

Tehnici Web

CURSUL 9

Semestrul I, 2018-2019
Carmen Chirita

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

(old) Client-side storage: cookies

Cookies sunt date depozitate de browser pe calculatorul utilizatorului (max 4KB) si sunt automat transmise serverului.

O cookie este o pereche **nume=valoare**

`document.cookie` intoarce un string care contine toate cookies atasate documentului

```
document.cookie ="mycookie = Hello"
```

.

Client-side Web Storage

Un obiect Storage este un array asociativ in care **cheile** si **valorile** sunt **stringuri**.

Urmatoarele proprietati ale obiectului window intorc obiecte din clasa Storage:

localStorage // permanent

sessionStorage //pana la inchiderea tabului

<http://www.w3.org/TR/2013/REC-webstorage-20130730/>

<https://developer.mozilla.org/en-US/docs/Web/Guide/API/DOM/Storage>

Client-side Web Storage

`localStorage.length` // nr de date pastrate în Storage

`localStorage.key(i)` // numele cheii cu indexul i

`localStorage.setItem(ume-cheie, ume-valoare)` //
adauga o cheie și valoarea ei sau înlocuiește
valoarea unei chei existente

`localStorage.getItem(ume-cheie)` // valoarea cheii

`localStorage.removeItem("x")` // șterge cheia din
Storage

`localStorage.clear()` // șterge toate cheile

`localStorage.propNoua=valoare`

Proprietatile și metodele sunt la fel și pentru
`sessionStorage`

```
<script>
window.onload = function()
{
    localStorage.setItem("nrc","0");
    var buton=document.getElementById("bt");
    buton.onclick= function()
    {
        var x = parseInt(localStorage.getItem("nrc"));
        if (x){
            localStorage.setItem("nrc", x + 1);
        }
        else{
            localStorage.setItem("nrc", "1");
        }
        document.getElementById("scie").value = localStorage.getItem("nrc");
    }
}
</script>
```

Numar de clickuri pe button

Click

```
<body>
  <p> Numar de clickuri pe button <input type="text" id="scie" value="0"> </p>
  <button id="bt"> Click</button>
</body>
```

Pentru a memora obiecte in localStorage se pot folosi metodele JSON.stringify si JSON.parse

```
<script type="text/javascript" >
window.onload = myMain;
function myMain() {document.getElementById('abuton').onclick= addob;
                    document.getElementById('sbuton').onclick= showob;};

function addob(){var x= parseInt(prompt("x"));
                  var y= parseInt(prompt("y"));
                  var ob = {px:x, py:[x,y]};
                  var stob=JSON.stringify(ob);
                  localStorage.setItem('obiect', stob);};

function showob() { var obst=JSON.parse(localStorage.getItem('obiect'));
                    alert(obst.py);
                    alert(typeof(obst.py[0]))}

</script>
```

```
<body>
<button type="button" id="abuton"> Add </button>
  <button type="button" id="sbuton"> Show </button>
</body>
```

HTML5 API: Geolocation

defineste obiectul navigator.geolocation
navigator.geolocation.getCurrentPosition()

```
window.onload = function () {  
document.getElementById("b1").addEventListener("click", showgeo);  
  
function showgeo () {  
if (!navigator.geolocation) { alert("Geolocation not supported"); }  
else {  
    navigator.geolocation.getCurrentPosition(  
        function(position) {  
            alert(position.coords.latitude + "/" + position.coords.longitude); } ,  
        function() { alert("Informatia nu e accesibila"); } );  
    }  
}  
}
```

Ajax

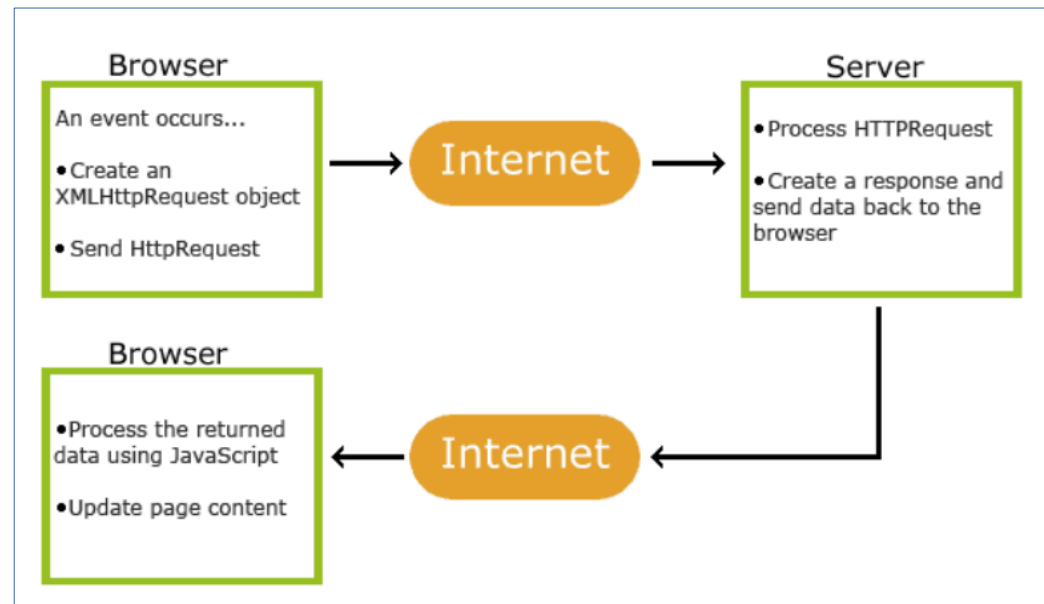
- este prescurtarea de la "Asynchronous JavaScript and XML"
- nu este o tehnologie, termenul se refera la un grup de tehnologii
- termenul a fost introdus de catre Jesse James Garrett in februarie 2005

