Programare web client-side, JavaScript

Ciprian Dobre
ciprian.dobre@cs.pub.ro

Obiective

- Scopul acestui capitol nu este acela de a face o descriere completa a JavaScript si a notiunilor de programare avansata pe care limbajul le suporta
- Exista numeroase documente şi carți publicate inclusiv în limba româna care conțin informații complete în domeniu.
- Prezentam în continuare doar o trecere in revista a limbajului.

Sumar

- JavaScript – fundamente
- JavaScript si HTML - gestiunea simpla a evenimentelor
- Exemple JavaScript
- Mai multe despre JavaScript
- JavaScript si AJAX

JavaScript

Fundamentele Limbajului

Despre JavaScript

- JavaScript *nu este Java,* nici macar *inrudit* cu Java
  - Numele original al JavaScript a fost “LiveScript”
  - Numele a fost modificat cand Java a devenit popular
  - Astazi cand Microsoft nu mai indrageste Java propriul nume acordat dialectului JavaScript este “Active Script”
- Instructiunile in JavaScript seamana cu instructiuni in Java deoarece ambele limbaje au imprumutat multe elemente chiar din limbajul C
  - JavaScript este relativ usor de invatat pentru programatori familiarizati cu Java
  - Totusi JavaScript *este* un limbaj de sine statator, complet si complex
- JavaScript este rar folosit pentru a scrie “programe” complete
  - Mici blocuri de instructiuni JavaScript sunt in general folosite pentru a adauga functionalitate paginilor HTML
  - JavaScript este adesea folosit in conjunctie cu formulare HTML
- JavaScript este *intr-o buna* masura independent de platforma

Folosirea JavaScript intr-un browser

- Codul JavaScript este inclus intre taguri <script>:
  - ```
    <script type="text/javascript">
        document.write("<h1>Hello World!</h1>") ;
    </script>
    ```
- Observatii:
  - Atributul type permite folosirea altor limbaje de scripting (chiar daca JavaScript este limbajul implicit)
  - Acest exemplu simplu face acelasi lucru ca a scrie <h1>Hello World!</h1> in acelasi loc in documentul HTML
  - Punct si virgula de la finalul instructiunii JavaScript este optional
    - Sfarsitul liniei marcheaza sfarsitul instructiunii, *daca* linia poate fi interpretata ca o instructiune completa
    - Se poate folosi punct si virgula pentru separarea mai multor instructiuni ce apar pe aceeasi linie
    - Este indicat totusi folosirea in permanenta a caracterului punct si virgula
    ```

JavaScript nu este intotdeauna disponibil

- Unele browsere mai vechi nu recunosc tag-ul script
  - Aceste browsere vor ignora tag-ul script dar vor *afisa* ce este inclus in interiorul tag-ului (codul JavaScript inclus)
  - Pentru a face aceste browsere sa ignore continutul putem folosi:

```
<script type="text/javascript">
<!--
    document.write("Hello World!")
//-->
</script>
```

  - Caracterul <!-- introduce un comentariu HTML
  - Pentru a face ca JavaScript sa ignore codul de sfarsit de comentariu HTML, -->, secventa // marcheaza un comentariu JavaScript, ce tine pana la sfarsitul liniei respective
- Unii utilizatori opresc JavaScript
  - Folositi <noscript>***message***</noscript> pentru a afisa un mesaj indiferent daca in interiorul acestuia apare si cod JavaScript

Exemplu

- John Smith's email:

