

# Tehnici Web

## CURSUL 8

Semestrul II, 2020-2021  
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 închiderea tabului

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

<https://www.w3.org/TR/webstorage/>

# 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 inlocuieste  
valoarea unei chei existente

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

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

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

`localStorage.propNoua=valoare`

Proprietatile și metodele sunt la fel și pentru  
`sessionStorage`

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

# Formulare

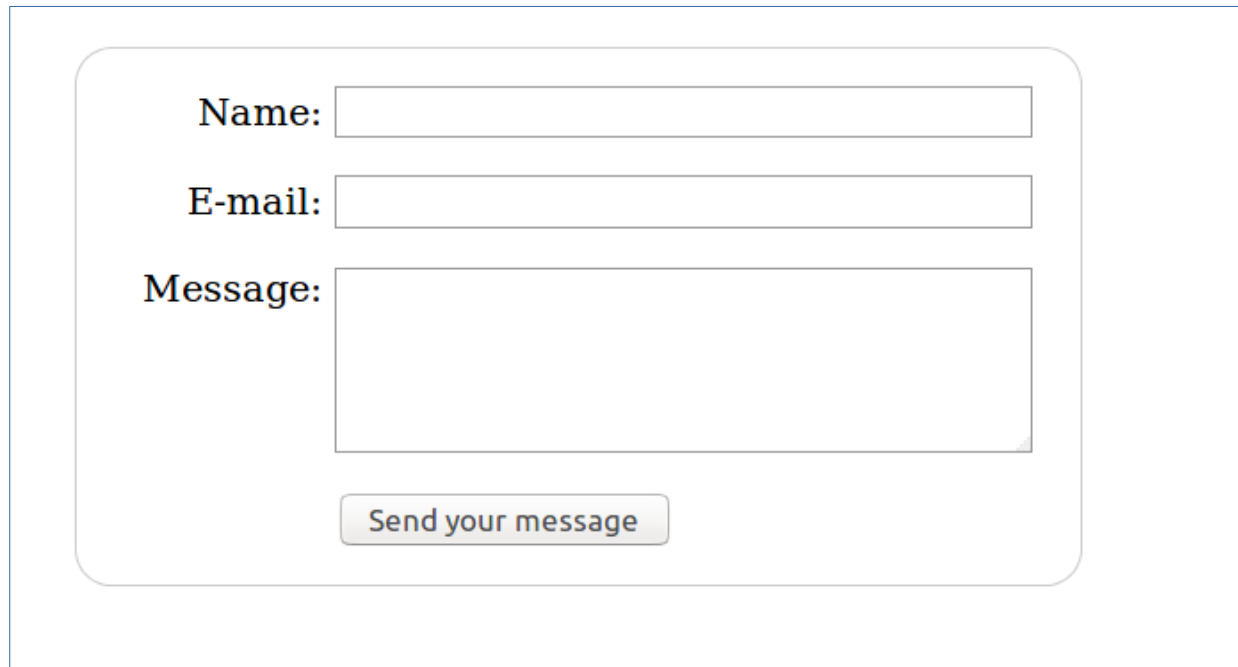
Elementul HTML `<form>` reprezintă o secțiune a documentului care conține controale interactive pentru a trimite informații către un server web.

```
<form>
```

```
<!-- se poate pune orice element HTML, dar importante  
sunt elementele speciale care creaza controale in  
browser -->
```

```
</form>
```

# Formulare



A web form with a light gray border and rounded corners. Inside, there are three input fields and a button. The first field is labeled 'Name:' and is a single-line text input. The second field is labeled 'E-mail:' and is a single-line text input. The third field is labeled 'Message:' and is a multi-line text area. Below the text area is a button labeled 'Send your message'.

Name:

E-mail:

Message:

Dupa ce forma este completata, datele sunt trimise serverului ca un **query string**.



# Forms

```
<form id="myform">
```

```
<!-- lista de elemente asociate formelor -->
```

```
<input>
```

```
<textarea>
```

```
<fieldset>
```

```
<select>
```

```
<button type="submit"> Submit </button>
```

```
</form>
```

```
<input type="submit" form="myform" value="Submit">
```

```
<button type="submit" form="myform"> Submit </button>
```

# Elemente specifice formelor: tagul **input**

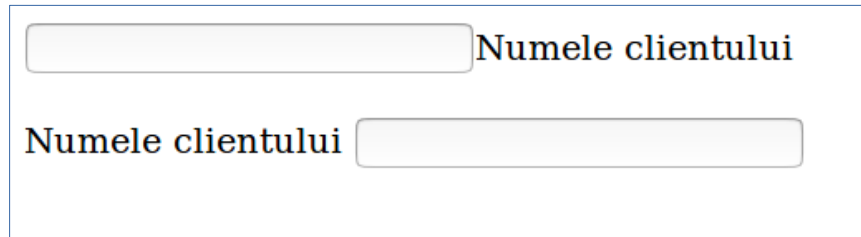
Elementul HTML **<input>** poate fi afișat în mai multe moduri, în funcție de valorile atributului **type**.

