

Tehnici Web

CURSUL 9

Semestrul I, 2019-2020
Carmen Chirita

<https://sites.google.com/site/fmitehnicweb/>

Ajax

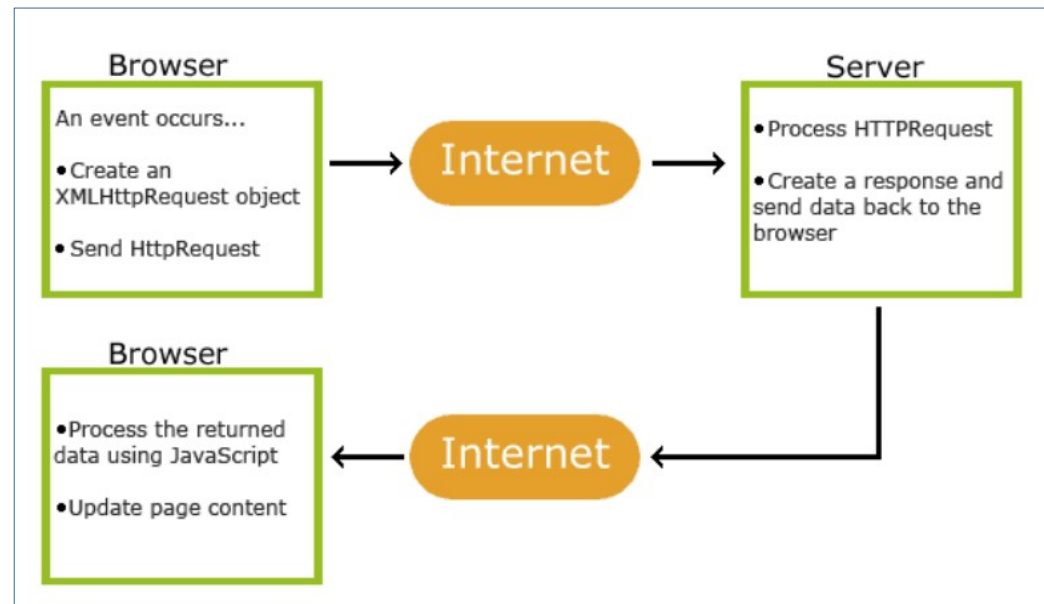
- este prescurtarea de la "Asynchronous JavaScript and XML"
- nu este o tehnologie, termenul se refera la un grup de tehnologii
- termenul a fost introdus de catre Jesse James Garrett in februarie 2005

Ajax

Permite actualizarea unor parti ale unei pagini web fără a reîncarca întreaga pagina

Trimite cereri către un server web și citește datele primite de la server

Nucleul sau îl reprezintă obiectul **XMLHttpRequest** care este folosit pentru a schimba date asincron cu serverul web



XMLHttpRequest

Scopul obiectului XMLHttpRequest este de a permite JavaScript sa formuleze cereri HTTP si sa le trimita la server, sa afiseze datele primite fara a fi necesara reincarcarea paginii.

In plus, pot fi procesate in paralel mai multe conexiuni cu serverul, fara a bloca browser-ul pana la primirea raspunsului.

Inainte de a putea utiliza XMLHttpRequest, trebuie creata o instanta a acestui obiect:

```
var xhr = new XMLHttpRequest()
```

("xhr" poate fi orice nume de variabila)

XMLHttpRequest

onreadystatechange
//functia care se executa
la schimbarea starii

readyState
// starea cererii
(0,1,2,3,4)

status
// codul de stare HTTP
200 pt OK

responseText
//raspunsul primit de la server în format text

responseXML
//raspunsul primit de la server în format XML

XMLHttpRequest

open()
//creaza cererea

send()
//trimite cererea

<https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest>

<http://www.javascriptkit.com/jsref/ajax.shtml>

Crearea unei cereri HTTP: metodele `open()` și `send()`

```
open( method, url, async ) // specifică tipul de cerere
```

`method`: poate fi GET sau POST

`url`: adresa serverului

`async`: true (asynchronous) sau false (synchronous)

```
send() // trimite cererea către server (se folosește cu GET )
```

```
send( string ) // trimite cererea către server (se folosește cu POST)
```

```
var xhr = new XMLHttpRequest();
```

```
xhr.open("GET", "test.txt", true);
```

```
xhr.send();
```

Gestionarea raspunsului de la server

Proprietatea **readyState** reprezintă starea XMLHttpRequest

0: neinitializat, 1: incarca, 2: incarcat (date trimise), 3: interactiv (incep sa se primeasca date de raspuns), 4: complet (raspuns primit complet))

Proprietatea **onreadystatechange** definește o funcție care trebuie executată când se schimbă readyState.

```
xhr.onreadystatechange = nume-functie;
```

Proprietatile:

status: codul de stare HTTP al raspunsului de la server, in format numeric (200 pt. "OK", 403 pt. "Interzis", 404 pt. "Negasit",etc)

statusText: statusul în format text ("OK", "Not Found")

Gestionarea raspunsului de la server

Când readyState este 4 și status este 200, răspunsul este pregătit

Accesarea datelor primite de la server

Proprietatea **responseText**: returneaza raspunsul primit de la server, in format text (string).

Proprietatea **responseXML**: returneaza raspunsul primit de la server in format XML.