```
<script language=javascript>
    <!--
    var name = "smithj"
    var host1 = "seas.up"
    var host2 = "enn.edu"
    var addr =
    document.write("<a href=mai" + "lto:" +
        name + "&#64;" + host1 + host2 +
        "?subject=CIT597:" +
        ">" + "John Smith" + "</a>")
    //-->
</script>
<noscript>
smithj at seas
</noscript>
```

Unde apare codul JavaScript

- Codul JavaScript poate apare fie in sectiunea <head>, fie in sectiunea <body> a unui document HTML
  - *Functiile* JavaScript ar trebui sa fie definite in sectiunea <head>
    - Acest lucru asigura ca functiile sunt incarcate chiar inainte ca ele sa fie cerute
  - Codul JavaScript din <body> este executat atunci cand este incarcata pagina
- Functiile JavaScript pot fi plasate si in fisiere separate avand in general extensia .js
  - <script src="myJavaScriptFile.js"></script>
  - Codul acesta se plaseaza in sectiunea <head>
  - Un fisier extern .js permite folosirea aceluiasi cod JavaScript in mai multe pagini HTML
  - Fisierul extern .js nu poate contine la randul sau alt tag <script>
- Codul JavaScript poate fi plasat si impreuna cu un *obiect al unui formular*, ca de exemplu un buton
  - Codul JavaScript va fi executat atunci cand respectivul obiect este folosit

Tipuri de date primitive

- JavaScript are trei tipuri “primitive”: number, string si boolean, si doua valori speciale, null si undefined
  - Orice altceva este un obiect
- Numerele sunt intotdeauna stocate ca valori de tip float
  - Numerele hexazecimale incep cu 0x
  - Unele platforme trateaza 0123 ca octal, altele il trateaza ca zecimal
    - Pentru ca nu puteti fi siguri, e mai indicat sa evitati cu totul folosirea valorilor in octal!
- String-urile pot fi delimitate prin apostroafe sau ghilimele
  - String-urile pot contine \n (newline), \" (double quote), etc.
- Valorile boolean sunt fie true fie false
  - 0, "0", stringul gol, undefined, null si NaN sunt false, restul valorilor sunt true

Variabile

- Variabilele pot fi declarate folosind var:
  - var pi = 3.1416, x, y, name = "Dr. Dave" ;
  - Numele de variabile trebuie sa inceapa cu o litera sau cu underscor
  - Numele de variabile sunt case-sensitive
  - Variabilele *nu au tip* (ele pot tine valori de orice tip)
- Exista doar doua scopuri ale variabilelor: local si global
  - Variabilele declarate intr-o functie sunt locale respectivei functii (accesibile numai din interiorul respectivei functii)
  - Variabilele declarate in afara unei functii sunt globale (accesibile de oriunde din pagina)
- Variabilele pot fi declarate implicit prin simpla adaugare a unei valori unor variabile
  - Variabilele declarate implicit sunt intotdeauna globale

Operatori (1)

- Deoarece majoritatea sintaxei JavaScript este imprumutata din C (si seamana prin urmare cu cea din Java), vom parcurge aceste lucru destul de repede

- Operatori aritmetici (toate numerele sunt in virgula mobila):
  `+ - * / % ++ --`
- Operatori de comparatie:
  `< <= == != >= >`
- Operatori logici:
  `&& || !` (`&&` and `||` are *short-circuit* operators)
- Operatori la nivel de biti:
  `& | ^ ~ << >> >>>`
- Operatori de asignare:
  `+= -= *= /= %= <<= >>= >>>= &= ^= |=`

Operatori (2)

- Operator la nivel de stringuri:
  +
- Operatorul de conditie:
  condition ? ***value_if_true*** : ***value_if_false***
- Teste de egalitate:
  - == si!= incearca convertirea operatorilor la acelasi tip inaintea efectuarii testului
  - **Nu ca in C sau Java:** === si !== considera operanzii *inegali* daca acestia sunt de tipuri diferite
- Operatori suplimentari (vor fi discutati):
  new typeof void delete

Comentarii

- Comentariile sunt similare precum in C sau Java:
  - Intre // si sfarsitul liniei
  - Intre /* si */
- Comentariile de tip javadoc din Java, /** … */, sunt tratate similar comentariilor de tip /* … */; ele nu au nici o semnificatie speciala in JavaScript

Instructiuni (1)

- Majoritatea sintaxelor de instructiuni JavaScript sunt imprumutate din C
  - Atribuirea: `greeting = "Hello, " + name;`
  - Instructiunea compusa:
    { ***statement***; …; ***statement*** }
  - Instructiunea If:
    if (***condition***) ***statement***;
    if (***condition***) ***statement***; else ***statement***;
  - Instructiuni de iterare:
    while (***condition***) ***statement***;
    do ***statement*** while (***condition***);
    for (***initialization***; ***condition***; ***increment***) ***statement***;

Instructiuni (2)

- Instructiunea switch:

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

- Alte instructiuni familiare:
  - break;
  - continue;
  - Instructiunea vida, precum;; sau { }

JavaScript nu este Java

- Deja realizati ca *stiti* destul de multe despre JavaScript
  - Pana acum am vazut aspecte ce sunt *similare* precum in Java
- JavaScript are unele constructii ce *seamana* cu constructiile corespondente din Java:
  - JavaScript are Obiecte si tipuri de date primitive
  - JavaScript are nume calificate; de exemplu, `document.write("Hello World");`
  - JavaScript are evenimente si gestiunea evenimentelor (event handlers)