## Input Type Text

(câmp pentru introducerea unui text pe o singura linie)

### Varianta 1 (eticheta label simpla)

```
<label><input type="text" name="client">  
Numele clientului </label>
```



A screenshot of a web form element. It consists of a light gray rectangular text input field followed by the text 'Numele clientului' in a dark gray font.

### Varianta 2 (eticheta label cu atributul for)

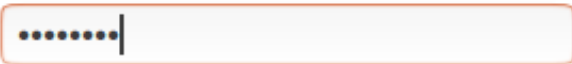
```
<label for ="user">Numele clientului </label>  
<input type="text" name="client" id="user">
```

Atribute specifice: size (width pentru câmp), maxlength, required

# Elemente specifice formelor: tagul **input**

## Input Type password

```
<label><input type="password" name="parola" maxlength="8">  
Parola</label>
```



Parola

## Input Type text cu pattern

```
<label><input type="text" id="cnp" pattern="[0-9]{13}"> CNP  
</label>
```

```
<label><input type="text" id="ci" pattern="[A-Z]{2}[0-9]{4}">C.I  
</label>
```

# Elemente specifice formelor: tagul **input**

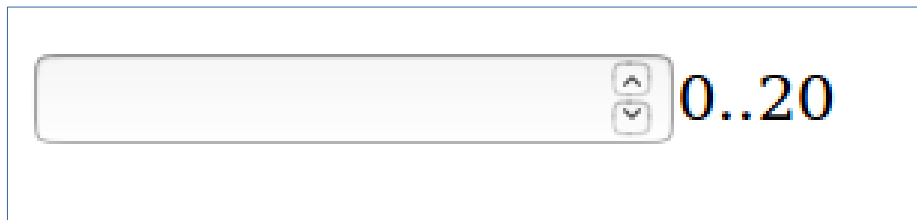
## Input Type file

`<label>Ataseaza fisier <input type="file" name="fileField"> </label>`

A screenshot of a web form element. It consists of the text "Ataseaza fisier" followed by a button labeled "Răsfoiește..." and then the text "Niciun fișier selectat." The entire element is enclosed in a light blue border.

## Input Type number

`<label><input type="number" min="0" max="20" name="numar">  
0..20 </label>`

A screenshot of a web form element. It shows a number input field with a light blue border. The field contains the text "0..20". To the right of the input field is a small button with up and down arrows. The entire element is enclosed in a light blue border.

# Elemente specifice formelor: tagul **input**

## Input Type range

<label>

```
<input type="range" min="0" max="20" step="2" name="numar">  
0..20 </label>
```



## Input Type email

```
<label><input type="email" name="adresa" required multiple>  
Email </label>
```

Atributul **multiple** funcționează doar pentru input cu type = file, email

# Elemente specifice formelor: tagul **input**

## Input Type radio

☐ iute  
☒ dulce

```
<label><input type="radio" name="gust" value="iute"> iute </label><br>
```

```
<label><input type="radio" name="gust" value="dulce" checked> dulce</label>
```

numai unul din butoanele radio poate fi selectat, de aceea atributul **name** trebuie sa aiba aceeaasi valoare pentru toate optiunile

## Input Type checkbox

```
<label><input type="checkbox" name="topping1" value="rosii"> rosii</label>
```

```
<label><input type="checkbox" name="topping2" value="branza">branza  
</label>
```

pot fi selectate simultan mai multe butoane checkbox

☒ rosii  
☒ branza

În HTML5 au fost introduse și alte valori pentru input type

<input type=

color \\ introducerea unei culori

date \\introducerea unei date în formatul  
yyyy-mm-dd

image \\creaza un buton grafic folosind o imagine

tel \\ introducerea unui numar de tel

time \\ data în formatul hh:mm

url \\ introducerea unui URL în formatul  
urlscheme://restofurl

week \\data în formatul yyyy-Www

( ex. 2017-W45)

# Elemente specifice formelor: tagul **fieldset**

Este folosit pentru gruparea elementelor in interiorul formelor

```
<fieldset>
```

```
<label><input type="radio" name="gust" value="iute"> iute </label><br>
```

```
<label><input type="radio" name="gust" value="dulce"> dulce</label>
```