Exemplu mdn

```
<script>
window.onload=function() {
  var httpRequest;
  document.getElementById("ajaxButton").addEventListener('click', makeRequest);

  function makeRequest() {
    httpRequest = new XMLHttpRequest(); //creaza un obiect XMLHttpRequest

    if (!httpRequest) {
      alert('Giving up :( Cannot create an XMLHTTP instance');
      return false;
    }
    httpRequest.onreadystatechange = alertContents;
    httpRequest.open('GET', 'test.html');
    httpRequest.send();
  }

  function alertContents() {
    if (httpRequest.readyState === 4) {
      if (httpRequest.status === 200) {
        alert(httpRequest.responseText); //continutul fis. test.html
      } else {
        alert('There was a problem with the request.');
```



The screenshot shows a standard browser alert dialog box with a white background and a gray border. The text inside the dialog is the HTML content of the file 'test.html'. At the bottom right of the dialog is a single button labeled 'Ok'.

```
<!DOCTYPE html>
<html lang="ro">
<head>
<meta charset="utf-8">
<title>Exemplu Ajax</title>
</head>
<body>
Acesta este un test.
</body>
</html>
```

Ok

XML- Extended Markup Language

Fișierul test.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<persoane>  
  <pers nume="Ion" varsta = "40"></pers>  
  <pers nume="Maria" varsta ="30"></pers>  
</persoane>
```

Obiectul **XMLDocument** in JavaScript
poate fi parcurs cu metode asemanatoare celor din DOM

```
var xml = httpRequest.responseXML;  
var vpers= xml.getElementsByTagName('pers');  
alert(vpers[0].getAttribute('nume'));
```

Exemplu XMLHttpRequest si XML

<script>

```
.....
httpRequest.open('GET', 'test.xml');
httpRequest.onreadystatechange = alertContents;
httpRequest.send();
}
function alertContents() {
    if (httpRequest.readyState === 4) {
        if (httpRequest.status === 200) {

            var xmlDoc = httpRequest.responseXML;
            var per = xmlDoc.getElementsByTagName('pers');
            var content = "";
            for (var i=0; i < per.length; i++)
                content = content + per[i].getAttribute("nume") + " are " + per[i].getAttribute("varsta")
            + " ani \n";
            alert(content);

            .....
        }
    }
}
</script>
```

test.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persoane>
    <pers nume="Ion" varsta = "40"></pers>
    <pers nume="Maria" varsta ="30"></pers>
</persoane>
```

Ion are 40 ani
Maria are 30 ani

Ok

JSON = JavaScript Object Notation

<http://www.json.org/>

Ofera o modalitate de reprezentare a datelor, ca alternativa la XML.
Bazat pe JavaScript, este in prezent un format independent de limbaj.
Multe limbaje pot prelucra date in format JSON.
Este folosit pentru schimbul de informații cu serverul.

Elemente de baza:

<i>Object:</i>	<code>{"cheie1":val1, "cheie2":val2}</code>
<i>Array:</i>	<code>[val1, val2, val3]</code>
<i>Value:</i>	string, number, object, array, true, false, null

date.json

<pre>{ "pers": { "nume": "Ion", "varsta": 42 } }, { "pers": { "nume": "Maria", "varsta": 30 } }]</pre>

Sintaxa JSON

Câmpul **cheie** trebuie să fie scris cu ghilimele

`"nume": "Ana"`

Câmpul **valoare** poate fi:

`string, number, obiect (JSON), array, boolean, null`

Valoare nu poate fi

`function`
`date`
`undefined`

Obiectele JSON sunt reprezentate între acolade

`{"nume": "Ana", "varsta": 30, "porecla": null }`

Elementele array sunt reprezentate între paranteze drepte

`["Ana", "Mihai", "Maria"]`

Valoare: string, number, object, array, true, false, null

JSON String: { "nume": "Andrei" }

JSON Number: { "varsta": 30 }

JSON Object: { "pers": { "nume": "Ion", "varsta": 42 } }

JSON Array: { "studenti": ["Ionut", "Mihai", "Dana"] }

JSON Boolean: { "promovat": true }

JSON null: { "porecla": null }

Obiecte JSON

```
myObj = {"cheie1":val1, "cheie2":val2, "cheie3":val3 };
```

Accesarea obiectelor: `myObj.cheie1` sau `myObj["cheie1"]`

Iterarea proprietatilor unui obiect

```
<script>
var myObj = { "student":"Popescu", "grupa":231, "promovat":true };
for (x in myObj) {
    document.getElementById("prop").innerHTML += x + "<br>";
}
</script>
```

```
.....
<p id="prop">
```

Paragraful va contine

student
grupa
promovat

Obiecte JSON

Iterarea valorilor proprietatilor unui obiect

```
<script>
var myObj = { "student":"Popescu", "grupa":231, "promovat":true };
for (x in myObj) {
    document.getElementById("val").innerHTML += myObj[x]+ " ";
}
</script>
```

```
.....
<p id="val">
```

Paragraful va contine

Popescu 231 true

Obiecte JSON încorporate

```
var myObj = { "student":"Ionescu",  
              "grupa":30,  
              "note": {"nota1":8,"nota2":9, "nota3":10}  
            }
```

Accesarea obiectelor încorporate:

```
myObj.note.nota2 // 9  
myObj.note["nota2"] // 9
```

Modificarea valorilor: `myObj.note.nota1="10";`

Stergerea proprietatilor: `delete myObj.note.nota1;`

JSON Arrays

[val1, val2, ..., valn]

val1,...,valn pot fi string, number, object, array, boolean or null.

Array în interiorul obiectelor JSON

```
var ob = { "student": "Ionescu",  
           "grupa": 30,  
           "note": [7, 8, 9]  
         }
```

Accesarea valorilor: ob.note[0] // 7

Iterarea valorilor în Array:

```
for (i în ob.note)  
{  
    x += ob.note[i];  
}
```

sau

```
for (i=0; i< ob.note.length; i++)  
{  
    x += ob.note[i];  
}
```

Exemplu (w3schools) (array incorporat în array)

```
<p id="demo"></p>
```

```
<script>
```

```
var myObj, i, j, x = "";
```

```
myObj = {
```

```
  "name": "John",
```

```
  "age": 30,
```

```
  "cars": [
```

```
    { "name": "Ford", "models": [ "Fiesta", "Focus", "Mustang" ] },
```

```
    { "name": "BMW", "models": [ "320", "X3", "X5" ] },
```

```
    { "name": "Fiat", "models": [ "500", "Panda" ] }
```

```
  ]
```

```
}
```

```
for (i in myObj.cars) {
```

```
  x += "<h2>" + myObj.cars[i].name + "</h2>";
```

```
  for (j in myObj.cars[i].models) {
```

```
    x += myObj.cars[i].models[j] + "<br>";
```

```
  }
```

```
}
```

```
document.getElementById("demo").innerHTML = x;
```

```
</script>
```