  - Gestiunea exceptiilor in JavaScript este *aproape* similara cu cea din Java
- Totusi JavaScript are unele constructii *diferite* de cele din Java:
  - *Numele de variabile* nu au tip: tipul unei variabile depinde de valoarea curenta pe care respectiva variabila o detine
  - Obiectele si arrayurile sunt definite intr-o maniera diferita de cea din Java
  - JavaScript are instructiunea `with` si o noua forma a instructiunii `for`

Gestiunea exceptiilor (1)

- Gestiunea exceptiilor in JavaScript este *aproape* la fel ca cea din Java
- ***expresia*** throw creaza si arunca o exceptie
  - ***Expresia*** este valoarea exceptiei si poate fi de *orice* tip (adesea este un String)
- try {
  statements to try
  } catch (***e***) { // Notice: no type declaration for ***e***
  exception handling statements
  } finally { // optional, as usual
  code that is always executed
  }
  - Folosind aceasta abordare exista o *singura* clauza catch

Gestiunea exceptiilor (2)

- try {
  statements to try
  } catch (***e*** if ***test1***) {
  exception handling for the case that test1 is true
  } catch (***e*** if ***test2***) {
  exception handling for when test1 is false and test2 is true
  } catch (***e***) {
  exception handling for when both test1and test2 are false
  } finally { // optional, as usual
  code that is always executed
  }
- In mod normal testul poate fi ceva de genul
  e == "InvalidNameException"
  dar poate fi orice alt tip de test

Obiecte

- In Java clasele descriu obiecte si toate obiectele aceleiasi clase au exact aceleasi campuri si metode
  - Obiectele JavaScript sunt mai flexibile decat obiectele Java
- JavaScript lucreaza cu "object *literals*", scrisi conform urmatoarei sintaxe:
  - { ***name1*** : ***value1*** , … , ***nameN*** : ***valueN*** }
- Exemplu (din documentatia Netscape):
  - car = {myCar: "Saturn", 7: "Mazda",
    getCar: CarTypes("Honda"), special: Sales}
    - Campurile sunt myCar, getCar, 7 (este chiar un nume legal de camp) si special
    - "Saturn" si "Mazda" sunt campuri de tip String
    - CarTypes este un apel de functie
    - Sales este o variabila ce a fost definita anterior
  - Exemplu de folosire: document.write("I own a " + car.myCar);

Trei moduri de a crea un obiect

- Puteti folosi un *object literal*:
  - `var course = { number: "CIT597", teacher: "Dr. Dave" }`
- Puteti folosi new pentru a crea un obiect “blank” object caruia sa ii adaugati campuri ulterior:
  -
```
var course = new Object();
course.number = "CIT597";
course.teacher = "Dr. Dave";
```
- Puteti scrie si folosi un constructor:
  -
```
function Course(n, t) {  // functions should be defined in <head>
    this.number = n;     // keyword "this" is required, not optional
    this.teacher = t;
}
```
  - `var course = new Course("CIT597", "Dr. Dave");`

Array literals

- JavaScript are array *literals,* scrisi cu paranteze drepte si virgule
  - Exemplu: color = ["red", "yellow", "green", "blue"];
  - Array-urile sunt initializate de la pozitia 0*:* color[0] este "red"
- Daca puneti doua virgule succesive array-ul va avea un element “empty” la acea pozitie
  - Exemplu: color = ["red", , , "green", "blue"];
    - color are 5 elemente
  - Totusi, o singura virgula la final este ignorata
    - Exempul: color = ["red", , , "green", "blue”,]; are tot 5 elemente

Patru moduri de a crea un array

- Puteti folosi un array literal:
  var colors = ["red", "green", "blue"];
- Puteti folosi new Array() pentru a crea un array gol:
  - var colors = new Array();
  - Puteti adauga elemente ulterior la acel array:
    colors[0] = "red"; colors[2] = "blue"; colors[1]="green";
- Puteti folosi new Array(***n***) cu un singur argument numeric pentru a crea un array de respectiva dimensiune
  - var colors = new Array(3);
- Puteti folosi new Array(...) cu *doua sau mai multe* argumente pentru a crea un array continand respectivele valori:
  - var colors = new Array("red","green", "blue");

Lungimea unui array

- Daca myArray este un array lungimea acestuia este intoarsa de catre myArray.length
- Dimensiunea unui array poate fi modificata prin adaugarea de elemente peste lungimea curenta a acestuia
  - Exemplu: var myArray = new Array(5); myArray[10] = 3;
- Array-urile sunt risipite, spatiul este alocat pentru elementele carora le-a fost cu adevarat asignata o valoare
  - Exemplu: myArray[50000] = 3; este perfect OK
  - Indicii trebuie sa fie intre 0 si 2^{32}-1
- Similar ca in C si Java, nu exista array-uri bidimensionale; dar exista array de array: myArray[5][3]

Array-uri si obiecte

- Array-urile *sunt* obiecte
- car = { myCar: "Saturn", 7: "Mazda" }
  - car[7] este echivalent cu car.7
  - car.myCar este acelasi cu car["myCar"]
- Daca *cunoasteti* numele unei proprietati puteti folosi notatia cu punct: car.myCar
- Daca *nu cunoasteti* numele proprietatii dar il aveti intr-o variabila (sau il puteti calcula), *folosi* notatia de tip array: car["my" + "Car"]

Functii aplicabile array-urilor

- Daca myArray este un array,
  - myArray.sort() sorteaza array-ul alfabetic
  - myArray.sort(function(a, b) { return a - b; }) sorteaza numeric
