Εισαγωγή στη JavaScript

Δρ. Γιάννης Χαμόδρακας
Μέλος ΕΔΙΠ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών

6-May-15

Γενικές Πληροφορίες

- Η JavaScript είναι διαφορετική γλώσσα από τη Java
- Η σύνταξη της μοιάζει με τη σύνταξη της Java και της C
- Τρέχει σε οποιαδήποτε πλατφόρμα παρέχει JavaScript interpreter (συνήθως κομμάτι των browser)
- Μπορεί να αποτελεί κομμάτι σελίδων HTML εντός στοιχείου script:
  - HTML5:

```
<script>
        document.write("<h1>Hello World!</h1>") ;
</script>
```

  - XHTML:

```
<script type="text/javascript">
        <![CDATA[
         if (variable < 10 && variable >= 0)
                action();
        ]]>
</script>
```

Γενικές Πληροφορίες

- Στοιχεία JavaScript τοποθετούνται τόσο εντός του στοιχείου <head> όσο και εντός του <body>
  - Οι ορισμοί των συναρτήσεων JavaScript πρέπει να βρίσκονται εντός του στοιχείου <head> ώστε να διασφαλίζεται ότι η συνάρτηση έχει φορτωθεί πριν τη χρήση της
  - Στοιχεία JavaScript εντός <body> εκτελούνται κατά τη φόρτωση της σελίδας
- Όταν στοιχεία JavaScript ορίζονται εντός στοιχείων input σε φόρμες (π.χ. submit) ο κώδικας εκτελείται κατά τη χρήση του στοιχείου input
- Ο κώδικας JavaScript μπορεί να περιλαμβάνεται σε εξωτερικό αρχείο. Εντός του στοιχείου <head>

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

Πρωτογενείς Τύποι

- number, string, και boolean
  - Μεταβλητές ή σταθερές που δεν ανήκουν στους παραπάνω τύπους είναι αντικείμενα
- Οι αριθμοί αποθηκεύονται πάντοτε ως τιμές κινητής υποδιαστολής
  - Οι δεκαεξαδικοί αριθμοί ξεκινούν με 0x
  - Οκταδικοί πρέπει να αποφευγονταί
- Οι συμβολοσειρές βρίσκονται εντός μόνων ή διπλών εισαγωγικών
- Boolean: true ή false
  - Οι σταθερές 0, "0", κενές συμβολοσειρές, undefined, null, και NaN (NotANumber) είναι επίσης false, όλες οι υπόλοιπες τιμές είναι true

Μεταβλητές

- Η δήλωση μεταβλητών ξεκινά με τη δεσμευμένη λέξη var:
  - var counter = 0, x, y, name = " John" ;
  - Τα ονόματα των μεταβλητών ξεκινούν με γράμμα ή κάτω παύλα
  - Όπως και στη C είναι case-sensitive
  - Οι μεταβλητές δεν έχουν τύπο, μπορούν να αποθηκεύουν τιμές οποιουδήποτε τύπου
  - Η λέξη var είναι προαιρετική αλλά πρέπει να χρησιμοποιείται για την αναγνωσιμότητα του κώδικα
- Τοπικές μεταβλητές: δηλώνονται εντός συναρτήσεων και είναι προσβάσιμες μέσα στο σώμα τους
- Καθολικές μεταβλητές: δηλώνονται έξω από κάθε συνάρτηση

Τελεστές (1)

- Αριθμητικοί τελεστές
  `+  -  *  /  %  ++  --`
- Τελεστές σύγκρισης
  `<  <=  ==  !=  >=  >`
- Λογικοί τελεστές
  `&&  ||  !`
- Δυαδικοί τελεστές
  `&  |  ^  ~  <<  >>  >>>`
- Τελεστές ανάθεσης
  `+=  -=  *=  /=  %=  <<=  >>=  >>>=  &=  ^=  |=`

Τελεστές (2)

- Τελεστής συνένωσης συμβολοσείρών
  `+`
- Τελεστής υπό συνθήκη
  condition `?` ***value_if_true*** `:` ***value_if_false***
- Τελεστές ισότητας και ανισότητας
  - Οι τελεστές `==` και `!=` μετασχηματίζουν τους τελεστέους στον ίδιο τύπο πριν τον έλεγχο
  - Οι τελεστές `===` και `!==` αν οι τελεστέοι έχουν διαφορετικό τύπο θεωρούνται άνισοι
- Τελεστές που θα συζητηθούν σε επόμενες διαφάνειες
  `new` `typeof` `void` `delete`

Σχόλια

- Όπως στην Java:
  - Σχόλια γραμμής //
  - Σχόλια block /* … */

Εντολές (1)

- Όπως στην C
  - Ανάθεση: greeting = "Hello, " + name;
  - Σύνθετη εντολή:
    { ***statement***; …; ***statement*** }
  - Εντολή ελέγχου if-else:
    if (***condition***) ***statement***;
    if (***condition***) ***statement***; else ***statement***;
  - Εντολές βρόχου:
    while (***condition***) ***statement***;
    do ***statement*** while (***condition***);
    for (***initialization***; ***condition***; ***increment***) ***statement***;

Εντολές (2)

- Εντολή switch:

```
switch (expression) {
   case label :
      statement;
      break;
   case label :
      statement;
      break;
   ...
   default : statement;
}
```

- Άλλες εντολές:
  - break;
  - continue;
  - Κενή εντολή, ;; ή { }

Σύγκριση με τη Java

- Ομοιότητες με τη Java
  - Υπάρχουν Objects και πρωτογενείς τύποι δεδομένων
  - Υποστηρίζονται γεγονότα και χειριστές γεγονότων
  - Ο χειρισμός των εξαιρέσεων (exceptions) είναι παρόμοιος με τη Java
- Διαφορές από τη Java
  - Οι μεταβλητές δεν έχουν προκαθορισμένο τύπο: ο τύπος της μεταβλητής εξαρτάται από την τιμή της
  - Τα αντικείμενα και οι πίνακες ορίζονται με διαφορετικό τρόπο
  - Εντολές with