Ford

Fiesta

Focus

Mustang

BMW

320

X3

X5

Fiat

500

Panda

Obiectul JSON in JavaScript

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON

`JSON.stringify(valoare)` // transforma un obiect JavaScript intr-un string JSON
`JSON.parse(text)` //transforma un string JSON într-un obiect JavaScript

Exemplu:

```
var o1 = {pers: {nume:"Ion", varsta:42}},  
    o2 = {pers: {nume:"Maria", varsta:30}},  
    o = [o1,o2];  
  
var s = JSON.stringify(o);  
// "[{"pers":{"nume":"Ion","varsta":42}},{"pers":{"nume":"Maria","varsta":30}}]"  
  
localStorage.setItem("myarray", s);  
var st = localStorage.getItem("myarray");  
  
var jo = JSON.parse(st);
```

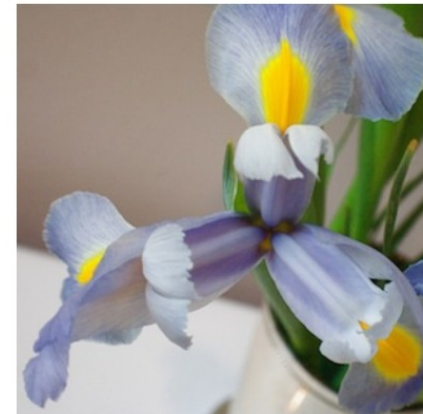
Poate fi folosit pentru memorare
in localStorage si sessionStorage

pictures.json

```
[  
  {"picture": {"caption": "Picture 1", "source": "images/flower1-300.jpg"} },  
  {"picture": {"caption": "Picture 2", "source": "images/flower2-300.jpg"} },  
  {"picture": {"caption": "Picture 3", "source": "images/flower3-300.jpg"} },  
  {"picture": {"caption": "Picture 4", "source": "images/flower4-300.jpg"} },  
  {"picture": {"caption": "Picture 5", "source": "images/flower5-300.jpg"} }  
]
```

```
var data ;  
var httpObj = new XMLHttpRequest();  
httpObj.open('GET', "pictures.json", true);  
  
httpObj.onreadystatechange = function() {  
  if (httpObj.readyState == 4) {  
    if (httpObj.status == 200)  
    {  
      data=JSON.parse(httpObj.responseText);  
      gallery(data);  
    }  
    else {alert("eroare");}  
  }  
  
  httpObj.send(null);  
};
```

Gallery 1



Picture 1

Picture 1

Picture 2

Picture 3

Picture 4

Picture 5

```
function gallery(data){
  var menu_links = document.querySelectorAll("#menu button");
  var img_gal =document.getElementById("gal");
  var img_cap =document.getElementById("caption");
  for (var i=0; i< menu_links.length; i++)
  {
    let j=i;

    menu_links[j].onclick = function () {

      img_gal.src=data[j].picture.source;

      img_cap.textContent=data[j].picture.caption;

    }

  }
}
```

```
<body>
<h1>Gallery 1</h1>
<div id="container">
  <p>
    
  </p>
  <p id="caption">
    Picture 1
  </p>
</div>
<p id="menu">
  <button type="button">Picture 1</button>
  <button type="button">Picture 2</button>
  <button type="button">Picture 3</button>
  <button type="button">Picture 4</button>
  <button type="button">Picture 5</button>
</p>
</body>
```

➤ Submiterea formelor

```
<body>
<form id="testform" method="post" action="http://localhost:8080/action">

<label>Nume:</label>
  <input type="text" name="name">
<label> Varsta:</label>
<input type="text" name="age">
<label>Localitate:</label>
<select name="city">
  <option value="Bucuresti">Bucuresti</option>
  <option value="Timisoara" selected>Timisoara</option>
</select>
<button type="submit" id="buton"> Trimite </button>
</form>
</body>
```

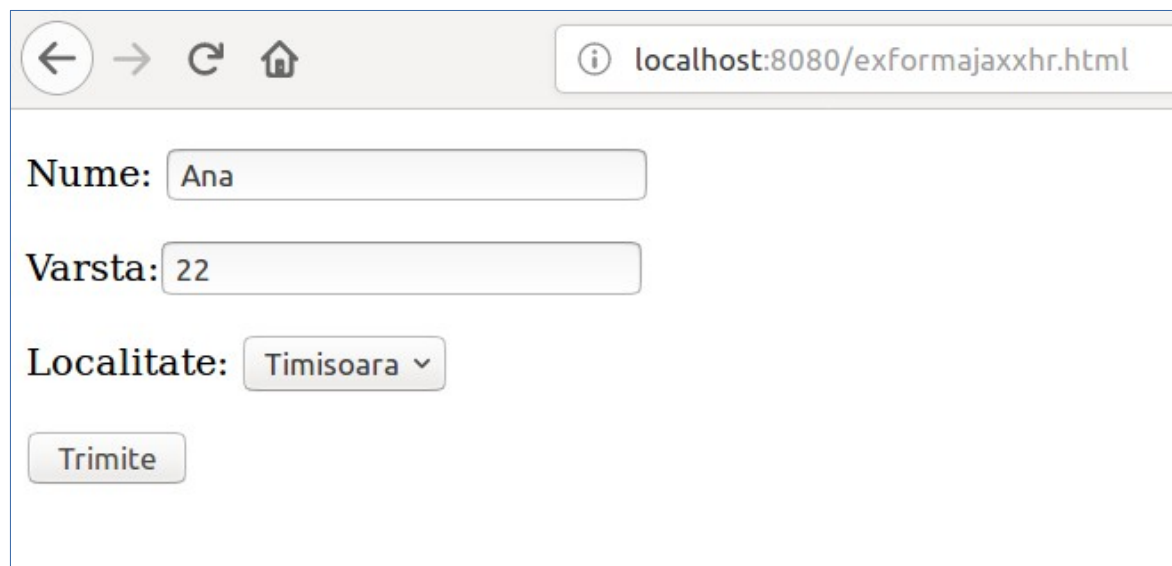
Aplicația server

```
const express = require('express')
const bodyParser = require('body-parser')
const multipart = require('connect-multiparty')
const app = express()
const multi = multipart()

app.use(bodyParser.urlencoded({extended: true}))
app.use(express.static('public'));

app.get('/', function(req, res) {res.send('hello');})
app.post('/action', multi, function(req, res) {res.send
(req.body.name + ' din ' + req.body.city + ' are ' + req.body.age +
' (de) ani');})
app.listen(8080, function() {console.log('listening')})
```