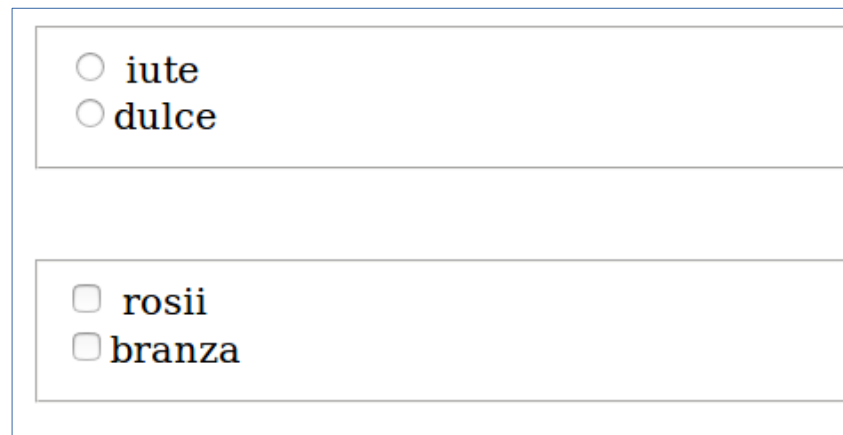
```
</fieldset>
```

```
<fieldset>
```

```
<label><input type="checkbox" name="topping1" value="rosii"> rosii</label><br>
```

```
<label><input type="checkbox" name="topping2" value="branza">branza  
</label>
```

```
</fieldset>
```



<input type="radio"/> iute <input type="radio"/> dulce
<input type="checkbox"/> rosii <input type="checkbox"/> branza



## Elemente specifice formelor: **textarea**

```
<textarea name="comments" rows="10" cols="48">  
Text initial </textarea>
```

Reprezintă un camp de editare a textului pe mai multe linii  
Implicit cols = 20

## Elemente specifice formelor: **select**

Reprezintă un control care ofera un meniu de optiuni

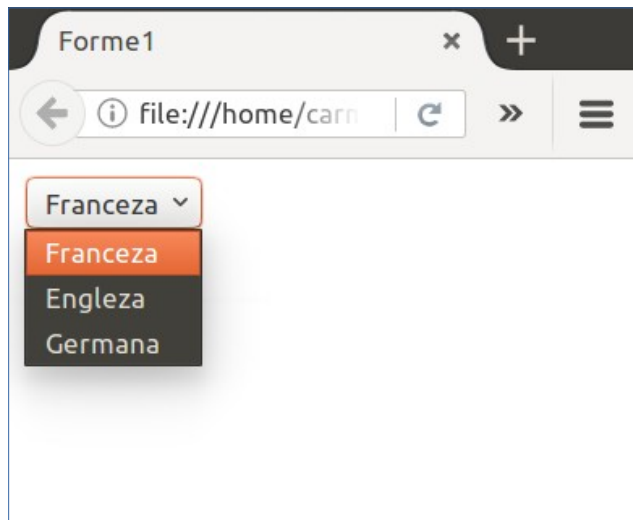
```
<select name="optional">
```

```
<option value="franceza"> Franceza </option>
```

```
<option value="engleza">Engleza </option>
```

```
<option value="germana">Germana</option>
```

```
</select>
```



# Submiterea formelor: elementele **input** si **button**

## Input Type submit

```
<input type="submit" value="Trimite">
```

## Input Type reset

```
<input type="reset" value="Reseteaza" size="12">
```

## Button Type submit

```
<button type="submit"> Trimite </button>
```

## Button Type reset

```
<button type="reset"> Reseteaza </button>
```

Elementul **button** poate fi folosit in afara formelor cu atributul `type="button"`

