

# JAVASCRIPT 1

---

Διδάσκοντες: Π. Αγγελάτος, Δ. Ζήνδρος

Επιμέλεια διαφανειών: Π. Αγγελάτος

Σχολή Ηλεκτρολόγων Μηχανικών  
και Μηχανικών Υπολογιστών



# Στόχος της ώρας

- Εισαγωγή της γλώσσας Javascript σε αρχάριο επίπεδο:
  - Βασική σύνταξη
  - Συνδυασμός Javascript και HTML
  - Μεταβλητές
  - Τελεστές
  - Έλεγχος ροής (if, else, switch, for, while)
  - Συναρτήσεις
  - Αντικείμενα
  - Βασικοί τύποι δεδομένων

# Javascript

- **Δεν** έχει κάποια σχέση με την Java
- Τρέχει στον **client** και **όχι** (απαραίτητα) στον server
- Στέλνεται στον browser μέσω HTTP
- Με την Javascript ορίζουμε την συμπεριφορά της σελίδας

# Εναλλακτικές λύσεις

- Δεν έχουμε άλλη εναλλακτική λύση που να είναι ανοικτό πρότυπο

# Τι μπορεί να κάνει;

- Animations, εφέ
- Φόρτωση περιεχομένου ασύγχρονα (χωρίς refresh)
- Έλεγχος φόρμας πριν το submit
- Πολλά, πολλά άλλα

# Συνδυασμός HTML και Javascript

- Ετικέτα `<script>`
- Χρησιμοποιείται για να φορτώσει κώδικα javascript
- Ιδιότητα `src`
  - Έχει τιμή το αρχείο JS που πρέπει να φορτώσει
- Ιδιότητα `type`
  - Έχει τιμή “text/javascript”

```
<script type="text/javascript" src="foo.js"></script>
```

# Συνδυασμός HTML και Javascript

```
<html>  
  <head>  
    <title>:)</title>  
  </head>  
  <body>  
    <script  
      type="text/javascript"  
      src="foo.js">  
    </script>  
  </body>  
</html>
```

# Συνδυασμός HTML και Javascript

- Ετικέτα `<script>`
- Το `src` μπορεί να παραληφθεί και να περιέχει τον κώδικα JS

```
<script type="text/javascript">
```

```
    //Javascript Code. This is a JS comment btw
```

```
</script>
```