➤ Submiterea formelor

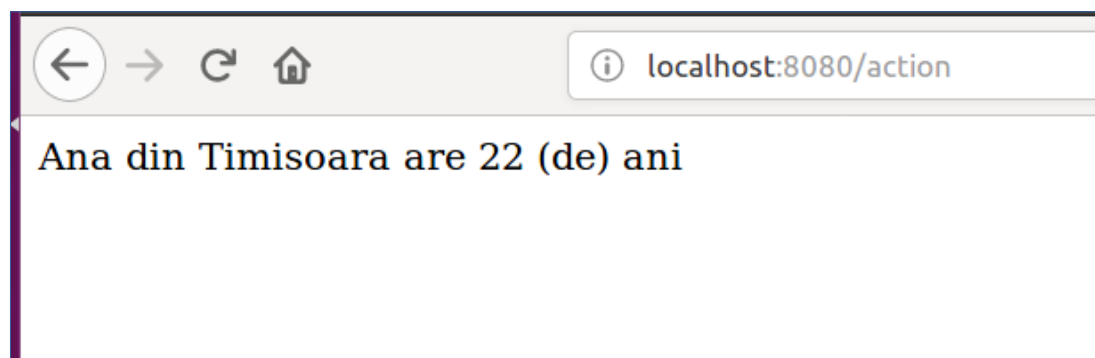


A screenshot of a web browser window. The address bar shows the URL `localhost:8080/exformajaxxhr.html`. The page contains a form with three input fields: "Nume:" with the value "Ana", "Varsta:" with the value "22", and "Localitate:" with a dropdown menu showing "Timisoara". Below the form is a button labeled "Trimite".

Nume:

Varsta:

Localitate:



A screenshot of a web browser window. The address bar shows the URL `localhost:8080/action`. The page displays the text "Ana din Timisoara are 22 (de) ani".

Ana din Timisoara are 22 (de) ani

➤ Submiterea formelor cu Ajax





JavaScript


```
window.onload=function () {  
    var forma=document.getElementById("testform");  
  
    forma.onsubmit= function (event){  
        event.preventDefault();  
  
        var data = new FormData(forma);  
        //datele din forma  
        var xhr = new XMLHttpRequest();  
        xhr.open('POST', forma.action, true);  
        xhr.onreadystatechange = function()  
        {  
            if (xhr.readyState == 4){  
                if (xhr.status == 200)  
                {document.getElementById("info").innerHTML=xhr.responseText;  
                    else alert("error");  
                }  
            }  
        };  
        xhr.send(data);  
    }  
}
```

HTML

```
<form id="testform" action="http://localhost:8080/action">  
    <p> <label>Nume:</label> <input type="text" name="name">  
    </p>  
    <p> <label> Varsta:</label><input type="text"  
    name="age"></p>  
    <p> <label>Localitate:</label> <select name="city"></p>  
        <option value="Bucuresti" >Bucuresti</option>  
        <option value="Timisoara" selected>Timisoara</option>  
    </select>  
    <p><button type="submit" id="buton"> Trimite </button> </p>  
</form>  
<article id="info">  
</article>
```

```
data= "name=Ana&age=22&city=Timisoara"
```



 localhost:8080/exformajaxxhr.html

Nume:

Varsta:

Localitate:

Timisoara ▾

Trimite

Ana din Timisoara are 22 (de) ani

jQuery (bibliotecă, creată de [John Resig](http://jquery.com/download/))
<http://jquery.com/download/>

Caracteristici:

- Metode eficiente pentru selecția elementelor; metode specifice de prelucrare a elementelor selectate.
- Metode pentru evenimente.
- Metode pentru efecte speciale și animații.
- Metode Ajax.

Cum se introduce JQuery?

```
<head>
```

(descărcarea de pe site-ul <http://jquery.com>)

```
<script type="text/javascript" src="jquery-3.3.1.min.js"></script>
```

sau

(inclusiunea din CDN)

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

```
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.3.1.min.js"></script>
```

```
</head>
```

Sintaxa jQuery

\$ sau jQuery() este unica functie globala si ea intoarce un **obiect jQuery**.

\$(selector).action()

selectarea elementelor si efectuarea unor acțiuni asupra elementelor

myjqname.js

```
$(document).ready(function(){  
    Cod JQuery  
});
```

sau

```
$(function(){  
    Cod JQuery  
});
```

Se executa codul JQuery după încărcarea paginii

jQuery

```
$(document).ready(function(){  
  
  $("p").css("background-color","gray");  
  
  $(".test").hide();  
  
  $(this).click(function(){ $(this).hide();  
});  
});
```

`$("p")` selecteaza toate elementele de tipul "p"

`$(".test")` selecteaza toate elementele cu clasa "test"

`$(document)`, `$(this)` intoarce un obiect **jQuery** caruia
ii pot fi aplicate metodele specifice **jQuery**

jQuery

```
$(document).ready(function(){  
  
var art = $("article");  
  
$("p", art). hover(function(){ alert(this.id);});  
  
});
```

`$(“selector”,el)` intoarce multimea descendentilor
elementelor el care verifica “selector”

jQuery

```
$(document).ready(function(){  
  
var art = $("article");  
  
$("p", art). click(function(){  
    $(this).css("background-color","gray");  
});  
});
```

`$(this)` intoarce un obiect **jQuery** caruia
ii pot fi aplicate metodele specifice **jQuery**

jQuery selectors

selectorii jQuery permit selectarea și manipularea elementelor HTML.

