STRUCTURA UNEI PAGINI HTML

```
<!DOCTYPE html>
<html>
<head>…</head>
<body>…</body>
</html>
```

JavaScript

```
<script type="text/javascript"> … </script>

<script type="text/javascript" src="xxx.js"></script>
```

Fisierul xxx.js poate fi creat de user.
Exista si biblioteci oficiale: jquery.min.js , prototype.js

Tipul type nu e necesar sa fie specificat, deoarece este implicit JavaScript (dar poate fi setat sa fie altul, de exemplu: "text/vbscript")

Fragmentul <script> poate fi plasat oriunde in cadrul <head> si <body>
Continut HTML poate exista doar in cadrul <body>, cu exceptia secventelor <script>

OBS: in cadrul <script> NU se scriu comenzi HTML, ci doar se proceseaza continutul HTML ; in cadrul <script> NU exista , <table> etc, ci doar comenzi JavaScript (functii). Din cadrul <script>, continut HTML nou poate fi generat doar prin comenzi specifice, de exemplu: document.write(" text "); Secventa text va fi plasata exact in acel loc in care apare (din punct de vedere secvential). In rest, doar se proceseaza si se modifica elementele HTML deja existente, in functie de numele sau eticheta lor.

EXEMPLU DE INTERACTIUNE HTML – JavaScript :

```
<!DOCTYPE html>
<html>
<body>
<div id="rezervat"></div>
<br>Ne aflam inainte de JavaScript.
<script>
var x = document.getElementById("rezervat");
x.innerHTML="click to make red";
x.onclick=function(){x.style.color="red";};
document.write("<br>Scriptul e gata.");
```

```
</script>
<br>Ne aflam in afara scriptului.
</body>
</html>
```

VARIANTA FARA <script>

```
<!DOCTYPE html>
<html>
<body>
<div id="rezervat" style="color:blue;"
onclick="this.style.color='red'">click to make red</div>
</body>
</html>
```

VARIANTA MAI COMPLEXA

```
<!DOCTYPE html>
<html>
<body>
<div style="color:blue;" onclick="this.style.color=
(this.style.color=='red')?'blue':'red';
this.innerHTML=Date()">click to show the time and to switch
the color (again and again)</div>
</body>
</html>
```

OBS: Regulile suprapunerii ghilimelelor:
1. ghilimelele de la sfarsit trebuie sa fie identice cu cele de inceput (ori ambele simple, ori ambele duble), iar orice alte ghilimele din interior trebuie sa fie de celalalt tip
2. daca se doreste neaparat acelasi tip de ghilimele peste tot, se poate folosi \" respectiv \'

OBS: Utilizarea operatorului \ :
- \' → '
- \" → "
- \\ → \
- \n → new line
- \r → carriage return
- \t → tab
- \b → backspace
- \f → form feed

OBS: Termenul this e un cuvant rezervat si face intotdeauna referire la obiectul curent. Pentru evitarea erorilor se recomanda ca in fiecare caz in parte sa se verifice daca this face intr-adevar referire la obiectul dorit (mai ales cand e vorba de programarea obiect-orientata).

In exemplele de mai sus this poate lipsi (e implicit). El e util doar cand se transmite ca parametru unei functii externe.

OBS: Pentru a afisa pe ecran instructiuni HTML (fara a fi procesate, ci doar afisate – in scop didactic, de exemplu) trebuie inlocuite caracterele:

- < cu <
- > cu >
- & cu &
- intreg fragmentul trebuie cuprins intre <pre><code> si </code></pre>

Sau se poate folosi direct:

```
<textarea disabled="true" style="border: none;background-color:white;">
    … text …
</textarea>
```

NOTIUNI DE LIMBAJ JavaScript (FARA INTERACTIUNE CU HTML)

1. INSTRUCTIUNI

IF … ELSE …

SWITCH

```
switch (day) {
case 6 : dayname = "Saturday"; break;
case 0 : dayname = "Sunday"; break;
default: dayname = "working day"; }
```

FOR

```
for ( -1- ; -2- ; -3- ) { … comenzi … }
```

OBS: Expresiile -1- -2- -3- pot fi multiple, complexe sau inexistente (dar nu si delimitatorul ;)
Daca -2- nu exista, trebuie sa existe un break in interior.
Comanda continue sare la urmatorul ciclu.