## Exemplu

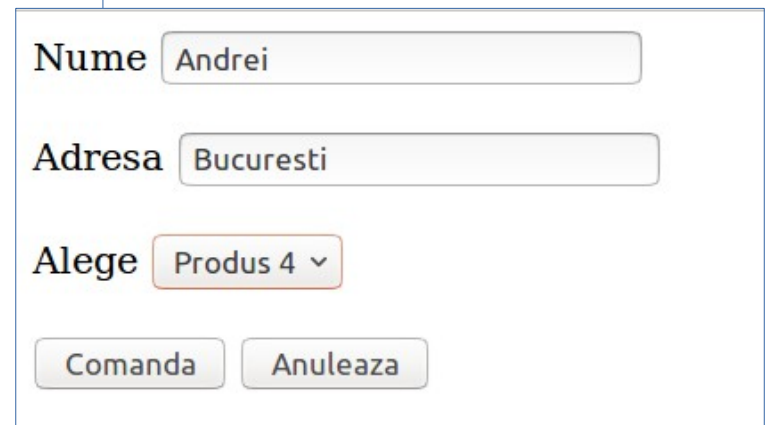
```
<form id="myform">
  <label for="nu">Nume</label>
  <input id="nu" type="text" name="nume"><br><br>

  <label for="adr">Adresa</label>
  <input id="adr" type="text" name="adresa"><br><br>

  <label for="adr">Alege</label>
  <select name="produs">
    <option value="p1">Produs 1</option>
    <option value="p2">Produs 2</option>
    <option value="p3">Produs 3</option>
    <option value="p4">Produs 4</option>
  </select><br><br>
  <button type="submit">Comanda</button>
  <input type="reset" value="Anuleaza">
</form>
```

sau în afara elementului <form>:

```
<button type="submit"
form="myform">Comanda</button>
<input type="reset" form="myform" value="Anuleaza">
```



Nume

Adresa

Alege

URSURI/EXEMPLE/form1.html?nume=Andrei&adresa=Bucuresti&produs=p4



## Atributele formelor

```
<form target="_blank" method="post" action="URL">
```

```
target = "_self" / "_blank" / "numeiframe"  
// locul in care este afisat raspunsul dupa  
submiterea formei (implicit: _self)
```

```
method = "post"/"get"  
// metoda folosita pentru comunicarea  
cu serverul (implicit: get)
```

```
action = URL-ul unui program care va procesa forma  
(implicit: URL paginii curente)
```

## Exemplu: target cu iframe

```
<form action="http://localhost:8080/cale" method="GET"
target="numeiframe">
  <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">Timisoara</option>
  </select>
<br>
<button type="submit" id="buton"> Trimite </button>
</form>
<br>
<iframe name="numeiframe"> <iframe>
</body>
```

Nume:

Varsta:

Localitate:

Buna George din Timisoara

# Atributul method = "post"/"get"

```
POST /cale HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ro-R0,ro;q=0.8,en-US;q=0.6,en-GB;q=0.4,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 33
Connection: keep-alive
Upgrade-Insecure-Requests: 1

name=George&age=20&city=Timisoara
```

POST  
Query string in corpul mesajului

```
GET /cale?name=George&age=20&city=Timisoara HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ro-R0,ro;q=0.8,en-US;q=0.6,en-GB;q=0.4,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

GET  
Query string in URL

Aplicația server  
(Node.js)

```
var http = require('http');
var url = require('url');

http.createServer(function (req, res) {
  var url_parts = url.parse(req.url, true);
  var query = url_parts.query;

  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Buna ' + query.name + ' din ' + query.city);
}).listen(8080);
```

# Atributul action cu mailto

```
<form action="mailto:mymail@yahoo.com" method="post" enctype="text/plain">  
Name:<br>  
<input type="text" name="name"><br>  
E-mail:<br>  
<input type="text" name="mail"><br>  
Comment:<br>  
<input type="text" name="comment" size="50"><br><br>  
<input type="submit" value="Send">  
<input type="reset" value="Reset">  
</form>
```

La submiterea formei  
se trimite un email

Name:

E-mail:

Comment:

De la: carmen <carmen\_stama...>

Către: mymail@yahoo.com

Subiect: Formular postat de Firefox

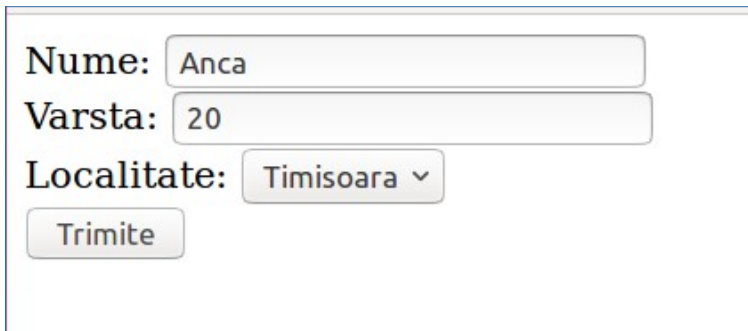
name=Carmen  
mail=carmen\_stama@yahoo.com  
comment=hello



# O forma este reprezentata în JavaScript de un obiect HTMLFormElement

proprietatea **elements** intoarce colectia campurilor formei (array-like object)

proprietatea **length** intoarce numarul de campuri



A screenshot of a web form. It contains three input fields: 'Nume:' with the value 'Anca', 'Varsta:' with the value '20', and 'Localitate:' with a dropdown menu showing 'Timisoara'. Below these fields is a button labeled 'Trimite'.

