Tehnologii Internet

CURSUL 08 – INTERACȚIUNEA ÎN PAGINILE WEB (2)

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

Cuprins

1. Paradigmele limbajului JavaScript
2. AJAX
3. jQuery
4. AngularJS

1. Paradigmele limbajului JavaScript

- Limbaj de scripting
- Sintaxa JavaScript s-a dezvoltat pornind de la limbajul C
- JavaScript nu are legătură cu limbajul Java
- Are la bază standardul ECMAScript și este asemănător cu limbajul ActionScript

1. Paradigmele limbajului JavaScript

Limbaj de scripting

- Limbaj de programare care utilizează script-uri

Script

- Program care este executat într-un mediu care interpretează și execută instrucțiunile/task-urile

Limbaj dinamic

- În timpul execuției se pot adăuga funcționalități pe care limbajele statice le adaugă la compilare; e.g, se pot extinde obiecte și definiții

1. Paradigmele limbajului JavaScript

Funcție first-class

- Limbajul permite transmiterea funcțiilor ca argumente ale altor funcții, permite ca o variabilă să aibă ca valoare o funcție. Funcțiile sunt variabile de tipul function

Programare orientată-obiect

- Folosește obiecte care au atribute și metode
- Moștenirea este realizată prin prototipuri

Programare imperativă

- Instrucțiunile schimbă starea programului

1. Paradigmele limbajului JavaScript

Programare funcțională

- Execuția unui program presupune evaluarea unor funcții (matematice) și evitarea schimbării stării și a datelor mutabile
- Valoarea returnată de o funcție depinde doar de argumentele de intrare ale funcției
- Apelarea de mai multe ori a unei funcții cu aceleași argumente va produce același rezultat de fiecare dată

1. Paradigmele limbajului JavaScript

Programare funcțională

```
var f = function() {
  console.log("Functii exprimate ca obiecte");
};

(function(nume) {
  console.log( "Functiile "+nume +" se pot
               auto-executa la definire");
})("Anonime");
```

1. Paradigmele limbajului JavaScript

Programare bazată pe prototip

- Programare orientată obiect în care nu există clase, iar moștenirea se realizează prin clonarea obiectelor existente, acestea având rolul de prototip

1. Paradigmele limbajului JavaScript

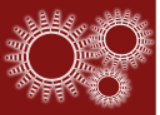
Programare bazată pe prototip

```
function Employee(name, dept) {
   this.name = name || "";
   this.dept = dept || "general";
}
function WorkerBee(projs) {
   this.projects = projs || [];
}
WorkerBee.prototype = new Employee;

function Engineer(name, projs, mach) {
   this.base = WorkerBee;
   this.base(name, "engineering", projs);
   this.machine = mach || "";
}
Engineer.prototype = new WorkerBee;
Employee.prototype.specialty = "none";

var jane = new Engineer("Doe, Jane", ["navigator",
"javascript"], "belau");
```

Sursa: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details_of_the_Object_Model



2. AJAX

2.1. Introducere

2.2. Elementele AJAX

2.3. Comunicarea cu serverul

2.4. Same-origin policy

2.1. AJAX - introducere

- **AJAX** - **A**synchronous **J**ava**S**cript and **X**ML
- Modalitate prin care se transmit și se primesc date la și de la server, și prin care se actualizează părți ale unei pagini web fără a reîncărca toată pagina
- Exemplu de utilizare: single-page application (SPA)
- Permite paginilor web să se actualizeze asincron
- Comunicare asincronă = procesare care permite altor procese/thread-uri să-și continue execuția înainte ca transmisia datelor să se termine

2.2. Elementele AJAX

- Obiectul XMLHttpRequest object
  - Pentru a comunica asincron cu serverul
- JavaScript + DOM
  - Pentru a afișa sau a interacționa cu datele
- CSS
  - Pentru a modifica datele din punct de vedere vizual
- XML sau JSON
  - Format folosit în transferul datelor