Exemplu: for (var i=0; i<10; i++)

WHILE

```
var i=0;
while (i<10) { … i++; }
```

DO … WHILE

```
var i=0;
do { … i++; }
while (i<10);
```

2. NOTIUNEA DE FUNCTIE SI DE PROGRAMARE OBIECT-ORIENTATA

DEFINIREA UNUI OBIECT

```
var person = { firstname : "John",
               lastname  : "Doe",
               age       : 50 };

                  ▲           ▲
   (PARAMETRU) NUME      VALOARE (CORESPONDENT)
```

SAU

```
var person = new Object();
    person.firstname = "John";
    person.lastname  = "Doe";
    person.age       = 50;
```

Utilizare:

```
name = person.lastname; SAU
name = person["lastname"];
```

OBS: Un obiect e un vector generalizat: in cadrul unui vector se definesc doar valorile, nu si denumirile elementelor (adica: vector[0]=… , vector[1]=… , etc – elementele sunt identificate prin ordinea lor: 0,1,2,…) , pe cand in cadrul unui obiect elementele au ele insele un nume (in exemplul dat: "firstname", "lastname", "age" – astfel incat identificarea se face prin person["lastname"], nu person[1]). Suplimentar, elementele unui obiect pot fi si functii.

Instructiunea FOR particularizata pentru obiecte:

```
var x, text="";
var person={firstname:"John", lastname:"Doe", age:50};
for (x in person) text = text + person[x];
alert(text);
```

DEFINIREA UNEI FUNCTII

- varianta simpla

→ definire directa (preferabil):

```
<script>
function myFunction(arg1,arg2)
{ … procesare date …
return [val1, val2, val3, this];}
</script>
```

OBS: return implica automat exit

Exemplul 1:

```
function myFunction() { return ("Hello!"); }
alert(myFunction())
```

Exemplul 2:

```
function myFunction(a,b) { return a*b; }
alert(myFunction(4,3))
```

→ definire ca (tip de) variabila (nerecomandat):

```
<script>
var myFunction = function()
{ … procesare date …
return [val1, val2, val3, this];};
</script>
```

OBS: Cuvantul rezervat var e optional (el reprezinta definirea locala ; fara el, definirea e globala).

→ definire combinata:

```
<script>
var xyz = function abc()
{ // xyz is visible here
  // abc is visible here};
// xyz is visible here
// abc is undefined here
</script>
```

Comparatie:

```
function xyz(){
  function abc(){};
  // abc is defined here...
}
// ...but not here
```

Din motive de browser, este optima definirea succesiva:

```
function abc(){};
var xyz = abc;
```

In acest caz xyz si abc sunt denumiri (echivalente) ale aceluiasi obiect.

Comparatie:

```
function abc(){};
alert(abc.name); // prints "abc"
```

```
var abc = function(){};
alert(abc.name); // prints ""
```

Explicatie: in cazul 2 functia e anonima

BIBLIOGRAFIE:

http://kangax.github.io/nfe/
http://ejohn.org/apps/learn/#9
http://www.permadi.com/tutorial/jsFunc/index.html
http://www.dustindiaz.com/javascript-function-declaration-ambiguity/
http://javascriptweblog.wordpress.com/2010/07/06/function-declarations-vs-function-expressions/

In urma executiei:
myFunction(arg1,arg2)[0] = val1
myFunction(arg1,arg2)[1] = val2
myFunction(arg1,arg2)[2] = val3

Denumirea obiectului curent (this) se poate adauga pentru verificarea corectitudinii.