Χειρισμός exception (1)

- Η έκφραση throw ***expression*** δημιουργεί exception
  - Η τιμή του exception είναι στο παραπάνω παράδειγμα *expression* οποιουδήποτε τύπου (συχνά μια σταθερή literal συμβολοσειρά)
- try {
  εντολές
  } catch (***e***) { // Δεν δηλώνεται τύπος
  εντολές χειρισμού exception
  } finally { // Προαιρετικά
  εντολές που πάντοτε εκτελούνται
  }
  - Σε αυτή την περίπτωση υπάρχει μόνο ένα μπλοκ σύλληψης και χειρισμού του exception

Χειρισμός exception (2)

- try {
  εντολές
  } catch (***e*** if ***test1***) {
  εντολές χειρισμού αν η έκφραση test1 είναι true
  } catch (***e*** if ***test2***) {
  εντολές αν η test1 είναι false και η test2 true
  } catch (***e***) {
  εντολές χειρισμού αν test1, test2 false
  } finally {
  εντολές που πάντοτε εκτελούνται
  }
- Πολλαπλά block χειρισμού βάσει ελέγχου:
  e == "SomeException"

Συναρτήσεις

- Πρέπει να ορίζονται εντός του στοιχείου <head> ώστε να φορτώνονται στην αρχή
- Ορισμός συνάρτησης:
  function ***name***(***arg1***, ***…***, ***argN***) { ***statements*** }
  - Η συνάρτηση μπορεί να περιέχει εντολές return ***value*** αλλιώς επιστρέφει undefined (function declaration);
  - Οι μεταβλητές που δηλώνονται εντός συνάρτησης είναι τοπικές
- Κλήση συνάρτησης
  name(***arg1***, ***…***, ***argN***);
- Όσον αφορά τις παραμέτρους δημιουργείται αντίγραφο της μεταβλητής που δέχονται εντός της συνάρτησης. Αν η μεταβλητή αναφέρεται σε αντικείμενο, μπορεί να αλλάξουν τα πεδία του εντός της συνάρτησης (call by-sharing).

Object literals

- Σύνταξη: παρόμοια με τη JavaScript Object Notation (JSON). Ωστόσο, δεν είναι απαραίτητα τα διπλά εισαγωγικά στα ονόματα των παραμέτρων. Η τιμή δεν είναι απαραίτητα literal
  - { ***name1*** : ***value1*** , … , ***nameN*** : ***valueN*** }
- Π.χ.
  - car = {type: "207", brand: “Peugeot", getCar: CarTypes(“Peugeot"), special: Sales}
    - Τα πεδία είναι type, getCar, brand, και special
    - "207" και "Peugeot" Strings
    - CarTypes κλήση συνάρτησης
    - Sales μεταβλητή που δηλώθηκε πρωτύτερα
  - Χρήση: document.write("I own a " + car.type);

Τρόποι δημιουργίας Object

- Μέσω literal:
  - var course = { number: "YS14", teacher: "IC" }
- Μέσω του τελεστή new δημιουργείται αντικείμενο χωρίς πεδία, τα οποία προστίθενται αργότερα
  - var course = new Object();
    course.number = "YS14";
    course.teacher = "IC";
- Μέσω της συγγραφής συνάρτησης constructor:
  - function Course(n, t) { // εντός του <head>
        this.number = n; // απαιτείται το this
        this.teacher = t;
    }
  - var course = new Course("YS14", "IC");

Literal πινάκων

- Π.χ.
  - color = ["red", "yellow", "green", "blue"];
  - color[0] είναι "red" (αρίθμηση από το 0)
- Αν υπάρχουν 2 συνεχόμενα κόμματα, στην εν λόγω θέση υπάρχει κενό στοιχείο
- Π.χ.: color = ["red", , , "green", "blue"];
  - Ο πίνακας color έχει 5 στοιχεία
  - Τα κόμματα στο τέλος αγνοούνται
    - Π.χ. ο πίνακας color = ["red", , , "green", "blue",]; έχει ακόμη 5 στοιχεία

Τρόποι δημιουργίας πινάκων

- Μέσω literal
  var colors = ["red", "green", "blue"];
- Μέσω της έκφρασης new Array() για τη δημιουργία πίνακα χωρίς στοιχεία:
  - var colors = new Array();
  - Στη συνέχεια μπορούν να προστεθούν. Το μέγεθος δεν είναι προκαθορισμένο
    colors[0] = "red"; colors[2] = "blue"; colors[1]="green";
- new Array(***n***) δημιουργεί πίνακα μεγέθους n
  - var colors = new Array(3);
- Η χρήση του new Array(…) με 2 ή περισσότερα ορίσματα, δημιουργεί πίνακα με στοιχεία τα ορίσματα:
  - var colors = new Array("red","green", "blue");

Το μέγεθος του πίνακα

- Αν η μεταβλητή myArray είναι πίνακας, το μέγεθος του είναι myArray.length
- Το μέγεθος του πίνακα μπορεί να αλλάξει με ανάθεση εκτός του τρέχοντος μεγέθους του
  - Π.χ.: var myArray = new Array(5); myArray[10] = 3;
- Οι πίνακες είναι sparse, δηλαδή εκχωρείται χώρος μόνο για τα στοιχεία στα οποία έχει ανατεθεί τιμή
  - Π.χ.: myArray[50000] = 3;
  - Οι δείκτες των πινάκων είναι εντός του διαστήματος $[0,2^{32}-1]$
- Δισδιάστατοι πίνακες, πίνακες πινάκων: myArray[5][3]

Πίνακες και αντικείμενα

- Οι πίνακες είναι αντικείμενα.
- car = { myCar: "Saturn", 7: "Mazda" }
  - Ισοδύναμα car[7] ~ car.7
  - Ισοδύναμα car.myCar ~ car["myCar"]
- Αν είναι γνωστό το όνομα ενός πεδίου προτιμάται: car.myCar
- Αν δεν είναι γνωστό το όνομα ενός πεδίου, αλλά υπάρχει σε μεταβλητή ή μπορεί να υπολογιστεί, **πρέπει** να χρησιμοποιηθεί το notation πίνακα car["my" + "Car"]