Ajax

Permite actualizarea unor parti ale unei pagini web fără a reincarca intreaga pagina

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

Nucleul sau il reprezinta 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.

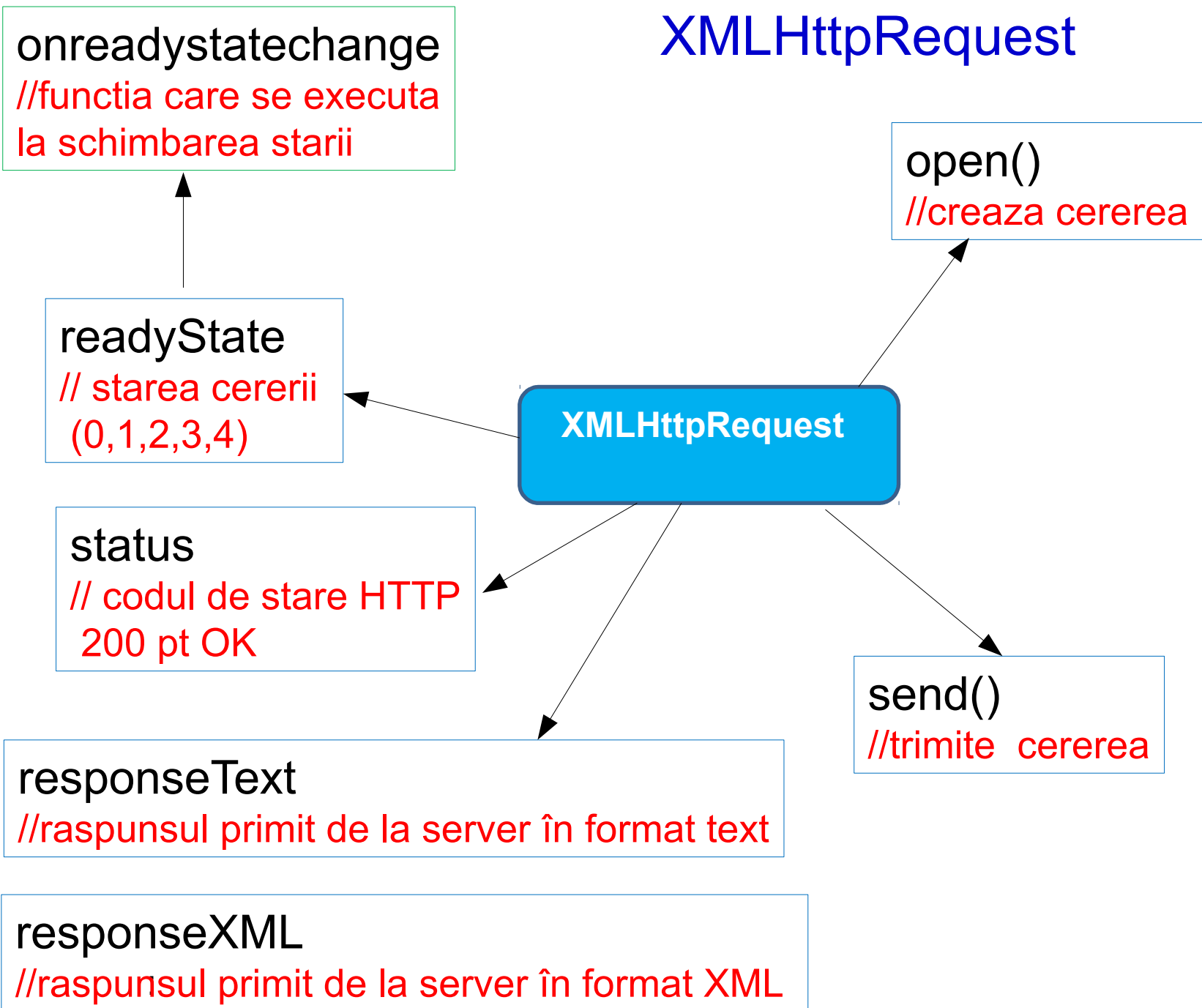
In plus, pot fi procesate in paralel mai multe conexiuni cu serverul, fara a bloca browser-ul pana la primirea raspunsului.