```
<script>
window.onload=function()
{
var f=document.getElementById("forma");
alert(f.elements); //[objectHTMLFormControlsCollection]
alert(f.length); // 4
alert(f.elements[0].value); //Anca
}
</script>
```

```
<form id="forma">
  <label>Nume:</label>
  <input type="text" name="name" value="Anca">
<br>
  <label> Varsta:</label>
  <input type="text" name="age" value="20">
<br>
  <label>Localitate:</label>
  <select name="city">
    <option value="Bucuresti" selected>Bucuresti</option>
    <option value="Timisoara">Timisoara</option>
  </select>
<br>
  <button type="submit" id="buton"> Trimite </button>
</form>
```

# Forme: proprietatea value

- pentru `<input type="text">` si `<textarea>` proprietatea value reprezinta valoarea introdusa de utilizator la submiterea formei si este de tipul string
- proprietatea value pentru `<option>` reprezinta valoarea optiunii
- altfel reprezinta valoarea atributului value

```
<input type="text" value="5">
```

```
<input type="submit" value="Trimite">
```

```
<input type="radio" value="v1">
```

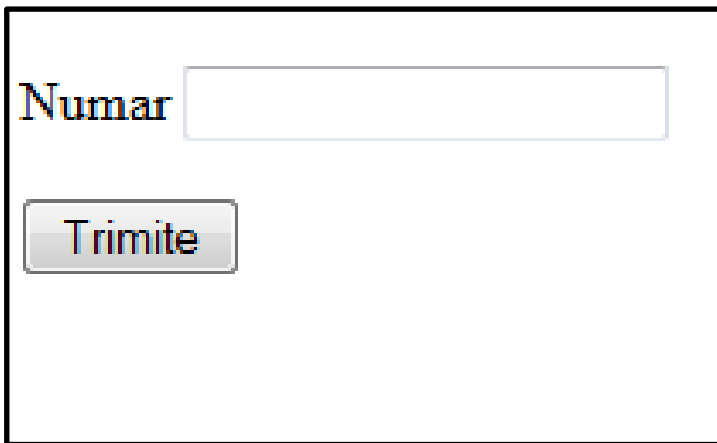
```
<input type="checkbox" value="v1">
```

# Evenimente emise pe <form>

- **submit:** evenimentul lansat la trimiterea formularului
- **reset:** evenimentul lansat când un formular este resetat  
(pentru a face reset, trebuie introdus un <input> sau <button> cu type="reset")

# Exemplu

Citesc un numar folosind  
campul input; la click pe  
buton afisez suma



Numar

```
window.onload = myMain;
```

```
function myMain() {  
  document.getElementById("citire").onsubmit  
    = suma;  
}
```

```
function suma() {  
  var x = document.getElementById("nr").value;  
  var s = 0;  
  for (var i=1; i<= parseInt(x) ; i++) s=s+i;  
  alert('Suma este ' + s);  
}
```

```
<form id="citire" >  
Numar <input type="text" id="nr" >  
<button type="submit"> Trimite </button>  
</form>
```

```
<form id="citire" >  
  Numar <input type="text" id="nr" >  
  <button type="submit"> Trimite  
</button>  
</form>
```

```
window.onload = myMain;  
  
function myMain() {  
  document.getElementById("citire").onsubmit  
    = suma;  
}  
  
function suma() {  
  var x = document.getElementById("nr").value;  
  var s = 0;  
  for (var i=1; i<= parseInt(x) ; i++) s=s+i;  
  alert('Suma este ' + s);  
}
```

## doua evenimente

Name	Target	Property	Handler
load	window	onload	myMain
submit	form	onsubmit	suma

# Forme: proprietati specifice campurilor

- proprietatea checked (boolean) pentru elemente

<input type="radio" value="v1">

<input type="checkbox" value="v1">

- proprietatile campului <select>

options array-like care contine optiunile

selectedIndex pentru determinarea

optiunii selectate

<select multiple> atributul multiple permite selectarea  
mai multor optiuni

- proprietatile campului <fieldset>

elements array-like care contine componentele

Desi formele au modalitati specifice de selectare este recomandat ca, atat formele, cat si campurile lor sa fie accesate folosind metoda `getElementById`

Exemplu: prelucrarea butoanelor radio