Συναρτήσεις πινάκων

- Έστω πίνακας myArray
  - myArray.sort() αλφαβητική ταξινόμηση
  - myArray.sort(function(a, b) { return a - b; }) αριθμητική ταξινόμηση βάσει συνάρτησης
  - myArray.reverse() αντιστροφή σειράς στοιχείων
  - myArray.push(...) εισαγωγή ενός ή περισσότερων νέων στοιχείων στο τέλος του πίνακα με αύξηση του μεγέθους του
  - myArray.pop() αφαίρεση και επιστροφή του τελευταίου στοιχείου ενός πίνακα με μείωση του μεγέθους του
  - myArray.toString() επιστρέφεται string με τις τιμές ων στοιχείων χωρισμένες με κόμματα

Ο βρόχος for...in

- Μπορεί εύκολα να διατρέξουμε τα πεδία ενός αντικειμένου με το βρόχο for (***variable*** in ***object***) ***statement***;
  - Π.χ.:
```
for (var prop in course) {
    document.write(prop + ": " + course[prop]);
}
```
  - Έξοδος:
```
teacher: IC
number: YS14
```
  - Η σειρά με την οποία διατρέχονται τα πεδία δεν είναι καθορισμένη
  - Αν προστίθενται ή αφαιρούνται πεδία στο αντικείμενο εντός του βρόχου, η συμπεριφορά είναι ακαθόριστη!

Αστοχίες βρόχου for…in

- Δεν πρέπει να εφαρμόζεται σε πίνακες!
- Δεν διατρέχει απαραίτητα όλα τα πεδία ενός αντικειμένου
  - Π.χ. τα built-in πεδία δεν είναι απαριθμήσιμα (βλ. myArr.length)

Η εντολή with

- Η εντολή with (***object***) ***statement*** ; έχει σαν αποτέλεσμα ότι το ***object*** αποτελεί default πρόθεμα των μεταβλητών στην εντολή ***statement***
- Π.χ. τα παρακάτω είναι ισοδύναμα:
  - with (document.myForm) {
    result.value = compute(myInput.value) ;
    }
  - document.myForm.result.value =
    compute(document.myForm.myInput.value);
- Η χρήση του with μπορεί να κάνει τον κώδικα δυσανάγνωστο

Regular expressions

- 2 τρόποι γραφής:
  - Εντός καθέτων re = /ab+c/<flags>
  - Με constructor re = new RegExp("ab+c")
- re.exec(***string***) εκτελεί αναζήτηση του ***regexp*** εντός του ***string***
  - Επιστρέφει null αν δεν βρεθεί
  - Αν έχει τεθεί το flag g /ab+c/g επιστρέφει πίνακα του οποίου το πρώτο στοιχείο είναι το πρώτο substring που κάνει match. Αν κληθεί εκ νέου ξεκινά την αναζήτηση μετά το πρώτο matched substring, μέχρι να φτάσει στο τέλος του string οπότε επιστρέφει null και ξεκινά από την αρχή
  - Αν δεν έχει τεθεί το flag g η αναζήτηση ξεκινά από την αρχή κάθε φορά

Υποστήριξη από browsers

- Παλιότεροι browser μπορεί να έχουν προβλήματα συμβατότητας

Ασφάλεια

- Δεν επιτρέπεται
  - Η ανάγνωση και η εγγραφή από/σε αρχεία στο σύστημα αρχείων του χρήστη
  - Η εκτέλεση εξωτερικών προγραμμάτων
  - Η σύνδεση με άλλους υπολογιστές εκτός από τη μεταφόρτωση HTML σελίδων και την αποστολή e-mail

Debugging

- Firefox
  - F12 -> debugger, console κλπ.
- Chrome
  - More Tools -> Developer Tools

Αριθμητικές Σταθερές

- Infinity, Number.POSITIVE_INFINITY – το αποτέλεσμα της διαίρεσης θετικού αριθμού με το μηδέν
- Number.NEGATIVE_INFINITY -- το αποτέλεσμα της διαίρεσης αρνητικού αριθμού με το μηδέν
- NaN, Number.NaN (Not a Number) – το αποτέλεσμα της διαίρεσης 0/0
  - Η σταθερά NaN έχει σχέση ανισότητας με ο,τιδήποτε (και με τον εαυτό της)
  - Υπάρχει καθολική συνάρτηση isNaN() function
- Number.MAX_VALUE – ο μεγαλύτερος αριθμός που μπορεί να αναπαρασταθεί
- Number.MIN_VALUE – ο μικρότερος αριθμός που μπορεί να αναπαρασταθεί εντός του [0,1]

Συμβολοσειρές και χαρακτήρες

- Συμβολοσειρές: πρωτογενής τύπος δεδομένων
- Εντός διπλών ή μονών εισαγωγικών
- Δεν υπάρχει τύπος για χαρακτήρες
- Escaped χαρακτήρες εντός συμβολοσειρών:

\0	NUL	\v	vertical tab
\b	backspace	\'	single quote
\f	form feed	\"	double quote
\n	newline	\\	backslash
\r	carriage return	\x*DD*	Unicode hex *DD*
\t	horizontal tab	\x*DDDD*	Unicode hex *DDDD*

Μέθοδοι String

- charAt(***n***)
  - Επιστρέφει τον n χαρακτήρα ενός string
  - concat(***string1***, …, ***stringN***)
  - Συνενώνει N string στο string όπου εφαρμόζεται
- indexOf(***substring***)
  - Επιστρέφει τη θέση του πρώτου χαρακτήρα της πρώτης εμφάνισης του ***substring*** στο string όπου εφαρμόζεται ή -1 αν δεν βρεθεί
- indexOf(***substring***, ***start***)
  - Όπως πριν αλλά ξεκινάει από τη θέση start
- lastIndexOf(***substring***), lastIndexOf(***substring***, ***start***)
  - Όπως η indexOf, αλλά η αναζήτηση ξεκινά από το τέλος του string