se bazează pe selectorii CSS existenți și, în plus, există anumiți selectori personalizați.

toți selectorii din jQuery încep cu semnul dolar și paranteze: \$ ().

selectorii CSS

```
$("#div"); $("#myid"); $(".myclass"); $("input[type='text']");
```

```
$("li:first-child"); $("li:last-child"); $("p:first-of-type"); $("p:nth-of-type(2)")
```

```
 $("div > p") ; $("div p") ; $("div + p") ; $("div ~ p")
```


Selectori JQuery

<http://api.jquery.com/category/selectors/jquery-selector-extensions/>

`$(":animated");` // selecteaza elementele care sunt intr-o animație la momentul executarii selectorului

`$(":button");` // selecteaza elementele input cu type="button"
`$(":radio");` // selecteaza elementele input cu type="radio"

`$(":header");` // selecteaza elementele header h1-h6

`$(":gt(n)");` // selecteaza elementele cu index $\geq n$

`$(":not(selector)");` //selecteaza toate elementele, cu excepția celor definite de selector

```
$("p:not(.intro)").css("background-color", "yellow");
```

jQuery: event handlers

Metode de evenimente:

.click(), .dblclick(), .submit(), .hover(), .keydown()

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

- ```
$(window).keydown(function(event){alert(event.key);});
```

## jQuery chaining

```
$("#b1").click(function(){
 $("#article p").hide();})
 .dblclick(function(){
 $("#article p").show();});
```

## Metoda on()

ataseaza una sau mai multe funcții handler de ev. pentru elementele selectate

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

```
$("#p").on({
 mouseenter: function(){
 $(this).css("background-color", "lightgray");
 },
 mouseleave: function(){
 $(this).css("background-color", "lightblue");
 },
 click: function(){
 $(this).css("background-color", "yellow");
 }
});
```

## Traversare DOM cu jQuery

`children()` // copii directi ai elementului selectat

`find()` // toți descendentii elementului selectat

`parent()` //parintele elementului selectat

`parents()` //stramosii elementului selectat

`parentsUntil()` //stramosii pana la elementul  
specificat

`siblings()` //toti fratii elementului selectat

`next()`, `nextAll()`, `nextUntil()`

`prev()`, `prevAll()`, `prevUntil()`

```
$("li").parent().css("border", "3px solid red");
```

# Traversare DOM cu jQuery - metode de filtrare

**first(), last(), eq(), filter(), not()**

**first()** // returneaza primul element al elementelor selectate

**last()** // returneaza ultimul element al elementelor selectate

```
$("#div").first(), $(".unu").last
```

**eq()** // returneaza elementul cu un anumit index  
al elementelor selectate (index >=0)

```
$("#p").eq(1)
```

**filter()** // selecteaza elementele care indeplinesc o conditie

**not()** // opusa metodei filter()

```
$("#ul li").filter(":gt(2)"), $("#ul li").not(":gt(2)")
```

# Manipulare DOM cu jQuery

## creare/ stergere/modificare

```
$(target).method(content)
```

```
append() //inserare la sfârșitul elem. selectat
prepend() //inserare la începutul elem. selectat
after() //inserare dupa elem. selectat
before() //inserare înainte de elem. selectat
remove()//stergerea elementului selectat
empty()//sterge copii elementului selectat
```

# Manipulare DOM cu jQuery

```
$(document).ready(function(){
 $("#apend").click(function(){
 $("ol").append("Element nou");
 });
 $("#prepend").click(function(){
 $("ol").prepend("Element nou");
 });

 $("#after").click(function(){
 $("ol").after("<p>Element nou</p>");
 });
 $("#before").click(function(){
 $("ol").before("<p>Element nou</p>");
 });
 $("#empty").click(function(){
 $("ol").empty();
 });
});
```

append.html

```
<body>
<button id="apend">Append</button>
<button id="prepend">Prepend</button>
<button id="after">After</button>
<button id="before">Before</button>
<button id="empty">Empty</button>

Item 1
Item 2
Item 3
Item 4

</body>
```

Append

Prepend

After

Before

Empty

1. Item 1
2. Item 2
3. Item 3
4. Item 4

# Selectarea și modificarea conținutului unui element

## Metodele `text()`, `html()`, `val()`

pentru campurile formelor

```
<p id="myelem">
Text vechi
</p>
<form> <input id="myfin" value="valoare initiala">
</form>
```

```
$("#myelem").text(); // intoarce conținutul ca text fara marcate
```

```
$("#myelem").html(); // intoarce conținutul ca text cu marcate
```

```
$("#myelem").text("Text nou"); // modifica conținutul
```

```
$("#myelem").html(" Text nou "); // modifica conținutul interpretand marcatele
```

```
$("#myfin").val (); // intoarce valoarea atributului value
```

```
$("#myfin").val ("valoare noua"); // modifica valoarea atributului value
```



## Manipularea atributelor unui element

`attr("nume-atr"),`

`attr("nume-atr","valoare")`

`attr({atribut1:val1, atribut2:val2,...})`

`removeAttr("nume-atr")`

```
$("#flori").attr("alt", "flori de vara");
```

```
$("#fmi").attr("src","http://www.fmi.unibuc.ro");
```

```
$("#a").attr("target","_blank");
```

```
$("#a").removeAttr("target");
```

```
$("#img").attr({width:150px;height:200px;});
```

# Modificarea proprietatilor CSS

css("nume-proprietate", "valoare"),  
hasClass("nume-clasa"),  
addClass("nume-clasa"),  
removeClass("nume-clasa")

```
.vechi {color:blue;}
```

```
.nou {color:red;}
```

```
<div id="deschimbata">
Text
</div>
```

```
de= document.getElementById("deschimbata");

function schimba(de){
 if ($(de).hasClass("vechi"))
 {$(de).removeClass("vechi");
 $(de).addClass("nou");}
}
```