```
<input type="radio" name="a" id="a1" value="0">  
<input type="radio" name="a" id="a2" value="1">  
<input type="radio" name="a" id="a3" value="2">
```

```
var sel = 0;  
for (var i = 1; document.getElementById('a'+i); i++) {  
  if (document.getElementById('a'+i).checked) {sel = i; break;}  
}
```

# Exemplu - quiz

```
<form name="quiz" id="quiz">
<fieldset><legend> 3+5=? </legend>
<input type="radio" name="q1" value="0">10<br>
<input type="radio" name="q1" value="1">8 <br>
<input type="radio" name="q1" value="0">55<br>
</fieldset>

<fieldset><legend> 3*5=? </legend>
<input type="radio" name="q2" value="0">10<br>
<input type="radio" name="q2" value="1">15 <br>
<input type="radio" name="q2" value="0">55<br>
</fieldset>

<button type="submit" id="buton"> Click me </button>
</form>
```

3+5=?

☐ 10

☐ 8

☐ 55

3\*5=?

☐ 10

☐ 15

☐ 55

Click me



# Exemplu - quiz

```
window.onload = myMain;
```

```
function myMain() {  
    document.getElementById("quiz").onsubmit = totalQuiz;}
```

```
function totalQuiz() {  
    var fe = document.getElementById("quiz").elements;  
    var q1 = fe[0].elements;  
    var q2 = fe[4].elements;  
    var x=0;  
    for (var i = 0; i<q1.length; i++) if (q1[i].checked) x = x + parseInt(q1[i].value);  
    for (var i = 0; i<q2.length; i++) if (q2[i].checked) x = x + parseInt(q2[i].value);  
    alert(x);  
}
```

# Evenimente pe controale pentru formulare

## ► input

- pentru `<input>`, `<textarea>` sau orice element cu conținut editabil
- se declanșează la fiecare modificare (ex: fiecare caracter scris într-un input)

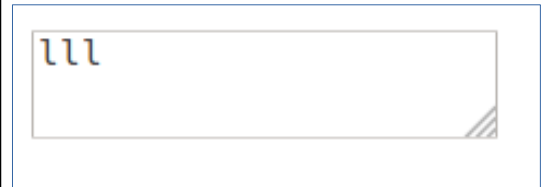
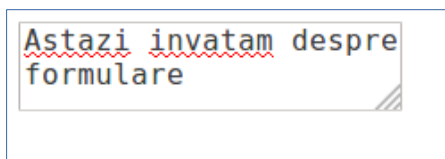
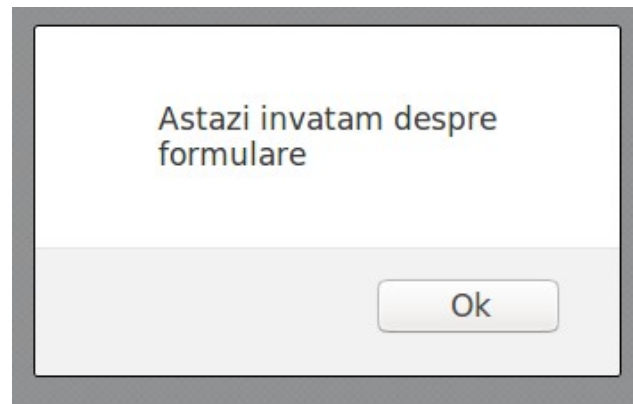
## ► change

- pentru `<input>`, `<select>`, `<textarea>`
- eveniment emis după ce o valoare nouă e aleasă
- pentru elementele `<textarea>` sau `<input type="text">` se declanșează când se scoate focus de pe element

# Evenimentul **change**

```
<head>
<script>
window.onload = function () {
    document.getElementById("tt").onchange = function () {
        alert(document.getElementById("tt").value); }
    }
</script>
</head>
<body>
    <textarea id="tt" width="50" height="20" >||| </textarea>
</body>
```

Înainte de ev. change

A small text area with a light blue border and a small diagonal line in the bottom right corner. It contains three vertical bars '|||' in a blue font.A small text area with a light blue border and a small diagonal line in the bottom right corner. It contains the text 'Astazi invatam despre formulare' in a blue font.An alert dialog box with a gray border. It has a white background with the text 'Astazi invatam despre formulare' in a blue font. At the bottom right, there is a button labeled 'Ok'.

