JavaScript

Η JavaScript είναι μια scripting γλώσσα, η οποία αναπτύχθηκε από τη Netscape για να επεκτείνει τις δυνατότητες της HTML και αρχικά λειτουργούσε μόνο στο browser της εταιρίας. Στη συνέχεια υποστηρίχθηκέ και από τους υπόλοιπους browsers οι οποίοι πρόσθεσαν τις δικές τους επεμβάσεις στη γλώσσα (JScript της Microsoft για Internet Explorer). Με τη JavaScript μπορείς εύκολα να αναπτύξεις μια interactive σελίδα (μια σελίδα δηλαδή που αλληλεπιδρά στον εαυτό της). Η εισαγωγή αυτή δείχνει τι μπορεί να γίνει μέσω της JavaScript - και πιο σημαντικό πώς μπορεί να γίνει. Να τονίσουμε ότι παρά την ομοιότητα του ονόματος η JavaScript δεν έχει καμία σχέση με τη Java τόσο όσο αφορά τη λειτουργικότητα και τη δομή όσο αφορά και το δημιουργό (η Java είναι δημιούργημα της Sun).

Απαιτήσεις της JavaScript

Τι χρειάζεται ώστε να τρέξουν scripts γραμμένα σε JavaScript; Ένας browser που υποστηρίζει JavaScript – κι αφού όλοι οι browsers πλέον υποστηρίζουν τη JavaScript μπορούμε να πούμε "έναν browser".

Εισαγωγή JavaScript σε μια HTML σελίδα

Ο κώδικας JavaScript εισάγεται απευθείας στην HTML σελίδα. Για να δείτε πώς αυτό δουλεύει, θα κοιτάξουμε ένα απλό παράδειγμα:

```
<html>
<body>
<br>
Αυτό είναι ένα απλό κανονικό HTML έγγραφο.
<br>
<script type="text/javascript" >
document.write("Αυτό είναι JavaScript!")
</script>
<br>
Ξανά σε HTML.
</body>
</html>
```

Από πρώτη ματιά φαίνεται σαν ένα κανονικό HTML έγγραφο. Το μόνο καινούργιο μέρος είναι το:

```
<script type="text/javascript">
document.write("Αυτό είναι JavaScript!")
</script>
```

Αυτό λέγεται JavaScript. Για να δείτε αυτό το script να δουλεύει, σώστε τον πάνω κώδικα σαν ένα απλό HTML έγγραφο και ανοίξτε το σε ένα browser.

Οτιδήποτε μεταξύ <script> και </script> (δηλαδή μέσα στην ετικέτα tag) μεταφράζεται σαν JavaScript κώδικας.

Εδώ βλέπετε την χρήση του document.write() - μία από τις πιο σημαντικές εντολές στη JavaScript

προγραμματισμό. Η document.write() χρησιμοποιείται για να γράψουμε κάτι στο πραγματικό έγγραφο (σε αυτή τη περίπτωση, το HTML έγγραφο). Έτσι το μικρό μας JavaScript πρόγραμμα γράφει Αυτό είναι JavaScript! στο HTML έγγραφο.

Non-JavaScript browsers

Πώς φαίνεται η σελίδα μας όταν ο browser δεν καταλαβαίνει JavaScript; Ένας non-JavaScript browser δεν ξέρει τι σημαίνει <script>. Αγνοεί το <script> και βγάζει όλο τον ακόλουθο κώδικα λες και είναι κανονικό κείμενο. Αυτό σημαίνει ότι ο χρήστης θα δει τον JavaScript κώδικα μέσα στο HTML έγγραφο. Αυτό βεβαίως δεν ήταν ο σκοπός μας. Υπάρχει τρόπος για να κρύψουμε τον JavaScript κώδικα από παλιούς browsers. Θα χρησιμοποιήσουμε τα HTML comments <!-- -->. Ο νέος μας κώδικας θα δείχνει ως εξής:

```
<html>
<body>
<br>
Αυτό είναι ένα απλό κανονικό HTML έγγραφο.
<br>
<script type="text/javascript">
<!-- Κρύβουμε τον κώδικα
document.write("Αυτό είναι JavaScript!")
// -->
</script>
<br>
Ξανά σε HTML.
</body>
</html>
```

Το αποτέλεσμα σε ένα non-JavaScript browser θα είναι το εξής:

```
Αυτό είναι ένα απλό κανονικό HTML έγγραφο.
Ξανά σε HTML.
```

Χωρίς τα HTML-comments το αποτέλεσμα σε ένα non-JavaScript browser θα ήταν:

```
Αυτό είναι ένα απλό κανονικό HTML έγγραφο.
document.write("Αυτό είναι JavaScript!")
Ξανά σε HTML.
```

Σημειώστε ότι δεν μπορείτε να κρύψετε τον JavaScript κώδικα τελείως. Αυτό που κάνουμε εδώ είναι να εμποδίσουμε το λάθος αποτέλεσμα του κώδικα σε παλιούς browsers - αλλά ο χρήστης μπορεί να δει τον κώδικα επιλέγοντας την λειτουργία 'View document source' του browser που χρησιμοποιεί. ΔΕΝ υπάρχει τρόπος να εμποδίσουμε κάποιον από το να δει τον πηγαίο κώδικα (με σκοπό να μη δει πώς γίνεται κάτι).

Events - Γεγονότα

Τα γεγονότα (Events) και οι χειριστές γεγονότων (event handlers) είναι ένα πολύ σημαντικό μέρος στον JavaScript προγραμματισμό. Τα Events προκαλούνται από τις πράξεις του χρήστη. Αν ο χρήστης πατήσει ένα κουμπί, τότε

συμβαίνει ένα Click-event. Αν ο δείκτης του mouse κινηθεί πάνω από μια διεύθυνση (link), τότε συμβαίνει ένα MouseOver-event. Υπάρχουν πολλά διαφορετικά events.

Θέλουμε το JavaScript πρόγραμμά μας να αντιδρά σε συγκεκριμένα events. Αυτό μπορεί να γίνει με την βοήθεια των event-handlers (χειριστές γεγονότων). Ένα κουμπί μπορεί να εμφανίζει ένα pop-up παράθυρο όταν πατιέται. Αυτό σημαίνει ότι το pop-up παράθυρο πρέπει να εμφανιστεί σαν απάντηση στο Click-event. Ο event-handler που χρειαζόμαστε λέγεται onClick. Αυτός λέει στον υπολογιστή τι να γίνει όταν συμβεί το ανάλογο event. Ο επόμενος κώδικας δείχνει ένα απλό παράδειγμα του event- handler onClick:

```
<form>
<input type="button" value="Πάτησε με" onClick="alert('Καλημέρα')">
</form>
```

Υπάρχουν πολλά νέα πράγματα σε αυτόν τον κώδικα - γι' αυτό ας τα δούμε βήμα- βήμα. Μπορείτε να δείτε ότι σχεδιάζουμε μία φόρμα με ένα κουμπί (αυτό γίνεται με HTML). Το νέο μέρος είναι onClick="alert('Καλημέρα')" μέσα στο <input>. Όπως είπαμε ήδη, αυτό καθορίζει τι θα συμβεί όταν το κουμπί πατηθεί. Έτσι αν ένα Click-event συμβεί, ο υπολογιστής θα εκτελέσει το alert('Καλημέρα'). Αυτό είναι JavaScript κώδικας (Σημειώστε ότι δεν χρησιμοποιούμε το <script> σε αυτή τη περίπτωση).

Η εντολή alert() σου επιτρέπει να δημιουργείς popup παράθυρα. Μέσα στη παρένθεση πρέπει να βάλετε το κείμενο του pop-up παραθύρου. Σε αυτή τη περίπτωση είναι 'Καλημέρα'. Έτσι το script δημιουργεί ένα παράθυρο με το κείμενο 'Καλημέρα' όταν ο χρήστης πατήσει το κουμπί. Ένα πράγμα που μπερδεύει λίγο: Στην εντολή document.write() χρησιμοποιήσαμε διπλό εισαγωγικό " και στο συνδυασμό alert() χρησιμοποιήσαμε απλό ' - γιατί; Βασικά μπορούμε να χρησιμοποιήσουμε και τα δύο. Όμως στο τελευταίο παράδειγμα γράψαμε onClick="alert('Καλημέρα')" - βλέπετε ότι χρησιμοποιούνται και τα δύο σύμβολα. Αν γράφαμε onClick="alert("Καλημέρα")" ο υπολογιστής θα μπερδευόταν μιας και δεν είναι εμφανές ποιο μέρος ανήκει στο onClick event-handler και ποίο όχι. Γι' αυτό πρέπει να εναλλαχθούν τα διπλά με μονό σε αυτή τη περίπτωση. Δεν έχει σημασία με ποια σειρά θα χρησιμοποιηθούν τα δύο σύμβολα. Αυτό σημαίνει ότι θα μπορούσα επίσης να γράψω onClick='alert("Καλημέρα")' και να έχω το ίδιο αποτέλεσμα.

Υπάρχουν πολλοί διαφορετικοί event-handlers που μπορούν να χρησιμοποιηθούν. Θα μάθετε μερικούς καθώς προχωράτε - μα όχι όλους. Απευθυνθείτε σε συγκεκριμένα έγγραφα αν θέλετε να ξέρετε όλους τους event-handlers που υπάρχουν.

Το popup παράθυρο θα περιέχει το κείμενο JavaScript alert στο caption Bar (στο πάνω πάνω αριστερά μέρος του παραθύρου). Αυτός είναι περιορισμός ασφαλείας. Μπορείτε να δημιουργήσετε ένα παρόμοιο popup παράθυρο με την εντολή prompt(). Αυτό το παράθυρο δέχεται και τιμή που καθορίζει ο χρήστης για να χρησιμοποιηθεί αργότερα απ' το script. Π.χ. μπορεί αν χρησιμοποιηθεί σε ένα script που μιμείται ένα σύστημα προστασίας και να ρωτά για ένα συγκεκριμένο password. Το κείμενο στο popup παράθυρο δείχνει ότι το παράθυρο έρχεται από τον browser και όχι από το Operating system. Μιας και το Caption Bar είναι περιορισμός ασφαλείας, δεν μπορεί να αλλάξει.

Χρησιμοποιούμε συναρτήσεις (functions) στα περισσότερα scripts.

Οι συναρτήσεις (functions) είναι ένας τρόπος για να ενσωματώνουμε πολλές εντολές μαζί. Το ακόλουθο script βγάζει τρεις φορές το ίδιο κείμενο:

```
<html>
<script type="text/javascript">
<!-- Κρύβουμε τον πηγαίο κώδικα
document.write("Καλωσήρθατε στη σελίδα!<br>");
document.write("Αυτό είναι JavaScript!<br>");
document.write("Καλωσήρθατε στη σελίδα!<br>");
document.write("Αυτό είναι JavaScript!<br>");
document.write("Καλωσήρθατε στη σελίδα!<br>");
document.write("Αυτό είναι JavaScript!<br>");
// -->
</script>
</html>
```

Αυτό θα γράψει το κείμενο

```
Καλωσήρθατε στη σελίδα!
Αυτό είναι JavaScript!
3 φορές
```

Κοιτάχτε τον πηγαίο κώδικα - γράφοντας τον κώδικα τρεις φορές βγάζει το σωστό αποτέλεσμα. Για μεγαλύτερη αποτελεσματικότητα θα μπορούσαμε να χρησιμοποιήσουμε τον ακόλουθο κώδικα:

```
<html>
<script type="text/javascript">
<!-- hide
function myFunction() { document.write("Καλωσήρθατε στη σελίδα!<br>"); document.write("Αυτό
είναι JavaScript!<br>");
}
myFunction(); myFunction(); myFunction();
// -->
</script>
</html>
```

Σε αυτό το script προσδιορίζουμε μία function. Αυτό γίνεται από τις γραμμές:

```
function myFunction() { document.write("καλωσήρθατε στη σελίδα!<br>"); document.write("Αυτό
είναι JavaScript!<br>");
}
```

Οι εντολές μέσα στα { } ανήκουν στην λειτουργία myFunction() που δημιούργησα. Αυτό σημαίνει ότι τα δύο document.write() ενοποιούνται, γίνονται ο ορισμός της function και μπορούν να εκτελεστούν αν καλέσουμε την ανάλογη function. Στο παράδειγμά μας, έχουμε τρεις κλίσεις του function myFuction. Βλέπετε ότι γράφουμε τρεις φορές myFunction() ακριβώς κάτω από τον ορισμό της function. Αυτό σημαίνει ότι το περιεχόμενο της function

εκτελείται τρεις φορές.

Αυτό είναι ένα πολύ εύκολο παράδειγμα μιας function. Ίσως αναρωτιέστε γιατί οι functions είναι τόσο σημαντικές. Καθώς διαβάζετε αυτή την εισαγωγή θα συνειδητοποιήσετε τα πλεονεκτήματα μιας function. Ειδικά η ρύθμιση μεταβλητών κάνει τα scripts πιο εύκαμπτα - θα καταλάβετε τι είναι αυτό αργότερα.

Οι functions μπορούν επίσης να χρησιμοποιηθούν σε συνδυασμό με event-handlers. Κοιτάξτε το παρακάτω παράδειγμα:

```
<html>
<head>
<script type="text/javascript">
<!-- Κρύβουμε τον κώδικα
function calculation() {
var x= 12;
var y= 5;
var result= x + y;
alert(result);
}
// -->
</script>
</head>
<body>
<form>
<input type="button" value="Calculate" onClick="calculation()">
</form>
</body>
</html>
```

Το κουμπί καλεί την function calculation().

