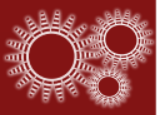
Tehnologii Internet

CURSUL 07 – INTERACȚIUNEA ÎN PAGINILE WEB

Universitatea Tehnică ”Gheorghe Asachi” din Iași
Facultatea de Automatică și Calculatoare
Departamentul de Calculatoare
Specializarea Tehnologia informației

Cuprins

1. Limbajul JavaScript
2. Document Object Model (DOM)
3. Browser Object Model (BOM)
4. Biblioteci și framework-uri JavaScript



1. Limbajul JavaScript

1.1. Introducere

1.2. Istoric

1.3. Sintaxa JavaScript

1.4. JavaScript în HTML

1.1. Limbajul JavaScript - introducere

JavaScript

- Client-side scripting
- Descrie comportamentul în cadrul paginilor web
- Interacțiunea cu utilizatorul
- Controlul elementelor din pagină în urma acțiunii utilizatorului și nu numai
- Comunicare asincronă cu serverul
- Modificarea dinamică a modului în care este afișat conținutul în web browser
- Manipularea elementelor HTML și a stilurilor

1.1. Limbajul JavaScript - introducere

- Exemplu de cod JavaScript

```
var currentHash = "no hash";
function checkForHashChange() {
	console.log("hash: " +
		window.location.hash.substring(1));
	if (window.location.hash.substring(1) !=
	currentHash) {
		preNavigateTo();
	}
}
```

1.2. Limbajul JavaScript - istoric

- 1995 – dezvoltat de Brendan Eich de la Netscape Communications Corporation
- 1995 – numele inițial al limbajului a fost Mocha, apoi a fost schimbat în LiveScript și în final, JavaScript
- 1996-1997 – specificațiile limbajului au fost făcute de European Computer Manufacturers Association (ECMA) => ECMAScript – standardul oficial
- 1998 – ECMAScript 2; 1999 – ECMAScript 3
- 2005 – Brendan Eich, ECMA și Macromedia încep să lucreze la ECMAScript 4, dar, după împotriviri din partea Microsoft => ECMAScript3.1
- 2005 – Jesse James Garrett – folosește termenul Ajax
- 2009 – ECMAScript 3.1 este redenumit în ECMAScript 5
- 2011 – ECMAScript 5.1

1.3. Sintaxa JavaScript

Comentarii

- Sunt ca în C/C++/Java
- Pentru a comenta o singură linie de cod: //
- Pentru a comenta mai multe linii de cod: /* */

1.3. Sintaxa JavaScript

Variabile

- Se definesc cu ajutorul cuvântului cheie **var**

```
var x = 10;
```

- Redeclararea unei variabile fără asignarea unei noi valori nu va duce la resetarea valorii
- O variabilă poate fi folosită înainte de a fi declarată
- Reguli de denumire a variabilelor:
  - Numele poate să înceapă cu o literă, $ sau _
  - Numele pot conține litere, cifre, $ și _
  - Numele sunt case-sensitive
  - Cuvintele rezervate nu pot fi folosite pentru a denumi o variabilă

1.3. Sintaxa JavaScript

Tipuri de dată

- Nu se specifică ce tip de dată are o variabilă
- Tipul este determinat în mod dinamic
- Aceeași variabilă poate fi număr, șir de caractere sau boolean
- Tipurile de dată: Number, String, Boolean, Null, Undefined, Symbol (ECMAScript 6), Object
- Toate valorile (exceptând de tipul Object) sunt imutabile
- Array este un obiect folosit pentru a crea vectori de elemente

1.3. Sintaxa JavaScript

Tipuri de dată

```
var x;                          // undefined
var x = null;                   // null
var x = 13.5;                   // Number
var x = "Ana";                  // String
var x = 'Ana are "mere"';       // String
var x = true;                   // Boolean
var x = ['Ana', 15, 3.5];       // Array (Object)
var x = {a: 10, b: 15};         // Object
var x = suma;                   // Function
```