OBS: O functie poate sa returneze doar un parametru (return val;) sau poate chiar sa nu returneze nimic, ci doar sa execute actiuni. De asemenea, poate sa nu aiba deloc parametri de intrare:
function myFunction() { … actiune … }

- varianta obiect-orientata

```
<script>

function myFunction(arg1,arg2)
{ … procesare date …
return [val1, val2, val3, this];}

var myObject = { property : "value" ,
                 make     : myFunction };

</script>
```

SAU DIRECT

```
<script>

var myObject = { property : "value" ,
                 make     : function (arg1,arg2)
                            { … procesare date …
                             return [val1,val2,val3,this];}
               };
</script>
```

OBS: Orice obiect are parametri (numiti si proprietati) si actiuni (numite si metode, proceduri). In exemplul dat, myObject are:
→ un parametru: myObject.property = "value"
→ o actiune: myObject.make(arg1,arg2) sau, echivalent, myObject["make"](arg1,arg2) ce returneaza valorile:

myObject.make(arg1,arg2)[0]=val1
myObject.make(arg1,arg2)[1]=val2

`myObject.make(arg1,arg2)[2]=val3`

- varianta generalizata (cu constructor)

```
<script>

function myObject(arg1,arg2)
{ this.property1 = "value";
  this.property2 = [this,arg1,arg2]; }

myObject.prototype =
{ make1: function() { alert("A fost apelat make1"); };
  make2: function() { return this.property2; }  };

</script>
```

Folosire:

```
var obj1 = new myObject("one","two");
var obj2 = new myObject("three","four");

obj1.make2() va returna [obj1,"one","two"]
```

EXEMPLU

```
function person(firstname, lastname, age) {
this.firstname = firstname;
this.lastname = lastname;
this.age = age;
this.changeName = changeName;
                  function changeName(name)
                  { this.lastname = name; } }
```

Utilizare:

```
var myFather = new person("John","Doe",50);
alert(myFather.firstname);
var myMother = new person("Sally","Rally",48);
myMother.changeName("Doe");
alert(myMother.lastname);
```

COMPARATIE:

```
function myDate(y,m,d)
{ this.year = y; this.month = m; this.day = d;
  return y+"-"+m+"-"+d; }

var d1 = new myDate(2010,11,30);
var d2 = myDate(2010,11,30);

typeof(d1)=object      d1.year=2010
typeof(d2)=string      d2.year=undefined    d2="2010-11-30"
```

OBSERVATIE:

Pentru orice obiect (predefinit sau definit de user) parametrul constructor returneaza obiectul parinte (tipul de obiect, functia care a generat obiectul).

Exemplu:

```
var cars = new Array("Saab","Volvo","BMW");
alert(typeof(cars))      → returneaza object
alert(cars.constructor)  → returneaza function Array()
```

Cu ajutorul prototype se pot adauga noi parametri si functii (actiuni) oricarui constructor (obiect).

Se pot adauga noi functii chiar si unor obiecte predefinite

Exemplul 1:

```
String.prototype.contains=function(it) { return this.indexOf(it) != -1; }
```

Exemplul 2:

```
Array.prototype.myUpperCase = function() {
                          for (i=0; i<this.length; i++)
                          this[i] = this[i].toUpperCase; }

var fruits=["Banana","Orange"];
fruits.myUpperCase();
alert(fruits);
```

3. TIPURI DE DATE

OBS: Orice variabila, indiferent de tipul ei, are implicit valoarea undefined. Daca este definita, atunci are valoarea null. Daca are deja

atribuita o valoare, atunci poate fi readusa la starea initiala prin comanda nume=null

OBS: Tipurile de date sunt ele insele niste obiecte predefinite.

BOOLEAN (LOGIC : TRUE / FALSE)

```
var name = new Boolean;
var name = new Boolean(1);
```

Operatori:
→ de comparatie: == !=
→ logici: && (and) || (or) ! (not)

NUMERE

- definire directa: var name = 123;
- definire ca obiect: var name = new Number;

Exemplu:

```
var x = 123;              → typeof(x)="number"
var y = new Number(123);  → typeof(y)="object"
```

PARAMETRI

```
Number.NaN                ← Not-a-Number
Number.MAX_VALUE
Number.MIN_VALUE
Number.NEGATIVE_INFINITY  ← -Infinity
Number.POSITIVE_INFINITY  ←  Infinity
```

OBS: Acesti parametri sunt statici. Nu se pot folosi decat cu obiectul Number. Cuvantul Number e implicit (nu trebuie specificat). Aceasta este o particularitate a obiectului Number (de exemplu String si Array nu o au). De asemenea, de obiectul Number apartine si biblioteca predefinita Math.