Η function αυτή κάνει συγκεκριμένους υπολογισμούς. Για αυτό χρησιμοποιούμε τις μεταβλητές x, y και result. Μπορούμε να καθορίσουμε την τιμή της μεταβλητής χρησιμοποιώντας την εντολή var. Οι μεταβλητές μπορούν να χρησιμοποιηθούν για να αποθηκεύσουν διαφορετικού είδους τιμές - όπως νούμερα, κείμενο, κλπ. Η γραμμή var result= x + y; λέει στον browser να δημιουργήσει μια μεταβλητή result και να αποθηκεύσει σ' αυτή το αποτέλεσμα της πράξης x + y (δηλαδή 5 + 12). Μετά από αυτό, η μεταβλητή result παίρνει την τιμή 17. Η εντολή alert(result) είναι σε αυτή την περίπτωση ίδια με την εντολή alert(17). Αυτό σημαίνει ότι το αποτέλεσμα είναι ένα popup παράθυρο με την τιμή 17 πάνω του.

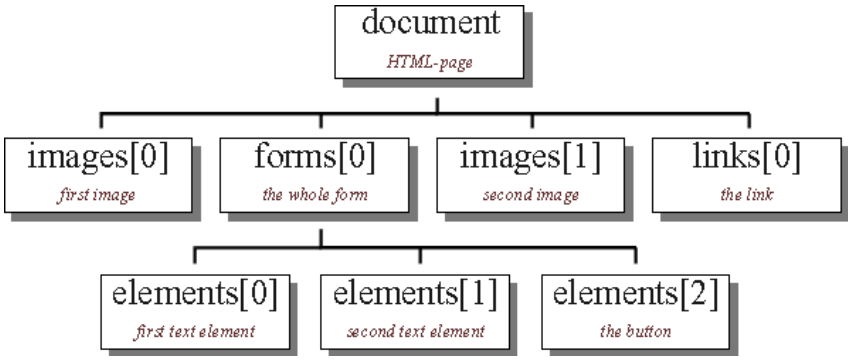
Ιεραρχία JavaScript

Η JavaScript οργανώνει όλα τα στοιχεία μιας web σελίδας σε μια ιεραρχία. Κάθε στοιχείο της σελίδας θεωρείται αντικείμενο. Κάθε αντικείμενο έχει τις δικές του ιδιότητες (properties) και μεθόδους (methods). Με τη βοήθεια του JavaScript μπορούμε εύκολα να διαχειριστούμε τα αντικείμενα αφού πρώτα καταλάβουμε πως αντιλαμβάνεται η JavaScript ένα έγγραφο HTML. Ο ακόλουθος κώδικας είναι μια απλή HTML σελίδα. Θα το

χρησιμοποιήσουμε ως παράδειγμα για να δείξουμε την ιεραρχική δομή των ιστοσελίδων και το μοντέλο διαχείρισής τους από την HTML.

```
<html>
<head>
<title>My homepage
</head>
<body bgcolor=#ffffff>
<center>
<img src="home.gif" name="pic1" width=200 height=100>
</center>
<p>
<form name="myForm"> Name:
<input type="text" name="name" value=""><br>
e-Mail:
<input type="text" name="email" value=""><br><br>
<input type="button" value="Push me" name="myButton" onClick="alert('Yo')">
</form>
<p>
<center>
<img src="ruler.gif" name="pic4" width=300 height=15>
<p>
<a href="http://www.uni.gr/~my/"> My homepage</a>
</center>
</body>
</html>
```

Έχουμε δύο εικόνες, ένα link και μία φόρμα με δύο πεδία κειμένου και ένα κουμπί. Από πλευράς JavaScript το παράθυρο του browser βλέπεται σαν ένα αντικείμενο, εν ονόματι window. Το αντικείμενο windows περιέχει συγκεκριμένα στοιχεία, όπως η StatusBar. Μέσα στο παράθυρο, μπορούμε να φορτώσουμε ένα HTML έγγραφο (ή άλλου τύπου αρχείου - προς το παρόν, θα περιοριστούμε σε HTML αρχεία). Αυτή η σελίδα είναι το αντικείμενο document. Αυτό σημαίνει ότι το αντικείμενο document αναπαριστά το HTML έγγραφο το οποίο βλέπει ο χρήστης εκείνη τη στιγμή. Το document είναι ένα πολύ σημαντικό αντικείμενο στη JavaScript - θα χρησιμοποιείται συχνά. Οι ιδιότητες (properties) του document είναι για παράδειγμα το χρώμα του φόντου. Αλλά το πιο σημαντικό είναι ότι όλα τα αντικείμενα HTML είναι ιδιότητες του document. Αντικείμενα είναι για παράδειγμα τα links, ή οι φόρμες. Η παρακάτω εικόνα παρουσιάζει την ιεραρχία του HTML εγγράφου που δείξαμε παραπάνω:



Θέλουμε να μπορούμε να πάρουμε πληροφορίες για τα διάφορα αντικείμενα και να τα διαχειριστούμε. Γι' αυτό πρέπει να ξέρουμε πώς να έχουμε πρόσβαση στα διάφορα αντικείμενα. Για να απευθυνθούμε στην πρώτη εικόνα (image) πρέπει να ξεκινήσουμε απ' την αρχή. Το πρώτο αντικείμενο λέγεται document. Η πρώτη εικόνα της σελίδας αντιπροσωπεύεται από το images[0]. Αυτό σημαίνει ότι μπορούμε να έχουμε πρόσβαση στην εικόνα τυπώνοντας document.images[0].

Αν για παράδειγμα θέλετε να ξέρετε τι τιμή έχει το πρώτο πεδίο κειμένου, πρέπει να σκεφτείτε ποια διαδρομή μας δίνει πρόσβαση σε αυτό το πεδίο κειμένου. Ξανά αρχίζουμε από το πάνω μέρος της ιεραρχίας. Ακολουθήστε το "μονοπάτι" που φτάνει στο elements[0] - βάλτε όλα τα ονόματα που συναντάτε στη διάρκεια του μονοπατιού μαζί. Αυτό σημαίνει ότι έχετε πρόσβαση στο πρώτο πεδίο κειμένου με: document.forms[0].elements[0] Αλλά τώρα πώς θα ξέρουμε το εισαχθέν κείμενο; Για να μάθουμε τις ιδιότητες και τις μεθόδους ενός αντικειμένου, πρέπει να διαβάσουμε την ανάλογη JavaScript reference. Εδώ βλέπετε ότι ένα πεδίο κειμένου έχει και την ιδιότητα value. Η ιδιότητα αυτή αντιπροσωπεύει το κείμενο (την τιμή του πεδίου). Τώρα μπορείτε να διαβάσετε το κείμενο που εισήγαγε ο χρήστης:

```
name= document.forms[0].elements[0].value;
```

Η τιμή του κειμένου αποθηκεύεται στη μεταβλητή name. Μπορούμε τώρα να δουλέψουμε με αυτή τη μεταβλητή. Για παράδειγμα, μπορούμε να δημιουργήσουμε ένα popup παράθυρο alert("Hi " + name). Αν η τιμή είναι 'Stefan' η εντολή alert("Hi " + name) θα ανοίξει ένα popup παράθυρο με το κείμενο 'Hi Stefan' Αν έχετε ιδιαίτερα μεγάλες σελίδες το να απευθύνεστε στα αντικείμενα με αριθμούς μπορεί να σας μπερδέψει αρκετά - π.χ. document.forms[3].elements[17] ή μήπως ήταν document.forms[2].elements[18]; Για να αποφύγετε αυτό το πρόβλημα μπορείτε να ονομάσετε τα διάφορα αντικείμενα. Ακολουθεί ένα τέτοιο παράδειγμα:

```
<form name="myForm"> Name:
<input type="text" name="name" value=""><br>
...
```

Αυτό σημαίνει ότι το forms[0] λέγεται επίσης myForm. Αντί για elements[0] μπορούμε να χρησιμοποιήσουμε το name (όπως προσδιορίζει η ιδιότητα NAME="" στο <input>). Έτσι αντί να γράφουμε

```
name= document.forms[0].elements[0].value;
```

μπορούμε να γράψουμε

```
name= document.myForm.name.value;
```

Αυτό το κάνει πολύ πιο εύκολο - ειδικά όταν έχουμε μεγάλες σελίδες με πολλά αντικείμενα. (Σημειώστε ότι το αν τα γράμματα είναι μικρά ή κεφαλαία έχει σημασία - αυτό σημαίνει ότι δεν μπορείτε να γράψετε myform αντί για myForm). Πολλές ιδιότητες των αντικειμένων δεν περιορίζονται στην δυνατότητα να τις "διαβάζεις". Μπορείς

κάλλιστα να ρυθμίσεις τις τιμές τους. Για παράδειγμα μπορείτε να γράψετε ένα νέο κείμενο το πεδίο κειμένου. Κοιτάχτε το παρακάτω παράδειγμα:

Ιδού ο κώδικας του παραδείγματος - το ενδιαφέρον κομμάτι είναι μέσα στην ιδιότητα onClick-property στο δεύτερο <input>:

```
<form name="myForm">
<input type="text" name="input" value="Μπλα, μπλα">
<input type="button" value="Write" onClick= "document.myForm.input.value= 'Καλημέρα!'; ">
```

Δεν μπορώ να περιγράψω κάθε λεπτομέρεια εδώ. Γίνεται όμως πολύ καθαρότερο αν προσπαθήσετε να καταλάβετε την ιεραρχία με τη βοήθεια μιας JavaScript reference. Έχω γράψει εδώ ένα μικρό παράδειγμα. Εδώ μπορείτε να δείτε τη χρήση διάφορων αντικειμένων. Προσπαθήστε να καταλάβετε το script με τη βοήθεια μιας reference.

Πρώτα ας δούμε τον πηγαίο κώδικα:

```
<html>
<head>
<title>Objects</title>
<script type="text/javascript">
<!-- hide
function first() {
// δημιουργεί ένα popup παράθυρο με το
// κείμενο που έχει εισαχθεί στο πεδίο κειμένου
alert("Η τιμή του πεδίου κειμένου είναι: " +
document.myForm.myText.value);
}
function second() {
// Το function κοιτάζει το Check Box
var myString= "The checkbox is ";
// Είναι το CheckBox τικαρισμένο ή όχι?
if (document.myForm.myCheckbox.checked)
myString+= "επιλεγμένο"
else myString+= "μη επιλεγμένο";
// output string alert(myString);
}
// -->
</script>
</head>
<body bgcolor=lightblue>
<form name="myForm">
<input type="text" name="myText" value="Μπλα, μπλα">
<input type="button" name="button1" value="Κουμπί 1"
onClick="first()">
<br>
```

```
<input type="checkbox" name="myCheckbox" CHECKED>
<input type="button" name="button2" value="Κουμπί 2"
onClick="second()">
</form>
<p><br><br>
<script type="text/javascript">
<!-- hide
document.write("Το χρώμα στο φόντο είναι: ");
document.write(document.bgColor + "<br>");
document.write("Το κείμενο στο κουμπί 2 είναι:");
document.write(document.myForm.button2.value);
// -->
</script>
</body>
</html>
```

Το αντικείμενο location

Εκτός από τα αντικείμενα window και document υπάρχει άλλο ένα σημαντικό αντικείμενο: το location. Αυτό το αντικείμενο αντιπροσωπεύει την διεύθυνση του HTML εγγράφου που απεικονίζεται. Έτσι αν φορτώσατε την σελίδα http://www.xyz.com/page.html τότε η τιμή του location.href είναι ίδια με την πάνω διεύθυνση.
Ακόμα πιο σημαντικό είναι ότι μπορείτε να βάλετε νέες τιμές στο location.href. Αυτό το κουμπί, παραδείγματος χάριν, ανοίγει ένα άλλο έγγραφο σε αυτό το παράθυρο:

```
<form>
<input type=button value="Yahoo" onClick="location.href='http://www.yahoo.com'; ">
</form>
```

Δημιουργώντας frames

Μια ερώτηση που εμφανίζεται αρκετά συχνά είναι πώς μπορούν frames και JavaScript να συνεργαστούν. Πρώτα, θέλω να εξηγήσω τι είναι τα frames και σε τι χρησιμεύουν. Μετά απ' αυτό, θα δούμε πώς μπορούμε να συνδυάσουμε frames και JavaScript. Το παράθυρο του browser μπορεί να χωριστεί σε πολλά frames. Αυτό σημαίνει ότι frame είναι μια τετράγωνη περιοχή μέσα σε ένα παράθυρο του browser. Κάθε frame δείχνει το δικό του έγγραφο (τις περισσότερες φορές HTML έγγραφα). Έτσι μπορείτε για παράδειγμα, να δημιουργήσετε δυο frames. Στο πρώτο να φορτώσετε τη σελίδα της Netscape και στο δεύτερο τη σελίδα της Microsoft.
Παρόλο που η δημιουργία frames είναι υπόθεση της HTML, θέλω να περιγράψω μερικά βασικά πράγματα. Για να δημιουργήσουμε frames, χρειαζόμαστε δύο εντολές: <frameset> και <frame>. Ένα έγγραφο με δύο frames μπορεί να δείχνει ως εξής:

```
<html>
<frameset rows="50%,50%">
```

```
<frame src="page1.htm" name="frame1">
<frame src="page2.htm" name="frame2">
</frameset>
</html>
```

Αυτό θα παράγει δύο frames. Μπορείτε να δείτε ότι χρησιμοποιούνται οι ιδιότητες rows στην εντολή <frameset>. Αυτό σημαίνει ότι τα δύο frames είναι το ένα πάνω στο άλλο. Το πάνω frame απεικονίζει την σελίδα page1.htm και το κάτω τη σελίδα page2.htm

Αν θέλετε να έχετε στήλες (columns) αντί για σειρές (rows), πρέπει να γράψετε cols αντί για rows στην εντολή <frameset>. Το "50%,50%" δείχνει πόσο μεγάλα θα είναι τα δύο έγγραφα. Μπορείτε επίσης να γράψετε "50%,*" αν δεν θέλετε να υπολογίσετε πόσο μεγάλο πρέπει να είναι το δεύτερο παράθυρο για να φτάσει το 100%. Μπορείτε επίσης να γράψετε το μέγεθος σε pixels, αν παραλείψετε το σύμβολο %. Κάθε frame παίρνει τη δικιά του name ιδιότητα στην εντολή <frame>. Αυτό θα μας βοηθήσει να έχουμε πρόσβαση στα frames μέσω JavaScript.