1.3. Sintaxa JavaScript

Expresii și operatori

- this – contextul de execuție a funcției
- function – definește o funcție
- [] – definirea unui array
- {} – definirea unui obiect
- /ab+c/i – expresii regulate

1.3. Sintaxa JavaScript

Operatori - *între paranteze este pusă precedența*

- Operatori de grupare (0)
  - ( ) – controlarea precedenței
- Operatori de acces (1)
  - . [“”] – accesarea proprietăților
  - new – crearea unei instanțe a unui constructor (cu listă argumente)
- Operatori de funcții (2)
  - ( ) – apelul unei funcții
  - new – crearea unei instanțe a unui constructor (fără listă de argumente)

1.3. Sintaxa JavaScript

Operatori

- Operatori postfixați (3)
  - a++, a-- – incrementare și decrementare
- Operatori prefixați (4)
  - ++a, --a – incrementare și decrementare
- Operatori unari (*operator expresie*) (4)
  - delete – șterge proprietatea unui obiect
  - void – evaluează expresia și returnează undefined
  - typeof – determină tipul unui obiect
  - +, -, ~, !

1.3. Sintaxa JavaScript

Operatori

- Operatori aritmetici (5 și 6)
  - *, /, % +, -
- Operatori binari de shift-are (7)
  - <<, >>, >>>
- Operatori relaționali (8)
  - in – (*proprietate* in *obiect*) – returnează true dacă proprietatea specificată se găsește în obiectul respectiv
  - instanceof – (*obiect* instanceof *constructor*) – testează dacă un obiect are în lanțul de prototipuri proprietatea prototype a unui constructor
  - <, >, <=, >=

1.3. Sintaxa JavaScript

Operatori

- Operatori de egalitate (9)
  - ==, != – convertește operanzii dacă nu sunt de același tip și apoi aplică comparația strictă
  - ===, !== – compară operanzii fără conversia de tip
- Operatori binari (10, 11, 12)
  - &, ^, |
- Operatori logici (13, 14)
  - &&, ||

1.3. Sintaxa JavaScript

Operatori

- Operatorul condițional (ternar) (15)
  - (*condiție*)?*dacă_este_adevărată*:*dacă_este_falsă*
- Operatori de atribuire (16)
  - =, *=, /=, %=, +=, -=, <<=, >>=, >>>=, &=, ^=, |=
- Operatorul virgulă (19)
  - , - permite evaluarea mai multor expresii și returnează rezultatul dat de ultima expresie

1.3. Sintaxa JavaScript

Operatori

- Exemplu

```
// definirea constructorilor
function A(){}
function B(){}

B.prototype = new A(); // moștenirea
var o = new B();
o instanceof A; // true
o instanceof B; // true
```

1.3. Sintaxa JavaScript

Șiruri de caractere

- Proprietăți:
  - constructor, length, prototype
- Metode:
  - charAt, charCodeAt, fromCharCode,
  - indexOf, lastIndexOf
  - concat, slice, split, substr, substring, trim
  - match, replace, search
  - toLocaleLowerCase, toLocaleUpperCase, toLowerCase, toUpperCase, toString, valueOf

1.3. Sintaxa JavaScript

Numere

- Sunt valori pe 64 de biți cu virgulă mobilă
- 1 bit semn, 11 biți exponent, 52 biți mantisă
- Numerele fără virgulă sunt considerate precise dacă au maxim 15 cifre

```
var x = 13;          // număr fără zecimale
var x = 13.00;       // Număr cu zecimale
var x = 123e5;       // 12300000
var x = 123e-5;      // 0.00123
var x = 0xF1;        // număr în baza 16
```

1.3. Sintaxa JavaScript

Numere – valori speciale

- Infinity

```
var x =  1 / 0;    // x va fi Infinity
var y = -1 / 0;    // y va fi -Infinity
```

- NaN (Not a Number)

```
var x =  1 / "Ana";     // x va fi NaN
var y = 100 / "10";    // y va fi 10
```