  - myArray.reverse() inverseaza elementele array-ului
  - myArray.push(...) adauga orice numar de elemente la finalul array-ului si creste dimensiunea acestuia
  - myArray.pop() inlatura si intoarce ultimul element al array-ului si decrementeaza dimensiunea acestuia
  - myArray.toString() intoarce un string continand valoarile elementelor array-ului, separate prin virgula

Instructiunea for … in

- Similara instructiunii iterative din Java for(***type var*** : ***collection***)
- Puteti itera prin proprietatile unui obiect folosind for (***variable*** in ***object***) ***statement***;
  - Exemplu:
```
for (var prop in course) {
    document.write(prop + ": " + course[prop]);
}
```
  - Posibil output:
```
teacher: Dr. Dave
number: CIT597
```
  - Proprietatile sunt accesate intr-o ordine *nedefinita*
  - Daca adaugati sau stergeti proprietati obiectului din loop este *nedefinit* daca instructiunea iterativa va traversa si proprietatile nou definite sau nu
  - Array-urile *sunt* obiecte; aplicata unui array, for…in va vizita "proprietatile" 0, 1, 2, …
  - Observati ca course["teacher"] este echivalent cu course.teacher
    - Trebuie sa folositi paranteze drepte daca numele proprietatii este *intr-o variabila*

Instructiunea with

- with (***object***) ***statement*** ; *foloseste* ***object*** ca prefix implicit al variabilelor din ***statement***
  - Daca campurile accesate nu exista prefixul nu va fi folosit
- De exemplu, urmatoarele sunt echivalente:
  - ```
    with (document.myForm) {
        result.value = compute(myInput.value) ;
    }
    ```
  - ```
    document.myForm.result.value =
        compute(document.myForm.myInput.value);
    ```
- Instructiunea with este utila atunci cand aveti de facut multe manipulari asupra unui acelasi obiect
  - Instructiunea with poate fi confuzanta si din acest motiv trebuie folosita cu discernamant

Functii

- Functiile trebuie definite in sectiunea <head> a paginii HTML pentru a asigura faptul ca ele sunt incarcate primele
- Sintaxa pentru definirea unei functii este:
  function ***name***(***arg1***, ***…***, ***argN***) { ***statements*** }
  - Functia poate contine instructiuni de tipul return ***value***;
  - Orice variabila declarata in interiorul unei functii este locala respectivei functii
- Sintaxa pentru apelarea unei functii este
  name(***arg1***, ***…***, ***argN***)
- Parametrii simpli sunt transmisi prin *valoare,* obiectele sunt transmise prin *referinta*

Expresii regulate

- O expresie regulata poate fi scrisa in oricare din urmatoarele forme:
  - Folosind un constructor, precum in re = new RegExp("ab+c")
  - Folosind slash-uri, precum in re = /ab+c/
- Expresiile regulate sunt aproape la fel precum in Perl sau Java (doar cateva elemente mai putin folosite lipsesc)
- ***string***.match(***regexp***) are rolul de a cauta in ***string*** aparitii ale ***regexp***
  - Intoarce null daca nu este gasit nimic
  - Daca ***regexp*** are setat flag-ul g (global search), match intoarce un array cu toate substringurile ce se potrivesc expresiei
  - Daca g nu este setat, match intoarce un array al carui element de pe pozitia 0 reprezinta textul ce se potriveste expresiei, iar extra elementele sunt subexpresiile paranterizate, si proprietatea index este setata pe pozitia de inceput a substringului ce se potriveste expresiei

Atentionari

- JavaScript este un limbaj mare, complex
  - Am prezentat doar elemente de la suprafata acestuia
  - E usor de pornit cu JavaScript, dar daca aveti nevoie sa il folositi intr-o maniera mai complexa trebuie sa va faceti timp pentru a-l invata cu adevarat
  - Scrierea si testarea programelor necesita un pic de efort din partea echipei de dezvoltare
- JavaScript nu este total independent de platforma
  - Trebuie sa va asteptati ca pe diverse browsere codul sa se comporte in mod diferit
  - Scrierea si testarea programelor necesita un pic de efort din partea echipei de dezvoltare
- Browserele in general nu raporteaza erori
  - Nu va asteptati sa primiti ceva mesaje ajutatoare care sa va ajute in munca de depanare a codului JavaScript
  - Scrierea si testarea programelor necesita un pic de efort din partea echipei de dezvoltare

Quiz

- Scrieti codul corespunzator unei pagini HTML in care sa aveti un obiect Car, un constructor declarat in <head> si care sa afiseze valoarea curenta a campului *type* in document

JavaScript si HTML

Gestiunea Simpla a Evenimentelor

JavaScript si DOM

- JavaScript se bazeaza pe un Document Object Model (DOM) ce descrie structura paginii web
  - Nu reprezinta acelasi lucru ca XML DOM
- Puteti face multe lucruri intelegand un pic conceptul de DOM
  - Puteti folosi DOM pentru a accesa elementele paginii web
  - Puteti capta evenimente fara sa fiti deloc familiarizati cu conceptul DOM
  - Aveti nevoie de DOM pentru a face eventuale modificari in pagina web

Evenimente

- Unele (dar nu toate) elementele paginii web raspund la interactivitatea din partea utilizatorului (keystrokes, mouse clicks) prin crearea de evenimente
  - Diverse tipuri de elemente produc evenimente diferite