Μπορείτε να έχετε πολλά <frameset> το ένα να φωλιάζεται μέσα στο άλλο.

```
<frameset cols="50%,50%">
<frameset rows="50%,50%">
<frame src="cell.htm">
<frame src="cell.htm">
</frameset>
<frameset rows="33%,33%,33%">
<frame src="cell.htm">
<frame src="cell.htm">
<frame src="cell.htm">
</frameset>
</frameset>
```

Μπορείτε να ρυθμίσετε το μέγεθος της διαχωριστικής γραμμής, αλλάζοντας την τιμή της ιδιότητας borderστο <frameset>. border=0 σημαίνει ότι δεν θέλετε να υπάρχει διαχωριστική γραμμή.

Frames και JavaScript

Τώρα θέλουμε να ρίξουμε μια ματιά στο πώς η JavaScript 'βλέπει' τα frames σε ένα παράθυρο του browser. Γι' αυτό θα δημιουργήσουμε δύο frames, όπως στο πρώτο παράδειγμα, του πρώτου μέρους. Έχουμε δει πως η JavaScript ταξινομεί όλα τα στοιχεία του παραθύρου σε μια ιεραρχική σειρά. Το ίδιο γίνεται και με τα frames. Η επόμενη εικόνα δείχνει την ιεραρχία του πρώτου παραδείγματος:

Στο πάνω μέρος της ιεραρχίας έχουμε το window. Το παράθυρο αυτό είναι χωρισμένο σε δύο frames. Το παράθυρο είναι ο γονέας (στην αγγλική γλώσσα parent) σε αυτή την ιεραρχία και τα δύο frames είναι τα παιδιά (children). Δώσαμε στα δύο frames τα ονόματα frame1 και frame2. με τη βοήθεια των δύο ονομάτων μπορούμε να ανταλλάσσουμε πληροφορίες μεταξύ των δύο frames. Ένα script πιθανώς θα πρέπει να λύσει το εξής πρόβλημα: ο χρήστης διαλέγει ένα link στο ένα frame - αλλά ο συγγραφέας θέλει η σελίδα να εμφανιστεί στο δεύτερο frame, κι όχι στο πρώτο. Αυτό μπορεί να χρησιμοποιηθεί για σελίδες Menu (εν ονόματι MenuBars ή

NavigationBars) όπου το ένα frame μένει συνέχεια το ίδιο και προσφέρει πολλά διαφορετικά links, για την περιπλάνηση μέσα στην σελίδα του συγγραφέα. Πρέπει να κοιτάξουμε τρεις περιπτώσεις:

1) Το parent window/frame αποκτά πρόσβαση στο child frame

2) Το child frame αποκτά πρόσβαση στο parent window/frame

3) Το child frame αποκτά πρόσβαση στο άλλο child frame

Από την πλευρά του παραθύρου τα δύο frames λέγονται frame1 κει frame2. Μπορείτε να δείτε στην πάνω εικόνα ότι υπάρχει απ' ευθείας σχέση μεταξύ parent window και κάθε frame. Έτσι αν έχουμε ένα script στο parent window - αυτό σημαίνει στην σελίδα που δημιουργεί τα frames - και θέλουμε να έχουμε πρόσβαση στα frames, μπορούμε να χρησιμοποιήσουμε απλώς το όνομα του frame. Για παράδειγμα, μπορείτε να γράψετε:

```
frame2.document.write("Ένα μήνυμα απ' το πατέρα παράθυρο.");
```

Μερικές φορές χρειάζεστε να έχετε πρόσβαση στο parent window από ένα frame. Αυτό χρειάζεται π.χ. αν θέλουμε να σβήσουμε τα frames. Σβήνοντας τα frames, σημαίνει να φορτωθεί μια νέα σελίδα στη θέση της σελίδας που δημιούργησε τα frames. Αυτό είναι στην περίπτωσή μας το parent window. Μπορούμε να έχουμε πρόσβαση στο parent window (ή parent frame) από τα child frames με το αντικείμενο parent. Για να φορτώσουμε ένα νέο document, πρέπει να δώσουμε νέα τιμή στο location.href. Εφόσον θέλουμε να διώξουμε τα frames, πρέπει να χρησιμοποιήσουμε το αντικείμενο location του parent window. Εφόσον το κάθε frame μπορεί να φορτώσει την δικιά του σελίδα έχουμε διαφορετικό αντικείμενο location για κάθε frame. Μπορούμε να φορτώσουμε νέα σελίδα στο parent window με την εντολή:

```
parent.location.href= "http://...";
```

Πολύ συχνά θα χρειαστείτε να έχετε πρόσβαση σε ένα child frame από ένα άλλο child frame. Πώς μπορείτε να γράψετε κάτι από το πρώτο frame στο δεύτερο frame - αυτό σημαίνει ποια εντολή πρέπει να χρησιμοποιήσετε στο κείμενο page1.htm; Στην εικόνα, βλέπετε ότι δεν υπάρχει απ' ευθείας σύνδεση μεταξύ των δύο frames. Αυτό σημαίνει ότι δεν μπορούμε απλώς να καλέσουμε frame2 από το frame1 μιας και αυτό το frame δεν ξέρει τίποτα για την ύπαρξη του άλλου frame. Για το parent window το δεύτερο frame λέγεται frame2 και το parent window λέγεται parent για το πρώτο frame. Γι' αυτό πρέπει να γράψουμε τα εξής για να αποκτήσουμε πρόσβαση στο αντικείμενο document του δεύτερου frame:

```
parent.frame2.document.write("Γειά, το frame1 επιφέρει αυτές τις αλλαγές.");
```

Navigation bars

Ας κοιτάξουμε ένα navigationbar. Θα έχουμε αρκετά links σε ένα frame. Αν ο χρήστης διαλέξει ένα απ' αυτά τα links, η σελίδα δεν θα εμφανιστεί στο ίδιο frame - θα εμφανιστεί στο άλλο frame.

Αρχίζοντας, χρειαζόμαστε ένα script το οποίο δημιουργεί τα frames. Αυτό το έγγραφο μοιάζει με το πρώτο παράδειγμα. :Ονομάστε το frames3.htm

```
<html>
<frameset rows="80%,20%">
<frame src="start.htm" name="main">
<frame src="menu.htm" name="menu">
</frameset>
```

```
</html>
```

Η σελίδα start.htm είναι η σελίδα που θα δείχνει το main frame στην αρχή.

Δεν υπάρχουν ιδιαίτερες απαιτήσεις γι' αυτή τη σελίδα. Η επόμενη σελίδα είναι το frame menu (menu.htm):

```
<html>
<head>
<script type="text/javascript">
<!-- hide
function load(url) {
parent.main.location.href= url;
}
// -->
</script>
</head>
<body>
<a href="javascript:load('first.htm')">Πρώτο link</a>
<a href="second.htm" target="main">Δεύτερο link</a>
<a href="third.htm" target="_top">Τρίτο link</a>
</body>
</html>
```

Πάνω μπορείτε να δείτε τρεις διαφορετικούς τρόπους για να φορτώσετε μια σελίδα στο frame main.

Το πρώτο link χρησιμοποιεί τη function load(). Κοιτάξτε πώς καλείται αυτή η function:

```
<a href="javascript:load('first.htm')">first</a>
```

Μπορείτε να δείτε ότι μπορούμε να αφήσουμε τον browser να εκτελέσει το κώδικα JavaScript αντί για να φορτώσει άλλη σελίδα - απλώς πρέπει να χρησιμοποιήσουμε την παράμετρο JavaScript: στην ιδιότητα href. Μπορείτε να δείτε πως γράφω 'first.htm' μέσα στη παρένθεση. Η τιμή αυτή περνά στην function load(). Η function load() έχει ρυθμιστεί ως:

```
function load(url) {
parent.main.location.href= url;
}
```

Το url μέσα στην παρένθεση σημαίνει ότι η τιμή 'first1.htm' αποθηκεύεται στη μεταβλητή url. Μέσα στη function load() μπορούμε τώρα να χρησιμοποιήσουμε την μεταβλητή. Θα δούμε πιο αναλυτικά παραδείγματα αυτής της ιδιότητας των μεταβλητών και των functions αργότερα. Το δεύτερο link χρησιμοποιεί την ιδιότητα target. Αυτό, στην πραγματικότητα δεν είναι JavaScript.

Είναι ενέργεια της HTML. Βλέπετε ότι πρέπει να καθορίσουμε το όνομα του frame. Σημειώστε ότι δεν πρέπει να βάλουμε parent πριν απ' το όνομα του frame. Αυτό μπορεί να σας μπερδέψει λίγο. Ο λόγος γι' αυτό είναι ότι το target είναι HTML και όχι JavaScript. Το τρίτο link δείχνει πώς να βγάλουμε τα frames χρησιμοποιώντας την ιδιότητα target. Αν θέλετε να βγάλετε τα frames με την function load() πρέπει απλώς να γράψετε parent.location.href= url μέσα στο function.

Ποιο τρόπο πρέπει να διαλέξετε; Αυτό εξαρτάται απ' το script και τι θέλετε να κάνετε. Η ιδιότητα target είναι πολύ απλή στη χρήση. Μπορείτε να τη χρησιμοποιήσετε αν θέλετε να φορτώσετε άλλη σελίδα σε κάποιο frame. Η λύση

JavaScript (όπως το πρώτο link) χρησιμοποιείται συνήθως όταν θέλουμε να κάνουμε πολλά πράγματα σαν απάντηση στο πάτημα του link. Ένα συνηθισμένο πρόβλημα είναι να φορτώσετε δύο σελίδες σε δύο διαφορετικά frames. Παρόλο που μπορείτε να το λύσετε αυτό χρησιμοποιώντας την ιδιότητα target, χρησιμοποιώντας JavaScript functions είναι πιο άμεσο. Ας πούμε ότι έχετε τρία frames, με ονόματα frame1,frame2 και frame3. Ο χρήστης πατά ένα link στο frame1. Τότε θέλετε να φορτώσετε δύο διαφορετικές σελίδες στα δύο άλλα frames. Μπορείτε, παραδείγματος χάριν, να χρησιμοποιήσετε την επόμενη function:

```
function loadtwo() { parent.frame1.location.href= "first.htm"; parent.frame2.location.href= "second.htm";
}
```

Αν θέλετε να διατηρήσετε την function πιο εύκαμπτη, μπορείτε να χρησιμοποιήσετε variable passing. Η function θα δείχνει ως εξής:

```
function loadtwo(url1, url2) { parent.frame1.location.href= url1;
parent.frame2.location.href= url2;
}
```

Μπορείτε να καλείτε αυτή τη function γράφοντας loadtwo("first.htm", "second.htm") ή loadtwo("third.htm", "forth.htm"). Χρησιμοποιώντας Variable passing κάνει τη function πιο εύκαμπτη. Μπορείτε να τη χρησιμοποιήσετε ξανά και ξανά με διαφορετικό περιεχόμενο.

Παράθυρα

Δημιουργώντας παράθυρα

Ανοίγοντας νέα παράθυρα είναι μια σπουδαία λειτουργία του JavaScript. Μπορείτε να φορτώσετε ένα νέο έγγραφο (για παράδειγμα ένα έγγραφο HTML) σε ένα νέο παράθυρο ή να δημιουργήσετε νέα έγγραφα (on-the-fly). Πρώτα, θα ρίξουμε μια ματιά στο πώς δημιουργούμε ένα νέο παράθυρο, φορτώνουμε μια HTML σελίδα σε αυτό το παράθυρο και μετά το κλείνουμε. Το ακόλουθο script ανοίγει ένα νέο παράθυρο και φορτώνει μια σελίδα:

```
<html>
<head>
<script type="text/javascript">
<!-- hide
function openWin() {
myWin= open("bla.htm");
}
// -->
</script>
</head>
<body>
<form>
<input type="button" value="Άνοιγμα νέου παραθύρου"
onClick="openWin()" >
</form>
```

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

Η σελίδα bla.htm εμφανίζεται στο νέο παράθυρο χρησιμοποιώντας την method open(). Μπορείτε να τροποποιήσετε τη μορφή του νέου παραθύρου. Για παράδειγμα, μπορείτε να αποφασίσετε αν θα έχει statusbar, toolbar ή menubar. Εκτός αυτού, μπορείτε να ρυθμίσετε το μέγεθος του παραθύρου. Το ακόλουθο script ανοίγει ένα νέο παράθυρο το οποίο έχει μέγεθος 400x300. Το παράθυρο δεν θα 'χει statusbar, toolbar ή menubar.

```
<html>
<head>
<script type="text/javascript">
<!-- hide
function openWin2() {
myWin= open("bla.htm", "displayWindow",
"width=400,height=300,status=no, toolbar=no,menubar=no");
}
// -->
</script>
</head>
<body>
<form>
<input type="button" value="Άνοιγμα νέου παραθύρου"
onClick="openWin2()">
</form>
</body>
</html>
```

Μπορείτε να δείτε ότι μπορούμε να προσδιορίσουμε τις παραμέτρους γράφοντας "width=400,height=300,status=no,toolbar=no,menubar=no". Σημειώστε ότι <u>δεν πρέπει να υπάρχουν κενά</u> (spaces) στο κείμενο αυτό!