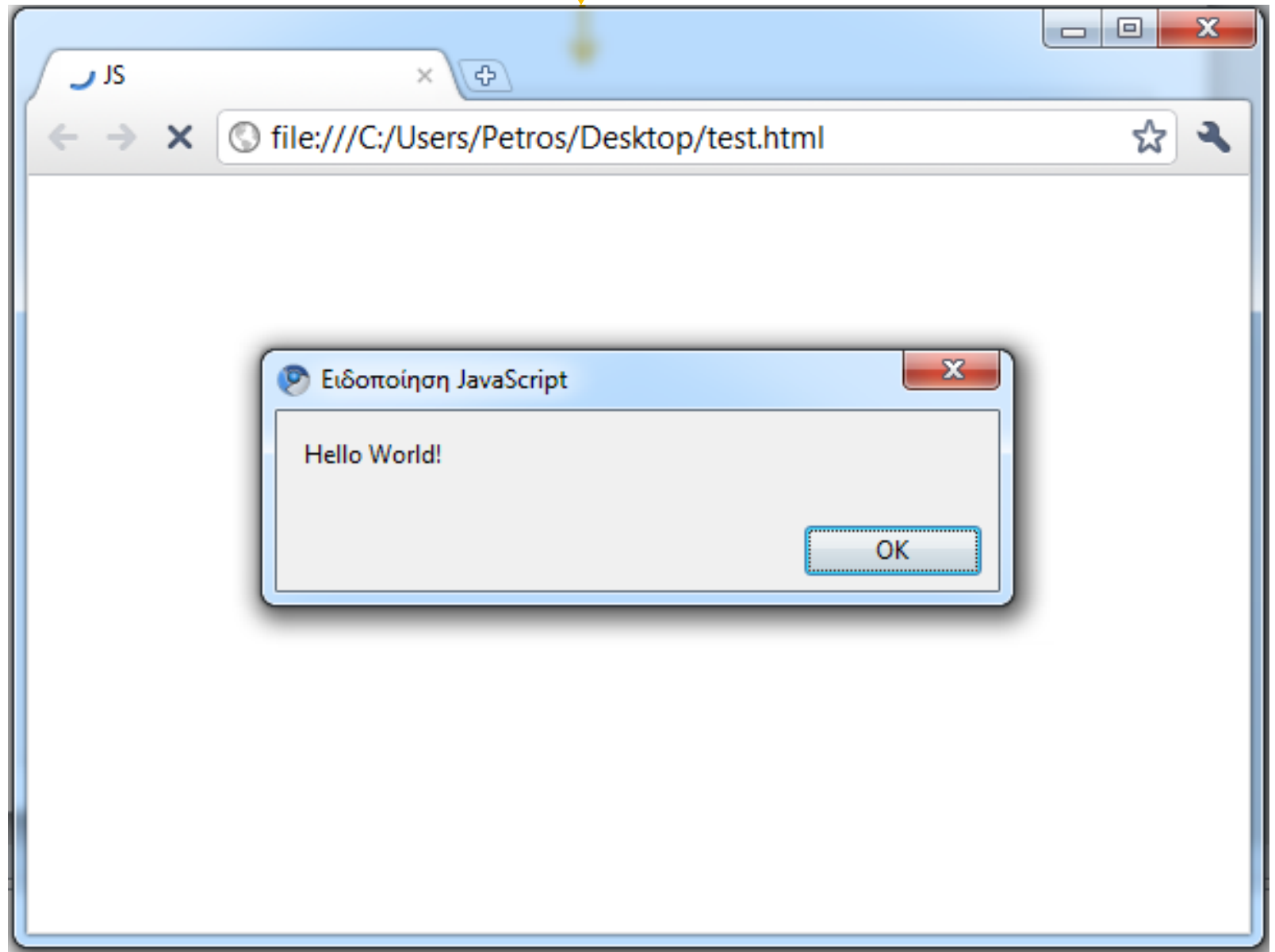


# Βασική σύνταξη

- Ο κώδικας σε κάθε script αρχίζει να εκτελείται την ώρα που τον «διαβάζει» ο browser
- Ο κώδικας εκτελείται σειριακά
  - Η μία εντολή μετά την άλλη
- Κάθε εντολή τελειώνει σε ;
- Δε λειτουργεί τυπώνοντας πράγματα

# Γεια σου κόσμε!

```
<html>  
  <head>  
    <title>JS</title>  
  </head>  
  <body>  
    <script type="text/javascript">  
      alert( 'Hello world!' );  
    </script>  
  </body>  
</html>
```



# alert()

- **alert**: Βγάζει ειδοποίηση με το κείμενο της πρώτης παραμέτρου.
- Χρήσιμη για να ξέρουμε τι συμβαίνει κατά την ανάπτυξη της εφαρμογής

# Εκτέλεση

- Η Javascript εκτελείται από τη μηχανή Javascript του εκάστοτε browser
- Οι νέες μηχανές Javascript κάνουν compile τον κώδικα πριν τον τρέξουν

Browser	Javascript Engine
Google Chrome	V8
Mozilla Firefox	SpiderMonkey
Opera	Caracan
Microsoft Internet Explorer	Chakra

# Τελεστές

Τελεστές	Λειτουργία
+, -, *, /, %	Αριθμητικές πράξεις
, &&, !	Λογικές πράξεις: ή, και, όχι
==, !=, <, >, <=, >=	Σύγκριση
++, --	Αύξηση, μείωση
+	Ένωση αλφαριθμητικών
=, +=, -=, *=, /=, %=	Ανάθεση τιμής