    - Browserele nu seamana nici cand vine vorba de tipurile de evenimente pe care le produc
  - Ne vom concentra pe evenimente ce tin de elementele specifice unor formulare HTML
- Puteti folosi handlere asociate unor elemente de formulare HTML
  - Daca evenimentul nu este generat handlerul nu face nimic
  - Un handler ar trebui sa fie foarte reduc ca dimensiune
    - Majoritatea handlerelor apeleaza o functie in care se produce toata functionalitatea specifica evenimentului

Un handler de evenimente simplu

- <form method="post" action="">
  <input type="button"
  name="myButton"
  value="Click me"
  onclick="alert('You clicked the button!');">
  </form>
  - Butonul este incorporat intr-un formular
    - method specifica *cum* sunt trimite datele formularului; action specifica *unde* sunt ele trimise
  - Tag-ul este input, avand atributul type="button"
  - Atributul name poate fi folosit de alt cod JavaScript
  - Atributul value va aparea pe buton
  - onclick este numele evenimentului ce se vrea a fi gestionat
    - Valoarea elementului onclick este codul JavaScript care va fi executat
    - alert are rolul de a declansa o fereastra de pop-up de tip alerta cu un mesaj furnizat ca argument

Conventii de scriere

- JavaScript este case sensitive
- HTML nu este case sensitive
- onclick="alert('You clicked the button!');"
  - Partile subliniate tin de codul HTML
  - Stringul este cod JavaScript
  - Veti vedea adesea metoda onclick scrisa si sub forma onClick
    - Conventiile de nume Java sunt mai usor de citit
    - Acest lucru este permis in HTML, dar daca este folosit in cod JavaScript este generat automat un cod de *eroare*
- Observatie: Deoarece avem un string in interiorul altui string avem nevoie de atat varianta cu ghilimele,cat si de cea cu apostroafe pentru reprezentarea stringurilor

Evenimente comune

- Majoritatea elementelor HTML produc urmatoarele evenimente:
  - onClick – elementul respectiv este clicked
  - onDblClick – pe elementul respectiv se executa dublu-click intr-o succesiune rapida
  - onMouseDown – butonul de mouse este apasat cand cursorul era deasupra elementului
  - onMouseOver – cursoul mouse-ului este mutat deasupra elementului
  - onMouseOut – cursorul mouse-ului este scos in afara ariei elementului
  - onMouseUp – butonul mouse-ului este eliberat cand cursorul era inca deasupra elementului
  - onMouseMove –mouse-ul este mutat
- In JavaScript, acestea trebuie scrise cu litere mici

Exemplu: un rollover simplu

- Urmatorul cod va face textul Hello
  red atunci cand cursorul mouse-ului trece deasupra lui si
  blue atunci cand cursorul mouse-ului iese din suprafata elementului

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

- Un rollover aplicabil unei imagini:

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

Evenimente si gestionare de evenimente (1)

- Urmatoarele tabele sunt preluate de la: http://developer.netscape.com/docs/manuals/js/client/jsguide/index.htm

Event	Applies to	Occurs when	Handler
Load	Document body	User loads the page in a browser	onLoad
Unload	Document body	User exits the page	onUnload
Error	Images, window	Error on loading an image or a window	onError
Abort	Images	User aborts the loading of an image	onAbort

Evenimente si gestionare de evenimente (2)

Event	Applies to	Occurs when	Handler
KeyDown	Documents, images, links, text areas	User depresses a key	onKeyDown
KeyUp	Documents, images, links, text areas	User releases a key	onKeyUp
KeyPress	Documents, images, links, text areas	User presses or holds down a key	onKeyPress
Change	Text fields, text areas, select lists	User changes the value of an element	onChange

Evenimente si gestionare de evenimente (3)

Event	Applies to	Occurs when	Handler
MouseDown	Documents, buttons, links	User depresses a mouse button	onMouseDown
MouseUp	Documents, buttons, links	User releases a mouse button	onMouseUp
Click	Buttons, radio buttons, checkboxes, submit buttons, reset buttons, links	User clicks a form element or link	onClick

Evenimente si gestionare de evenimente (4)

Event	Applies to	Occurs when	Handler
MouseOver	Links	User moves cursor over a link	onMouseOver
MouseOut	Areas, links	User moves cursor out of an image map or link	onMouseOut
Select	Text fields, text areas	User selects form element's input field	onSelect

Evenimente si gestionare de evenimente (5)

Event	Applies to	Occurs when	Handler
Move	Windows	User or script moves a window	onMove
Resize	Windows	User or script resizes a window	onResize
DragDrop	Windows	User drops an object onto the browser window	onDragDrop

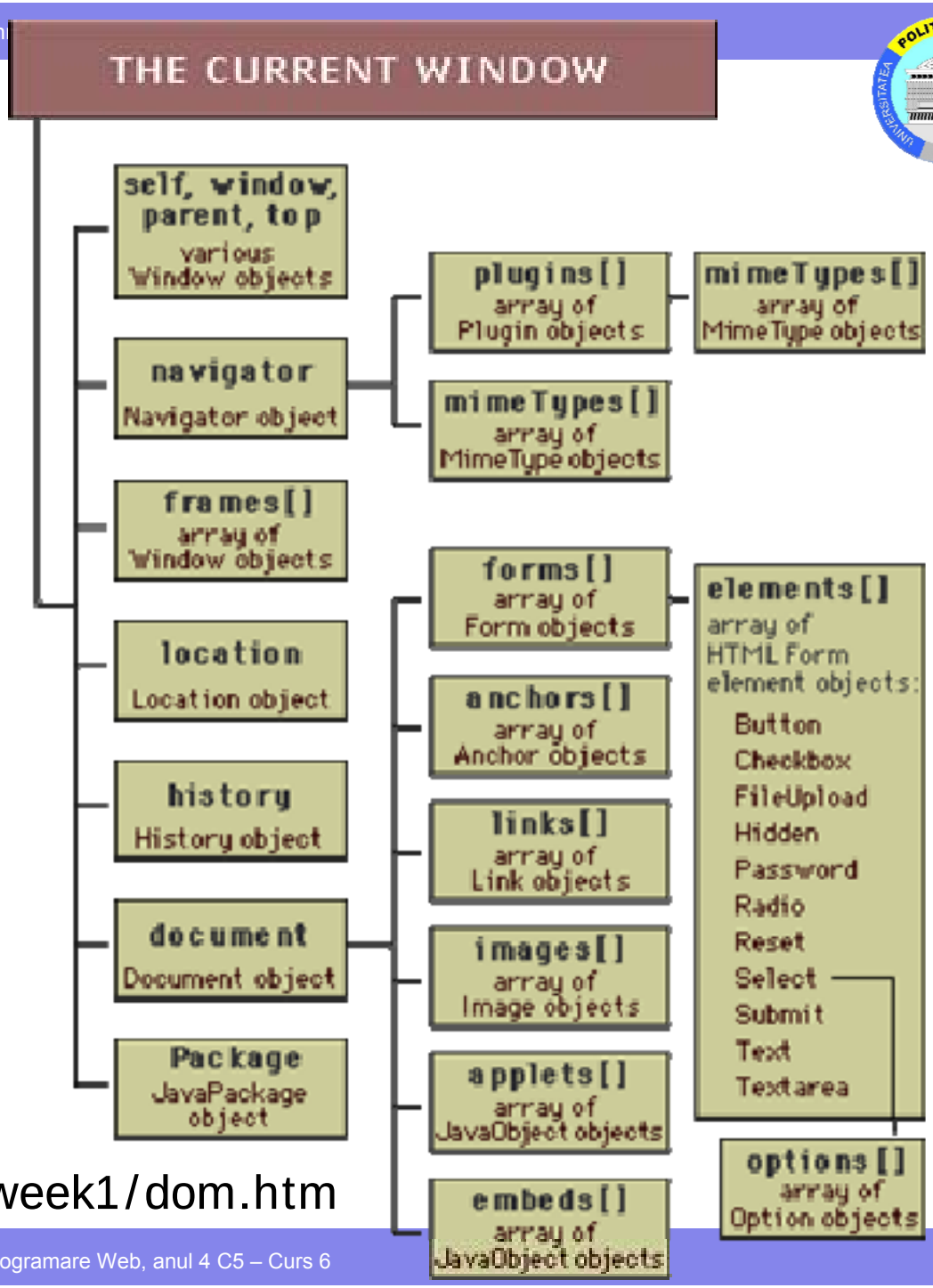
Evenimente si gestionare de evenimente (6)

Event	Applies to	Occurs when	Handler
Focus	Windows and all form elements	User gives element input focus	onFocus
Blur	Windows and all form elements	User moves focus to some other element	onBlur
Reset	Forms	User clicks a Reset button	onReset
Submit	Forms	User clicks a Submit button	onSubmit

Inapoi la DOM

- Puteti atasa handlere de evenimente elementelor HTML avand putine cunostinte legate de DOM