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



<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.');
```

```
<!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);  
  
            .....  
        }  
    }  
}
```

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) // intoarce un JSON string  
JSON.parse(text)       // transforma un string in JSON
```

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>
```

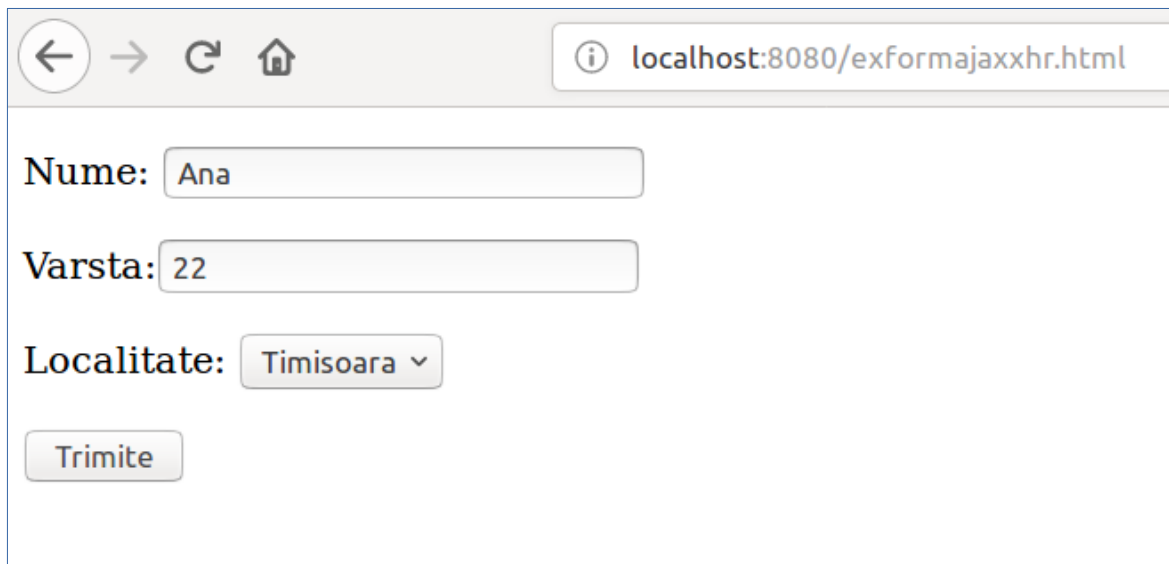
Aplicația server

```
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

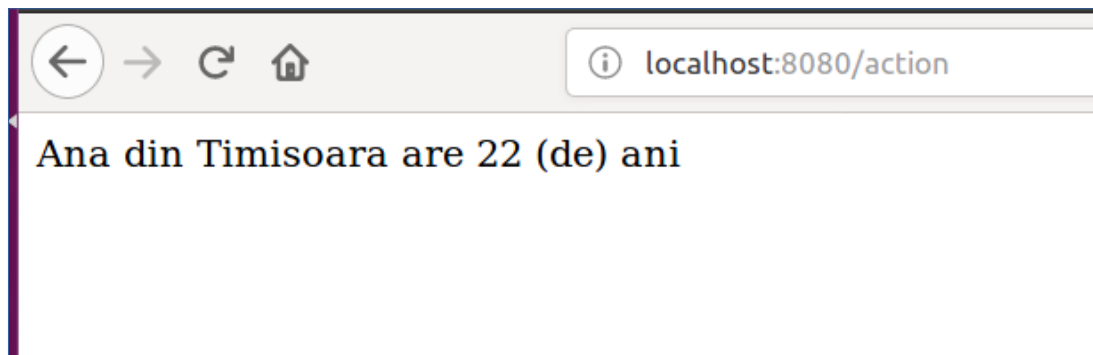


A screenshot of a web browser window. The address bar shows the URL `localhost:8080/exformajaxxhr.html`. The page contains a form with three input fields and a submit button. The first field is labeled "Nume:" and contains the text "Ana". The second field is labeled "Varsta:" and contains the number "22". The third field is labeled "Localitate:" and is a dropdown menu with "Timisoara" selected. Below the fields 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 result of the form submission: "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: ▼

Ana din Timisoara are 22 (de) ani