# Σύγκριση

Με `==`, `!=`

- Ο τύπος των τιμών μετατρέπεται ώστε να γίνει η σύγκριση
- Αν κάποιο από τα ορίσματα είναι Boolean μετατρέπεται σε Number
- Αν κάποιο από τα ορίσματα είναι String τότε το άλλο όρισμα μετατρέπεται σε String
- `1 == 1` → true
- `1 == 2` → false
- `0 == "hello"` → false
- `"1" == 1` → true

# Σύγκριση

- Η σύγκριση αλφαριθμητικών γίνεται εύκολα και **σωστά**:

```
a = "hello";  
if ( a == "hello" ) {  
    alert( 'a is hello.' );  
}  
else {  
    alert( 'a is not hello.' );  
}
```



if

```
if ( συνθήκη ) {  
    σώμα 1  
}  
else if ( συνθήκη ) {  
    σώμα 2  
}  
...  
else {  
    σώμα 3  
}
```

# switch

```
switch ( παράσταση ) {  
    case τιμή1:  
        σώμα 1;  
        break;  
    case τιμή2:  
        σώμα 1;  
        break;  
    ...  
    default:  
        εναλλακτικό σώμα  
}
```

# switch

- **Επιλέγει** ένα σώμα με βάση την τιμή μίας παράστασης
- Παρόμοιο με C, C++, Java, ...
  - Τα cases **μπορούν** να είναι και **όχι σταθερές**
- Αν η παράσταση έχει τιμή1
  - Εκτέλεση σώματος 1
- Αν η παράσταση έχει τιμή2
  - Εκτέλεση σώματος2
- ...
- Αλλιώς
  - Εκτέλεση εναλλακτικού σώματος
- Το **default** είναι προαιρετικό

# switch

- Παράληψη του break οδηγεί σε **fall-through**
- Εκτελούνται τα σώματα που ακολουθούν μέχρι το επόμενο break

for

```
for ( αρχικοποίηση; συνθήκη; βήμα ) {  
    σώμα  
}
```

```
for ( αρχικοποίηση; συνθήκη; βήμα ) {  
    σώμα  
}
```

# for

- Ίδιο σε C, C++, Java...
- Επαναλαμβάνει ένα σώμα σύμφωνα με κάποια συνθήκη
- Αρχικά τρέχει η **αρχικοποίηση**
- Αν η **συνθήκη** είναι **ψευδής**, τελειώσαμε
- Αν η **συνθήκη** είναι **αληθής**, τρέχει το **σώμα**
- Μετά το σώμα τρέχει το **βήμα**
- Η **συνθήκη** ελέγχεται ξανά, κ.ό.κ.

# while

```
while ( συνθήκη ) {  
    σώμα  
}
```

```
while ( συνθήκη ) {  
    σώμα  
}
```

# while

- Ίδιο σε C, C++, Java, ...
- Επαναλαμβάνει ένα σώμα σύμφωνα με κάποια συνθήκη
- Αρχικά ελέγχεται η συνθήκη
- Αν η **συνθήκη** είναι **ψευδής**, τελειώσαμε
- Αν η **συνθήκη** είναι **αληθής**, τρέχει το **σώμα**
- Η **συνθήκη** ελέγχεται ξανά, κ.ό.κ.



# do... while

```
do {  
    σώμα  
} while ( συνθήκη );
```

```
do {  
    σώμα  
} while ( συνθήκη );
```

## do... while

- Ίδιο σε C, C++, Java, ...
- Επαναλαμβάνει ένα σώμα σύμφωνα με κάποια συνθήκη
- Αρχικά τρέχει μία φορά το σώμα
- Στη συνέχεια ελέγχεται η συνθήκη
- Αν η συνθήκη είναι **ψευδής**, τελειώσαμε
- Αν η συνθήκη είναι **αληθής**, τρέχει το **σώμα**
- Η συνθήκη ελέγχεται ξανά, κ.ό.κ.