Εδώ είναι μια λίστα με τις παραμέτρους που μπορεί να έχει ένα παράθυρο:

- directories yes|no
- height αριθμός pixels
- location yes|no
- menubar yes|no
- resizable yes|no
- scrollbars yes|no
- status yes|no
- toolbar yes|no
- width αριθμός pixels
- alwaysLowered yes|no
- alwaysRaised yes|no

- dependent yes|no
- hotkeys yes|no
- innerWidth αριθμός pixels (αντικαθιστά το width)
- innerHeight αριθμός pixels (αντικαθιστά το height)
- outerWidth αριθμός pixels
- outerHeight αριθμός pixels
- screenX θέση σε pixels
- screenY θέση σε pixels
- titlebar yes|no
- z-lock yes|no

Μπορείτε να βρείτε την εξήγηση των νέων παραμέτρων σε κάποιο οδηγό για JavaScript 1.2. Στο μέλλον, θα παρουσιάσω παραδείγματα και τις εξηγήσεις των παραμέτρων. Με την βοήθεια αυτών των παραμέτρων μπορούμε να προσδιορίσουμε σε ποια θέση θα ανοίξει το παράθυρο. Δεν μπορείτε να το κάνετε αυτό σε παλιές εκδόσεις της JavaScript.

Το όνομα του παραθύρου

Όπως είδατε χρησιμοποιήσαμε τρεις παραμέτρους για να ανοίξουμε το παράθυρο:
myWin= open("bla.htm", "displayWindow", "width=400,height=300,status=no, toolbar=no,menubar=no");
Γιατί είναι η δεύτερη παράμετρος (displayWindow); Αυτό είναι το όνομα του παραθύρου. Είχαμε δει πώς να χρησιμοποιούμε την ιδιότητα target πρωτύτερα. Αν ξέρετε το όνομα ενός παραθύρου, μπορείτε να φορτώσετε μια νέα σελίδα σε αυτό, γράφοντας
<a href="bla.html" target="displayWindow"> Εδώ χρειάζεστε το όνομα του παραθύρου (αν το παράθυρο δεν υπάρχει, ένα νέο παράθυρο θα δημιουργηθεί με αυτό τον κώδικα).
Σημείωση: το myWin ΔΕΝ είναι το όνομα του παραθύρου. Μπορείτε απλώς να έχετε πρόσβαση σε αυτό το παράθυρο, μέσω της μεταβλητής. Επειδή είναι μια απλή μεταβλητή, είναι ισχύουσα μόνο μέσα στο script στο οποίο έχει προσδιορισθεί. Το όνομα του παραθύρου (εδώ displayWindow) είναι ένα μοναδικό όνομα το οποίο μπορεί να χρησιμοποιηθεί από όλα τα ανοιχτά παράθυρα.

Κλείνοντας παράθυρα

Μπορείτε να κλείσετε παράθυρα χρησιμοποιώντας JavaScript. Γι' αυτό χρειάζεστε την μέθοδο close(). Ας ανοίξουμε ένα νέο παράθυρο όπως πριν. Σε αυτό το παράθυρο θα φορτώσουμε την ακόλουθη σελίδα:

```
<html>
<script type="text/javascript">
<!-- hide
function closeIt() {
close();
}
// -->
```

```
</script>
<center>
<form>
<input type=button value="Κλείστε αυτό το παράθυρο"
onClick="closeIt()">
</form>
</center>
</html>
```

Αν πατήσετε το κουμπί στο νέο παράθυρο, το παράθυρο θα κλείσει. Οι μέθοδοι open() και close() ανήκουν στο αντικείμενο window. Κανονικά, πρέπει να γράψουμε window.open() και window.close() αντί για open() και close(). Αυτό είναι αλήθεια - Αλλά το αντικείμενο window είναι μια εξαίρεση. Δεν χρειάζεται να γράψετε window αν θέλετε να καλέσετε μια method του αντικείμενου window (αυτό συμβαίνει μόνο σε αυτό το αντικείμενο).

Δημιουργώντας δυναμικά έγγραφα

Ερχόμαστε τώρα σε μια έξυπνη λειτουργία του JavaScript - η δημιουργία εγγράφων on-the-fly. Αυτό σημαίνει ότι μπορείτε να αφήσετε τη JavaScript κώδικα να δημιουργήσει μια νέα HTML σελίδα. Επιπλέον, μπορείτε να δημιουργήσετε άλλα έγγραφα - όπως VRML-scenes, κλπ.. Μπορείτε να δημιουργήσετε τα έγγραφα αυτά, σε νέα παράθυρα ή σε frames. Πρώτα πρέπει να δημιουργήσουμε μια απλή HTML σελίδα, η οποία θα εμφανίζεται στο νέο παράθυρο. Εδώ είναι ένα script που θα πρέπει να του ρίξουμε μια ματιά.

```
<html>
<head>
<script type="text/javascript">
<!-- hide
function openWin3() {
myWin= open("", "displayWindow", "width=500,height=400,status=yes, toolbar=yes,menubar=yes");
// Άνοιγμα του εγγράφου για παραπέρα μετατροπή
myWin.document.open();
// Δημιουργία εγγράφου myWin.document.write("<html><head><title>Dynamic");
myWin.document.write("</title></head><body>"); myWin.document.write("<center><font
size=+3>");
myWin.document.write("Αυτό το HTMLέγγραφο δημιουργήθηκε ");
myWin.document.write("με χρήση της JavaScript!");
myWin.document.write("</font></center>");
myWin.document.write("</body></html>");
// Κλείσιμο του εγγράφου - (όχι του παραθύρου)
myWin.document.close();
}
// -->
</script>
</head>
<body>
```

```
<form>
<input type=button value="dynamic" onClick="openWin3()">
</form>
</body>
</html>
```

Ας κοιτάξουμε την function winOpen3(). Αρχικά βλέπουμε ότι ανοίγουμε ένα νέο παράθυρο. Όπως βλέπετε η πρώτη παράμετρος έχει κενή τιμή "" - αυτό σημαίνει ότι δεν καθορίζουμε ένα URL. Ο browser δεν πρέπει απλώς να φορτώσει ένα έγγραφο - η JavaScript θα δημιουργήσει ένα καινούργιο έγγραφο. Προσδιορίζουμε την μεταβλητή myWin. Με τη βοήθεια αυτής της μεταβλητής, μπορούμε να έχουμε πρόσβαση στο νέο παράθυρο. Σημειώστε ότι δεν μπορούμε να χρησιμοποιήσουμε το όνομα του παραθύρου (displayWindow) για αυτή τη λειτουργία. Εφόσον ανοίξουμε το παράθυρο, πρέπει να ανοίξουμε το έγγραφο. Αυτό γίνεται ως εξής:

```
// Άνοιγμα εγγράφου για παραπέρα μετατροπή
myWin.document.open();
```

Καλούμε την μέθοδο open() του αντικειμένου document - αυτή είναι διαφορετική μέθοδος από την open() του αντικείμενου window! Αυτή η εντολή δεν ανοίγει ένα νέο παράθυρο - ετοιμάζει το έγγραφο για μορφοποίηση.

Πρέπει να βάλουμε myWin πριν από document.open() για να έχουμε πρόσβαση στο παράθυρο. Οι επόμενες γραμμές δημιουργούν το έγγραφο, χρησιμοποιώντας την μέθοδο document.write():

```
// Δημιουργία εγγράφου
myWin.document.write("<html><head><title>Dynamic");
myWin.document.write("</title></head><body>");
myWin.document.write("<center><font size=+3>");
myWin.document.write("Αυτό το HTML έγγραφο δημιουργήθηκε ");
myWin.document.write("με χρήση JavaScript!"); myWin.document.write("</font></center>");
myWin.document.write("</body></html>");
```

Μπορείτε να δείτε τις κανονικές HTML εντολές του εγγράφου. Δημιουργούμε έναν HTML κώδικα! Μπορείτε να γράψετε οποιαδήποτε HTML εντολή. Μετά την μετατροπή πρέπει να κλείσουμε το έγγραφο. Ο επόμενος κώδικας το κάνει αυτό:

```
// Κλείσιμο του εγγράφου - (όχι του παραθύρου!)
myWin.document.close();
```

Όπως είπα πριν, μπορούμε να δημιουργήσουμε on-the-fly έγγραφα και να τα εμφανίσουμε σε frame. Αν, π.χ. έχετε δύο frames με τα ονόματα frame1 και frame2 και θέλετε να δημιουργήσετε ένα νέο έγγραφο στο frame2, μπορείτε να γράψετε τα ακόλουθα στο frame1:

```
parent.frame2.document.open();
parent.frame2.document.write("HTML κώδικας");
parent.frame2.document.close();
```

Το statusbar

Τα JavaScript προγράμματα μπορούν να γράψουν κείμενο στο statusbar - statusbar είναι η μπάρα στο κάτω μέρος

του browser. Το μόνο που πρέπει να κάνετε είναι να ρυθμίσετε την τιμή window.status. Το επόμενο παράδειγμα δείχνει δύο κουμπιά, τα οποία χρησιμοποιούνται για να γράψουν κείμενο στο statusbar και για να το σβήσουνε. Το script δείχνει ως εξής:

```
<html>
<head>
<script type="text/javascript">
<!-- hide
function statbar(txt) {
window.status = txt;
}
// -->
</script>
</head>
<body>
<form>
<input type="button" name="look" value="Γράψε!"
onClick="statbar(' Hi from the statusbar!');">
<input type="button" name="erase" value="Σβήσε!"
onClick="statbar('');">
</form>
</body>
</html>
```

Δημιουργούμε μια φόρμα με δύο κουμπιά. Και τα δύο κουμπιά καλούν την function statbar(). Βλέπετε ότι το κάλεσμα της function, που δημιουργείται στο Write! δείχνει ως εξής:

```
statbar('Γεια! Εγώ είμαι το statusbar!');
```

Μέσα στην παρένθεση ρυθμίζουμε την τιμή 'Γεια! Εγώ είμαι το statusbar!' Αυτό σημαίνει ότι η τιμή εισάγεται στο function statbar(). Μπορείτε να δείτε ότι έχουμε ρυθμίσει την function statbar() ως εξής:

```
function statbar(txt) {
window.status = txt;
}
```

Το καινούργιο είναι το txt μέσα στη παρένθεση του function. Αυτό σημαίνει ότι η τιμή που βάλαμε στο function αποθηκεύεται στη μεταβλητή txt. Δίνοντας μεταβλητές στα functions είναι ένας συνήθης τρόπος για να κάνουμε τα functions πιο εύκαμπτα. Μπορείτε να περάσετε πολλές μεταβλητές στα functions - Απλώς πρέπει να τις διαχωρίσετε με κόμματα. Η τιμή txt φαίνεται στο statusbar, χάρη στη γραμμή

```
window.status = txt.
```

Το να σβηστεί το κείμενο από το statusbar είναι εφικτό αν καθορίζουμε μια κενή τιμή στο window.status.
Το να δείχνουμε κείμενο στο statusbar μπορεί εύκολα να χρησιμοποιηθεί σε συνδυασμό με τα links. Αντί να δείχνει το URL του link μπορείτε να το κάνετε να εξηγεί τι περιέχει η σελίδα που ακολουθεί. Ο κώδικας γι' αυτό το παράδειγμα είναι ο εξής:

```
<a href="./dontclck.htm" onMouseOver="window.status='Don\'t click me!';
```

```
return true;" onMouseOut="window.status='';">link</a>
```

Εδώ χρησιμοποιούμε τα onMouseOver και onMouseOut για να ξέρουμε πότε ο δείκτης του mouseκινείται πάνω από το link. Η χρήση του return true μέσα στο event onMouseOver σημαίνει ότι ο browser δεν θα εκτελέσει τον δικό του κώδικα σαν απάντηση στο event MouseOver. Κανονικά ο browser δείχνει το URL του link στο statusbar. Αν δεν χρησιμοποιήσουμε το return true, ο browser θα γράψει στο statusbar αμέσως μόλις ο δικός μας κώδικας έχει εκτελεστεί - αυτό σημαίνει ότι θα έγραφε πάνω στο δικό μας κείμενο και, ως αποτέλεσμα, ο χρήστης δεν θα μπορούσε να το διαβάσει. Γενικά μπορούμε να εμποδίσουμε τις ενέργειες του browser, χρησιμοποιώντας το return true στον event- handler. onMouseOut.

Αν θέλουμε να γράψουμε το Don't click me – όπως στο παράδειγμα - καθώς καθορίζουμε την τιμή του onMouseOver event-handler, χρησιμοποιούμε απλά εισαγωγικά ('). Αλλά η λέξη Don't χρησιμοποιεί και αυτή απλό εισαγωγικό μεταξύ n και t! Ο browser μπερδεύεται αν απλώς γράψετε 'Don't ...' γιατί θεώρεότι tostring είναι το Don και το υπόλοιπο του περισσεύει και δεν ξέρει τι να το κάνει!. Για να λύσουμε αυτό το πρόβλημα, απλώς γράφουμε την κάθετο backslash \, πριν απ' το εισαγωγικό - που σημαίνει ότι το εισαγωγικό ανήκει στη τιμή (μπορείτε να κάνετε το ίδιο με τα διπλά εισαγωγικά ").

Timeouts

Με την βοήθεια των timeouts (ή αλλιώς timer, χρονόμετρα) μπορείτε να διατάξετε τον υπολογιστή να εκτελέσει τον κώδικα, εφόσον περάσει καθορισμένος χρόνος. Έχω δημιουργήσει ένα κουμπί - αν το πατήσετε, ένα παράθυρο θα εμφανιστεί μετά από 3 δευτερόλεπτα:

Ο κώδικας αυτής της περίπτωσης είναι:

```
<script type="text/javascript">
<!-- hide
function timer() {
setTimeout("alert('Τέλος χρόνου!')", 3000);
}
// -->
</script>
...
<form>
<input type="button" value="Timer"
onClick="timer()">
</form>
```

Το setTimeout() είναι method του αντικείμενου window. Καθορίζει ένα timeout - νομίζω ότι αυτό το έχετε καταλάβει ως τώρα. Η πρώτη τιμή είναι ο JavaScript κώδικας που πρέπει να εκτελεστεί μετά από τον καθορισμένο χρόνο. Στη περίπτωσή μας, η τιμή αυτή είναι "alert('Τέλος χρόνου!')". Σημειώστε ότι ο κώδικας πρέπει να είναι μέσα σε άνω τελείες. Η δεύτερη τιμή λέει στον υπολογιστή πότε ο κώδικας θα εκτελεστεί. Πρέπει να καθορίσετε τον χρόνο σε milliseconds (3000 milliseconds = 3 δευτερόλεπτα).