```
<script>
window.onload=main;

function main() {
obp=document.getElementById("scie");
obsel=document.getElementById("sel");
obsel.onchange = function () {
                obp.innerHTML+=obsel.options[obsel.selectedIndex].value;
            }
}
</script>
```

```
<body>
<p id="scie">Ati selectat:</p>
<form>
    <select id="sel">
        <option> val1 </option>
        <option> val2 </option>
        <option> val3 </option>
    </select>
</form>
</body>
```

Înainte de declansarea ev. change

Ati selectat:

val1 ▾

dupa declansarea ev. change

Ati selectat: val3

val3 ▾

# Obiecte

Un obiect este o colectie de proprietati, fiecare avand *nume* si *valoare*. Proprietatile care au ca valori functii se numesc *metode*.

Prototipul unui obiect este desemnat prin `obiect.prototype`

Orice obiect mosteneste proprietatile obiectului prototip (“prototypal inheritance”).

Toate obiectele care au acelasi prototip formeaza o clasa.

Toate obiectele sunt descendenti ai obiectului generic `Object`

**`Object.getPrototypeOf()`** //prototipul obiectului specificat

# Crearea obiectelor

- obiectele literal

```
var ob = {p1: v1, p2: v2, ... ,pn: vn};
```

- cu ajutorul obiectului generic

```
var ob = new Object();  
ob.p1=v1; ob.p2=v2,....
```

- folosind o functie constructor si operatorul new

- cu metoda Object.create()



```
function student(n,g) { this.num= n;  
                        this.grupa=g;  
                        }  
  
var p1=new student("Popescu",232);  
var p2=new student("Ionescu",242);
```

# Object.create(ob)

creaza un nou obiect care are ca prototip ob

```
var interval = {mx:2, my:4,  
                apartine: function(z){  
                    return (z <= this.my) && (z >= this.mx);};  
                }; //clasa
```

```
var obi = Object.create(interval); // obiect din clasa interval  
obi.mx =5; obi.my=7; //obi suprascrie proprietatile prototipului
```

```
var intervalD= Object.create(interval);  
intervalD.apartine = function(z){return (z < this.my) && (z > this.mx)};  
                        //subclasa
```

```
var obid= Object.create(intervalD); // obiect din clasa intervalD  
obid.mx = 5;obid.my=10;
```

```
interval.valid = function(){return (this.my >= this.mx)};  
intervalD.vid = function(){return (this.mx == this.my)};
```

```
alert(obid.valid()); alert(obid.vid());
```

# Cuvantul cheie this

```
var interval = {mx: 2, my: 4,  
    apartine: function(z){  
        return (z <= this.my) && (z >= this.mx);}  
    };
```

```
var obi = Object.create(interval);  
obi.mx = 5; obi.my=7;  
alert(obi.apartine(6));
```

In interiorul unui constructor sau al unei metode asociate unui obiect, **this** se refera la obiectul curent  
altfel, **this** se refera la obiectul **window**

```
var intervalD= Object.create(interval);  
intervalD.apartine = function(z){  
    return (z < this.my) && (z > this.mx);};
```

```
var obid= Object.create(intervalD);  
obid.mx = 5; obid.my = 10;  
alert (obid.apartine(5));
```



## Crearea obiectelor folosind o functie constructor si new

```
function Interval(x, y) {  
    this.mx= x;  
    this.my= y; } // clasa
```

```
Interval.prototype.apartine = function(z){  
    return (z <= this.my) && (z >= this.mx);}
```

// metoda adaugata în prototipul obiectelor create cu functia constructor

```
var obi = new Interval(1,4); // obiect din clasa Interval
```

```
Interval.prototype.valid = function(){return (this.my >= this.mx);};
```

```
alert(obi.valid()); //true
```

# Crearea obiectelor folosind o functie constructor si new

## Definirea subclaselor

```
function Interval(x, y) {  
    this.mx= x; this.my= y; } // clasa  
  
Interval.prototype.apartine = function(z){  
    return (z <= this.my) && (z >= this.mx);}
```

```
function IntervalD(x,y) {  Interval.call(this,x,y); } //this este obiectul care  
                        se construiește
```

```
IntervalD.prototype = Object.create(Interval.prototype); //am schimbat prototipul  
                        obiectelor create cu IntervalD
```

```
IntervalD.prototype.constructor = IntervalD; //restaurez proprietatea constructor
```

```
IntervalD.prototype.apartine = function(z){  
    return (z < this.my) && (z > this.mx);};
```