- Totusi, pentru a *schimba* ceea ce este afisat in cadrul paginii necesita cunoasterea modului in care puteti face referire la diversele elemente constituente ale respectivei pagini
- DOM-ul de baza este un standard W3C si este consistent intre diversele browsere existente
  - Unele aspecte mai complexe pot fi totusi dependente de browser
- Elementul de pe nivelul cel mai inalt (in cazul paginii curente) este window, si orice altceva descinde de la acesta
  - Orice variabila JavaScript reprezinta un camp al unui obiect
  - In DOM, toate variabilele se presupune ca pornesc cu “window.”
  - Toate celelalte elemente pot fi regasite mergand in jos pornind de la acest element de nivel cel mai inalt

Ierarhia DOM



Sursa:
http://sislands.com/coin70/week1/dom.htm

Campurile lui window (1)

- window
  - Fereastra curenta (nu se foloseste direct prea des).
- self
  - Acelasi lucru ca si window.
- parent
  - Daca ne referim la un frame, fereastra imediat superioara in care acesta este inclus.
- top
  - Daca ne referim la un frame, fereastra cea mai superioara dpdv ierarhic in care acesta este inclus.
- frames[ ]
  - Un array de frame-uri (daca exista) din cadrul ferestrei curente. Frame-urile sunt si ele la randul lor ferestre.
- length
  - Numarul de frame-uri continute in fereastra curenta.

Campurile lui window (2)

- document
  - Documentul HTML ce este curent afisat in fereastra.
- location
  - URL-ul documentului ce este curent afisat in fereastra. Daca setati aceasta proprietate la un nou URL, respectivul URL va fi incarcat in fereastra curenta. Apeland location.reload() va face refresh la fereastra.
- navigator
  - O referinta la obiectul Navigator (browser). Unele proprietati ale obiectului Navigator sunt:
    - appName – numele browserului, precum "Netscape"
    - platform – numele platformei pe care ruleaza browserul, precum "Win32"
- status
  - Un string ce poate fi citit/scris si care este afisat in zona de status a ferestrei browserului. Poate fi modificat cu o simpla instructiune de atribuire.

Metodele lui window (1)

- alert(string)
  - Afiseaza un dialog de alerta ce contine string-ul dat ca argument si un buton de OK.
- confirm(string)
  - Afiseaza o fereastra de dialog continand string-ul primit ca argument impreuna cu doua butoane de Cancel si de OK. Intoarce true daca a fost apasat butonul OK si false daca a fost apasat butonul Cancel.
- prompt(string)
  - Afiseaza un dialog de confirmare continand string-ul primit ca argument, un camp de tip text si doua butoane de Cancel si OK. Intoarce string-ul introdus de catre utilizator daca a fost apasat OK si null daca a fost apasat butonul Cancel.

Metodele lui window (2)

- open(URL)
  - Deschide o noua fereastra continand documentul specificat de la adresa URL data ca argument.
- close()
  - Inchide o anumita fereastra (care ar trebui sa fie o top-level window, nu un frame).

Campurile lui document (1)

- Trebuie sa prefixati aceste campuri cu document.
- anchors[ ]
  - Un array de obiecte Anchor (obiecte reprezentand tag-uri de forma <a name=…>)
- applets[ ]
  - Un array de obiecte Applet
    - Proprietatile sunt campurile publice definite pentru respectivele appleturi
    - Metodle sunt metodele publice ale appleturilor

Campurile lui document (2)

- forms[ ]
  - Un array de elemente de tip Form
    - Daca documentul contine un singur formular, acesta este intors in forms[0]
- images[ ]
  - Un array de obiecte Image
    - Pentru schimbarea unei imagini asignati un nou URL proprietatii src
- links[ ]
  - Un array de obiecte Link
    - Un link are cateva proprietati, incluzand href, ce contine URL-ul complet al legaturii

Campurile lui document (3)

- bgColor
  - Culoarea de fundal a documentului
    - Poate fi schimbata oricand
- title
  - Un string read-only ce contine titlul documentului
- URL
  - Un string read-only ce contine URL-ul documentului

Campurile obiectului form

- elements[ ]
  - Un array de elemente ale formularului

Exemple JavaScript

Obtinerea datei

- <script type="text/javascript">
  var d = new Date()
  document.write(d.getDate() + "/")
  document.write((d.getMonth() + 1) + "/")
  document.write(d.getFullYear())
  </script>

  - 27/09/2004

Obtinerea si formatarea datei

- <script type="text/javascript">
```
var d=new Date()
var weekday=new Array("Sunday", "Monday", "Tuesday",
      "Wednesday", "Thursday", "Friday","Saturday")
var monthname=new Array("Jan", "Feb", "Mar","Apr",
     "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
document.write(weekday[d.getDay()] + ", ")
document.write(monthname[d.getMonth()] + " " +
                          d.getDate() + ", ")
document.write(d.getFullYear())
```
</script>

- Monday, Sep 27, 2004

Obtinerea unui numar aleator

- Urmatorul cod genereaza un numar aleator in virgula mobila cuprins intre 0 si 1:
- <script type="text/javascript">
  document.write(Math.random())
  </script>

- 0.728762788388911

Obtinerea unei valori intregi aleatoare

- Urmatorul cod genereaza un numar aleator intreg cuprins intre 0 si 10:
- <script type="text/javascript">