# break

- Ίδιο σε C, C++, Java, ...
- Εμφανίζεται μέσα σε μία ροή ελέγχου
  - **for, while, do... while, switch**
- Διακόπτει την ροή και συνεχίζει αμέσως μετά
- Δεν γίνονται άλλες επαναλήψεις μετά το **break**

# continue

- Ίδιο σε C, C++, Java, ...
- Εμφανίζεται μέσα σε μία ροή επανάληψης
  - **for, while, do... while**
- Διακόπτει την ροή και συνεχίζει ελέγχοντας την συνθήκη
- Μπορεί να γίνουν και άλλες επαναλήψεις μετά το **continue**

# Σχόλια

- `//` η υπόλοιπη γραμμή είναι σχόλιο
- Το πολύ 1 γραμμή

```
a = 5; // assign a to be 5
```

- `/*` τα περιεχόμενα είναι σχόλιο `*/`
- 1 ή περισσότερες γραμμές

# Μεταβλητές στην Javascript

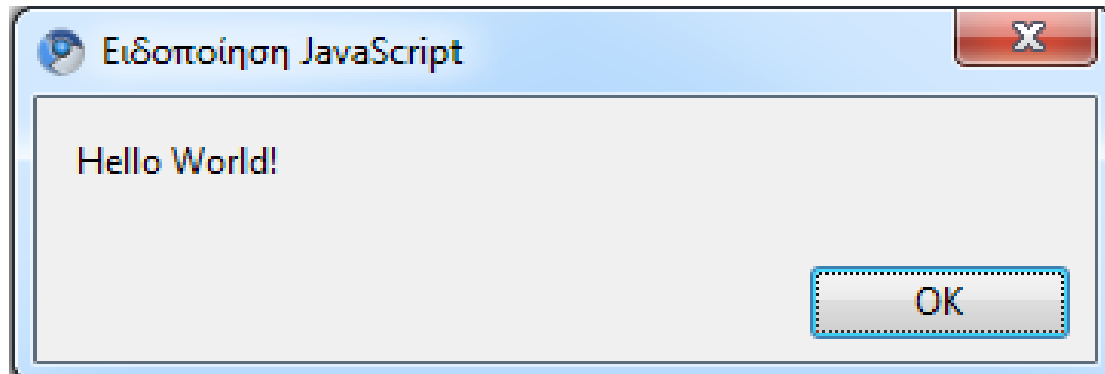
- Αποθηκεύουν μία τιμή
  - Νούμερα, κείμενα, πίνακες, αντικείμενα ...
- Δηλώνονται με την λέξη-κλειδί **var**
  - **var a;**
- Το όνομα...
  - Αρχίζει με γράμμα ή **\_**
  - Περιέχει γράμματα, αριθμούς, **\_**
  - Έχει ευαισθησία σε πεζά-κεφαλαία
- Παρόμοιες με C, C++, Java, Pascal, ...

# Μεταβλητές

- Τιμή μεταβλητής ορίζεται με τον τελεστή =
- **a = 5;**
- Δίνει στην μεταβλητή **a** την τιμή **5**
- Οι μεταβλητές μπορούν να **αλλάξουν** τιμή
- Μπορούν να χρησιμοποιηθούν μέσα σε παραστάσεις