## jQuery effects (metode pentru efecte speciale)

show("fast", [callback]), hide(), toggle(),  
fadeIn(), fadeOut(), fadeToggle(), fadeTo()  
slideUp(), slideDown("slow"), slideToggle(),  
animate()

# jQuery effects (metode pentru efecte speciale)

## Exemplul 1 (show\_hide.html)

```
$(document).ready(function(){
 $("#hide").click(function(){
 $("p").hide(2000);
 });
 $("#show").click(function(){
 $("p").show(3000,function(){alert("Hello");});
 });
});
```

Paragrafele se vor ascunde la click pe Hide

Paragrafele vor aparea la click pe Show

Hide

Show

<body>

<p>Paragrafele se vor ascunde la click pe Hide</p>

<p>Paragrafele vor aparea la click pe Show</p>

<button id="hide">Hide</button>

<button id="show">Show</button>

</body>

# jQuery effects (metode pentru efecte speciale)

## Exemplul 2 (fade.html)

```
$(document).ready(function(){
 $("button").click(function(){
 $("#div1").fadeToggle();
 $("#div2").fadeToggle("slow");
 $("#div3").fadeToggle(3000);
 });
});
```

```
<body>
<button>Click</button>

<div id="div1" style="width:80px;height:80px;
background-color:red;"></div>

<div id="div2" style="width:80px;height:80px;
background-color:green;"></div>

<div id="div3" style="width:80px;height:80px;
background-color:blue;"></div>

</body>
```

fadeToggle()

Click



# jQuery effects (metode pentru efecte speciale)

## Exemplul 3 (slideUp.html)

```
$(document).ready(function(){
 $("#par").click(function(){
 $("li").slideUp("slow");
 });
});
```

Click pentru a vedea efectul slideUp()

1. Item 1
2. Item 2
3. Item 3
4. Item 4

```
<p id="par">Click pentru a vedea efectul slideUp()</p>

Item 1
Item 2
Item 3
Item 4

```

## jQuery effects: animate()

creaza efect de animatie simultan pentru mai multe proprietati CSS care au valori numerice

```
$(#elanim).animate(
{ left: 200px;
width: "+=100"; // proprietati
border: "hide"}, // CSS

2000, // durata animatiei

function (){ ... } // callback
)
```

proprietati CSS care au  
valori numerice:

border, margin, padding,  
height,width,  
font size, line height,  
background position,

bottom, left, right, top  
(pentru elemente cu position)

# jQuery chaining

```
<body>
<button>Click</button>

<div id="box"></div>
</body>
```

```
#box {
 background: red;
 height: 100px;
 width: 100px;
 position: relative;
 top: 200px;
 left: 200px;
}
```

animate.html

```
$("#button").click(function(){

 $("#box").animate({opacity: "0.1", left: "+=400"}, 1200)
 .animate({opacity: "0.4", top: "+=160", height: "20", width: "20"}, "slow")
 .animate({opacity: "1", left: "0", height: "100", width: "100"}, "slow")
 .animate({top: "0"}, "fast")
 .slideUp()
 .slideDown("slow", function() {alert("Gata")});
 return false;
});
```



AJAX se ocupa cu schimbul de date cu un server și actualizarea unor parti dintr-o pagină web fără a reîncărca întreaga pagină.

jQuery oferă mai multe metode pentru funcționalitatea AJAX.

Cu ajutorul metodelor jQuery AJAX, se pot face cereri pentru date în format text, HTML, XML sau JSON de la un server folosind atât GET, cât și POST și se pot încărca datele externe direct în elementele HTML selectate

## Ajax: metoda `load()`

încarcă datele de pe un server și pune datele returnate în elementul selectat.

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

**URL** specifică adresa URL care se va încarca (parametru obligatoriu)

**date** reprezintă query stringul care se va trimite împreună cu cererea (optional)

**callback** este numele unei funcții care urmează a fi executată după ce metoda de încărcare a fost terminată (optional)

parametrii

**raspuns** - conține răspunsul de la server.  
**status** - conține status-ul (starea) cererii.  
**xhr** - obiectul XMLHttpRequest.

## Exemplu

```
<script>
$(document).ready(function(){
 $("button").click(function(){
 $("#div1").load("toload.html #doi", function(raspuns, status, xhr){
 if(status == "success")
 alert(raspuns);
 if(status == "error") alert(xhr.status);
 });
 });
});
</script>
```

```
<body>
<div id="div1"><h2>Continutul div-ului</h2></div>
<button>Load</button>
</body>
```

**Continutul div-ului**

Load

continut nou 2

Load

```
<!doctype html>
<html>
<article id="unu">
continut nou 1
</article>
```

```
<article id="doi">
continut nou 2
</article>
```

```
<article id="trei">
continut nou 3
</article>
</html>
```