Τώρα που ξέρετε πώς να γράψετε στο statusbar και πώς δουλεύουν τα timeouts, θα ρίξουμε μια ματιά στους scrollers (κινούμενο κείμενο στο statusbar). Οι scrollers χρησιμοποιούνται συχνά στο Internet. Θα δούμε πώς να προγραμματίσουμε έναν απλό scroller. Εκτός αυτού, θα σκεφτούμε πιθανές βελτιώσεις του scroller.

Οι Scrollers μπορούν να εισαχθούν εύκολα. Ας δούμε πώς θα δημιουργήσουμε το κινούμενο κείμενο στο statusbar. Πρέπει να γράψουμε ένα κείμενο στο statusbar. Μετά από μια χρονική περίοδο, πρέπει να γράψουμε το ίδιο κείμενο στο statusbar - αλλά πρέπει να το μετακινήσουμε σε πιο αριστερή θέση. Αν το επαναλαμβάνουμε αυτό ξανά και ξανά, ο χρήστης θα βλέπει αν τέλει το κινούμενο κείμενο.

Πρέπει να σκεφτούμε πώς θα καθορίσουμε ποιο μέρος του κειμένου θα εμφανιστεί, μιας και το όλο κείμενο είναι κανονικά μεγαλύτερο του statusbar.

Αυτό το κουμπί θα εμφανίζει ένα παράθυρο και θα δείξει τον scroller. Ο κώδικας είναι ο ακόλουθος - έχω προσθέσει σχόλια:

```
<html>
<head>
<script type="text/javascript">
<!-- hide
// Καθορισμός του κειμένου
var scrtxt = "This is JavaScript! " + "This is JavaScript! " +
"This is JavaScript!";
var length = scrtxt.length;
var width = 100;
var pos = -(width + 2);
function scroll() {
// Εμφάνιση του κειμένου στη δεξιά θέση και καθορισμός του timeout
// Μετακίνηση της θέσης ένα βήμα μακρύτερα
pos++;
// Υπολογισμός του μέρους του κειμένου που θα εμφανιστεί
var scroller = "";
if (pos == length) {
pos = -(width + 2);
}
// Αν το κείμενο δεν έχει φτάσει την αριστερή πλευρά ακόμα, πρέπει
// να εισάγουμε κενά - αλλιώς πρέπει να κόψουμε το πρώτο μέρος
// του κειμένου (το οποίο μετακινήθηκε στην αριστερή πλευρά)
if (pos < 0) {
for (var i = 1; i <= Math.abs(pos); i++) {
scroller = scroller + " ";}
scroller = scroller + scrtxt.substring(0,width –
i+1);
}
```

```
else {
scroller = scroller +
scrtxt.substring(pos,width+pos);
}
// Εισαγωγή του κειμένου στο statusbar window.status = scroller;
// Κάλεσμα της function μετά από 100 milliseconds setTimeout("scroll()", 100);
}
// -->
</script>
</head>
<body onLoad="scroll()">
Η HTML σελίδα μπαίνει εδώ.
</body>
</html>
```

Το κύριο μέρος του scroll() χρειάζεται για τον υπολογισμό του ποιου μέρους του κειμένου θα εμφανιστεί. Δεν εξηγώ τον κώδικα με λεπτομέρειες - απλώς πρέπει να καταλάβετε γενικά πώς ενεργεί αυτός ο scroller. Για να αρχίσουμε τον scroller χρησιμοποιούμε το onLoad event-handler της ετικέτας <body>. Αυτό σημαίνει ότι η function scroll() μπαίνει σε λειτουργία, εφόσον έχει φορτωθεί το έγγραφο.
Καλούμε την function scroll() με την ιδιότητα onLoad. Το πρώτο βήμα του scroller υπολογίζεται και εμφανίζεται. Στο τέλος της function scroll() καθορίζουμε ένα timeout. Αυτό κάνει την function scroll() να εκτελεστεί ξανά μετά από 100 milliseconds. Το κείμενο μετακινείται ένα βήμα μπροστά και ένα άλλο timeout καθορίζεται. Αυτό συνεχίζεται για πάντα. Οι Scrollers χρησιμοποιούνται συχνά στο Internet. Υπάρχει ο "κίνδυνος" ότι μπορεί να γίνουν γρήγορα μη δημοφιλείς. Πρέπει να παραδεχτώ ότι δεν μου αρέσουν τόσο πολύ. Ειδικά ενοχλητικό σε πολλές σελίδες είναι ότι το URL δεν μπορεί να εμφανιστεί όταν μετακινούμε το δείκτη πάνω από link. Αυτό μπορεί να αποφευχθεί, αν σταματούμε τον scroller όταν ένα MouseOver event συμβαίνει - μπορείτε να το ξεκινάτε ξανά με το onMouseOut. Αν θέλετε να έχετε ένα scroller, προσπαθήστε να μη χρησιμοποιήσετε τον απλό scroller - προσπαθήστε να προσθέσετε μερικά εφέ. Ίσως, ένα μέρος του κειμένου να έρχεται από τα αριστερά, και ένα άλλο απ' τα δεξιά - όταν συναντούνται στη μέση να σταματάνε για μερικά δευτερόλεπτα. Με μερική φαντασία, μπορείτε να βρείτε εναλλακτικές λύσεις.

Built – In αντικείμενα

Το αντικείμενο Date

Η JavaScript διαθέτει μερικά έτοιμα αντικείμενα όπως για παράδειγμα, τα αντικείμενα Date, Array και Math. Υπάρχουν πολλά άλλα αντικείμενα - απευθυνθείτε σε έγγραφα της Netscape για πλήρη λίστα αντικειμένων. Θα ρίξουμε μια ματιά στο αντικείμενο Date, πρώτα. Όπως φαίνεται απ' το όνομα, το αντικείμενο αυτό έχει να κάνει με το χρόνο και την ημερομηνία. Παραδείγματος χάριν, μπορείτε να υπολογίσετε πόσες μέρες μένουν ως τα επόμενα Χριστούγεννα, με τη βοήθεια αυτού του αντικειμένου. Ή να προσθέσετε τη πραγματική ώρα, στο έγγραφο HTML. Γι' αυτό λοιπόν, ας ξεκινήσουμε με ένα παράδειγμα που δείχνει την ώρα. Πρώτα πρέπει να

δημιουργήσουμε το νέο αντικείμενο Date. Γι' αυτό το λόγο χρησιμοποιούμε την εντολή new. Κοιτάξτε αυτή τη γραμμή κώδικα:

```
today= new Date()
```

Αυτό δημιουργεί ένα νέο αντικείμενο Date, που ονομάζεται today. Αν δεν καθορίσετε μια νέα ημερομηνία ή ώρα όταν δημιουργείτε το νέο αντικείμενο Date η πραγματική ώρα και ημερομηνία χρησιμοποιείται. Αυτό σημαίνει ότι μετά την εκτέλεση του today= new Date() το νέο αντικείμενο today αντιπροσωπεύει την ημερομηνία και ώρα που εκτελέστηκε ο κώδικας αυτός. Το αντικείμενο Date αναφέρεται σε μερικές methods που μπορούν να χρησιμοποιηθούν με το αντικείμενο today. Αυτές είναι π.χ. getHours(), setHours(), getMinutes(), setMinutes(), getMonth(), setMonth(), κ.λ.π. Μπορείτε να βρείτε πλήρη στοιχεία στο διαδίκτυο.

Σημειώστε ότι το αντικείμενο Date αντιπροσωπεύει μια καθορισμένη ημερομηνία και ώρα. Δεν είναι είδος ρολογιού που αλλάζει την ώρα κάθε δευτερόλεπτο, ή λεπτό. Για να πάρουμε νέο ημερομηνία και ώρα, χρησιμοποιούμε μια άλλη μέθοδο (αυτή είναι η Date() method, που καλείται απ' το new, όταν καθορίζουμε ένα νέο Date αντικείμενο:

```
today= new Date(1997, 0, 1, 17, 35, 23)
```

Αυτό θα δημιουργήσει ένα Date αντικείμενο, που αντικαθιστά την 1η Ιανουαρίου 1997, και ώρα 17:35 και 23 δευτερόλεπτα. Έτσι ρυθμίζετε την ημερομηνία και ώρα:

```
Date(year, month, day, hours, minutes, seconds)
```

Σημειώστε ότι πρέπει να χρησιμοποιήσετε 0 για τον Ιανουάριο - και όχι 1, όπως πιθανώς νομίζατε. Το 1 σημαίνει Φεβρουάριο, το 2 Μάρτιο, κ.ο.κ.

Τώρα θα γράψουμε ένα script το οποίο δείχνει την ημερομηνία και ώρα σε πραγματικό χρόνο. Ο κώδικας είναι ο εξής:

```
<script type="text/javascript">
<!-- hide
now= new Date();
document.write("Time: " + now.getHours() + ":" + now.getMinutes() + "<br>");
document.write("Date: "+(now.getMonth()+1)+"/"+ now.getDate() + "/" + (1900 +
now.getYear()));
// -->
</script>
```

Εδώ χρησιμοποιούμε methods, όπως getHours() για να δείξουμε την ώρα και ημερομηνία που έχει καθοριστεί στο αντικείμενο Date τώρα. Όπως βλέπετε προσθέτουμε 1900 στο χρόνο. Η method getYear() επιστρέφει ως αποτέλεσμα τα χρόνια που πέρασαν απ' το 1900. Αυτό σημαίνει ότι αν η χρονιά είναι 1997 θα επιστρέψει την τιμή 97, αν ο χρόνος είναι 2010 θα επιστρέψει την τιμή 110 – όχι 10! Θυμηθείτε ότι πρέπει να αυξήσουμε κατά ένα τον αριθμό που λαμβάνουμε απ' τη μέθοδο getMonth().

Το script δεν ελέγχει αν ο αριθμός των λεπτών είναι μικρότερος από 10. Αυτό σημαίνει ότι θα πάρετε αποτέλεσμα: 14:3, που στη πραγματικότητα σημαίνει 14:03. Θα δούμε στο επόμενο script πώς να λύσουμε αυτό το πρόβλημα.

Τώρα θα κοιτάξουμε το script που δείχνει ένα ρολόι εν λειτουργία. Εδώ είναι ο κώδικας:

```
<html>
<head>
<script Type="text/javascript">
<!-- Κρύβουμε τον κώδικα από παλιούς browsers
var timeStr, dateStr;
function clock() {
now= new Date();
// time
hours= now.getHours(); minutes= now.getMinutes(); seconds= now.getSeconds(); timeStr= "" +
hours;
timeStr+= ((minutes < 10) ? ":0" : ":") + minutes; timeStr+= ((seconds < 10) ? ":0" : ":") +
seconds; document.clock.time.value = timeStr;
// date
date= now.getDate(); month= now.getMonth()+1; year= now.getYear(); dateStr= "" + month;
dateStr+= ((date < 10) ? "/0" : "/") + date; dateStr+= "/" + year; document.clock.date.value
= dateStr;
Timer= setTimeout("clock()",1000);
}
// -->
</script>
</head>
<body onLoad="clock()">
<form name="clock">
Ώρα: <input type="text" name="time" size="8" value=""><br>
Ημερομηνία: <input type="text" name="date" size="8" value="">
</form>
</body>
</html>
```

Χρησιμοποιούμε την μέθοδο setTimeout() για να ρυθμίζουμε την ώρα και ημερομηνία κάθε δευτερόλεπτο. Γι' αυτό δημιουργούμε κάθε δευτερόλεπτο ένα νέο Date με την πραγματική ώρα. Μπορείτε να δείτε ότι η function clock() καλείτε στο onLoad event-handler στην ετικέτα <body>. Στο κύριο μέρος του εγγράφου μας έχουμε δύο πεδία κειμένου. Η function clock() γράφει την ώρα και ημερομηνία στα δύο στοιχεία της φόρμας, με το σωστό τύπο. Μπορείτε να δείτε ότι χρησιμοποιούμε δύο μεταβλητές timeStr και dateStr γι' αυτό το σκοπό. Έχω αναφέρει πριν ότι υπάρχει πρόβλημα όταν τα λεπτά λιγότερα απ' το 10 - αυτό το script λύνει αυτό το πρόβλημα, με την παρακάτω γραμμή στο κώδικα:

```
timeStr+= ((minutes < 10) ? ":0" : ":") + minutes;
```

Εδώ, ο αριθμός των λεπτών προσθέτεται στην μεταβλητή timeStr. Αν τα λεπτά είναι λιγότερα από 10, πρέπει να προσθέσουμε ένα 0. Αυτή η γραμμή του κώδικα μπορεί να φαίνεται λίγο παράξενη σε σας. Μπορείτε επίσης, να γράψετε κάτι σαν το ακόλουθο, που θα σας φανεί πιο οικείο:

```
if (minutes < 10) timeStr+= ":0" + minutes else timeStr+= ":" + minutes;
```

Το αντικείμενο Array

Τα arrays είναι σημαντικά. Σκεφτείτε ένα παράδειγμα, όπου πρέπει να αποθηκεύσετε 100 διαφορετικά ονόματα. Πώς θα το κάνετε αυτό στη JavaScript? Μπορείτε να δημιουργήσετε 100 μεταβλητές και να τους εισάγετε τα 100 ονόματα. Αυτό, όμως είναι πολύπλοκο. Τα Arrays μπορείτε να τα δείτε σαν μεταβλητές ενωμένες μεταξύ τους. Μπορείτε να έχετε πρόσβαση σ' αυτές μέσω του ονόματος και ενός αριθμού. Ας πούμε ότι το όνομα του array είναι names. Τότε, έχουμε πρόσβαση στο πρώτο όνομα γράφοντας names[0]. Το δεύτερο όνομα λέγεται name[1] κ.ο.κ.