# Μεταβλητές

```
<script type="text/javascript">  
    var a = "Hello world!";  
    alert( a );  
</script>
```

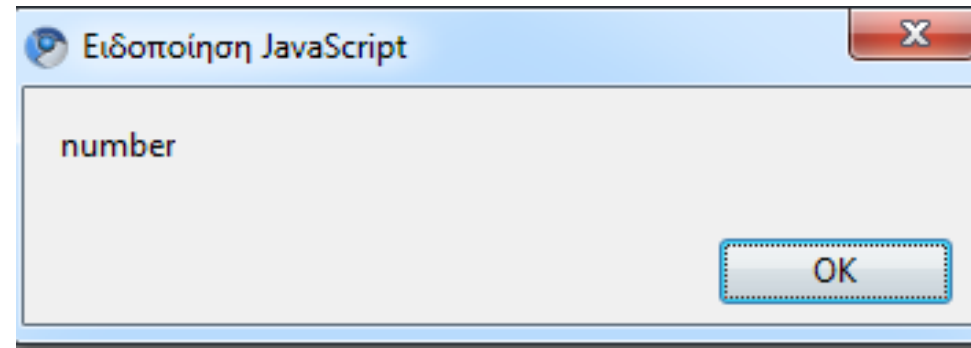
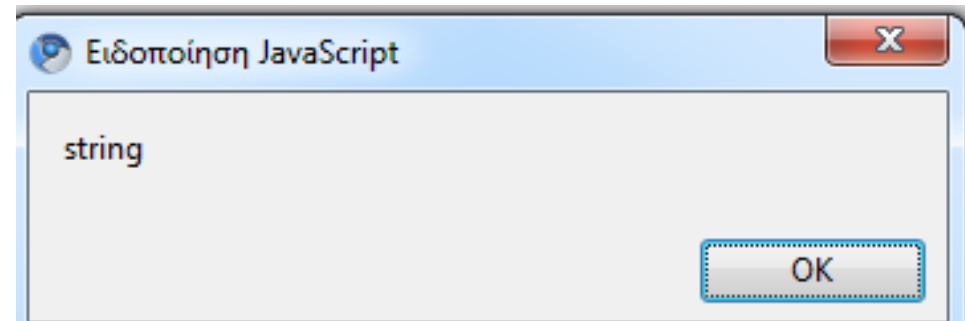




# Ασθενές σύστημα τύπων

- Τύποι στην Javascript:
  - Number
  - Boolean
  - String
  - Function
  - Object
- Μία μεταβλητή μπορεί να αλλάζει τύπο

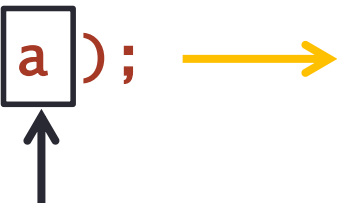
```
<script type="text/javascript">  
    var a = "Hello";  
    alert( typeof( a ) );  
    a = 5;  
    alert( typeof( a ) );  
</script>
```



# Δυναμικό σύστημα τύπων

- Μεταβλητές παίρνουν τύπο τιμής
- Διευκόλυνση στη συγγραφή κώδικα
- Σφάλματα **χρόνου εκτέλεσης** αντί συντακτικά
  - Πιο δύσκολα στον εντοπισμό
- Οι **μετατροπές** τύπων γίνονται **αυτόματα**

```
<script type="text/javascript">  
    var a = "5";  
    var b = 7;  
    alert( b - a );  
</script>
```



Μετατρέπεται σε Number για να γίνει η πράξη

# Συναρτήσεις

```
function όνομα_συνάρτησης( ορίσματα ) {  
    σώμα;  
}
```

# Συναρτήσεις

- Παρόμοιες με συναρτήσεις σε C, C++, Java, ...
- Ορίζουν υπο-ρουτίνες που κάνουν συγκεκριμένη δουλειά
- Ορίζονται με την λέξη-κλειδί function
- Ακολουθεί το **όνομα** της συνάρτησης
- Ακολουθούν τα ονόματα των **ορισμάτων** σε ( ) χωρισμένα με κόμματα

# Επιστροφή τιμής

- Οι συναρτήσεις **επιστρέφουν** τιμή με **return**
- Η τιμή επιστροφής χρησιμοποιείται όπου έγινε η κλήση
- Επιστροφή σημαίνει **τερματισμός** συνάρτησης
- Δεν ορίζουμε **τύπο** επιστροφής
- Δεν είναι υποχρεωτικό

