εκτελεστεί το script.

Σύνταξη:

```
for (μεταβλητή=τιμή_έναρξης; μεταβλητή<=τιμή_λήξης; μεταβλητή=μεταβλητή+βήμα)
 {
  Εντολές προς εκτέλεση
 }
```

Δομή επανάληψης (for)

Η εντολή for

Το παρακάτω παράδειγμα ορίζει ένα βρόγχο που ξεκινάει με **i = 0**. Ο βρόχος θα εκτελείται όσο το **i** είναι μικρότερο ή ίσο με **5**. Το **i** αυξάνεται κάθε φορά κατά **1**.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"><title>Demo Script</title>
  </head>
  <body>
  <script>
    let i;
    for (i=0;i<=5;i++)
    {
      document.write('Ο αριθμός είναι: ' + i + '<br>');
    }
  </script>
  </body>
</html>
```

Εναλλακτικά το **i** μπορεί να μειώνεται: `for (i=5;i>=0;i--)`.

Επίσης μπορεί να αυξάνεται/μειώνεται με διαφορετικό βήμα από 1. ***Άσκηση:*** *Προπαίδεια.*

Δομή επανάληψης (while)

Η εντολή while

Η **while** χρησιμοποιείται για επανάληψη, **όσο μια συγκεκριμένη συνθήκη είναι αληθής**.

```
while (Συνθήκη)
   {
    Εντολές προς εκτέλεση
   }
```

Παρατήρηση: Πρώτα γίνεται ο έλεγχος της συνθήκης και μετά εκτελούνται οι εντολές. Σε περίπτωση που **η συνθήκη είναι ψευδής** από την αρχή, **δεν εκτελείται καμία φορά** ο βρόγχος.

Παράδειγμα 1:

```
<script>
 let i=0;
 while (i<=5)
  {
   document.write('Αριθμός: ' + i);
   document.write('<br>');
   i++;
  }
</script>
```

Παράδειγμα 2:

```
<script>
 let answer, stop;
 stop = false;
 while (!stop) {
   answer=prompt("Να κάνουμε μάθημα;");
   if (answer=="ναι" || answer=="NAI")
        stop = true;
 }
</script>
```

Quiz

```
<script>
let euro, kostos, metritis;
euro = parseInt(prompt("Πόσα χρήματα έχεις;"));
kostos = parseInt(prompt("Πόσο κοστίζει το κάθε τεμάχιο;"));
metritis = 0;
while ( ??? ) {
  euro = euro - kostos;
  metritis++;
}
document.write("Αγόρασες " + metritis +
               " τεμάχια. Σου έμειναν " + euro + "€");
</script>
```

Ποια είναι η σωστή συνθήκη που πρέπει να μπει στη θέση των ???;

- `euro > 0`
- `euro >= kostos` ⇐
- `euro > kostos`
- `euro == kostos`

Δομή επανάληψης (do...while)

Η εντολή do…while

Η **do…while** χρησιμοποιείται για επανάληψη, όσο μια συγκεκριμένη συνθήκη είναι αληθής.

```
do
   {
    Εντολές προς εκτέλεση
   }
while (Συνθήκη)
```

Παρατήρηση: Πρώτα εκτελούνται οι εντολές και μετά γίνεται ο έλεγχος της συνθήκης. **Ο βρόγχος εκτελείται τουλάχιστον μια φορά**.

Παράδειγμα:

```
<script>
 let i=0;
 do
  {
   document.write('Ο αριθμός είναι: ' + i);
   document.write('<br>');
   i++;
  }
 while (i<=5)
</script>
```

Διακοπή βρόγχων

Η εντολή break

Η **break** θα σταματήσει το βρόγχο και θα συνεχίσει την εκτέλεση των εντολών μετά από αυτόν (εάν υπάρχουν).

Παράδειγμα:

```
<script>
let i=0;
for (i=0;i<=10;i++)
  {
  if (i==4) break;
  document.write('Αριθμός ' + i);
  document.write('<br>');
  }
</script>
```

Η εντολή continue

Η **continue** θα σταματήσει την **τρέχουσα** επανάληψη και θα συνεχίσει τον βρόγχο με την επόμενη τιμή.

Παράδειγμα:

```
<script>
let i=0;
for (i=0;i<=10;i++)
  {
  if (i==4) continue;
  document.write('Αριθμός ' + i);
  document.write('<br>');
  }
</script>
```

* Οι εντολές **break** και **continue** χρησιμοποιούνται επίσης για τη διακοπή βρόγχων **while**

Quiz

```
<script>
  let sum = 0;
  for (i = 1; i < 10; i++) {
    if (i == 3) continue;
    if (i == 6) break;
    sum = sum + i;
  }
  alert("Η τιμή της result είναι: " + sum);
</script>
```

Ποια είναι τιμή της sum που θα εμφανιστεί στο alert box;

- 7
- 10
- 12
- 13
- 18
- 45

ΤΕΛΟΣ Γ' ΜΕΡΟΥΣ

Ευχαριστώ για την προσοχή σας

ΜΕΡΟΣ Δ' - Περιεχόμενα

- Πίνακες (arrays)
- Τυχαίοι αριθμοί (random numbers)

με πολλά απλά παραδείγματα

Πίνακες (Arrays)

- **Τι είναι:** Ο **πίνακας** είναι μια "*ειδική μεταβλητή*"* που μπορεί να αποθηκεύσει περισσότερες από μια τιμές.

- Έστω ότι έχουμε μια λίστα στοιχείων π.χ. λίστα με ζώα. Αποθηκεύοντάς τα σε μεταβλητές θα έδειχναν ως εξής:

```
let animal1 = "Γάτα";
let animal2 = "Σκύλος";
let animal3 = "Αγελάδα";
let animal4 = "Πρόβατο";
```

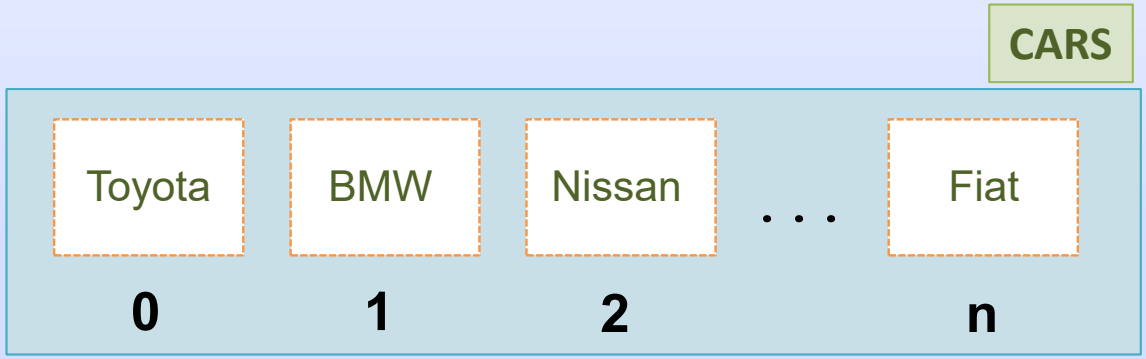
- Αν όμως υπήρχε η ανάγκη να διατρέξουμε τη λίστα των στοιχείων και να βρούμε κάποιο συγκεκριμένο;
- Και αν τα στοιχεία δεν ήταν μόνο τέσσερα αλλά 400 ή 4000;

Τότε τη λύση μας δίνει ο **πίνακας**!

* Στη JavaScript ο πίνακας είναι τύπου δεδομένων: **object** (αντικείμενο)

Πίνακες (Arrays)

- Ο **πίνακας** είναι μια συνεχόμενη διάταξη αποθηκευτικού χώρου.
- Φανταστείτε τον πίνακα σαν μια ομάδα από "κουτιά". Στο καθένα από αυτά μπορείς να αποθηκεύσεις μια τιμή.



- Κάθε "κουτί" έχει μοναδική αρίθμηση. Η αρίθμηση αυτή καλείται **δείκτης (index).**
- **Η αρίθμηση του δείκτη ξεκινάει από το 0**. Συνεπώς το 1ο κουτί έχει δείκτη 0, το δεύτερο δείκτη 1 κοκ.
- Όπως κάθε μεταβλητή, έτσι και ο πίνακας διαθέτει ένα **όνομα**.

Δημιουργία Πίνακα

- Για να δημιουργήσουμε ένα νέο άδειο πίνακα:

```
let cars = [];
```

- Για να δημιουργήσουμε ένα πίνακα με 3 στοιχεία και να θέσουμε τιμή σε κάθε στοιχείο:

```
let cars = ["Toyota", "BMW", "Nissan"];
```

Εναλλακτικά:

- Για να δημιουργήσουμε ένα νέο άδειο πίνακα:

```
let cars = new Array();
```

- Για να δημιουργήσουμε ένα πίνακα με 3 στοιχεία και να θέσουμε τιμή σε κάθε στοιχείο:

```
let cars = new Array("Toyota", "BMW", "Nissan");
```

Δημιουργία Πίνακα

Προσοχή:

- Η ακόλουθη εντολή δημιουργεί τον πίνακα numbers: [1, 2, 3, 4, 5]

```
let numbers = new Array(1, 2, 3, 4, 5);
```

- Η ακόλουθη εντολή **δεν δημιουργεί** τον πίνακα numbers: [10] αλλά ένα πίνακα **με 10 κενές θέσεις!**

```
let numbers = new Array(10);
```

- Για δημιουργία του πίνακα numbers: [10] γράφουμε:

```
let numbers = Array.of(10);
```

Δημιουργία Πίνακα

Κάθε στοιχείο του πίνακα μπορεί να είναι **οποιουδήποτε τύπου δεδομένων** π.χ.:

```
let mixArray = [1, 28, 'Νίκος', 3.14, true, 'Μαρία'];
```

Πυκνός πίνακας (dense). Αυτός που περιέχει συνεχόμενα στοιχεία π.χ.:

```
let denseArray = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];
```

Αραιός πίνακας (sparse). Αυτός που περιέχει μη συνεχόμενα (κενά - empty) στοιχεία π.χ.:

```
let sparceArray = [0, 1, 2, , 4, 5, , , 8, 9];
```

Σημείωση: Οι δύο παραπάνω πίνακες denseArray και sparceArray έχουν το ίδιο πλήθος στοιχείων (μέγεθος πίνακα)!

Μέγεθος Πίνακα (length)

Έστω ο πίνακας **cars**:

```
let cars = ["Toyota", "BMW", "Nissan", "Fiat", "Seat"];
```

Για να διαβάσουμε το **μέγεθος** του πίνακα:

```
let x = cars.length;
console.log(x); // Θα εμφανίσει: 5
```

Μπορούμε να θέσουμε τιμή στην ιδιότητα `length`!

- Αν είναι μικρότερη από τις θέσεις των στοιχείων του πίνακα, τότε τα επιπλέον στοιχεία αφαιρούνται (διαγράφονται) από τον πίνακα.
- Αν είναι μεγαλύτερη από τις θέσεις των στοιχείων του πίνακα, οι επιπλέον θέσεις θα είναι **κενές** (θα επιστρέφουν undefined).

Π.χ. με την δήλωση:

```
cars.length = 3; // Θα περιέχει μόνο τα 3 πρώτα στοιχεία
```

Πίνακες - JOIN()

- Για μετατροπή πίνακα σε αλφαριθμητικό (string) χρησιμοποιήστε: *πίνακας*.**join(***διαχωριστής*)**:**

```
let cars = ["Toyota", "BMW", "Nissan"];
console.log(cars.join(" και "));
// Θα εμφανίσει: Toyota και BMW και Nissan
```

- Ο προεπιλεγμένος (default) διαχωριστής είναι το κόμμα (,):

```
let cars = ["Toyota", "BMW", "Nissan"];
console.log(cars.join());
// Θα εμφανίσει: Toyota,BMW,Nissan
```

Πρόσβαση σε στοιχεία Πίνακα

- Έστω ο πίνακας:

```
let cars = ["Toyota", "BMW", "Nissan"];
```

- Για να **διαβάσουμε** ένα στοιχείο του πίνακα:

```
let oneCar = cars[1];
console.log(oneCar);
// Θα εμφανίσει: BMW
```

- Για να **γράψουμε** ένα στοιχείο του πίνακα:

```
cars[1] = "Mercedes";
console.log(cars);
// Θα εμφανίσει: 'Toyota', 'Mercedes', 'Nissan'
```

Παράδειγμα με Πίνακα