ACTIUNI

toFixed()
toPrecision()

toString()

OBS: numerele adaugate la siruri devin automat siruri (nu e nevoie de conversie explicita): "5"+5 devine automat "55"

SIRURI

- definire directa: var sir = "text";

- definire ca obiect: var sir = new String;

Exemplu:

```
var x = "John";              → typeof(x)="string"
var y = new String("John"); → typeof(y)="object"

var masina = "Volvo";
var litera = masina[2];

var text = new String("Hello!");

exemplu de parametru: var x = text.length;
exemplu de actiune  : var y = text.indexOf("el");
                      var z = text.toUpperCase();
```

ACTIUNI:

var_sir.match("xxx") returneaza null daca nu exista "xxx"
var_sir.replace("xxx","yyy") returneaza sirul modificat

var_sir.split(",") returneaza un vector

Exemplu:

```
var sir = "a bb c";
var vec = sir.split("");
```

→ vec[0]="a", vec[1]="bb", vec[2]="c"

VECTORI

- Metoda 1

```
var cars = new Array();
    cars[0] = "Saab";
    cars[1] = "Volvo";
    cars[2] = "BMW";
```

- Metoda 2

```
var cars = new Array("Saab","Volvo","BMW");
```

- Metoda 3

```
var cars = ["Saab","Volvo","BMW"];
```

OBS: Primul element al unui vector are indicele 0.

OBS: Elementele unui acelasi vector pot fi de tipuri diferite (de exemplu un element sa fie numar, iar altul sa fie sir).

```
cars.length = 3;
cars.indexOf("Volvo") = 1;
```

PARAMETRI

length

ACTIUNI

OBS: Unele actiuni modifica vectorul initial, in timp ce altele creeaza unul nou.

valueOf() → returneaza insusi vectorul
(e functia implicita a obiectului Array)

indexOf() → returneaza -1 daca nu gaseste
lastIndexOf() → cauta de la coada

join("separator") → uneste elementele intr-un sir
toString()

concat()

```
reverse()
pop()       ← dispare ultimul element (returneaza elementul)
shift()     ← dispare primul  element (returneaza elementul)
unshift("new1","new2",…)    ← adaugare la inceput
push("new1","new2",…)       ← adaugare la sfarsit

slice(de la inclusiv, pana la exclusiv) ← extract
splice(pozitie, nr. elemente sterse, "new1", "new2") ADD

sort()
```

Sortarea se poate face:

- pentru vector de siruri: sort()
- pentru vector de numere:

```
var x = [40,100,1,5,25,10];

x.sort(function(a,b){return a-b}); ← ascending
x.sort(function(a,b){return b-a}); ← descending
```

DATA

```
var time = new Date();
var hour = new Date().getHours();
var day  = new Date().getDay();

var d = new Date();
var year = d.getFullYear();
var day  = d.getDay();
var h    = d.getHours();
var m    = d.getMinutes();
var s    = d.getSeconds();
var time = d.toLocaleTimeString();
```

OBIECTELE PAGINII HTML

```
window

window.document
```

```
window.screen
window.location
window.history
window.navigator
```

(termenul "window" e implicit; nu trebuie specificat)

```
window
```

PARAMETRI

```
window.innerHeight
window.innerWidth
window.outerHeight
window.outerWidth
```

OBS: Fiecare browser in parte mai are si proprii lui parametri. De exemplu doar Opera are parametrul window.opera (aceasta e o metoda de a detecta browserul).

```
window.top
window.self
```

OBS: cuvinte rezervate corespunzatoare lui window : top , self

ACTIUNI

```
window.open()
window.close()
```

Exemple (dar nu se recomanda astfel de folosiri, deoarece browserele in general le ignora):

```
myWindow = window.open('','','width=200,height=100');
myWindow.document.write("<p>This is 'myWindow'</p>");

window.open("http://www.w3schools.com","_blank","toolbar=ye
s, location=yes, directories=no, status=no, menubar=yes,
scrollbars=yes, resizable=no, copyhistory=yes, width=400,
height=400");
```