Ok

## Ajax: metoda `get()`

solicita date de la un server cu o cerere HTTP GET

```
$.get(URL,callback)
```

**URL** specifică URL-ul unde se va trimite cererea (parametru obligatoriu)

**callback** este numele unei funcții care va fi executată dacă cererea este reusita (optional)

parametrii

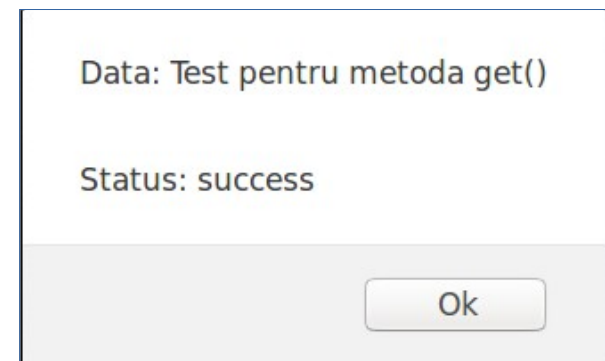
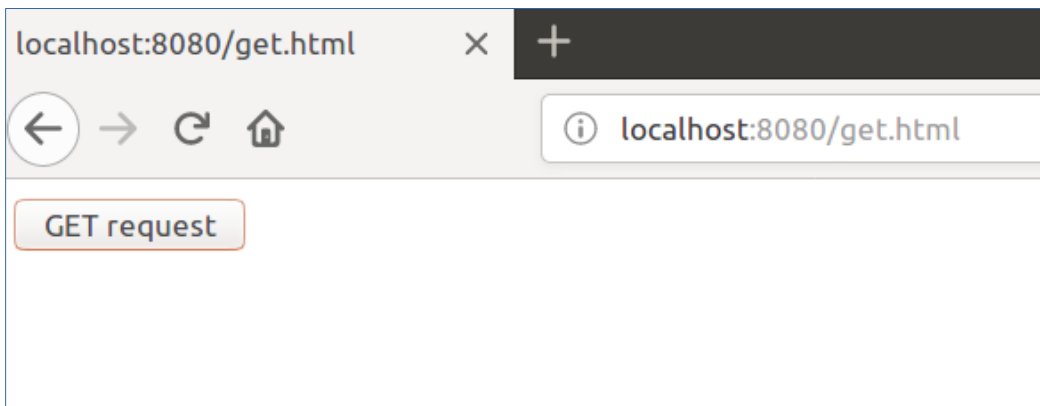
**raspuns** - contine raspunsul de la server.  
**status** - contine status-ul (starea) cererii.  
**xhr** - obiectul XMLHttpRequest.

## Exemplu: metoda get()

```
<script>
$(document).ready(function(){
 $("button").click(function(){
 $.get("http://localhost:8080/get.txt", function(data, status){
 alert("Data: " + data + "\nStatus: " + status);
 });
 });
});
</script>
```

get.txt

Test pentru metoda get()



# Ajax: metoda `post()`

solicita date de la un server cu o cerere HTTP POST

```
$.post(URL,data,callback)
```

**URL** specifică adresa URL la care se va trimite cererea (parametru obligatoriu)

**date** reprezintă query stringul care se va trimite împreună cu cererea (optional)

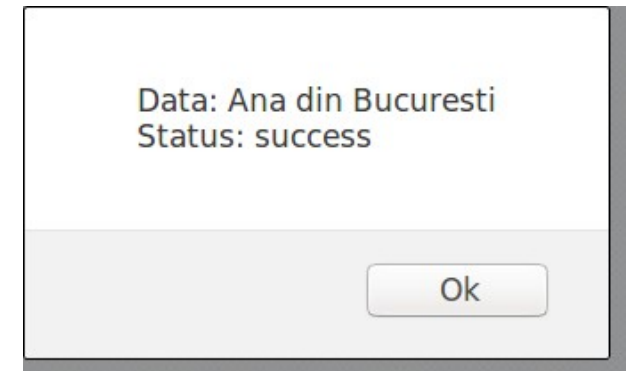
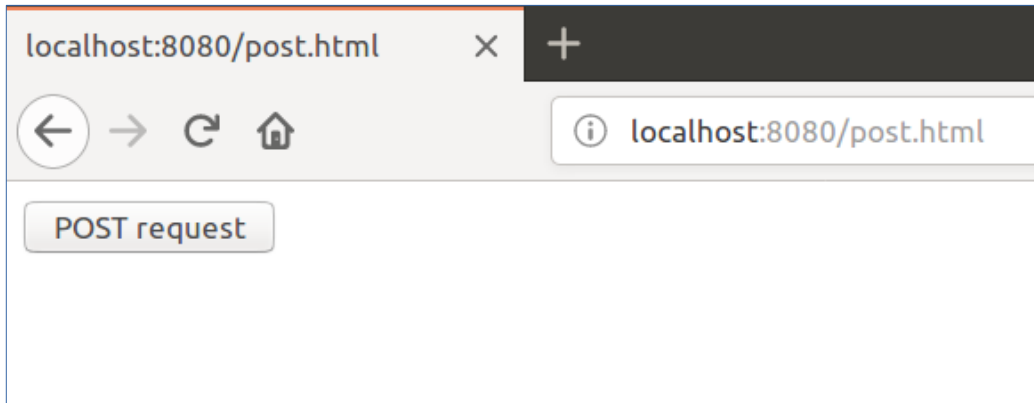
**callback** este numele unei funcții care va fi executată dacă cererea este reusita (optional)

parametrii

**raspuns** - contine raspunsul de la server.  
**status** - contine status-ul (starea) cererii.  
**xhr** - obiectul XMLHttpRequest.

Exemplu: metoda post()

```
<script>
$(document).ready(function(){
 $("button").click(function(){
 $.post("http://localhost:8080/pp", {nume: "Ana", city:"Bucuresti"},
 function(data, status){
 alert("Data: " + data + "\nStatus: " + status);
 });
 });
});
</script>
```



## Ajax: metoda `ajax()`

Se foloseste pentru a efectua o cerere AJAX (asynchronous HTTP) la server.

Toate functiile Ajax jQuery: `load()`, `get()`, `post()` folosesc metoda `ajax()`. Aceasta contine mai multe optiuni de configurare si prelucrare a datelor, iar sintaxa ei este mai complexa.

```
$.ajax({
 type: "GET"//"POST",
 url: url,
 data: {"nume":"valoare", "nume2":valoare2},
 // string sau obiect
 dataType: "text", //"xml", "json"
 success: function (raspuns, status){},
 error: function (raspuns, status){},
})
```



```
<script>
var persoane;
$(document).ready(function(){
 $("button").click(function(){
 $.ajax({
 type: "GET",
 url: "date.json",
 dataType: "json",
 success: function (data){
 persoane = data;
 for (var i=0; i< persoane.length; i++)
 $("#div1").append ("<p>" + persoane[i].pers.numa + " are varsta " +
 persoane[i].pers.varsta + "</p>");
 }
 });
 });
});
</script>
```

```
<body>
<div id="div1">
<h2>Aici se vor incarca date
noi</h2>
</div>
<button>Load</button>
</body>
```

date.json

```
[
 {
 "pers": {
 "numa": "Ion",
 "varsta": "42"
 }
 },
 {
 "pers": {
 "numa": "Maria",
 "varsta": "30"
 }
 }
]
```