2.3. Comunicarea cu serverul

- Obiectul XMLHttpRequest este un obiect JavaScript folosit pentru a comunica (a)sincron cu serverul
- Pașii comunicării
  1. Se creează un obiect XMLHttpRequest
  2. Se setează funcția de callback *onreadystatechange* care va procesa răspunsul
  3. Se apelează metodele *open* și *send* pentru a trimite cererea
  4. Când este primit răspunsul se apelează metoda de callback. Răspunsul este conținut în proprietatea responseText. Dacă răspunsul este XML se folosește proprietatea responseXML și se parsează folosind metode din DOM

2.3. Comunicarea cu serverul

```
var xmlhttp;
if (window.XMLHttpRequest) {
    xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange =
    function() {
        if (xmlhttp.readyState == 4 && xmlhttp.status ==
            200) {
            document.getElementById("myDiv").innerHTML =
                    xmlhttp.responseText;
        }
    }
    xmlhttp.open("GET", "info.txt", true);
    xmlhttp.send();
}
```

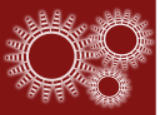
2.4. Same-origin policy

- Permite script-urilor care sunt executate într-un browser ca să acceseze DOAR resurse care au URI-uri cu aceeași schemă (protocol), același hostname și același port

- Alternative:
  - document.domain
  - web server pe post de proxy
  - JSONP
  - Cross-Origin Resource Sharing (CORS)

2.4. Same-origin policy

- https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy
- http://stackoverflow.com/questions/3076414/ways-to-circumvent-the-same-origin-policy
- http://www.hunlock.com/blogs/Howto_Dynamically_Insert_Javascript_And_CSS
- http://www.sitepoint.com/working-around-origin-policy/
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS



3. jQuery

3.1. Introducere

3.2. Utilizarea jQuery

3.3. Sintaxa jQuery

3.4. Exemple jQuery

3.5. Manipularea DOM

3.6. AJAX cu jQuery

3.1. jQuery - introducere

- Bibliotecă JavaScript
- Simplifică task-urile uzuale care necesitau mai multe linii de cod prin înglobarea codului în metode care pot fi apelate printr-o singură linie de cod
- Caracteristici:
  - Traversarea și modificarea documentelor HTML
  - Modificarea stilurilor CSS
  - Manipularea evenimentelor
  - Animații
  - Apeluri AJAX
  - Compatibilitatea cu majoritatea browser-elor

3.2. Utilizarea jQuery

```
<head>
<script src="jquery-1.11.1.min.js"></script>
</head>
```

- CDN – Content Delivery Network

```
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
<script
src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.11.1.min.js"></script>
```

3.3. Sintaxa jQuery

Sintaxa: `$(selector).action()`

- Exemplu: `$("#test").hide()`

Evenimentul onready

```
$(document).ready(function(){
   // metode jQuery și cod JavaScript
});
```

sau

```
$(function(){
   // metode jQuery și cod JavaScript
});
```

3.4. Exemple jQuery

```
$("p").click(function(){
    $(this).hide();
});

$("button").click(function(){
    $("div").animate({left:'250px'});
});

$("#p1").css("color","red").slideUp(2000).
    slideDown(2000);
```

3.5. Manipularea DOM

- Metode aplicate elementelor selectate:
  - text() – get/set pe conținutul text
  - html() – get/set pe conținut inclusiv marcaje
  - val() – get/set pe valoarea din câmpurile formularelor
  - attr() – get/set pe valorile atributelor
  - css() – get/set pe valoarea proprietăților CSS

```
$(".myButton").click(function(){
    alert("Text: " + $("#test").text());
    $("#test").text("Hello world!");
    $("p").css("background-color","red");
});
```

3.6. AJAX cu jQuery

Sintaxa:

```
$(selector).load(URL, data, callback);
```

- Exemplu
  - adăugarea în interiorul elementului div cu id-ul `div1` a conținutului fișierului `demo_test.txt` de la server

```
$("#div1").load("demo_test.txt");
```

Varianta avansată - sintaxa:

```
$.ajax(URL, settings)
```

3.6. AJAX cu jQuery

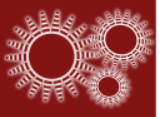
- Exemplu - transmiterea unor date la server și notificarea utilizatorului

```
$.ajax({
    type: "POST",
    url: "some.php",
    data: {name : "John", location : "Boston"}
}).done(function(msg) {
    alert("Data Saved: " + msg);
});
```