1.3. Sintaxa JavaScript

Numere

- Proprietăți care pot fi accesate doar prin obiectul Number:
  - MAX_VALUE, MIN_VALUE, POSITIVE_INFINITY, NEGATIVE_INFINITY
- Metode globale:
  - Number, parseFloat, parseInt
- Metode aplicate asupra numerelor:
  - toString, toExponential, toFixed, toPrecision, valueOf
- Funcția isNaN

1.3. Sintaxa JavaScript

Funcții

```
function numeFuncție(argumente) {
      // codul functiei
}
```

Exemplu:

```
var x = suma(5, 7);

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

1.3. Sintaxa JavaScript

Obiecte

```
var carteaMea = {
      titlu: "The shinning",
      autor: "Stephen King",
      anAparitie: 1977
}
```

- Accesarea proprietăților unui obiect:
  - `carteaMea.titlu`
  - `carteaMea["titlu"]`
- Metoda `hasOwnProperty("titlu")`

1.3. Sintaxa JavaScript

Obiectul Math

- Metode:
  - abs, ceil, floor, round
  - exp, log, pow, sqrt
  - sin, cos, tan, asin, acos, atan, atan2
  - min, max – numărul cu valoarea cea mai mică/mare dintr-o listă de valori
  - random – număr aleator între 0 și 1

1.3. Sintaxa JavaScript

Obiectul Date

- Crearea unui obiect de tipul Date:

```
new Date()
new Date(milliseconds)
new Date(dateString)
new Date(year, month, day, hours, minutes,
      seconds, milliseconds)
```

- Metode:
  - getDate, getDay, getFullYear, getHours, getMiliseconds, getMinutes, getMonth, getSeconds, getTime (similar pentru set)
  - Date.parse() – numărul de milisecunde
- Datele se pot compara folosind: ==, !=, <, >, <=, >=

1.3. Sintaxa JavaScript

Obiectul Array

- Exemplu

```
var x = ['Ana', 15, 3.5];
x[0] = 3;
x[x.length] = 15;
```

- Nu se recomandă crearea vectorilor folosind constructorul Array

```
new Array('Ana', 15, 3.5)
```

1.3. Sintaxa JavaScript

Obiectul Array

- Sparse Array

```
var x = [];
x[0] = 13;
x[3] = 14;
x.length // 4
x[1]     // undefined
```

- Associative Array
  - JavaScript NU suportă array-uri asociative (în care indexul din vector este șir de caractere)

1.3. Sintaxa JavaScript

Obiectul Array

- Proprietăți:
  - constructor, length, prototype
- Metode:
  - push, pop, splice, shift, unshift
  - concat, reverse, slice, sort
  - indexOf, lastIndexOf
  - join, toString, valueOf

1.3. Sintaxa JavaScript

Obiectul RegExp

- Expresie regulată = secvență de caractere care formează un pattern de căutare
- Sintaxa: */pattern/modificatori*
- Exemple:
  - `/internet/i`
  - `/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.(?:[A-Z]{2}|com|org|net|edu|gov|mil|biz|info|mobi|name|aero|asia|jobs|museum)$/g`

1.3. Sintaxa JavaScript

Obiectul RegExp

- Metode aplicate unui obiect de tipul String care folosesc și expresii regulate
  - search, replace
  - `str.replace(/internet/i, "TI")`
- Metode ale obiectului RegExp
  - test – caută pattern-ul într-un șir de caractere și returnează true/false
  - exec – caută pattern-ul într-un șir de caractere și returnează textul găsit sau null

1.3. Sintaxa JavaScript

- Instrucțiunile (cuvintele cheie) următoare se comportă la fel ca în limbajele C/C++
  - if, else, switch,
  - break, continue
  - for, while, do... while
- Se pot defini etichete (*etichetă*: *instrucțiuni*) la fel ca în C/C++/Java, dar se folosesc ca în Java
  - break *etichetă*;
  - continue *etichetă*;

1.3. Sintaxa JavaScript

- Instrucțiunea for... in