Εναλλακτικά μπορείτε να χρησιμοποιήσετε το αντικείμενο Array. Μπορείτε να δημιουργήσετε ένα νέο array γράφοντας myArray= new Array(). Τώρα, μπορείτε να εισάγετε τις τιμές στο array:

myArray[0]= 17; myArray[1]= "Stefan"; myArray[2]= "Koch";

Τα JavaScript arrays είναι πολύ εύκαμπτα. Δεν είναι ανάγκη να ασχολείστε με το μέγεθος του array το μέγεθός του ρυθμίζεται αυτόματα. Αν γράψετε myArray[99]= "xyz", το μέγεθος του array παίρνει 100 στοιχεία (ένα JavaScript array μπορεί μόνο να μεγαλώσει - δεν έχει τη δυνατότητα να μικρύνει. Γι' αυτό κρατήστε τα arrays όσο το δυνατόν μικρότερα).Δεν έχει σημασία αν αποθηκεύετε αριθμούς, κείμενο ή άλλα αντικείμενα σε ένα array. Δεν αναφέρομαι με λεπτομέρεια για τα arrays σε αυτό το σημείο, αλλά θα δείτε πόσο χρήσιμα μπορούν να γίνουν τα arrays.

Σίγουρα, τα πράγματα γίνονται πιο ξεκάθαρα κοιτώντας ένα παράδειγμα. Το αποτέλεσμα του επόμενου παραδείγματος είναι:

```
Πρώτη τιμή
Δεύτερη τιμή
Τρίτη τιμή
```

Ο κώδικας είναι ο ακόλουθος:

```
<script type="text/javascript">
<!-- hide
myArray= new Array();
myArray[0]= "Πρώτη τιμή";
myArray[1]= "Δεύτερη τιμή";
myArray[2]= "Τρίτη τιμή";
for (var i= 0; i< 3; i++) {
document.write(myArray[i] + "<br>");
}
// -->
</script>
```

Πρώτα, δημιουργούμε ένα array, με την ονομασία myArray. Τότε, εισάγουμε τρεις διαφορετικές τιμές σε αυτό. Μετά απ' αυτό δημιουργούμε ένα loop. Αυτό το loop εκτελεί την εντολή document.write(myArray[i] + "
"); τρεις φορές. Η μεταβλητή i μετρά από το 0 ως το 2 με αυτό το for-loop. Μπορείτε να δείτε ότι χρησιμοποιούμε το myArray[i] μέσα στο for-loop. Καθώς το i παίρνει τις τιμές από το 0 ως το 2, παίρνουμε τρεις document.write() κλήσεις. Θα μπορούσαμε επίσης να το γράψουμε ως εξής:

```
document.write(myArray[0] + "<br>"); document.write(myArray[1] + "<br>");
```

```
document.write(myArray[2] + "<br>");
```

Το αντικείμενο Math

Αν χρειάζεστε να κάνετε μαθηματικούς υπολογισμούς, θα βρείτε μερικές methods στο αντικείμενο Math. Μία συνοπτική παρουσίαση ακολουθεί:

Ιδιότητες

Όνομα	**Περιγραφή**
E	Σταθερά Euler (περίπου 2,718)
LN2	Φυσικός λογάριθμος του 2 (περίπου 0,693)
LN10	Φυσικός λογάριθμος του 10 (περίπου 2,302)
LOG2E	Νεπέριος βάση-2 (περίπου 1,442)
LOG10E	Νεπέριος βάση-10 (περίπου 0,434)
PI	π (περίπου 3,14)
SQRT1_2	Ρίζα το 1/2(περίπου 0,707)
SQRT2	Ρίζα του 2 (περίπου 1,414)

Μέθοδοι

Method	**Description**
abs(x)	Απόλυτη τιμή
acos(x)	Συνημίτονο τόξου (x σε ακτίνια)
asin(x)	ημίτονο τόξου (x σε ακτίνια)
atan(x)	Τόξο εφαπρωμένης (x σε ακτίνια μεταξύ –π/2 και π/2)
atan2(y,x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Άνω στρογγυλοποίηση σε ακέραιο
cos(x)	συνημίτονο (x σε ακτίνια)
exp(x)	E^x
floor(x)	Κάτω στρογγυλοποίηση σε ακέραιο
log(x)	Φυσικός λογάριθμος βάση E
max(x,y,z,...,n)	Μέγιστος
min(x,y,z,...,n)	Ελάχιστος
pow(x,y)	Το x υψωμένο στην y
random()	Τυχαίος αριθμός μεταξύ 0 και 1
round(x)	Στρογγυλοποίηση στον κοντινότερο ακέραιο
sin(x)	Ημίτονο (x σε ακτίνια)
sqrt(x)	Τετραγωνική ρίζα

tan(x)	Εφαπτωμένη (x σε ακτίνια)

Το αντικείμενο Image

Εικόνες σε μια σελίδα

Τώρα, θα ρίξουμε μια ματιά στο αντικείμενο Image, με τη βοήθεια του οποίου μπορείτε να αλλάξετε τις εικόνες σε μια σελίδα. Αυτό σας επιτρέπει, π.χ. να δημιουργήσετε animations.

Πρώτα, να δούμε πώς οι εικόνες σε μια σελίδα μπορούν να χρησιμοποιηθούν μέσω JavaScript. Όλες οι εικόνες παρουσιάζονται μέσω ενός array. Αυτό το array λέγεται images. Είναι ιδιότητα του αντικείμενου document. Κάθε εικόνα παίρνει ένα αριθμό. Η πρώτη εικόνα παίρνει τον αριθμό 0, η δεύτερη τον αριθμό 1 κ.ο.κ. Έτσι, έχουμε πρόσβαση στην πρώτη εικόνα γράφοντας document.images[0]. Κάθε εικόνα σε ένα HTML έγγραφο θεωρείται σαν αντικείμενο Image. Ένα αντικείμενο Image έχει καθορισμένες ιδιότητες που μπορούν να χρησιμοποιηθούν μέσω JavaScript. Μπορείτε παραδείγματος χάριν να ξέρετε το μέγεθος της εικόνας χάρη στις ιδιότητες width και height. Το document.images[0].width σας δίνει το πλάτος (σε pixel) της πρώτης εικόνας στη σελίδα. Ειδικά αν έχετε πολλές εικόνες σε μια σελίδα είναι δύσκολο να θυμάστε τον αριθμό της κάθε εικόνας. Δίνοντας ονόματα στην κάθε εικόνα λύνουμε το πρόβλημα. Για την εικόνα:

```
<img src="img.gif" name="myImage" width=100 height=100>
```

μπορείτε να έχετε πρόσβαση σε αυτή γράφοντας

```
document.myImage
```

ή

```
document.images["myImage"].
```

Φορτώνοντας νέες εικόνες

Παρόλο που είναι καλό να ξέρουμε πώς να πάρουμε το μέγεθος της εικόνας, αυτό δεν είναι αυτό που βασικά θέλουμε να ξέρουμε. Θέλουμε να αλλάζουμε τις εικόνες σε μια σελίδα. Γι' αυτό το σκοπό χρησιμοποιούμε την ιδιότητα src. Όπως και στο <img> η ιδιότητα src καθορίζει την εικόνα που δείχνεται. Μπορείτε να ρυθμίσετε νέες διευθύνσεις σε μια ήδη φορτωμένη εικόνα της σελίδας. Το αποτέλεσμα θα είναι ότι η νέα εικόνα θα φορτώσει στη θέση της παλιάς. Κοιτάξτε το παρακάτω παράδειγμα:

```
<img src="./img1.gif" name="myImage" width=100 height=100>
```

Η εικόνα img1.gif φορτώνεται και παίρνει το όνομα myImage. Η ακόλουθη γραμμή κώδικα αντικαταστεί τη img1.gif με τη νέα εικόνα img2.gif:

```
document.myImage.src= "img2.gif";
```

Η νέα εικόνα έχει το ίδιο μέγεθος με την παλιά. Δεν μπορείτε να αλλάξετε το μέγεθος της περιοχής που βρίσκεται η εικόνα.

Φορτώνοντας τις εικόνες από πριν

Ένα μειονέκτημα είναι ότι η εικόνα φορτώνει αφού έχει τεθεί η νέα τιμή της ιδιότητας src. Εφόσον η εικόνα δεν έχει φορτωθεί από πριν, παίρνει κάποιο καιρό μέχρι να φορτωθεί απ' το Internet. Σε μερικές περιπτώσεις δεν

υπάρχει πρόβλημα - αλλά συχνά αυτές οι καθυστερήσεις δεν είναι αποδεκτές. Τι μπορούμε να κάνουμε γι' αυτό; Να φορτώσουμε τις εικόνες από πριν είναι η λύση. Γι' αυτό το σκοπό πρέπει να δημιουργήσουμε ένα νέο Image αντικείμενο. Κοιτάξτε αυτές τις γραμμές κώδικα:

```
hiddenImg= new Image();
hiddenImg.src= "img3.gif";
```

Η πρώτη γραμμή δημιουργεί το αντικείμενο Image. Η δεύτερη γραμμή καθορίζει την εικόνα η οποία θα καλείται μέσω του hiddenImg. Έχουμε ήδη δει ότι όταν θέτουμε νέα τιμή στην ιδιότητα src αναγκάζεται ο browser να φορτώσει την νέα εικόνα. Έτσι η εικόνα img2.gif φορτώνει όταν η δεύτερη γραμμή του κώδικα φορτώνει. Όπως υπονοεί το όνομα hiddenImg, η εικόνα δεν δείχνεται εφόσον ο browser τελειώσει το φόρτωμά της. Απλώς κρατείται στην μνήμη (ή καλύτερα στο cache) για να χρησιμοποιηθεί αργότερα. Για να δείξουμε την εικόνα, μπορούμε να γράψουμε:

```
document.myImage.src= hiddenImg.src;
```

Τώρα η εικόνα έρχεται απ' το cache και δείχνεται αμέσως. Έχουμε καταφέρει να φορτώσουμε από πριν την εικόνα. Φυσικά ο browser πρέπει να έχει τελειώσει με το να φορτώνει τις εικόνες, ώστε να μπορούμε να τις δούμε χωρίς καθυστέρηση. Έτσι, αν έχετε πολλές εικόνες θα υπάρχει καθυστέρηση έτσι κι αλλιώς, επειδή ο browser είναι απασχολημένος κατεβάζοντας όλες τις άλλες εικόνες. Πρέπει πάντα να σκέφτεστε την ταχύτητα του Internet - το κατέβασμα των εικόνων δεν γίνεται γρηγορότερο με αυτή τη μέθοδο. Προσπαθούμε μόνο να φορτώσουμε τις εικόνες από πριν - έτσι ο χρήστης θα τις δει από πριν. Αυτό κάνει την όλη διαδικασία πιο εύκολη.

Αλλάζοντας εικόνες σύμφωνα με την ενέργεια του χρήστη

Μπορείτε να δημιουργήσετε ωραία εφέ με την αλλαγή εικόνων ως απάντηση σε ορισμένα events. Μπορείτε, π.χ. να αλλάζετε την εικόνα όταν ο δείκτης του mouse είναι πάνω από μια συγκεκριμένη περιοχή. Δοκιμάστε το ακόλουθο παράδειγμα μετακινώντας το δείκτη πάνω απ΄ τις εικόνες:

Ο πηγαίος κώδικας είναι ο εξής:

```
<a href="#" onMouseOver="document.myImage2.src='img2.gif'"
nMouseOut="document.myImage2.src='img1.gif'">
<img src="./img1.gif" name="myImage2;" width=160 height=50 border=0></a>
```

Αυτός ο κώδικας δημιουργεί κάποια προβλήματα:

- Η δεύτερη εικόνα δεν φορτώνεται από πριν.
- Πρέπει να ξαναγράψουμε τον κώδικα για κάθε εικόνα στη σελίδα.
- Θέλουμε ένα script το οποίο να μπορεί να χρησιμοποιηθεί και σε άλλες σελίδες χωρίς μεγάλες αλλαγές.

Τώρα θα κοιτάξουμε ένα script που λύνει όλα αυτά τα προβλήματα. Το script παίρνει πιο πολύ ώρα να γραφεί - αλλά όταν έχει γραφεί, δεν χρειάζεται να ασχοληθείτε ξανά μαζί του. Υπάρχουν δύο απαιτήσεις για να κρατήσουν το script εύκαμπτο:

- Μη καθορισμένος αριθμός εικόνων - δεν έχει σημασία αν είναι 10 ή 100 εικόνες.
- Μη καθορισμένη σειρά των εικόνων - θα πρέπει να είναι εφικτή η αλλαγή των εικόνων σε οποιαδήποτε θέση.