3.6. AJAX cu jQuery

- Exemplu - încărcarea și execuția unui fișier JavaScript

```
$.ajax({
    type: "GET",
    url: "test.js",
    dataType: "script"
});
```



4. AngularJS

4.1. Introducere

4.2. Exemplu

4.3. Concepte de bază*

4.4. AJAX cu AngularJS

* Doar câteva concepte AngularJS

4.1. AngularJS - introducere

- Framework și bibliotecă JavaScript
- Ușor de utilizat, în special în aplicațiile single-page
- Versiunea 1.0 a fost lansată în 2012
- Dezvoltat de Miško Hevery de la Google
- Extinde vocabularul HTML cu atribute noi numite directive și leagă datele de HTML prin intermediul expresiilor

4.2. AngularJS - exemplu

```
<html>
<head><script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.26/angular.min.js"></script></head>
<body>
<div ng-app="" ng-init="nume='Ion'">
  <p>Name: <input type="text" ng-model="nume"></p>
  <p ng-bind="nume"></p>
</div>
</body>
</html>
```

4.3. AngularJS – concepte de bază

Directive (en., ng-directives)

- **ng-app** – definește o aplicație AngularJS
- **ng-model** – face legătura dintre controale HTML (input, select, textarea) și datele aplicației
- **ng-bind** – face legătura dintre datele aplicației și partea vizuală a HTML
- **ng-init** – inițializează variabilele aplicației AngularJS
- **ng-controller** – controlează datele unei aplicații

- În loc de prefixul **ng-** se poate folosi **data-ng-** pentru ca documentul să fie valid HTML5

4.3. AngularJS – concepte de bază

Expresii

- Sintaxa: **{{ *expresie* }}**
- Au aceeași funcționalitate ca **ng-bind**
- Sunt asemănătoare cu expresiile JavaScript
- Pot conține numere, șiruri de caractere, operatori și variabile – se pot accesa proprietățile unui obiect sau elementele unui vector

```
<div ng-app=""
  ng-init="persoana={nume: 'Pop', prenume: 'Ion'}">
   <p>
      Nume <span ng-bind="persoana.nume"></span>
      Prenume {{ persoana.prenume }}
   </p>
</div>
```

4.3. AngularJS – concepte de bază

Filtre

- Sunt folosite pentru a transforma datele
- Pot fi adăugate directivelor și expresiilor
- Sintaxa: | ***filtru***
- Exemple: currency, filter, orderBy, lowercase, uppercase

```
<div ng-app=""
  ng-init="persoana={nume: 'Pop', prenume: 'Ion'}">
   <p> Nume
   <span ng-bind="persoana.nume | uppercase"></span>
   Prenume {{ persoana.prenume | lowercase }}
   </p>
</div>
```

4.3. AngularJS – concepte de bază

```
<div ng-app="" ng-controller="controllerNume">
    <ul>
        <li ng-repeat="x in listaNume | orderBy:
        'tara'">
                {{ x.nume + ', ' + x.tara }}</li>
    </ul>
</div>
<script>
    function controllerNume($scope) {
        $scope.listaNume = [ {nume: 'Ion', tara:
                'Romania'}, {nume: 'Hans', tara:
                'Germania'}, {nume: 'Tom', tara: 'UK'}
        ];
    }
</script>
```

4.4. AJAX cu AngularJS

```
<div ng-app="" ng-controller="controllerNume">
    <ul>
        <li ng-repeat="x in listaNume | orderBy:
        'tara'">
                {{ x.nume + ', ' + x.tara }}</li>
    </ul>
</div>
<script>
    function controllerNume($scope, $http) {
        $http.get("clienti.json").success(
            function(raspuns) {
                $scope.listaNume = raspuns;
            }
        );
    }
</script>
```

Bibliografie

- http://www.json.org/
- http://www.json.org/xml.html
- http://www.yaml.org/spec/1.2/spec.html
- http://www.johnpapa.net/pageinspa/
- http://singlepageappbook.com/
- http://www.w3schools.com/ajax/
- http://jquery.com/
- http://www.w3schools.com/jquery/default.asp
- http://www.paulirish.com/2010/the-protocol-relative-url/
- https://angularjs.org/
- http://www.w3schools.com/angular/default.asp