# Συναρτήσεις

```
function add( a, b ) {  
    return a + b;  
}
```

```
alert( add( 4, 7 ) ); //11
```

# Συναρτήσεις

- Οι συναρτήσεις είναι μεταβλητές!

```
function foo() {  
    alert( 'hello' );  
}
```

```
var foo = 5;
```

```
foo(); //Σφάλμα. Το 5 δεν είναι συνάρτηση.
```



# Αντικείμενα

- Στην Javascript όλα είναι αντικείμενα!
- Τι είναι ένα αντικείμενο;
  - Ένα λεξικό
  - Αντιστοίχιση τιμών σε κλειδιά
- Τα κλειδιά είναι Strings
- Τιμές μπορεί να είναι
  - άλλο αντικείμενο
  - αριθμός
  - αλφαριθμητικό
  - συνάρτηση

# Αντικείμενα

- Δήλωση αντικειμένου

//Κενό αντικείμενο

```
var a = { };
```

//Αντικείμενο με 2 ιδιότητες

```
var b = {  
    foo: 5,  
    bar: 'hello'  
}
```

# Αντικείμενα

- Διαβάζουμε τις τιμές ενός αντικειμένου με
- `object[ 'key' ]`
- Ή αλλιώς `object.key` (syntactic sugar)

```
var a = {  
    foo: 'bar'  
}
```

```
var b = a.foo; //Το b έχει την τιμή 'bar'
```

# Αντικείμενα

- Μπορούμε να προσθέτουμε ιδιότητες μετά τη δημιουργία

```
var a = {  
    foo: 0
```

```
}
```

```
a.bar = 1; //Προσθήκη ιδιότητας bar με τιμή  
1
```

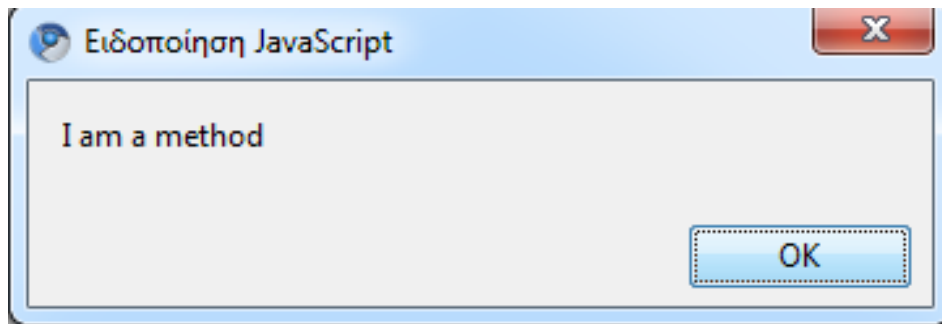
# Αντικείμενα

- Μέθοδοι αντικειμένων
- Είναι κλειδιά που η τιμή τους είναι συνάρτηση

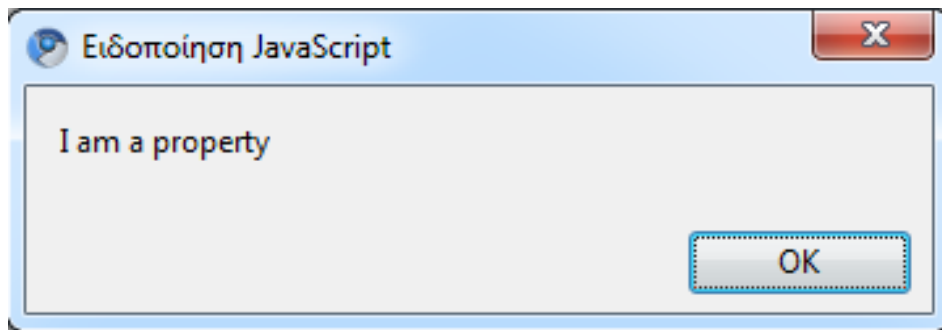
```
var a = {  
    foo: function() {  
        return 'I am a method';  
    },  
    bar: 'I am a property'  
}
```