Ιδού ο κώδικας με σχόλια:

```
<html>
<head>
<script type="text/javascript">
<!-- hide
var pics;
pics = new Array();
var objCount = 0; // αριθμός των εικόνων στην σελίδα
function preload(name, first, second) {
// φόρτωμα των εικόνων από πριν σε array
pics[objCount] = new Array(3);
pics[objCount][0] = new Image();
pics[objCount][0].src = first;
pics[objCount][1] = new Image();
pics[objCount][1].src = second;
pics[objCount][2] = name;
objCount++;
}
function on(name){
for (i = 0; i < objCount; i++) {
if (document.images[pics[i][2]] != null)
if (name != pics[i][2]) {
// set back all other pictures
document.images[pics[i][2]].src = pics[i][0].src;
} else {
// απεικόνιση της εικόνας στη θέση του δείκτη
document.images[pics[i][2]].src = pics[i][1].src;
}
}
}
function off(){
for (i = 0; i < objCount; i++) {
// θέτουμε πίσω όλες τις εικόνες
if (document.images[pics[i][2]] != null)
document.images[pics[i][2]].src = pics[i][0].src;
}

}
// Φόρτωμα των εικόνων από πριν - πρέπει να καθοριστεί
// ποιες εικόνες θα φορτωθούν από πριν και σε ποιο Image
// αντικείμενο θα ανήκουν (αυτό είναι το πρώτο argument).
preload("link1", "img1f.gif", "img1t.gif");
preload("link2", "img2f.gif", "img2t.gif");
```

```
preload("link3", "img3f.gif", "img3t.gif");
// -->
</script>
<head>
<body>
<a href="./link1.htm" onMouseOver="on('link1')" onMouseOut="off()">
<img name="link1" src="./link1f.gif" width="140" height="50" border="0"></a>
<a href="./link2.htm" onMouseOver="on('link2')" onMouseOut="off()">
<img name="link2" src="./link2f.gif"
width="140" height="50" border="0"></a>
<a href="./link3.htm" onMouseOver="on('link3')" onMouseOut="off()">
<img name="link3" src="./link3f.gif" width="140" height="50" border="0"></a>
</body>
</html>
```

Αυτό το script βάζει όλες τις εικόνες στο array pics. Η preload() function η οποία καλείται στην αρχή χτίζει αυτό το array. Μπορείτε να δείτε ότι καλούμε την preload() function ως εξής:

```
preload("link1", "img1f.gif", "img1t.gif");
```

Αυτό σημαίνει ότι το script θα πρέπει να φορτώσει τις δυο εικόνες img1f.gif και img1t.gif. Η πρώτη εικόνα είναι η εικόνα η οποία θα εμφανίζεται όταν ο δείκτης του ποντικιού δεν είναι μέσα στη περιοχή της εικόνας. Όταν ο χρήστης θα μετακινήσει το δείκτη πάνω απ' την εικόνα, η δεύτερη εικόνα θα εμφανιστεί. Με τη πρώτη παράμετρο "img1" όταν καλούμε την preload() function καθορίζουμε σε ποιο αντικείμενο Image στη σελίδα θα ανήκουν οι δυο εικόνες. Αν κοιτάξετε στο κύριο μέρος του εγγράφου, θα δείτε μια εικόνα με το όνομα img1. Χρησιμοποιούμε το όνομα της εικόνας (και όχι τον αριθμό της) για να μπορούμε να αλλάξουμε την σειρά των εικόνων χωρίς να αλλάζουμε το script. Οι δύο functions on() και off() καλούνται απ' τους event-handlers onMouseOver και onMouseOut. Επειδή οι εικόνες δεν μπορούν να αντιδράσουν στα MouseOver και MouseOut, βάλαμε ένα link γύρω απ' τις εικόνες. Όπως βλέπετε, η on() function βάζει πίσω όλες τις άλλες εικόνες. Αυτό είναι αναγκαίο γιατί μπορεί να είναι πολλές εικόνες highlighted (το event MouseOut δεν συμβαίνει, π.χ. όταν ο χρήστης μετακινήσει το δείκτη απ' την εικόνα κατ' ευθείαν έξω απ' το παράθυρο).

Οι εικόνες είναι σίγουρα ένας ωραίος τρόπος για να εμπλουτίσουμε τις σελίδες. Το αντικείμενο Image σας αφήνει να δημιουργήσετε πολύ ωραία εφέ. Αλλά σημειώστε ότι όχι όλες οι εικόνες και τα JavaScript προγράμματα κάνουν ωραίες τις σελίδες. Αν ρίξετε μια ματιά σε πολλές σελίδες, θα δείτε ότι χρησιμοποιούνται τέτοιες τεχνικές με πάρα πολύ άσχημο τρόπο. Δεν είναι η ποσότητα των εικόνων που κάνει τη σελίδα ωραία - είναι η ποιότητα. Είναι ιδιαίτερα ενοχλητικό να περιμένεις να κατεβούν 50 kB κακών γραφικών. Αν θυμηθείτε αυτό όταν φτιάχνετε JavaScript, οι επισκέπτες/ πελάτες είναι πιο πιθανό να ξαναδούν τη σελίδα.

Φόρμες

Εξετάζοντας την φόρμα

Οι φόρμες χρησιμοποιούνται συχνά στο Internet. Η φόρμα στέλνεται συνήθως μέσω του server ή μέσω

ταχυδρομείου σε ένα συγκεκριμένο e-mail λογαριασμό. Μα πώς μπορούμε να είμαστε σίγουροι ότι ο χρήστης συμπλήρωσε σωστά τα πεδία της φόρμας? Με τη βοήθεια του JavaScript, η φόρμα μπορεί εύκολα να ελεγχθεί πριν σταλθεί. Πρώτα, θέλω να σας δείξω πώς μπορούμε να εξετάσουμε μια φόρμα. Μετά, θα ρίξουμε μια ματιά στο να στέλνουμε πληροφορίες μέσω Internet.

Πρώτα απ 'όλα, θέλουμε να δημιουργήσουμε ένα script. Η HTML θα περιέχει δύο πεδία κειμένου. Ο χρήστης θα γράφει το όνομά του στο πρώτο και την e-mail διεύθυνση του στο δεύτερο. Μπορείτε να εισάγετε οτιδήποτε στα πεδία κειμένου και να πατήσετε το κουμπί. Επίσης, προσπαθήστε να μη βάλετε τίποτα και να πατήσετε το κουμπί.

```
Το όνομά σας:
Η e-mail διεύθυνση σας:
```

Όσον αφορά το πρώτο πεδίο κειμένου, θα λάβετε ένα μήνυμα λάθους, αν δεν γράψετε τίποτα. Κάθε άλλη τιμή του πεδίου θεωρείται σωστή. Φυσικά, αυτό δεν εμποδίζει τον χρήστη απ' το να γράψει ένα ψεύτικο όνομα. Ο browser δέχεται ακόμα και αριθμούς. Αν θέλετε λοιπόν να βάλετε την τιμή '17' θα πάρετε αποτέλεσμα 'Γεια 17!'. Γι' αυτό λοιπόν, αυτός δεν είναι ο καλύτερος έλεγχος. Η δεύτερη φόρμα είναι πιο έξυπνα φτιαγμένη. Προσπαθήστε να βάλετε ένα απλό κείμενο - το όνομά σας για παράδειγμα.

Δεν θα δουλέψει (εκτός αν έχετε ένα @ στο όνομά σας...). Αυτό το κριτήριο για να δεχθεί την τιμή σαν κανονική e-mail διεύθυνση είναι το @. Ένα απλό @ θα κάνει την δουλειά - αλλά αυτό δεν έχει και πολύ νόημα. Κάθε Internet e-mail διεύθυνση περιέχει @, έτσι φαίνεται σωστό να ελέγχουμε για @ εδώ.

Πώς δείχνει το script γι' αυτά τα δύο πεδία κειμένου και την εξέτασή τους? Ιδού ο κώδικας:

```
<html>
<head>
<script type="text/javascript">
<!-- Hide
function test1(form) {
if (form.text1.value == "")
alert("Παρακαλώ, εισάγετε μια τιμή!")
else {
alert("Γεια "+form.text1.value+"! Όλα ok!");
}
}
function test2(form) {
if (form.text2.value == "" ||
form.text2.value.indexOf('@', 0) == -1)
alert("Το e-mail δεν είναι σωστά γραμμένο!");
else alert("OK!");
}
// -->
</script>
</head>
<body>
<form name="first">
```

```
Enter your name:<br>
<input type="text" name="text1">
<input type="button" name="button1"
value="Δοκιμή πεδίου" onClick="test1(this.form)">
<P>
Enter your e-mail address:<br>
<input type="text" name="text2">
<input type="button" name="button2"
value="Δοκιμή πεδίου" onClick="test2(this.form)">
</body>
</html>
```

Πρώτα πρέπει να ρίξουμε μια ματιά στον HTML κώδικα στο κύριο μέρος. Δημιουργούμε δύο πεδία κειμένου και δύο κουμπιά. Τα κουμπιά καλούν τις functions test1(...) και test2(...), ανάλογα με ποιο κουμπί πατιέται. Περνάμε την τιμή this.form στις functions για να μπορούμε να καθορίσουμε τα σωστά πεδία στις functions αργότερα.
Η function test1(form) δοκιμάζει αν το πεδίο είναι κενό. Αυτό γίνεται γράφοντας if (form.text1.value == "")... . Το 'form' είναι η μεταβλητή που λαμβάνει την τιμή 'this.form' όταν καλείται η function. Μπορούμε να πάρουμε την τιμή του πεδίου κειμένου χρησιμοποιώντας το 'value' σε συνδυασμό με το form.text1. Για να κοιτάξουμε αν η τιμή είναι κενή, την συγκρίνουμε με "". Αν η τιμή είναι ίση με "", τότε δεν υπάρχει τιμή. Ο χρήστης θα πάρει το μήνυμα λάθους. Αν η τιμή δεν ισούται με "", τότε ο χρήστης θα λάβει ok. Το πρόβλημα εδώ είναι ότι ο χρήστης μπορεί να εισάγει μόνο κενά. Αυτό θα βλέπεται σαν σωστή τιμή! Αν θέλετε, μπορείτε να ελέγχετε γι' αυτές τις πιθανότητες και να τις αποκλείετε. Πιστεύω ότι θα 'ναι εύκολο με τις πληροφορίες που δίνονται εδώ. Τώρα, πρέπει να κοιτάξουμε στη function test2(form). Αυτή η function συγκρίνει ξανά την τιμή με το "", για να δει αν έχει εισαχθεί τιμή ή όχι. Αλλά έχουμε προσθέσει κάτι στο if. Το || λέγεται OR operator. Το if ελέγχει αν το πρώτο ή το δεύτερο κριτήριο σύγκρισης είναι σωστό. Αν τουλάχιστον το ένα απ' τα δύο είναι σωστό, η όλη if εντολή είναι σωστή και το ακόλουθο κείμενο θα εκτελεστεί. Αυτό σημαίνει ότι θα πάρετε ένα μήνυμα λάθους αν η τιμή είναι κενή ή δεν υπάρχει @ στο κείμενο. Το δεύτερο operation στην εντολή if κοιτάζει αν υπάρχει @ στη τιμή.

Ελέγχοντας για συγκεκριμένους χαρακτήρες

Μερικές φορές θέλετε να περιορίσετε τη φόρμα σε συγκεκριμένους χαρακτήρες ή αριθμούς. Σκεφτείτε ένα τηλεφωνικό αριθμό - η τιμή θα έπρεπε να έχει μόνο αριθμούς (υπολογίζουμε ότι ο αριθμός δεν περιέχει άλλους χαρακτήρες).Θα μπορούσαμε να ελέγξουμε αν το κείμενο είναι μόνο αριθμοί. Αλλά πολλοί άνθρωποι χρησιμοποιούν άλλους χαρακτήρες στους αριθμούς - π.χ.: 01234-56789, 01234/56789 ή 01234 56789 (με κενό, δηλαδή, ενδιάμεσα). Ο χρήστης δεν πρέπει να αναγκαστεί να μην χρησιμοποιήσει αυτά τα σύμβολα. Έτσι πρέπει να επεκτείνουμε το script να ελέγχει για ψηφία και σύμβολα. Ο πηγαίος κώδικας:

```
<html>
<head>
<script type="text/javascript">
<!-- hide
function check(input) {
```

```
var ok = true;
for (var i = 0; i < input.length; i++) {
var chr = input.charAt(i);
var found = false;
for (var j = 1; j < check.length; j++) {
if (chr == check[j]) found = true;
}
if (!found) ok = false;
}
return ok;
}
function test(input) {
if (!check(input, "1", "2", "3", "4",
"5", "6", "7", "8", "9", "0", "/", "-", " "))
{
alert("Input not ok.");
}
else {
alert("Input ok!");
}
}
// -->
</script>
</head>
<body>
<form> Telephone:
<input type="text" name="telephone" value="">
<input type="button" value="Έλεγχος"
onClick="test(this.form.telephone.value)">
</form>
</body>
</html>
```

Η function test() προσδιορίζει ποιοι χαρακτήρες επιτρέπονται.