**Aici se vor incarca date noi**

Ion are varsta 42

Maria are varsta 30

Load

<body>

<form id="testform" action="http://localhost:8080/action" method="POST">

<label>Nume:</label>

<input type="text" name="name"><br>

<label> Varsta:</label>

<input type="text" name="age"><br>

<label>Localitate:</label>

<select name="city">

<option value="Bucuresti" selected>Bucuresti</option>

<option value="Timisoara" selected>Timisoara</option>

</select><br>

<button type="submit" id="buton"> Trimite </button>

</form>

</body>

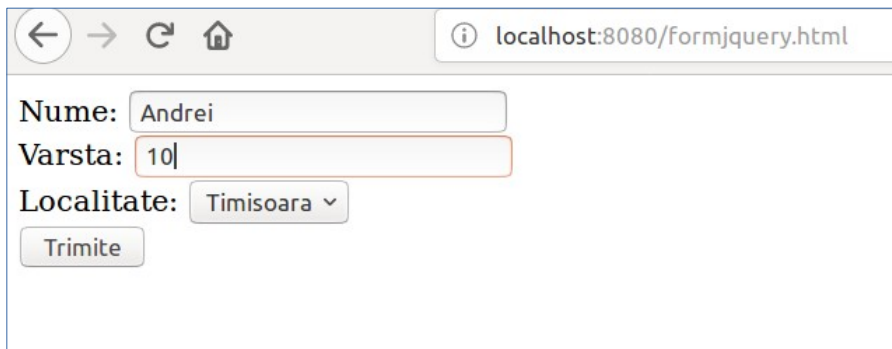
Aplicația server

```
const express = require('express')
const bodyParser = require('body-parser')
const multipart = require('connect-multiparty')
const app = express()
const multi = multipart()

app.use(bodyParser.urlencoded({extended: true}))
app.use(express.static('public'));

app.post('/action', multi, function(req, res) {res.send(req.body.name + ' din '
| + req.body.city + ' are ' + req.body.age + ' (de) ani');})

app.listen(8080, function() {console.log('listening')})
```

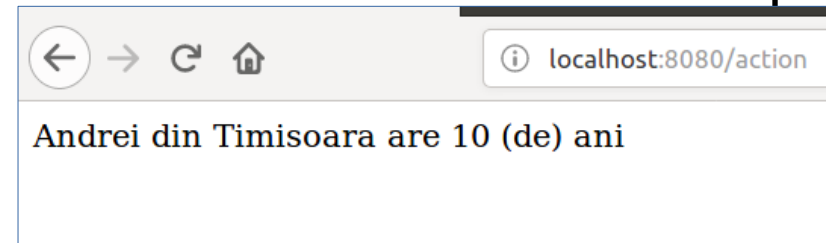


← → ↻ 🏠 localhost:8080/formjquery.html

Nume:

Varsta:

Localitate:



← → ↻ 🏠 localhost:8080/action

Andrei din Timisoara are 10 (de) ani

## ➤ Submiterea formelor folosind JQuery ajax()

```
$(document).ready(function(){

$("#testform").submit(function (event){

var dataform = $("#testform").serialize();

$.ajax({
 type: "POST",
 url:"http://localhost:8080/action",
 data: dataform,
 success: function (data, textStatus){
 $("#info").html(data);},
 error: function (data, statusCode){
 alert(statusCode);}
 });
 event.preventDefault();
}) })
```



dataform= "name=Maria&age=20&city=Bucuresti"

← → ↻ 🏠 localhost:8080/formjquery.html

Nume:

Varsta:

Localitate:

← → ↻ 🏠 localhost:8080/formjquery.html

Nume:

Varsta:

Localitate:

Maria din Bucuresti are 20 (de) ani

## Subiect dat la examen

S1. Sa se implementeze urmatoarea functionalitate:  
la click pe orice paragraf se construiesc o lista cu continutul elementelor **<b>**  
care sunt copii ai paragrafului, lista fiind adaugata in document imediat dupa paragraf;  
urmatoarele clickuri pe paragraf nu vor avea nici un efect.

Ex: daca p este

`<p id="pgf">Ion are o <b>pisica</b>, un <b>catel</b>, un <b> papagal<b>. </p>`  
atunci lista va contine: **pisica, catel, papagal.**

### liste.html

Ion are o **pisica**, un **catel**, un **papagal**.

George are o **bicicleta** si un **scuter**.

Ana are **mere**, cateva **pere**, si multe **nuci**.

Vine **vacanta**, de Craciun voi primi **cadouri** si voi rezolva **exercitii** la **Tehnici Web** pentru **examen**.

A venit vacanta!

Ion are o **pisica**, un **catel**, un **papagal**.

- pisica
- catel
- papagal

George are o **bicicleta** si un **scuter**.

Ana are **mere**, cateva **pere**, si multe **nuci**.

- mere
- pere
- nuci

Vine **vacanta**, de Craciun voi primi **cadouri** si voi rezolva **exercitii** la **Tehnici Web** pentru **examen**.

- vacanta
- cadouri
- exercitii
- Tehnici Web
- examen

A venit vacanta!

liste.js

```
function copiiB(elem){ \creeaza lista cu continutul copiilor de tip b ai elementului elem
 var copii=elem.getElementsByTagName("b");
 var ul=document.createElement("ul");
 for(var i=0;i<copii.length;i++){
 var li=document.createElement("li");
 li.innerHTML=copii[i].innerHTML;
 ul.appendChild(li);
 }
 var frate=elem.nextElementSibling;
 document.body.insertBefore(ul,frate); \inserare lista imediat după elem
}

window.onload=function(){
 var colectie=document.getElementsByTagName("p"); \selectarea paragrafelor
 for(var i=0;i<colectie.length;i++){
 let k=i;
 colectie[i].nr=0; \proprietate noua care reține nr. clickului
 colectie[i].onclick = function(){ \clickul pe paragrafe cu apelarea functiei copiiB
 if(colectie[k].nr==0){
 copiiB(colectie[k]);
 colectie[k].nr=1;
 }
 }
 }
}
```