Μέθοδοι String (2)

- match(***regexp***)
  - Επιστρέφει πίνακα με τα αποτελέσματα, ή null αν δεν γίνει match
- replace(***regexp***, ***replacement***)
  - Επιστρέφει νέο string όπου τα substring που ταιριάζουν αντικαθίστανται από τη συμβολοσειρά ***replacement***
- search(***regexp***)
  - Επιστρέφει τη θέση του πρώτου substring που ταιριάζει στο string ή -1 αν δεν βρεθεί

boolean

- Τιμές true και false
- Κατά τον μετασχηματισμό τους σε boolean μόνο οι κάτωθι τιμές μετατρέπονται σε false
  - 0
  - "0" και '0'
  - Το κενό string, " ή ""
  - undefined
  - null
  - NaN

undefined και null

- Ειδικές τιμές undefined και null
- undefined έχει δικό της τύπο του οποίου είναι η μοναδική τιμή
  - Αυτή είναι η τιμή μιας μεταβλητής που έχει δηλωθεί αλλά δεν έχει οριστεί και η τιμή του πεδίου ενός αντικειμένου που δεν υπάρχει
- void τελεστής που όταν εφαρμοστεί σε οποιαδήποτε τιμή επιστρέφει undefined
- null είναι πρωτογενής τύπος δεδομένων
- Οι τιμές null και undefined είναι == αλλά όχι ===

Εύρεση τύπου

- Ο μοναδιαίος τελεστής typeof επιστρέφει ένα εκ των ακόλουθων string "number", "string", "boolean", "object", "undefined", και "function"
  - typeof null δίνει "object" για λόγους συμβατότητας με παλιότερο κώδικα
  - Αν myArray πίνακας , typeof myArray είναι "object"
- Για να γίνει διάκριση μεταξύ διαφορετικών τύπων αντικειμένων χρησιμοποιείται ο τελεστής instanceof,
  - ***myObject*** instanceof ***Constructor***
    - Το αντικείμενο πρέπει να έχει δημιουργηθεί με συνάρτηση ***Constructor***

Wrappers και μετατροπές τύπου

- Η JavaScript έχει “wrapper” αντικείμενα όταν θέλουμε να χειριστούμε έναν πρωτογενή τύπο ως αντικείμενο
  - var s = new String("Hello"); // s is now a String
  - var n = new Number(5); // n is now a Number
  - var b = new Boolean(true); // b is now a Boolean
  - Σπανίως χρησιμοποιούνται λόγω των αυτόματων μετατροπών που κάνει εξ ορισμού η JavaScript
- Δεν υπάρχει “cast”
  - var s = x + ""; // s is now a string
  - var n = x + 0; // n is now a number
  - var b = !!x; // b is now a boolean

Μεταβλητές

- Κάθε μεταβλητή είναι πεδίο αντικειμένου
- Όταν ξεκινά το πρόγραμμα δημιουργείται *global αντικείμενο*
- Στην client-side JavaScript, το *window* είναι το global αντικείμενο
  - Παίρνουμε αναφορά σε αυτό με το window ή με το this
  - Οι “built-in” μεταβλητές και μέθοδοι είναι ορισμένες εδώ
- Οι τοπικές μεταβλητές σε μια συνάρτηση είναι πεδία του αντικειμένου της συνάρτησης που δημιουργείται κατά την κλήση της

Ονόματα HTML στη JavaScript

- Στην HTML *window* είναι το global αντικείμενο
  - Το πιο σημαντικό πεδίο του window είναι το document
- Μπορούμε να πάρουμε αναφορά στα στοιχεία form μέσω document.forms[***formNumber***].elements[***elementNumber***]
- Κάθε στοιχείο form μπορεί να έχει πεδίο name
- Το name μπορεί να χρησιμοποιηθεί αντί για τον πίνακα
  - Π.χ.
    - <form name="myForm">
      <input type="button" name="myButton" id="myButton"...>
    - Αντί για document.forms[0].elements[0]
    - document.myForm.myButton ή
    - document.myForm.elements["myButton"]

Συναρτήσεις

- Στη JavaScript, κάθε συνάρτηση είναι *αντικείμενο*
- Υποστηρίζεται η αναδρομή:
  - ```
    function factorial(n) {
        if (n <= 1) return 1;
        else return n * factorial(n - 1);
    }
    ```
- Υποστηρίζεται ο εμφωλιασμός:
  - ```
    function hypotenuse(a, b) {
        function square(x) { return x * x; }
        return Math.sqrt(square(a) + square(b));
    }
    ```

Ο constructor Function()

- Εφόσον οι συναρτήσεις είναι αντικείμενα έχουν constructor:
  - Function(***arg1***, ***arg2***, …, ***argN***, ***body***)
  - Όλα τα ορίσματα του constructor είναι συμβολοσειρές
  - Π.χ.:

```
var f = new Function("x", "y", "return x * y;");
```

- Μέσω της ανάθεσης σε μεταβλητή η συνάρτηση λαμβάνει όνομα
- Επομένως, μπορούν να δημιουργηθούν συναρτήσεις δυναμικά
  - Η μεταγλώττιση είναι υπολογιστικά ακριβή σε αυτή την περίπτωση
- Όταν ορίζονται συναρτήσεις με constructor είναι πάντα καθολικές

Function expressions

- Μπορεί μια συνάρτηση να οριστεί literally ως εξής
  - var f = function(x, y) { return x * y; }
  - Δεν είναι απαραίτητα global
- Για να υποστηρίζεται η αναδρομή πρέπει να της δοθεί όνομα
  - var f = function fact(n) { if (n <= 1) return n;
    else return n * fact(n - 1) ; };
  - Το όνομα αυτό δεν διατηρείται μετά τη δημιουργία της συνάρτησης