```
<script>
  let i, answer, counter=0;
  let lessons = ["Γλώσσα", "Μαθηματικά", "Φυσική", "Βάσεις Δεδομένων",
"Προγραμματισμός", "Δίκτυα Η/Υ"];

  for (i=0; i<lessons.length ;i++) {
    answer = confirm("Σου αρέσει το μάθημα: " + lessons[i] + ";\n Πάτησε ΟΚ
αν ναι, Άκυρο αν όχι.");
    if (answer) counter++;
  }

  switch (counter) {
    case 0:
      alert("Κανένα μάθημα; Χμμ... Μάλλον προτιμάς το διάλλειμα!");
      break;
    case 1:
      alert("Μόνο ένα μάθημα; Ίσως να σου αρέσει και η Γυμναστική!");
      break;
    default:
      alert("Σου αρέσουν " + counter + " μαθήματα");
  }
</script>
```

Προσθήκη στοιχείων στο τέλος

- Έστω ο πίνακας:

```
let cars = ["Toyota", "BMW", "Nissan"];
```

- **Προσθέτουμε** στοιχεία **στο τέλος του πίνακα** με τη μέθοδο: *πίνακας*.**push()**

```
cars.push("Fiat");
console.log(cars);
// Θα εμφανίσει: 'Toyota', 'BMW', 'Nissan', 'Fiat'
```

- Ο πίνακας διαθέτει τώρα 4 στοιχεία.
- Ο δείκτης για το νέο στοιχείο ενημερώνεται αυτόματα.

Αφαίρεση στοιχείου από το τέλος

- Έστω ο πίνακας:

```
let cars = ["Toyota", "BMW", "Nissan"];
```

- **Αφαιρούμε** το **τελευταίο στοιχείο του πίνακα** με τη μέθοδο: *πίνακας*.**pop()**.

- Η μέθοδος αυτή, **επιστρέφει** το στοιχείο που αφαιρείται.

```
let x = cars.pop();
console.log(cars);
// Θα εμφανίσει: 'Toyota', 'BMW'
// Η μεταβλητή x θα έχει την τιμή: 'Nissan'
```

Αφαίρεση στοιχείου από την αρχή

- Έστω ο πίνακας:

```
let cars = ["Toyota", "BMW", "Nissan"];
```

- **Αφαιρούμε** το **πρώτο στοιχείο του πίνακα** με τη μέθοδο: *πίνακας*.**shift()**.

- Η μέθοδος αυτή, **επιστρέφει** το στοιχείο που αφαιρείται.

```
let x = cars.shift();
console.log(cars);
// Θα εμφανίσει: 'BMW', 'Nissan'
// Η μεταβλητή x θα έχει την τιμή: 'Toyota'
```

- Ο δείκτης για κάθε στοιχείο ενημερώνεται αυτόματα.

Προσθήκη στοιχείων στην αρχή

- Έστω ο πίνακας:

```
let cars = ["Toyota", "BMW", "Nissan"];
```

- **Προσθέτουμε** στοιχεία **στην αρχή του πίνακα** με τη μέθοδο: *πίνακας*.**unshift()**

```
cars.unshift("Fiat");
console.log(cars);
// Θα εμφανίσει: 'Fiat', 'Toyota', 'BMW', 'Nissan'
```

- Ο πίνακας διαθέτει τώρα 4 στοιχεία.
- Ο δείκτης για κάθε στοιχείο ενημερώνεται αυτόματα.

Προσθήκη/αφαίρεση στοιχείων σε ενδιάμεση θέση

- Έστω ο πίνακας:

```
let cars = ["Toyota", "BMW", "Nissan"];
```

- **Προσθέτουμε** ένα στοιχείο **σε οποιαδήποτε θέση του πίνακα**: *πίνακας*.**splice(***start, deleteCount*[,*item1*,…]**)**

```
cars.splice(1,0,"Fiat");
console.log(cars);
// Θα εμφανίσει: 'Toyota', 'Fiat', 'BMW', 'Nissan'
```

- Ο πίνακας διαθέτει τώρα 4 στοιχεία.
- Ο δείκτης για κάθε στοιχείο ενημερώνεται αυτόματα.
- Με την **splice** μπορούμε επίσης να **αφαιρέσουμε** στοιχεία, θέτοντας στην *deleteCount* το πλήθος των στοιχείων προς αφαίρεση.

Συνένωση Πινάκων

- **Ενώνουμε** δύο (ή περισσότερους) πίνακες (ή τιμές) με τη μέθοδο:
  var *new_array = old_array*.**concat(***value1[,value2,…]***)**