```
var max = 10;
number=Math.random()*max + 1;
document.write(Math.floor(number));
```

</script>

- 5

Afisarea unui mesaj de alerta

- Urmatorul cod afiseaza un mesaj de alerta atunci cand un buton este apasat:
- <form> // Buttons can only occur within forms

```
    <input type="button" name="Submit"
           value="Alert!“
           onclick="alert('Oh oh, something
     happened!');">
</form>
```

Depanare

- Urmatorul cod arata ce evenimente au fost declansate atunci cand utilizatorul a intreprins diverse actiuni
- In sectiunea <head> a paginii HTML definim:
  - <script>

```
<!--
function tell(a, b) {
    document.forms[0].result.value+="\n"+a+": " + b;
}
//-->
</script>
```

- Pentru fiecare element al formularului adaugam un handler pentru oricare eveniment (plauzibil)

Depanarea (pentru un Buton)

- <input type="button" name="plainButton" value="Plain Button"
  onMouseDown="tell(this.name, 'onmousedown');"
  onMouseUp="tell(this.name, 'onmouseup');"
  onClick="tell(this.name,'onclick');"
  onDblClick="tell(this.name,'ondblclick');"
  onFocus="tell(this.name, 'onfocus');"
  onBlur="tell(this.name, 'onblur');"
  onMouseOver="tell(this.name, 'onmouseover');"
  onMouseOut="tell(this.name, 'onmouseout');"
  onChange="tell(this.name, 'onchange');"
  onKeyPress="tell(this.name, 'onkeypress');"
  onKeyDown="tell(this.name, 'onkeydown');"
  onKeyUp="tell(this.name, 'onkeyup');"
  onSelect="tell(this.name, 'onselect');"
  onReset="tell(this.name, 'onreset');"
  >

JavaScript - advanced

Suportul browserului

- JavaScript functioneaza pe aproape toate browserele
- Internet Explorer foloseste JScript (referit in meniuri ca “Active Scripting”), ce reprezinta dialectul Microsoft-ului de JavaScript ☺
- Browserele mai vechi nu suporta unele constructii JavaScript
  - Vom presupune suportul unui browser modern
- Activarea si dezactivarea JavaScript:
  - Daca nu stiti cum sa faceti asta din browser, vedeti si http://www.mistered.us/tips/javascript/browsers.shtml

Ce NU puteti face

- Pentru protectia vizitatorilor la paginile voastre, folosind JavaScript nu puteti:
  - Executa alte programe
  - Sa va conectati la alte computere, exceptand download-ul altor pagini HTML sau trimiterea de e-mail
  - Sa determinati ce alte site-uri a vizitat utilizatorul
  - Citi sau scrie fisiere
    - Totusi, JScript in IE permite scripting ASP, modalitatea prin care viermele foarte distructiv JS.Gigger.A@mm s-a raspundit de exemplu
    - Implicit, Outlook Express permite ca mail-ul primit sa ruleze scripturi
    - Pentru a dezactiva scripting-ul in Outlook Express, vedeti http://support.microsoft.com/support/kb/articles/Q192/8/46.ASP

Numere

- In JavaScript, toate numerele sunt de tip floating point
- Numere speciale predefinite:
  - Infinity, Number.POSITIVE_INFINITY – rezultatul impartirii unui numar pozitiv la zero
  - Number.NEGATIVE_INFINITY – rezultatul impartirii unui numar negativ la zero
  - NaN, Number.NaN (Not a Number) – rezultatul impartirii 0/0
    - NaN este diferit de orice, chiar si de sine insusi
    - Exista o functie globala isNaN()
  - Number.MAX_VALUE – cel mai mare numar reprezentabil
  - Number.MIN_VALUE – cel mai mic (apropiat de zero) numar reprezentabil

String-uri si caractere

- In JavaScript, *string* este un tip primitiv
- Stringurile sunt incadrate de ghilimele sau apostroafe
- Nu exista tipul “character”
- Caractere speciale:

\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*

Cateva metode de lucru cu string-uri

- charAt(***n***)
  - Intoarce al ***n***-lea caracter din string
- concat(***string1***, …, ***stringN***)
  - Concateneaza string-urile primite ca argumente
- indexOf(***substring***)
  - Intoarce pozitia primului caracter al ***substring*** in string-ul recipient sau -1 daca nu este gasit
- indexOf(***substring***, ***start***)
  - Intoarce pozitia primului caracter al ***substring*** in stringul dat ca argument incepand de la pozitia ***start***, sau -1 daca nu este gasit
- lastIndexOf(***substring***), lastIndexOf(***substring***, ***start***)
  - Similar indexOf, dar cauta string-ul incepand de la sfarsit spre inceput

Variabile

- Orice variabila este o proprietate a unui obiect
- Atunci cand JavaScript porneste, creaza un *obiect global* – obiectul *window*
  - Poate fi referit ca window sau ca this
- Pot exista mai mult de un singur obiect “global”
  - De exemplu, un frame poate referi un alt frame printr-un cod precum parent.frames[1]
- Elementele HTML form pot fi referite prin document.forms[***formNumber***].elements[***elementNumber***]
- Orice element HTML form are un atribut name
  - Numele poate fi folosit in locul refererintei de tip array
  - De exemplu, da
    - <form name="myForm">
      <input type="button" name="myButton" ...>
    - Atunci in loc de document.forms[0].elements[0]
    - Putem spune document.myForm.myButton

Functii

- In Java, metodele sunt *asociate cu* obiecte
- In JavaScript, o functie *este* un obiect
- Functiile pot fi recursive:
  - ```