Function names

- Το όνομα μιας συνάρτησης είναι η μεταβλητή που αναφέρεται σε αυτήν
  - var square = function(x) { return x * x; };
  - var a = square(4); // a έχει τιμή 16
  - var b = square; // b αναφέρεται στη συνάρτηση square
  - var c = b(5); // c έχει τιμή 25
  - var d = [b]; // d είναι πίνακας, πρώτο στοιχείο το αντικείμενο d της συνάρτησης
  - var e = d[0](6); // e έχει τιμή 36

Το function object

- Όταν καλείται μια συνάρτηση αποδίδονται τιμές στα πεδία του αντικειμένου της συνάρτησης
- Τα πεδία του αντικειμένου περιλαμβάνουν:
  - Τις παραμέτρους της συνάρτησης
  - Τις τοπικές μεταβλητές
  - Το αντικείμενο arguments

arguments

- Το αντικείμενο arguments μοιάζει με πίνακα
  - arguments[***n***] είναι συνώνυμο του ***n*** ορίσματος
  - arguments.length είναι ο αριθμός των ορισμάτων με τα οποία καλέστηκε η συνάρτηση
    - ***function***.length είναι ο αριθμός των ορισμάτων με τα οποία ορίστηκε η συνάρτηση
  - arguments.callee αναφέρεται στο ίδιο το αντικείμενο της συνάρτησης

Παραδείγματα χρήσης arguments

- function max() {
  var m = Number.NEGATIVE_INFINITY;
  for (var i = 0; i < arguments.length; i++) {
  if (arguments[i] > m) m = arguments[i];
  }
  return m;
  }
- function(n) {
  if (n <= 1) return 1;
  return n * arguments.callee(n - 1);
  }

Πεδία συναρτήσεων (1)

- length – Ο αριθμός των *τυπικών* παραμέτρων
- arguments – Το αντικείμενο Arguments που είναι σαν πίνακας των πραγματικών παραμέτρων της κλήσης
- caller – Η καλούσα συνάρτηση ή null αν η κλήση έγινε στο ανώτερο επίπεδο του καθολικού αντικειμένου

Πεδία συναρτήσεων (2)

- Εφόσον οι συναρτήσεις είναι αντικείμενα μπορούν να τους προστεθούν πεδία
  - Καλή εναλλακτική σε σχέση με τις καθολικές μεταβλητές
  - Π.χ.

```
uniqueInteger.counter = 0;
function uniqueInteger() {
    return uniqueInteger.counter++;
}
```

  - Μοιάζουν με τις στατικές μεταβλητές

Συναρτήσεις και μέθοδοι

- Όταν μια συνάρτηση είναι πεδίο ενός αντικειμένου ονομάζεται μέθοδός του
- Πρώτα δημιουργούμε ένα αντικείμενο:
  - function Point(xcoord, ycoord) {
    this.x = xcoord; // το this είναι υποχρεωτικό
    this.y = ycoord;
    }
  - myPoint = new Point(3, 5);
- Μέθοδος είναι μια συνάρτηση που σχετίζεται και καλείται μέσω ενός αντικειμένου (επομένως χρησιμοποιεί το this)
- Η συνάρτηση που ακολουθεί δεν έχει από μόνη της νόημα:
  - function distance(x2, y2) {
    function sqr(x) { return x * x; }
    return Math.sqrt(sqr(this.x – x2) + sqr(this.y – y2));
    }

Μέθοδοι

- Μπορεί η συνάρτηση να μετατραπεί σε μέθοδο αν προστεθεί ως πεδίο ενός αντικειμένου:
  - myPoint.dist = distance;
- Τώρα η αναφορά this στη συνάρτηση αναφέρεται στο myPoint, και μπορούμε να γράψουμε:
  - document.write("The distance is " + myPoint.dist(6, 9));
- Εναλλακτικά:
  - Οι μέθοδοι καλούνται ως εξής:
    call(***object***, ***arg1***, ..., ***argN***) και
    apply(*object*, [***arg1***, ..., ***argN***])
  - document.write("The distance is " + distance.call(myPoint, 6, 9));
  - document.write("The distance is " + distance.apply(myPoint, [6, 9]));

Μέθοδοι (2)

- Οι συναρτήσεις call και apply είναι ορισμένες για όλες τις συναρτήσεις
  - call λαμβάνει οσοδήποτε αριθμό παραμέτρων
  - apply λαμβάνει πίνακα παραμέτρων
- Έτσι δεν γίνεται μόνιμη συσχέτιση μιας συνάρτησης με ένα αντικείμενο, αλλά μια συνάρτηση καλείται *σαν* να ήταν μέθοδος ενός αντικειμένου *object*
- Η αναφορά this εντός της συνάρτησης αναφέρεται στο αντικείμενο *object*

Μέθοδοι (3)

- Με τον προηγούμενο τρόπο μια συνάρτηση προσαρτάται σε ένα συγκεκριμένο αντικείμενο
- Αν θέλουμε να προσαρτήσουμε μία μέθοδο σε **όλα** τα αντικείμενα που δημιουργούνται από έναν constructor, την προσθέτουμε ως πεδίο, στο πεδίο prototype του constructor
  - Point.prototype.dist = distance;
- Π.χ. είναι πολύ χρήσιμο να προσθέσουμε τη μέθοδο toString

```
Point.prototype.toString =
    function() {
        return "(" + this.x + ", " + this.y + ")";
    };
```

JavaScript και DOM

- Η JavaScript εργάζεται πάνω σε ένα Document Object Model (DOM) που περιγράφει τη δομή μιας ιστοσελίδας
  - Δεν πρέπει να συγχέεται με το XML DOM
- Το DOM χρησιμοποιείται για:
  - να γίνει ανάγνωση, επεξεργασία και αλλαγή των στοιχείων της ιστοσελίδας
  - για τη σύλληψη γεγονότων πάνω στην ιστοσελίδα

Γεγονότα (events)