Exemplu: se deschide un nou tab in browser, apoi e inchis

```
var myWindow = window.open();
myWindow.document.write("hello");
myWindow.close();

window.moveTo()
```

```
window.resizeTo()

window.print()
window.scrollBy()
window.scrollTo()

window.alert("mesaj") → cere Ok

window.confirm("mesaj") → true if Ok , false if Cancel

window.prompt("mesaj","text")
→ textul (re)introdus (eventual "") if Ok, null if Cancel

window.event

window.onload = functie
```

onload din JavaScript nu trebuie confundat cu cel din HTML
in HTML e suportat de :
<body>, <frame>, <frameset>, <iframe>, <img>, <input type="image">,
<link>, <style>
si are sintaxa : onload = "myFunction()" sau "direct comenzi"
Indiferent de varianta insa, onload nu reprezinta apelare de functie,
ci reincarcare a paginii.

OBS: nu e nici o deosebire intre functiile predefinite si cele definite
de user ; asa cum se poate apela direct <script> alert("mesaj");
</script> la fel se poate apela direct <script> myFunction(); </script>

```
var myVar = window.setInterval ( functie , milisecunde )
var myVar = window.setTimeout  ( functie , milisecunde )

window.clearInterval(myVar)
window.clearTimeout(myVar)

[doar cu acest scop e necesara definirea cu myVar]

functie poate sa fie:
→ function() {myFunction()} sau {direct comenzi}
→ direct doar myFunction [daca nu are parametri]
```

OBS: setTimeout , pentru a fi echivalent cu setInterval , se foloseste
in functii care se auto-apeleaza:

```
function myFunction()
{ … comenzi …
t=setTimeout (function(){myFunction()},1000); }
```

SAU

```
function myFunction()
{ … comenzi …
t=setTimeout("myFunction()",1000); }
```

SAU (daca myFunction nu are parametri)

```
function myFunction()
{ … comenzi …
t=setTimeout(myFunction,1000); }
```

OBS: clearTimeout(t); (plasat in afara lui myFunction) inceteaza executia

screen

```
screen.width
screen.height
screen.availWidth
screen.availHeight
screen.colorDepth
screen.pixelDepth
```

location

PARAMETRI

```
location.href
location.host
location.hostname
location.pathname
location.port       ← 80 sau 443
location.protocol   ← http sau https
```

ACTIUNI

```
location.assign("adresa URL")   ← se adauga la istorie
location.replace("adresa URL")  ← totodata sterge istoria
location.reload()
```

history

```
history.length
history.back()
history.forward()
history.go()
```

document

O pagina HTML ("documentul" HTML) are o structura arborescenta (parinte-copil):

```
html → header
     → body   → div → div
                    → p
              → div
              → table → tr → td
                             → td
                      → tr → td
                             → td
```

Orice element mosteneste implicit caracteristicile parintelui (si pot fi modificate pe cale explicita).

Fie o pagina HTML de genul:

```
<!DOCTYPE html>
<html>
<body>
Text. Text. Text.
</body>
</html>
```

Ceea ce este scris in cadrul <body> nu e lipsit de caracteristici (asa cum ar putea parea), ci are toate caracteristicile elementului parinte <body>: culoarea, fondul, stilul de scriere etc.

STRUCTURA UNUI ELEMENT (NOD) HTML

```
<xxx id="alfa">innerHTML</xxx>
```

xxx poate fi dintre cele predefinite (body, div, table etc) sau poate fi definit de user:

```
document.createElement("xxx");
```

OBS: nu se mai folosesc: <font>, , <frame>, <frameset>, <noframes>, <acronym>, <applet>, <basefont>, <big>, <dir>, <strike>, <tt>

Caracteristicile unui element (nod) HTML
din punct de vedere al structurii arborescente:

```
innerHTML    [acelasi lucru cu nodeValue]
nodeValue
nodeName
nodeType

parentNode
childNodes[]
firstChild = childNodes[0]
lastChild
nextSibling
previousSibling
```

node	nodeType	nodeName	nodeValue
element	1	tag name	undefined
attribute	2	attribute name	attribute value
text	3	#text	text itself
comment	8		
document	9	#document	

Oricarui element (inclusiv lui <body>) i se poate atribui un id de identificare, ce trebuie sa fie unic pe intreaga pagina HTML,astfel incat respectivul element sa poata fi procesat din cadrul unui <script>.