```
var obj = {a:1, b:2, c:3};
for (var prop in obj) {
     console.log("o." + prop + " = " +
                 obj[prop]);
}
// Output:
// "o.a = 1"
// "o.b = 2"
// "o.c = 3"
```

1.3. Sintaxa JavaScript

Tratarea erorilor

- se face la fel ca în Java cu mențiunea că excepția generată poate fi de tipul String, Number, Boolean sau Object
- Cuvinte cheie: try, catch, finally, throw

1.3. Sintaxa JavaScript

Recomandări privind scrierea codului

- Similare cu limbajele C/C++/Java

Alte recomandări

- Variabilele să fie declarate înainte de a fi folosite
- Trebuie evitată folosirea variabilelor globale
- Variabilele locale dintr-o funcție se declară cu **var**
- Numerele, șirurile de caractere și valorile de tipul boolean trebuie tratate ca primitive și nu ca obiecte
- Dacă la apelul unei funcții se omite un argument atunci respectivul argument are valoarea **undefined**

1.3. Sintaxa JavaScript

```
var x1 = {};                // new object
var x2 = "";                // new primitive string
var x3 = 0;                 // new primitive number
var x4 = false;             // new primitive boolean
var x5 = [];                // new array object
var x6 = /()/;              // new regexp object
var x7 = function(){};      // new function object
```

1.3. Sintaxa JavaScript

Domeniul de vizibilitate/accesibilitate

- Este similar cu limbajulele C/C++/Java
- O variabilă declarată într-un bloc poate fi folosită doar în cadrul blocului respectiv
- *O variabilă care NU a fost declarată (folosind var) devine automat variabilă globală*

```
function f() {
        nume = "Ana";
}
```

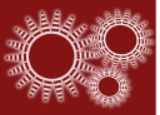
1.4. JavaScript în HTML