- Ορισμένα στοιχεία της ιστοσελίδας είναι διαδραστικά σε ενέργειες του χρήστη (πληκτρολόγιο, mouse) με την έννοια ότι δημιουργούν γεγονότα
- Διαφορετικά είδη στοιχείων παράγον διαφορετικά γεγονότα
  - Υπάρχουν διαφορές από browser σε browser
  - Θα επικεντρωθούμε σε γεγονότα που δημιουργούνται από στοιχεία εντός στοιχείων form και άλλα συχνά δημιουργούμενα γεγονότα
- Μπορεί να εισαχθεί κώδικας JavaScript για τη διαχείριση των γεγονότων
  - Οι χειριστές πρέπει να έχουν μικρό μέγεθος καλώντας κατάλληλες συναρτήσεις

Παράδειγμα

- <form method="post" action="">
  <input type="button"
  name="myButton"
  value="Click me"
  onClick="alert('You clicked the button!');">
  </form>
  - onClick το γεγονός του οποίου θα γίνει η διαχείριση
    - Η τιμή του χαρακτηριστικού onClick είναι ο κώδικας JavaScript που θα εκτελεστεί
    - Η built-in συνάρτηση alert δημιουργεί ένα alert box με το εν λόγω κείμενο

Εισαγωγικά

- onClick="alert('You clicked the button!');"
- Όταν έχουμε string εντός string πρέπει να χρησιμοποιηθούν μονά εισαγωγικά

Συνήθη γεγονότα

- onClick
- onDblClick
- onMouseDown – το κουμπί του mouse πατιέται όταν ο κέρσορας βρίσκεται πάνω από ένα στοιχείο
- onMouseOver – ο κέρσορας μετακινείται πάνω από το στοιχείο
- onMouseOut – ο κέρσορας μετακινείται εκτός στοιχείου
- onMouseUp – αφήνουμε το κουμπί του mouse πάνω από ένα στοιχείο
- onMouseMove – ο κέρσορας μετακινείται
- Στη JavaScript πρέπει να χρησιμοποιούνται πεζά

Παράδειγμα

- Ο κάτωθι κώδικας κάνει το κείμενο Hello εντός του h1
  κόκκινο όταν ο κέρσορας μετακινείται πάνω του
  μπλε όταν ο κέρσορας απομακρύνεται

```
<h1 onMouseOver="style.color='red';"
    onMouseOut="style.color='blue';">Hello </h1>
```

- Εναλλαγή εικόνων

```
<img src="../Images/duke.gif"
     width="55" height="68"
     onMouseOver="src='../Images/duke_wave.gif';"
     onMouseOut="src='../Images/duke.gif';">
```

Γεγονότα και χειριστές (1)

Γεγονός	Εφαρμογή	Συμβαίνει	Χειριστής
Load	body	Φόρτωση της σελίδας	onLoad
Unload	body	Έξοδος από τη σελίδα	onUnload
Error	Images, window	Σφάλμα κατά τη φόρτωση εικόνας, παραθύρου	onError
Abort	Images	Διακοπή φόρτωσης εικόνας	onAbort

Γεγονότα και χειριστές (2)

Γεγονός	Εφαρμογή	Συμβαίνει	Χειριστής
KeyDown	Documents, images, links, text areas	Πάτημα πλήκτρου	onKeyDown
KeyUp	Documents, images, links, text areas	Άφημα κουμπιού	onKeyUp
KeyPress	Documents, images, links, text areas	Πληκτρολόγηση χαρακτήρα	onKeyPress
Change	Text fields, text areas, select lists	Αλλαγή τιμής στοιχείου	onChange

Γεγονότα και χειριστές (3)

Γεγονός	Εφαρμογή	Συμβαίνει	Χειριστής
MouseDown	Documents, buttons, links	Πάτημα κουμπιού mouse	onMouseDown
MouseUp	Documents, buttons, links	Το κουμπί αφήνεται	onMouseUp
Click	Buttons, radio buttons, checkboxes, submit buttons, reset buttons, links	Πάτημα αριστερού κουμπιού	onClick

Γεγονότα και χειριστές (4)

Γεγονός	Εφαρμογή	Συμβαίνει	Χειριστής
MouseOver	Links	Μετακίνηση κέρσορα πάνω από στοιχείο	onMouseOver
MouseOut	Areas, links	Μετακίνηση κέρσορα εκτός στοιχείου	onMouseOut
Select	Text fields, text areas	Επιλογή στοιχείου εισόδου	onSelect

Γεγονότα και χειριστές (5)

Γεγονός	Εφαρμογή	Συμβαίνει	Χειριστής
Move	Windows	Μετακίνηση παραθύρου	onMove
Resize	Windows	Αλλαγή μεγέθους παραθύρου	onResize
DragDrop	Windows	Drag and Drop στο παράθυρο	onDragDrop

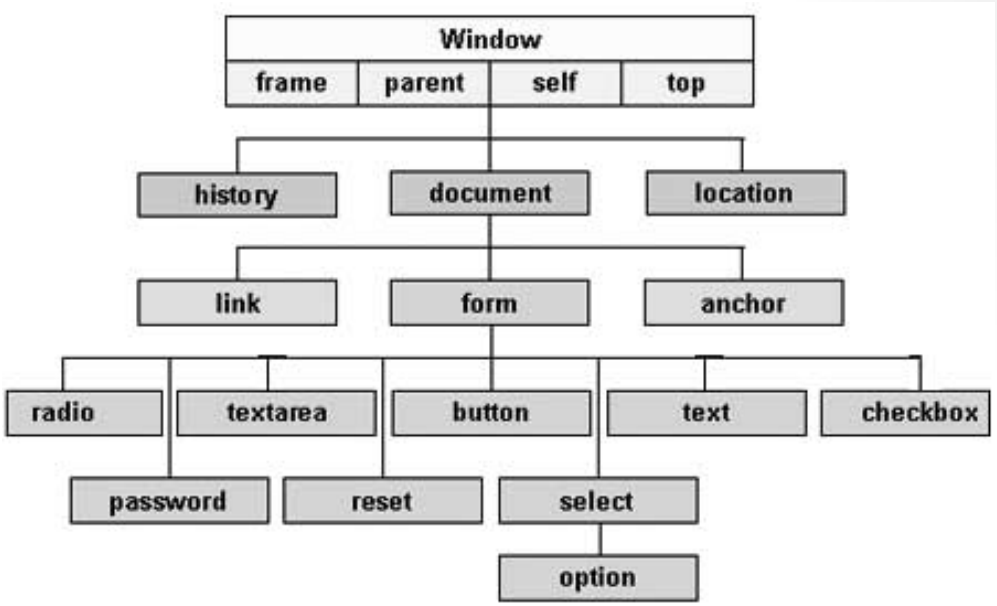
Γεγονότα και χειριστές (6)