```
var obid = new IntervalD(5,10);
```

**Se pot adauga proprietăți noi:** obid.mz = 1;

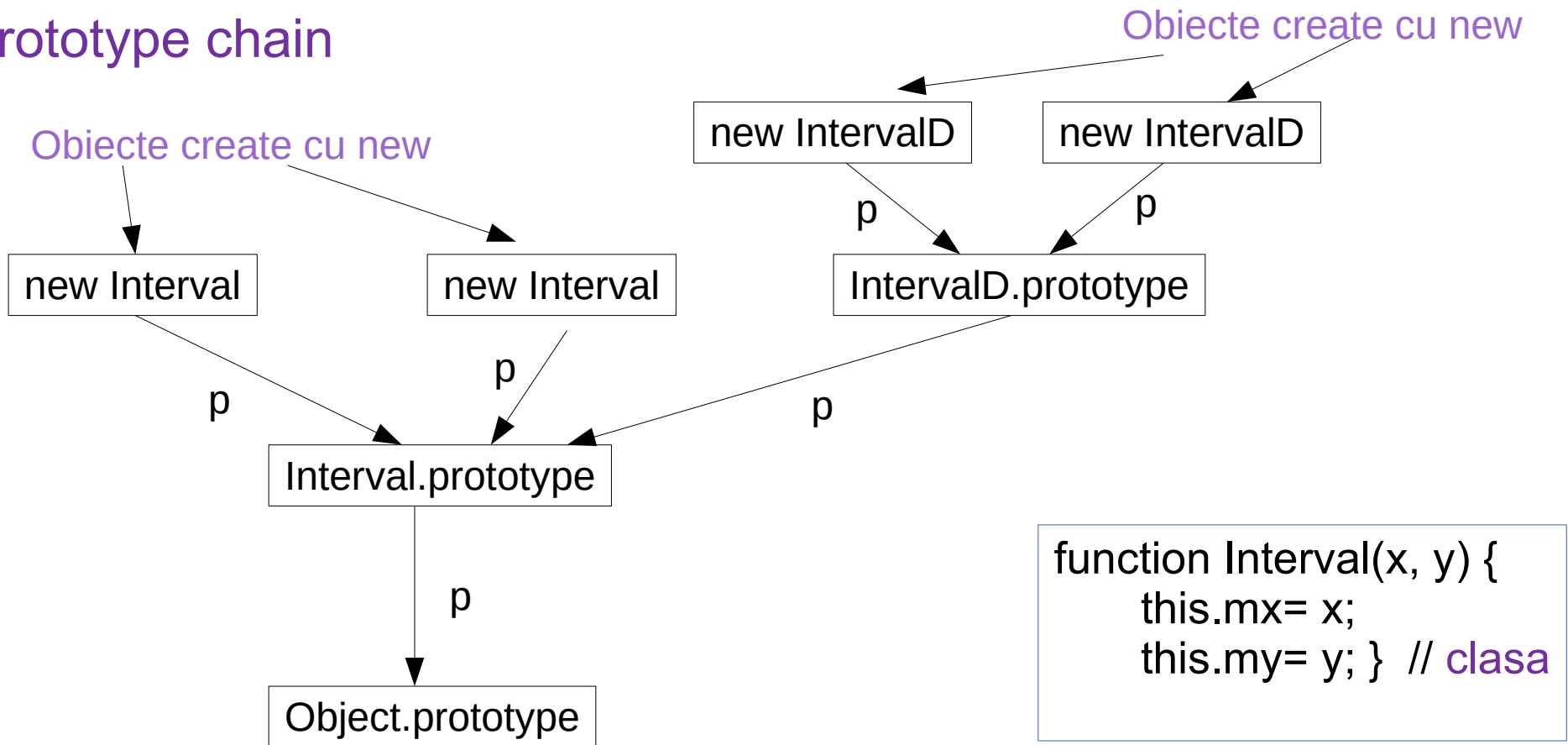
# Crearea obiectelor folosind o functie constructor si new

## Definirea subclaselor

```
function Interval(x, y) {  
    this.mx= x; this.my= y; } // clasa  
  
Interval.prototype.apartine = function(z){  
    return (z <= this.my) && (z >= this.mx);};  
Interval.prototype.valid = function(){return (this.my >= this.mx);};
```

```
function IntervalD(x,y) { Interval.call(this,x,y); }  
  
IntervalD.prototype = Object.create(Interval.prototype);  
IntervalD.prototype.constructor = IntervalD;  
IntervalD.prototype.apartine = function(z){  
    return (z < this.my) && (z > this.mx);};  
IntervalD.prototype.valid = function(){  
    return (Interval.valid.call(this) && (this.mx != this.my));  
}
```

# Prototype chain



```
function IntervalD(x,y) {Interval.call(this,x,y); }  
  
IntervalD.prototype = Object.create(Interval.prototype);  
//subclasa
```