```
<!DOCTYPE html>
<html>
<head>
<script>
    function schimbaContinut() {
        document.getElementById("myDiv").innerHTML =
                    "Noul continut";
    }
</script>
</head>
<body>
    <a onclick="schimbaContinut()">Schimba</a>
    <div id="myDiv">Ana are mere</div>
</body>
</html>
```

1.4. JavaScript în HTML

- Codul JavaScript se inserează:
  - Ca și conținut a tag-ului HTML <script>
  - În fișiere externe cu extensia .js
- Tag-ul <script> poate să apară fie în <head>, fie în <body>
- De obicei, codul JavaScript se execută ca urmare a interacțiunii utilizatorului
- Pentru a referi un fișier .js din HTML se folosește atributul *src* a tag-ului <script>

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



2. Document Object Model

2.1. Introducere

2.2. Obiecte DOM

2.3. Evenimente HTML

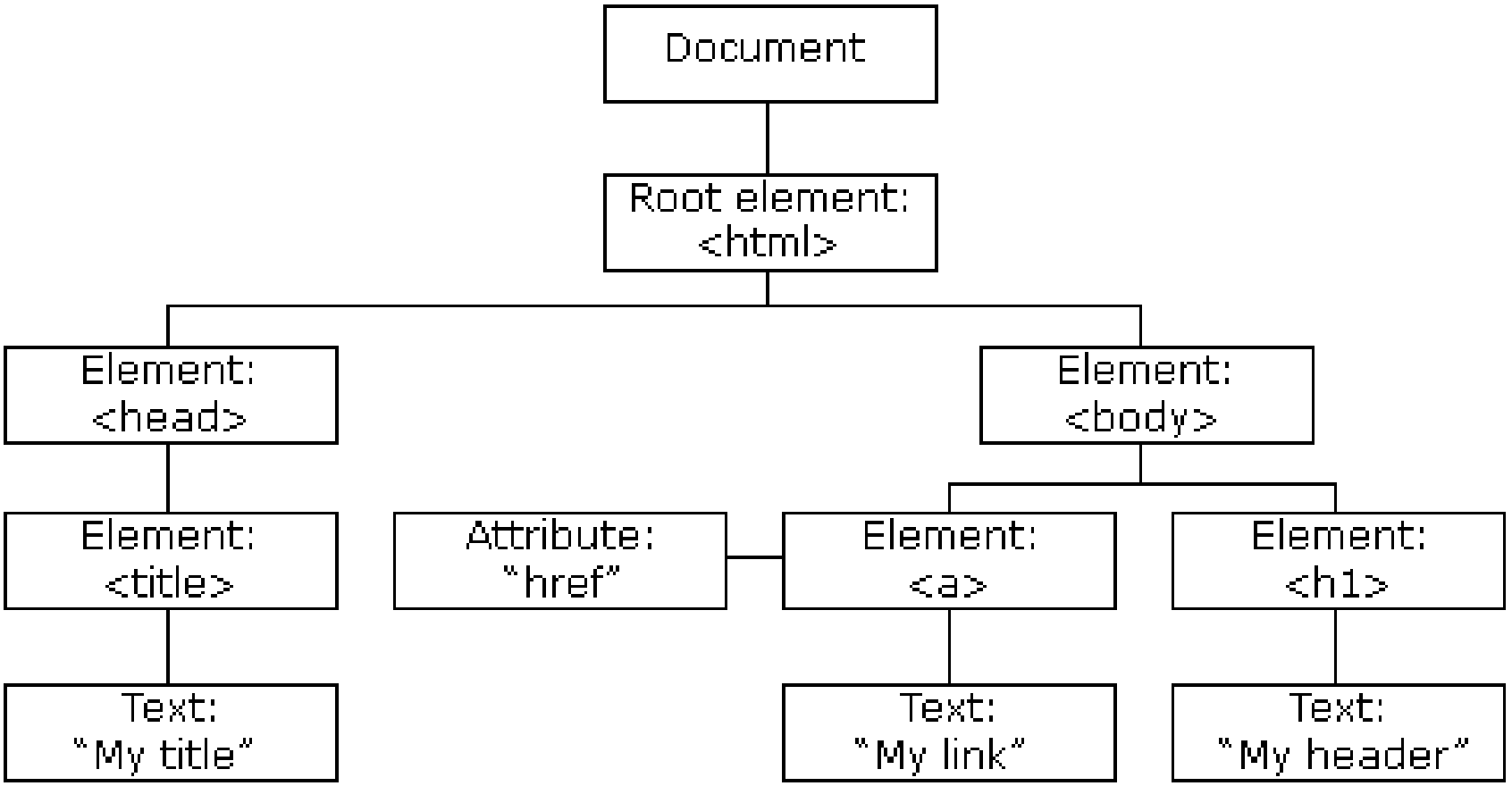
2.4. Noduri DOM

2.1. DOM - introducere

Document Object Model (DOM)

- DOM = Platformă și interfață care permite scripturilor să acceseze în mod dinamic și să schimbe conținutul, structura și stilul unui document
- Model creat când pagina web s-a încărcat
- Prin intermediul DOM, limbajul JavaScript poate accesa, schimba, adăuga, șterge elementele, atributele și stilurile CSS ale unui document HTML

2.1. DOM - introducere



Sursa: http://www.w3schools.com/js/js_htmldom.asp

2.2. Obiecte DOM

Obiectul document

Metodă	Descriere
document.getElementById()	Găsește un element după id
document.getElementsByTagName()	Găsește elemente după tag
document.getElementsByClassName()	Găsește elemente după clasă
document.createElement()	Creează un element HTML
document.removeChild()	Șterge un element HTML
document.appendChild()	Adaugă un element HTML
document.replaceChild()	Înlocuiește un element HTML
document.write(*text*)	Scrie la ieșirea HTML

2.2. Obiecte DOM

Modificarea elementelor HTML

Metodă	Descriere
element.innerHTML=	Înlocuiește conținutul unui element
element.attribute=	Schimbă atributul unui element
element.setAttribute*(attribute,value)*	Schimbă atributul unui element
element.style.*property*=	Schimbă stilul unui element

2.3. Evenimente HTML

- O secvență JavaScript poate fi executată atunci când apare un eveniment (e.g., utilizatorul apasă pe un element HTML)
- Atribute ale elementelor HTML:
  - onclick, onmouseenter, onmouseleave, onmousemove, onmouseover, ...
  - onkeydown, onkeypress, onkeyup
  - onload, onchange, ...
  - ...
- Mai multe detalii:
  - http://www.w3schools.com/jsref/dom_obj_event.asp

2.3. Evenimente HTML

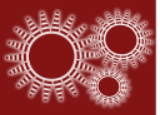
Event Listener

- *element*.addEventListener(*event, function, useCapture*);