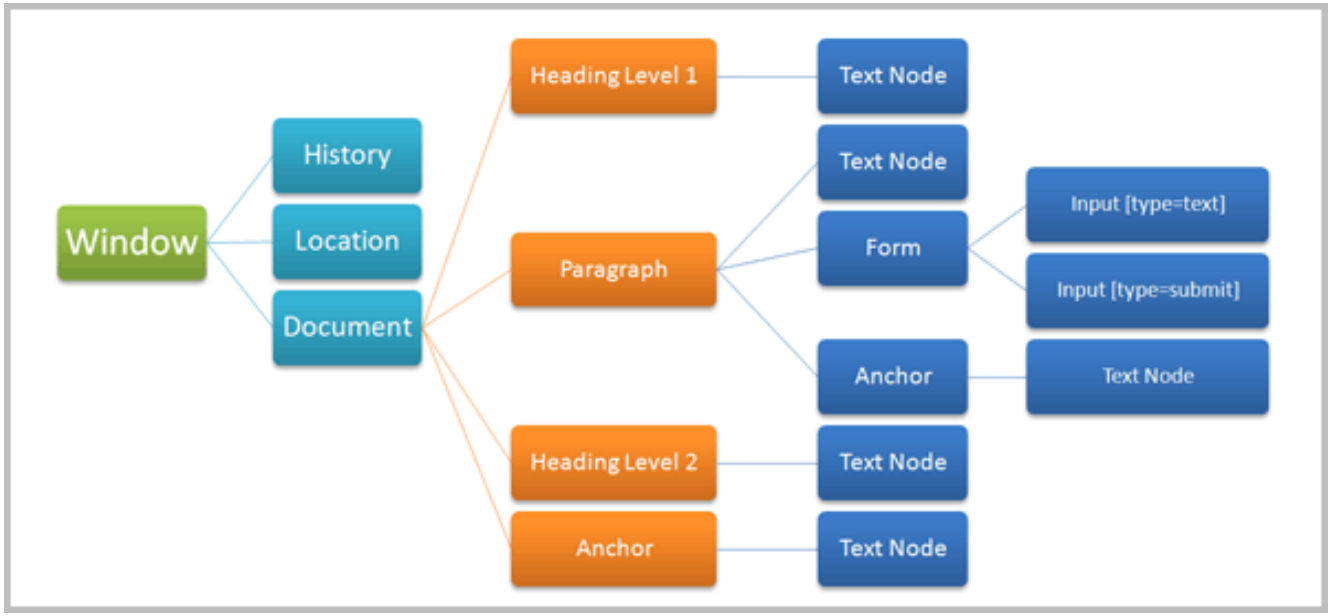
Γεγονός	Εφαρμογή	Συμβαίνει	Χειριστής
Reset	Forms	Πάτημα κουμπιού reset	onReset
Submit	Forms	Πάτημα κουμπιού submit	onSubmit

HTML DOM

- Πρότυπο του W3C ώστε να συμμορφώνονται οι browsers
- Το υψηλότερο στοιχείο στην ιεραρχία του DOM για την τρέχουσα σελίδα είναι το window
  - Κάθε μεταβλητή JavaScript είναι πεδίο κάποιου αντικειμένου
  - Επομένως όλες οι μεταβλητές για τα στοιχεία μιας ιστοσελίδας ξεκινούν από το αντικείμενο "window."

Ιεραρχία DOM





Πεδία του window (1)

- window
  - Το τρέχον παράθυρο
- self
  - Το τρέχον παράθυρο
- document
  - Το έγγραφο HTML που προβάλλεται εντός του παράθυρου
- location
  - Το URL του εγγράφου που προβάλλεται. Αν αυτό το πεδίο αλλάξει και κληθεί η συνάρτηση location.reload() το παράθυρο θα ανανεωθεί

Μέθοδοι του window (1)

- alert(string)
  - Εμφανίζει ένα κουτί διαλόγου προειδοποίησης που περιέχει το string και ένα κουμπί OK
- confirm(string)
  - Εμφανίζει ένα κουτί διαλόγου επιβεβαίωσης που περιέχει το string και 2 κουμπιά OK και Cancel. Επιστρέφει true αν πατηθεί το OK, false αν πατηθεί το Cancel
- prompt(string)
  - Εμφανίζει κουτί διαλόγου που περιέχει το string, ένα textbox και κουμπιά OK και Cancel. Επιστρέφει το κείμενο που εισάγει ο χρήστης αν πατηθεί το OK, null αν πατηθεί το Cancel.