```
let alpha = ["a", "b", "c"];
let numeric = [1, 2, 3];
let alphanumeric = alpha.concat(numeric);
console.log (alphanumeric);
// Θα εμφανίσει: 'a', 'b', 'c', 1, 2, 3
```

```
let alpha = ["α", "β", "γ"];
let alphanumeric = alpha.concat("ένα","δύο","τρία");
console.log(alphanumeric);
// Θα εμφανίσει: 'α', 'β', 'γ', 'ένα', 'δύο', 'τρία'
```

Παραγωγή Τυχαίων Αριθμών

- Για δημιουργία τυχαίων αριθμών χρησιμοποιούμε:

```
let tyxaios_arithmos = Math.random();
```

- Η μέθοδος random() μας επιστρέφει έναν αριθμό **[0,1)**
- Δηλαδή: από το 0 (συμπεριλαμβάνεται – κλειστό διάστημα) μέχρι το 1 (δε συμπεριλαμβάνεται – ανοικτό διάστημα).

```
let random1 = Math.random();
let random2 = Math.random();
let random3 = Math.random();
console.log(random1, random2, random3);
```

Math: ένα ενσωματωμένο αντικείμενο (built-in object) της JavaScript που διαθέτει ιδιότητες και μεθόδους για μαθηματικές συναρτήσεις και σταθερές.

Ρύθμιση εμβέλειας

- Η μέθοδος random() μας επιστρέφει έναν αριθμό **[0,1)**
- Πολλαπλασιάζοντας το αποτέλεσμα με έναν αριθμό αλλάζουμε την εμβέλεια του τυχαίου αριθμού π.χ.:

```
let randomValue = Math.random() * maxValue;
```

- Μετά την εκτέλεση του παραπάνω, η μεταβλητή **randomValue** θα έχει έναν αριθμό μεταξύ **[0, maxValue)**

```
let x = Math.random() * 5;
console.log( x );
// Η τιμή της x: 0 <= x < 5
```

Ρύθμιση εμβέλειας

- Τις περισσότερες φορές θέλουμε έναν τυχαίο ακέραιο αριθμό
- Χρησιμοποιώντας τη μέθοδο **`floor()`** αποκόπτουμε τα δεκαδικά ψηφία του αριθμού *
- Π.χ. το 3.782345693 γίνεται **3** ενώ το -4.345672319 γίνεται **-5**

```
let x
x = Math.random() * 10;
x = Math.floor(x);
console.log(x);
// Η τιμή της x θα είναι από 0 μέχρι 9
```

* **`Math.floor(x)`**: Επιστρέφει τον μεγαλύτερο ακέραιο που είναι μικρότερος ή ίσος του αριθμού που δίνεται (x).

Άσκηση

- Έστω ότι θέλουμε να φτιάξουμε ένα επιτραπέζιο παιχνίδι με ζάρια. Δημιουργήστε τον κώδικα σε JavaScript ο οποίος παράγει και εμφανίζει έναν αριθμό μεταξύ του 1 και του 6 **[1-6].**

Λύση Νο 1

```
let zari;
zari = Math.random() * 6; // 0 έως 5.99999999
zari = Math.floor(zari);  // 0 έως 5
zari += 1;                // 1 έως 6
console.log(zari);
```

Λύση Νο 2

```
let zari = Math.floor(Math.random() * 6) + 1;
console.log(zari);
```

Εισαγωγή στη JavaScript

Πηγές & χρήσιμοι σύνδεσμοι:

https://www.w3schools.com/js/
http://dide.flo.sch.gr/Plinet/Tutorials/Tutorials-JavaScript.html
http://www.eeei.gr/programming/javascript/
https://www.codecademy.com/catalog/language/javascript
https://developer.mozilla.org/en-US/docs/Web/JavaScript
https://javascript.info
https://el.wikipedia.org/wiki/JavaScript
https://www.youtube.com/@psounis

ΤΕΛΟΣ ΠΑΡΟΥΣΙΑΣΗΣ