    function factorial(n) {
        if (n <= 1) return 1;
        else return n * factorial(n - 1);
    }
    ```
- Functiile pot fi imbricate:
  - ```
    function hypotenuse(a, b) {
        function square(x) { return x * x; }
        return Math.sqrt(square(a) + square(b));
    }
    ```

Constructorul Function()

- Deoarece functiile sunt obiecte, ele au si un constructor:
  - Function(***arg1***, ***arg2***, …, ***argN***, ***body***)
  - Toate argumentele *constructorului* sunt string-uri
  - Exemplu:

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

- Functia nu are nici un nume
  - Dar se poate asigna unei variabile si folosi respectivul nume
  - Numele poate fi folosit pentru apelarea functiei in modul uzual
- Se pot construi functii in mod *dinamic* in JavaScript (ele sunt automat compilate)
  - Totusi, compilarea este o activitate computational-intensiva
- Functiile definite in acest fel sunt intotdeauna globale

Functii

- O functie poate fi definita prin folosirea unui constructor:
  - var f = new Function("x", "y", "return x * y;");
- O functie poate fi scrisa sub forma *literala,* precum in urmatorul exemplu:
  - var f = function(x, y) { return x * y; }
  - Aceasta functie nu este in mod necesar globala
- Pentru a scrie o functie recursiva, se poate proceda altfel:
  - var f = function fact(n) { if (n <= 1) return n;
    else return n * fact(n - 1) ; };
  - Numele nu persista dupa ce functia este creata

Nume de functii

- "Numele" unei functii este o variabila ce contine respectiva functie
  - var square = function(x) { return x * x; };
  - var a = square(4); // a now holds 16
  - var b = square; // b now holds square
  - var c = b(5); // c now holds 25
  - var d = [b]; // d is an array
  - var e = d[0](6); // e now holds 36

Proprietati ale functiilor

- Deoarece o functie este un obiect ii puteti adauga proprietati
  - Proprietatile functiilor sunt adesea o buna alternativa la variabilele globale
  - Exemplu:

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

  - Proprietatile functiilor sunt cumva similare variabilelor statice din Java

Variabile locale si globale

- O variabila este *locala* unei functii daca
  - Este un parametru formal al functiei
  - Este declarata cu var in interiorul unei functii (e.g. var x = 5)
- Altfel, variabilele sunt *globale*
- Mai exact, o variabila este globala daca
  - Este declarata in afara oricarei functii (cu sau fara var)
  - Este declarata prin *asignare* in interiorul unei functii (e.g. x = 5)

Functii si metode

- Atunci cand o functie reprezinta o proprietate a unui obiect, o numim “metoda”
  - O metoda poate fi invocata fie
    call(***object***, ***arg1***, …, ***argN***) fie
    apply(*object*, [***arg1***, …, ***argN***])
  - call si apply sunt definite pentru toate functiile
    - call primeste orice numar de argumente
    - apply primeste un array de argumente
  - Ambele permit invocarea functiei *ca si cum* ar fi o metoda a unui alt obiect, ***object***
  - In interiorul unei functii cuvantul cheie this se refera la ***object***

Metode (2)

- Putem apoi transforma functia intr-o metoda:
  - myPoint.dist = distance;
- this in interiorul unei functii se refera la myPoint, deci putem spune:
  - document.write("The distance is " + myPoint.dist(6, 9));
- Daca nu dorim sa asociem permanent functia cu myPoint, putem spune:
  - document.write("The distance is " + distance.call(myPoint, 6, 9));
- Sau:
  - document.write("The distance is " + distance.apply(myPoint, [6, 9]));

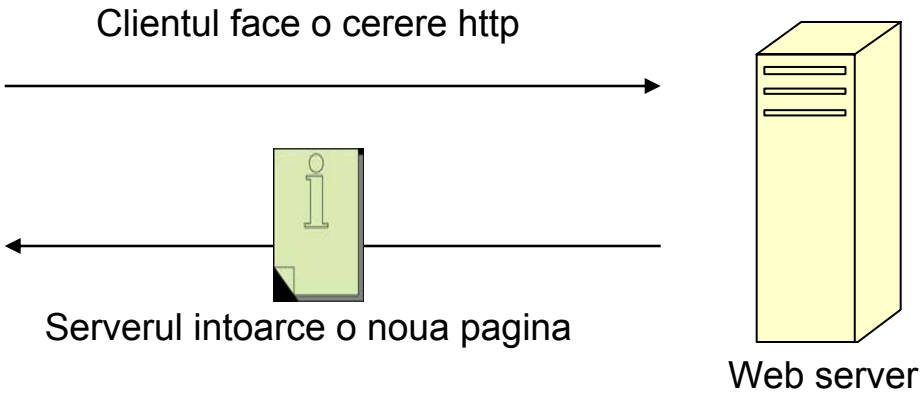
JavaScript si AJAX

Ce este AJAX?

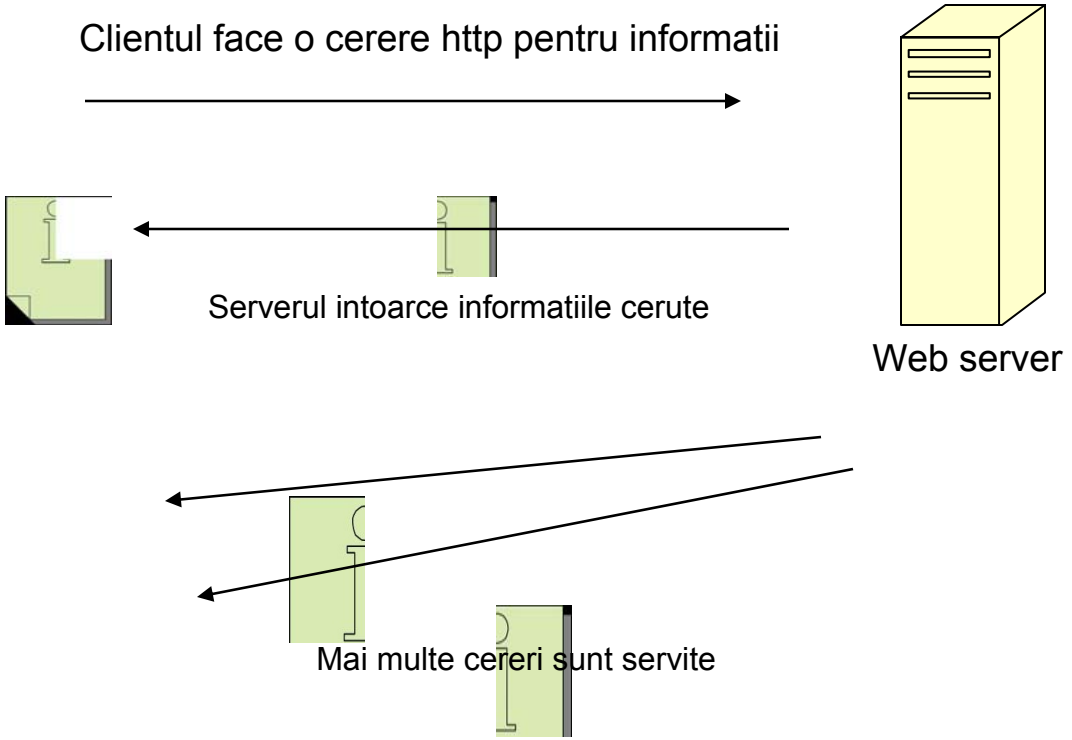
- AJAX = Asynchronous JavaScript and XML
- Permite actualizarea incrementala a paginilor Web.
- Construita folosind tehnologii standard web - HTTP, (X)HTML, CSS, JavaScript, Document Object Model (DOM), XML
- Folosit de multe companii populare astazi
- Exemple:
  - Google Suggests, Google & Yahoo! Maps
  - Amazon A9 Search
  - Flickr, BaseCamp, Kayak
  - Yahoo! AJAX Library

Exemplu – Yahoo! Sports

Interactiunea Web Traditionala



Cum functioneaza AJAX



De ce ne intereseaza AJAX?

- Permite construirea de aplicatii Rich Internet Applications (RIA)
- Permite interactiunea dinamica pe Web
- Imbunatateste performantele
- Actualizari real-time
- Nu necesita plug-in-uri

Cod Simplu pentru Schimbarea Stilului

```
<style type="text/css">
.notice {
  background-color:#FFFFCC;
}
.roInput {
   border:none;
  background-color:#FFFFFF;
}
</style>
```

```
<script type="text/javascript">
// get the element to hold updated data
var priceLoc = document.getElementById('priceLoc");
// update the data in price loc
priceLoc.value = "new data";
// set the style so change will be noticed
priceLoc.className = "notice";
// create timer to call clearActive() with element id and style name
setTimeout("clearActive('priceLoc','roInput');", 5000);

function clearActive(activeId, resetStyle) {
  var curActive = document.getElementById(activeId);
  curActive.className = resetStyle;
}
</script>
```