```
var p1 = 5, p2 = 7;
document.getElementById("myBtn").
  addEventListener("click", function() {
    myFunction(p1, p2);
});
function myFunction(a, b) {
    var result = a * b;
    document.getElementById("demo").
       innerHTML = result;
}
```

2.4. Noduri DOM

- Nod = tot dintr-un document HTML (întreg documentul, elemente, atribute, conținutul (textul) elementelor, comentariile)
- Proprietăți folosite pentru a naviga printre noduri:
  - parentNode
  - childNodes[*index*]
  - firstChild
  - lastChild
  - nextSibling
  - previousSibling



3. Browser Object Model

3.1. Introducere

3.2. Obiecte DOM

3.1. BOM - introducere

Browser Object Model (BOM)

- Prin intermediul BOM, limbajul JavaScript poate “comunica” cu browser-ul
- Nu există standarde oficiale pentru BOM

3.2. Obiecte BOM

Obiectul window

- Reprezintă fereastra browser-ului
- Toate obiectele, funcțiile și variabilele globale devin automat membrii ale obiectului window (inclusiv obiectul document)
- Proprietăți:
  - innerWidth, innerHeight
- Metode:
  - open, close, moveTo, resizeTo

3.2. Obiecte BOM

Obiectul screen

- Reprezintă ecranul utilizatorului
- Proprietăți:
  - width, height, availWidth, availHeight,
  - colorDepth, pixelDepth

3.2. Obiecte BOM

Obiectul location

- Reprezintă adresa din browser-ul utilizatorului
- Proprietăți:
  - location.href – URL-ul paginii curente
  - location.hostname – numele domeniului
  - location.pathname – calea și numele fișierului
  - location.protocol – protocolul folosit (http:// sau https://)
- Metode:
  - location.assign – încarcă un nou document

3.2. Obiecte BOM

Obiectul history

- Reprezintă istoricul browser-ului
- Metode:
  - back, forward

Obiectul navigator

- Conține informații despre browser-ul utilizatorului
- Proprietăți:
  - appName, appCodeName, appVersion, ...
  - cookieEnabled, geolocation, language, userAgent

3.2. Obiecte BOM

Ferestre pop-up

- alert
- confirm
- prompt

Evenimente de timing

- setInterval(*funcție, x*) – execută o funcție la fiecare x milisecunde (returnează un obiect interval)
- setTimeout(*funcție, x*) - execută o funcție o singură dată după un număr x de milisecunde
- clearInterval(*obiectInterval*) – oprește execuțiile ulterioare

3.2. Obiecte BOM

Obiecte document.cookie

- Cookies = date stocate în fișiere text la client
- Perechi: cheie=valoare;
- Exemplu:

```
document.cookie="username=John Doe;
expires=Thu, 18 Dec 2013 12:00:00 UTC";
```

4. Biblioteci și framework-uri JavaScript

Biblioteci

- jQuery
- MooTools
- YUI
- Dojo

Framework-uri

- AngularJS
- Backbone.JS
- Ember.js
- Knockout
- Agility.js
- CanJS
- Yahoo! Mojito

Bibliografie

- https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details_of_the_Object_Model
- http://www.regular-expressions.info/tutorial.html
- http://www.srirangan.net/2011-12-functional-programming-in-javascript
- http://javascript.info/tutorial/inheritance
- http://www.w3schools.com/jsref/dom_obj_event.asp
- http://www.airpair.com/js/javascript-framework-comparison
- http://jquery.com/
- http://www.w3schools.com/jquery/
- https://www.eclipsecon.org/na2014/sites/default/files/slides/Top%2010%20JavaScript%20Frameworks%20FINAL.pdf