Μέθοδοι του window (2)

- open(URL)
  - Άνοιγμα νέου παραθύρου με ανάγνωση του εγγράφου που βρίσκεται στη διεύθυνση URL
- close()
  - Κλείσιμο του παραθύρου

Πεδία του document (1)

- Τα πεδία αυτά ακολουθούν το document.
- anchors[ ]
  - Πίνακας αντικειμένων Anchor (που αντιστοιχούν στα στοιχεία <a name=...>)
- forms[ ]
  - Πίνακας αντικειμένων Form
    - Αν υπάρχει μόνο μία φόρμα: forms[0]
- images[ ]
  - Πίνακας αντικειμένων Image objects
    - Για την αλλαγή της εικόνας, εισάγεται νέο URL στο πεδίο src
- links[ ]
  - Πίνακας αντικειμένων Link

Πεδία του document (2)

- title
  - Συμβολοσειρά read-only με τον τίτλο του εγγράφου
- URL
  - Συμβολοσειρά read-only με το URL του εγγράφου

Πεδία του αντικειμένου form

- elements[ ]
  - Πίνακας των στοιχείων της φόρμας

AJAX

- **A**synchronous **J**avaScript **A**nd **X**ML

Τρόπος λειτουργίας

1. Ο χρήστης αλληλεπιδρά με μια σελίδα HTML μέσω π.χ. μιας φόρμας
2. Η JavaScript στη σελίδα HTML στέλνει διαφανώς για τον χρήστη μια αίτηση HTTP στον server
3. Ο Server απαντά: συνήθως δεν επιστρέφεται μια ολόκληρη ιστοσελίδα αλλά κάποια δεδομένα
4. Η JavaScript χρησιμοποιεί αυτά τα δεδομένα για να τροποποιήσει τη σελίδα: ο χρήστης δεν αντιλαμβάνεται αίτηση / απόκριση

- Είναι ταχύτερο γιατί λιγότερα δεδομένα μεταφέρονται και ο browser εκτελεί λιγότερες ενέργειες

Τεχνολογίες που εμπλέκονται

- JavaScript
- HTML
- CSS
- XML / JSON
- HTTP

XMLHttpRequest

- Η JavaScript χειρίζεται γεγονότα από τις φόρμες, δημιουργεί αντικείμενο XMLHttpRequest και το στέλνει στον server
  - Παρά το όνομά του το αντικείμενο XMLHttpRequest δεν απαιτεί XML
  - var request = new XMLHttpRequest();
  - Παλιότερα υπήρχε ασυμβατότητα στον Internet Explorer
  - ***request***.open(***method***, ***URL***, ***asynchronous***)
    - Η παράμετρος ***method*** είναι συνήθως 'GET' ή 'POST' (όχι απαραίτητα με REST API)
  - Η παράμετρος ***URL*** είναι η διεύθυνση του server
  - Η παράμετρος ***asynchronous*** είναι boolean. Αν είναι true, ο browser δεν περιμένει την απόκριση του server (αναμενόμενη συμπεριφορά)
  - ***request***.open(***method***, ***URL***) (***asynchronous*** είναι true)

Αποστολή αντικειμένου XMLHttpRequest

- ***request***.send(null);
  - Όταν η μέθοδος είναι GET
- ***request***.send(***content***);
  - Μπορούν να τεθούν και άλλες ιδιότητες του Request
  - Π.χ.:

```
request.setRequestHeader('Content-Type',
                         'application/x-www-form-urlencoded');
request.send('var1=' + value1 + '&var2=' + value2);
```

Server

- Συνήθως έχουμε REST API. Δεν είναι απαραίτητο όμως
- Αντί να επιστρέφεται μια πλήρης σελίδα επιστρέφεται String (JSON, XML, plain text)

Λήψη της απόκρισης

- Ο κώδικας Ajax εκτελείται ασύγχρονα. Δεν αναμένεται η αποστολή της απόκρισης
- Η απόκριση λαμβάνεται μέσω του χειρισμού γεγονότος
  - request.onreadystatechange = someFunction;
    - Ανάθεση συνάρτησης σε πεδίο που αντιστοιχεί σε γεγονός
  - function someFunction() {
    if(request.readyState == 4 && request.status == 200){
    var response = request.responseText;
    // ***Ektelesi kapoias allagis tis istoselidas***
    }
    }

readyState
0: request not initialized
1: server connection established
2: request received
3: processing request
4: request finished and response is ready

Χρήση του response

- Η συνάρτηση που καλείται με την αλλαγή του readystate δεν λαμβάνει παραμέτρους
  - Χρήση ανώνυμης συνάρτησης:
    request.onreadystatechange = function() {
    someFunction(request); }
    - Η ανώνυμη συνάρτηση καλεί κάποια άλλη someFunction με παράμετρο το αντικείμενο request

Εμφάνιση του response

- http_request.onreadystatechange =
  function() { alertContents(http_request); };
  http_request.open('GET', url, true);
  http_request.send(null);

- function alertContents(http_request) {
  if (http_request.readyState == 4) {
  if (http_request.status == 200) {
  alert(http_request.responseText);
  } else {
  alert('There was a problem with the request.');
  }
  }
  }

Πεδίο readyState

- Κατάσταση του άντικειμένου XMLHttpRequest
  - **readyState=0** αφού δημιουργηθεί το αντικείμενο XMLHttpRequest πριν κληθεί η μέθοδος open()
  - **readyState=1** αφού κληθεί η open() method, προτού κληθεί η send().
  - **readyState=2** αφού κληθεί η send().
  - **readyState=3** αφού εγκαθιδρυθεί επικοινωνία με τον server, προτού ληφθεί η απόκριση
  - **readyState=4** αφού ληφθεί η απόκριση

Πεδίο innerHTML

- innerHTML χρησιμοποιείται για αλλαγές μεγάλης κλίμακας στη σελίδα (non-standard προ HTML5)
  - Όταν τίθεται το πεδίο innerHTML ενός αντικειμένου που αντιστοιχεί σε στοιχείο HTML, το string που δίνεται αντικαθιστά πλήρως το υπάρχον περιεχόμενο του στοιχείου
- Σύνταξη:
  var ***markup*** = element.innerHTML;
  element.innerHTML = ***markup***;
- Παράδειγμα:
  document.getElementById(***someId***).innerHTML = ***response***;

Περίληψη

- Δημιουργία αντικειμένου XMLHttpRequest έστω ***request***
- Χρήση κατάλληλου URL προσθέτοντας παραμέτρους αν χρειάζεται (GET) ?***var1***=***value1***&***var2***=***value2***
- ***request***.open('GET', ***URL***)
- ***request***.onreadystatechange = ***handlerMethod***;
- ***request***.send(null);
- function handlerMethod() {
  if (***request***.readyState == 4) {
  // do stuff
  }
  }
  }