# Αντικείμενα

- `alert( a.foo() );`



- `alert( a.bar );`



# Αλφαριθμητικά

- Μία τιμή που είναι μία σειρά από αριθμούς, χαρακτήρες, γράμματα, ...
- Το μήκος μπορεί να αλλάζει
- Δεν υπάρχει περιορισμός μήκους
- **Δεν** πρόκειται για πίνακες από χαρακτήρες
- Επιτρέπονται Unicode χαρακτήρες άμεσα:  
**a = “Χαίρε, κόσμε!”;**
- Μπαίνουν σε διπλά ή μονά εισαγωγικά

# Συνένωση αλφαριθμητικών

- Με τον τελεστή **+**
- Παράγει ένα αλφαριθμητικό που είναι η παράθεση δύο άλλων
- **“Hello, “ + “world” → Hello, world**



# Αλφαριθμητικά ως αντικείμενα

- Κάθε αλφαριθμητικό έχει μεθόδους
  - **search** – Αναζήτηση μέσα στο String, επιστρέφει το index
  - **split** – Χωρίζει το string σε κομμάτια, επιστρέφει πίνακα
  - **substr** – Παίρνει κομμάτι από το string
  - **substring** – Παίρνει κομμάτι από το string
  - και άλλες
- Παράδειγμα

```
var a = '1,2,3,4,5';  
var b = a.split( ',' );//Το b είναι ίσο με  
[1,2,3,4,5]
```

# Αριθμοί

- Όλοι οι αριθμοί είναι τύπου Number
- 64bit κινητής υποδιαστολής
- Οι ακέραιοι είναι αξιόπιστοι μέχρι τα 15 δεκαδικά ψηφία

# Πίνακες

- Δηλώνονται με `[ ]`
- Οι τιμές του χωρίζονται με κόμματα
- Η αρίθμηση ξεκινάει από το 0

```
var a = [ 1, 'two', 3, 4 ];
```

```
a[ 0 ]; //1
```

```
a[ 1 ]; //two
```

# Πίνακες

- Έχουν χρήσιμες ιδιότητες και μεθόδους
- Ιδιότητα `length`
  - Περιέχει το μήκος του πίνακα
  - `var a = [ 1, 2, 3, ];`
  - `a.length; //3`
- Μέθοδοι
  - `pop()`
  - `push()`
  - `shift()`
  - `slice()`
  - `reverse()`
  - `join()`
  - `sort()`

# Πίνακες

```
var a = [ 1, 2, 3, 4 ];  
var b = a.pop();  
//a = [ 1, 2, 3 ] και b = 4  
a.unshift( b );  
// a = [ 4, 1, 2, 3 ] και b = 4  
a.push( 40 );  
//a = [ 4, 1, 2, 3, 40 ]  
a.sort();  
//a = [ 1, 2, 3, 4, 40 ]
```

# Το script μου δε τρέχει 😞

- Βλέπουμε συνακτικά σφάλματα και σφάλματα χρόνου εκτέλεσης στην κονσόλα σφαλμάτων του browser
- Στον Firefox πατάμε Control + Shift + J
- Χρησιμοποιούμε το alert για να δούμε την τιμή μιας μεταβλητής

# Μάθαμε

- Εισαγωγή της γλώσσας Javascript σε αρχάριο επίπεδο:
  - Βασική σύνταξη
  - Συνδυασμός Javascript με HTML
  - Μεταβλητές
  - Τελεστές
  - if, else, switch, for, while
  - Συναρτήσεις
  - Αντικείμενα
  - Αλφαριθμητικά

# Την επόμενη φορά...

- Η βιβλιοθήκη jQuery
  - Πως να κάνουμε πραγματικά cool πράγματα στον browser