Στέλνοντας μια φόρμα

Ποιοι διαφορετικοί τρόποι υπάρχουν για να στείλεις μία φόρμα; Ο απλός τρόπος είναι η αποστολή της φόρμας μέσω e-mail. Αυτή τη μέθοδο θα την αναλύσουμε σ' αυτό το σημείο. Αν θέλετε τη φόρμα να τη χειριστεί ο server, χρειάζεστε να χρησιμοποιήσετε CGI (Common Gateway Interface). Αυτό επιτρέπει την διαχείριση της φόρμας αυτόματα. Ο server μπορεί να δημιουργήσει για παράδειγμα, μια database από τις φόρμες που παραλήφθηκαν απ' τους πελάτες. Άλλο παράδειγμα είναι οι index-pages όπως το Yahoo. Έχουν μια φόρμα για να ψάχνουν στην database. Ο χρήστης παίρνει απάντηση γρήγορα, όταν πατηθεί το κουμπί submit. Δεν είναι ανάγκη να περιμένει

μέχρι οι άνθρωποι που συντηρούν τον server να διαβάσουν την φόρμα και να κοιτάξουν για την πληροφορία. Αυτή η δουλειά γίνεται αυτόματα απ' το server. Το JavaScript δεν μπορεί να κάνει τέτοια πράγματα.
Δεν μπορείτε να δημιουργήσετε guestbooks με τη JavaScript, γιατί η JavaScript δεν είναι ικανό να γράψει ένα αρχείο στον server. Αυτό μπορεί να γίνει μόνο μέσω CGI. Φυσικά, μπορείτε να δημιουργήσετε ένα guestbook μέσω e-mail. Αλλά θα πρέπει να βάζετε τις τιμές μόνοι σας. Αυτό είναι εντάξει αν δεν περιμένετε 1000 γράμματα την ημέρα. Το script είναι μόνο HTML. Έτσι, η JavaScript δεν χρειάζεται εδώ! Μόνο, αν χρειασθείτε να ελέγξετε τις τιμές των πεδίων θα χρειαστεί η JavaScript.

```
<form method=post action="mailto:your.address@goes.here" enctype="text/plain">
Σας αρέσει αυτή η σελίδα?
<input name="choice" type="radio" value="1"> καθόλου.<br>
<input name="choice" type="radio" value="2" CHECKED> Σπατάλημα χρόνου είναι.<br>
<input name="choice" type="radio" value="3">Η χειρότερη σελίδα που έχω δει.<br>
<input name="submit" type="submit" value="Αποστολή">
</form>
```

Η ιδιότητα enctype="text/plain" χρησιμοποιείται για να στείλει απλό κείμενο, κι όχι μέρη κώδικα. Αυτό κάνει την ανάγνωση του γράμματος πιο εύκολη.
Αν θέλετε να ελέγξετε την φόρμα πριν σταλθεί, μπορείτε να χρησιμοποιήσετε το onSubmit event- handler. Μπορείτε να βάλετε αυτόν τον event-handler στην εντολή <form>. Ο κώδικας θα δείχνει ως εξής:

```
function validate() {
// Έλεγχος της φόρμας
// ...
if (inputOK) return true else return false;
}
...
<form ... onSubmit="return validate()">
...
```

Με αυτό τον κώδικα, η φόρμα δεν στέλνεται, αν έχει γίνει κάποιο λάθος.

Πώς να κάνετε focus σε ένα συγκεκριμένο αντικείμενο

Με τη βοήθεια της μεθόδου focus(), μπορείτε να κάνετε την φόρμα σας πιο φιλική ως προς τον χρήστη. Μπορείτε να καθορίσετε ποιο στοιχείο της φόρμας είναι εστιασμένη στην αρχή. Ή μπορείτε να πείτε στον browser να εστιάσει στο πεδίο της φόρμας που είχε λάθος τιμή. Αυτό σημαίνει ότι ο browser θα τοποθετήσει τον κέρσορα στο στοιχείο της φόρμας, έτσι ο χρήστης δεν είναι ανάγκη να κάνει click στο πεδίο πριν εισάγει τιμή Το ακόλουθο script θα κάνει την δουλειά αυτή:

```
function setfocus() {
document.first.text1.focus();
}
bla bla bla
```

Αυτό το script θα εστίαζε το πρώτο πεδίο κειμένου χάρις στο script που έδειξα πάνω. Πρέπει να καθορίσετε το όνομα της φόρμας - που εδώ λέγεται first - και το όνομα του κάθε στοιχείου - εδώ text1. Αν θέλετε να εστιάσετε

αυτό το στοιχείο της φόρμας όταν φορτώνεται η σελίδα, μπορείτε να θέσετε την ιδιότητα onLoad στην εντολή <body>. Αυτό θα έδειχνε ως εξής:

```
<body onLoad="setfocus()">
```

Μπορούμε επίσης να μεγαλώσουμε το κώδικα ως εξής:

```
function setfocus() { document.first.text1.focus(); document.first.text1.select();
}
```

Άλλα ενδιαφέροντα συμβάντα

Resize

Με τη βοήθεια αυτού του event ξέρουμε πότε ο χρήστης αλλάζει το μέγεθος του παραθύρου. Το ακόλουθο script δείχνει αυτό:

```
<html>
<head>
<script type="text/javascript"> window.onresize= message; function message() {
alert("Το παράθυρο έχει αλλάξει μέγεθος!");
}
</script>
</head>
<body>
Παρακαλώ, αλλάξτε το μέγεθος του παραθύρου.
</body>
</html>
```

Με τη γραμμή window.onresize= message; ρυθμίζουμε τον event-handler. Αυτό σημαίνει ότι η function message() καλείται όταν το παράθυρο αλλάζει μέγεθος. Αν, για παράδειγμα, έχετε ένα κουμπί, μπορείτε να ρυθμίσετε το event-handler ως εξής:

```
<form name="myForm">
<input type="button" name="myButton" onClick=" alert('Κάποιος με πάτησε!')">
</form>
```

Αλλά μπορείτε και να γράψετε το εξής:

```
<form name="myForm">
<input type="button" name="myButton">
</form>
...
<script language="JavaScript>
document.myForm.myButton.onclick= message;
function message() {
alert('Κάποιος με πάτησε!');
}
</script>
```

Μπορεί να βρίσκετε την δεύτερη περίπτωση πιο πολύπλοκη. Γιατί λοιπόν την χρησιμοποιούμε στο πρώτο script?

Το πρόβλημα είναι ότι το αντικείμενο window δεν καθορίζεται μέσω μιας HTML εντολής - έτσι, πρέπει να χρησιμοποιήσουμε τον δεύτερο τρόπο. Δύο σημαντικά πράγματα: Πρώτον, δεν πρέπει να γράψετε window.onResize - δηλαδή, πρέπει να χρησιμοποιήσετε μικρά γράμματα, όχι κεφαλαία. Δεύτερον, δεν πρέπει να γράψετε παρένθεση μετά το message. Αν γράψετε window.onresize= message() ο browser μεταφράζει το message() σαν κάλεσμα function. Αλλά, σε αυτή τη περίπτωση δεν θέλουμε να καλέσουμε την function απ' ευθείας - θέλουμε απλώς να ρυθμίσουμε το event-handler.

Το αντικείμενο Event

Ένα νέο αντικείμενο, το Event έχει προστεθεί στη JavaScript 1.2. Περιλαμβάνει properties τα οποία περιγράφουν γεγονότα. Κάθε φορά που ένα event συμβαίνει, το αντικείμενο Event περνά στο event handler: ένα μήνυμα θα εμφανιστεί με τις συντεταγμένες του ποντικιού και πατήσετε πάνω στην εικόνα, όταν συνέβηκε το event: Ιδού και ο πηγαίος κώδικας:

```
<layer>
<a href="#" onClick="alert('x: ' + event.x + ' y: ' + event.y); return false;">
<img src="./davinci.jpg" width=209 height=264 border=0></a>
</layer>
```

Μπορείτε να δείτε ότι χρησιμοποιώ τον event handler onClick μέσα στη <a> εντολή, όπως θα κάναμε και με παλιότερες JavaScript εκδόσεις. Αυτό που είναι καινούργιο είναι ότι χρησιμοποιούμε το event.x και το event.y για τη δημιουργία της τιμής του παραθύρου. Το αντικείμενο Event είναι αυτό που χρειαζόμαστε για να ξέρουμε την θέση του mouse πάνω στην εικόνα. Έβαλα τα πάντα μέσα σε ένα <layer>. Έτσι θα πάρουμε τις συντεταγμένες σε σχέση με το layer, δηλαδή την εικόνα. Διαφορετικά θα παίρναμε τις συντεταγμένες του ποντικιού πάνω στο παράθυρο. (Το return false; χρησιμοποιείται ώστε ο browser να μην πάει στην διεύθυνση.)
Το αντικείμενο Event έχει τις ακόλουθες ιδιότητες (θα δούμε κάποιες απ' αυτές στα ακόλουθα παραδείγματα):

Ιδιότητα	**Περιγραφή**
Data	Array με URLs των dropped αντικειμένων όταν ένα DragDrop event συμβαίνει.
layerX	Οριζόντια θέση του ποντικιού σε pixel πάνω σε layer. Σε συνδυασμό με το Resize event, αυτή η ιδιότητα παρουσιάζει το μήκος του παραθύρου.
layerY	Κάθετη θέση του ποντικιού σε pixel πάνω σε layer. Σε συνδυασμό με το Resize event, αυτή η ιδιότητα παρουσιάζει το ύψος του παραθύρου.
modifiers	Μεταβλητή που δείχνει την κατάσταση των πλήκτρων αλλαγής (Shift, Caps Lock, Alt, κ.λ.π.) - παίρνει τις τιμές ALT_MASK, CONTROL_MASK, META_MASK, SHIFT_MASK.
pageX	Οριζόντια θέση του ποντικιού σε pixel πάνω στο παράθυρο. pageY Κάθετη θέση του ποντικιού σε pixel πάνω στο παράθυρο. screenX Οριζόντια θέση του ποντικιού σε in pixel πάνω στην οθόνη. screenY Κάθετη θέση του ποντικιού σε pixel πάνω στην οθόνη.
target	Μεταβλητή που αντιπροσωπεύει το event στο οποίο στάλθηκε.
type	μεταβλητή που παρουσιάζει τον τύπο του event.
which	ASCII τιμή του πλήκτρου που πατήθηκε, ή του πλήκτρου του ποντικιού.

x	Συνώνυμο με το layerX
y	Συνώνυμο με το layerY

Καταλαβαίνοντας τα Events

Μια σημαντική λειτουργία είναι η "φυλάκιση" του event (δηλαδή να καταλαβαίνουμε πότε ένα γεγονός συμβαίνει και να ενεργούμε ανάλογα). Αν κάποιος για παράδειγμα πατήσει ένα κουμπί, το onClick event handler του κουμπιού αυτού καλείται. Με τη βοήθεια της "φυλάκισης" του event μπορείτε να έχετε ως αποτέλεσμα το παράθυρο, έγγραφο ή layer να φυλακίζει το event πριν το χειριστεί το ίδιο το κουμπί. Έτσι το παράθυρο, έγγραφο ή layer μπορεί να χειριστεί το γεγονός πριν φτάσει στο στόχο του (το κουμπί). Ας κοιτάξουμε το επόμενο παράδειγμα, για να δούμε πώς αυτό χρησιμεύει:

```
<html>
<head>
<script type="text/javascript"> window.captureEvents(Event.CLICK); window.onclick= handle;
function handle(e) {
alert("Το παράθυρο φυλάκισε το γεγονός!");
return true; // δηλ. ακολουθία του link
}
</script>
</head>
<body>
<a href="test.htm">Πατήστε αυτό το link</a>
</body>
</html>
```

Μπορείτε να δείτε ότι δεν θέτουμε ένα event handler μέσα στο <a>. Απεναντίας, χρησιμοποιούμε το

```
window.captureEvents(Event.CLICK);
```

για να φυλακίσουμε το Click event μέσω του αντικειμένου window. Κανονικά το window δεν ξέρει για το Click event. Αλλά, με το να φυλακίσουμε το event μπορούμε να το μεταθέσουμε στο window. Σημειώστε τη γραφή του Event.CLICK. Το CLICK πρέπει να γραφεί με κεφαλαία. Αν θέλετε να φυλακίσετε πολλά γεγονότα, πρέπει να τα διαχωρίσετε με | - παράδειγμα:

```
window.captureEvents(Event.CLICK | Event.MOVE);
```

Μπορείτε να δείτε ότι χρησιμοποιούμε το return true; μέσα στη function handle() την οποία ρυθμίσαμε σαν event handling function. Αυτό σημαίνει ότι ο browser θα ακολουθήσει το link μετά την εκτέλεση της handle() function. Αν γράψετε return false;, όλες οι ακόλουθες ενέργειες παραλείπονται.

Αν ορίσετε ένα onClick event handler μέσα στο <a> θα δείτε ότι αυτός ο event handler δεν καλείται. Αυτό είναι φανερό, μιας και το window φυλάκισε το event πριν φτάσει το link. Αν ορίσετε τη handle() function ως εξής

```
function handle(e) {
alert("Το παράθυρο φυλάκισε το γεγονός!");
window.routeEvent(e);
return true;
}
```

ο υπολογιστής κοιτάζει αν υπάρχουν άλλοι event handlers καθορισμένοι απ' αυτό το αντικείμενο. Η μεταβλητή e είναι το αντικείμενο Event το οποίο προσπελάζεται όταν εκτελείται η function. Μπορείτε επίσης να στείλετε ένα event απ' ευθείας σε ένα συγκεκριμένο αντικείμενο. Γι' αυτό μπορούμε να χρησιμοποιήσουμε τη μέθοδο handleEvent(). Αυτό θα δείχνει ως εξής:

```
<html>
<script type="text/javascript"> window.captureEvents(Event.CLICK); window.onclick= handle;
function handle(e) {
document.links[1].handleEvent(e);
}
</script>
<a href="test.htm">Πατήστε αυτό το link</a><br>
<a href="test.htm"
onClick="alert('Event handler στο δεύτερο link!');"> Δεύτερο link</a>
</html>
```

Όλα τα Click events στέλνονται στο δεύτερο link - ακόμα κι αν δεν πατήσετε απ' ευθείας στα links! Το ακόλουθο script δείχνει ότι το script μπορεί να αντιδράσει σε πατήματα των πλήκτρων. Απλώς πατήστε ένα κουμπί για να δείτε τη λειτουργία εν δράση.

```
<html>
<script type="text/javascript"> window.captureEvents(Event.KEYPRESS); window.onkeypress=
pressed;
function pressed(e) {
alert("Ενα κουμπί πατήθηκε! Η ASCII τιμή του: " + e.which);
}
</script>
</html>
```

Παραδειγματικά σενάρια

Καταγραφή ημερομηνίας τροποποίησης

Απόκρτυψη σεναρίων

Άνοιγμα δευτερογενών παραθύρων

Αυτόματη προώθηση των ιστοσελίδων

Αυτόματη προώθηση με επιβεβαίωση

Πίνακας περιεχομένων