OBS: Similar cu *id* exista si *name* , dar se recomanda sa se evite atributul *name* , deoarece este invechit si nu se mai foloseste.

Orice element are (chiar daca nu explicit, dar ele exista) atribute (directe sau prin CSS style):

```
<body bgcolor="white">
```

accesat prin:

```
<script>
document.body.bgcolor = "white";
</script>
```

RESPECTIV

```
<body style="background-color:white;">
```

accesat prin:

```
<script>
document.body.style.backgroundColor = "white";
</script>
```

OBS: de evitat atributele directe, deoarece se tinde sa se lucreze doar prin intermediul CSS style

Forma generala:

```
<xxx id="alfa" atribut1="…" atribut2="…"
style="atribut_1:…;atribut_2:…;">innerHTML</xxx>
```

In cadrul elementului HTML stilurile sunt definite astfel:
<xxx style="nume1:valoare1; nume2:valoare2; …">
Din cadrul <script> sunt preluate astfel:
style.nume1 = … ; style.nume2 = …

OBS: in general se foloseste aceeasi denumire si in cadrul HTML si in cadrul <script> , dar exista si exceptii

PARAMETRI

```
document.domain  (www.xxx.com)
document.referrer (full link: http …)
document.URL      (full HTML link)
document.title
document.lastModified
document.cookie
```

OBS: Este absolut necesar sa se seteze o data de expirare a cookie-ului deoarece implicit cookie-ul expira (e sters) la inchiderea browserului.

Exemplul 1:

```
<!DOCTYPE html>
<html>
<head><script>

function setCookie(cname,cvalue,exdays)
{
var d = new Date();
d.setTime(d.getTime()+(exdays*24*60*60*1000));
var expires = "expires="+d.toGMTString();
document.cookie = cname+"="+cvalue+"; "+expires;
}

function getCookie(cname)
{
var name = cname + "=";
var ca = document.cookie.split(';');
for(var i=0; i<ca.length; i++)
  {
  var c = ca[i].trim();
  if (c.indexOf(name)==0) return
c.substring(name.length,c.length);
  }
return "";
}

function checkCookie()
{
var user=getCookie("username");
if (user!="")
  {
  alert("Welcome again " + user);
  }
else
  {
  user = prompt("Please enter your name:","");
  if (user!="" && user!=null)
    {
    setCookie("username",user,30);
    }
  }
}

</script></head>
```

```
<body onload="checkCookie()"></body>
</html>
```

Exemplul 2:

```
<!DOCTYPE html>
<html>
<head>
<script>

function setCookie(c_name,value,exdays)
    {
    var exdate=new Date();
    exdate.setDate(exdate.getDate() + exdays);
    var c_value=escape(value) + ((exdays==null) ? "" : ";
    expires="+exdate.toUTCString());
    document.cookie=c_name + "=" + c_value;
}

function getCookie(c_name)
{
var c_value = document.cookie;
var c_start = c_value.indexOf(" " + c_name + "=");
if (c_start == -1)
  {
  c_start = c_value.indexOf(c_name + "=");
  }
if (c_start == -1)
  {
  c_value = null;
  }
else
  {
  c_start = c_value.indexOf("=", c_start) + 1;
  var c_end = c_value.indexOf(";", c_start);
  if (c_end == -1)
  {
c_end = c_value.length;
}
c_value = unescape(c_value.substring(c_start,c_end));
}
return c_value;
}

function checkCookie()
{ var username=getCookie("username");
  if (username!=null && username!="")
```

```
  {
  alert("Welcome again " + username);
  }
else
  {
  username=prompt("Please enter your name:","");
  if (username!=null && username!="")
    {
    setCookie("username",username,365);
    }
  }
}
```

```
</script>
</head>
<body onload="checkCookie()">
</body>
</html>
```

ACTIUNI

```
document.documentElement      [se refera la <html>]
document.body                 [se refera la <body>]
document.getElementById()
document.getElementsByTagName()[]
document.getElementsByClassName()[]
document.getElementsByName()[]

document.forms[][]   ← numarul de <form>
document.images[]    ← numarul de <img>
document.links[]     ← numarul de "href" ( <a> si <area> )
document.anchors[]   ← numarul de <a>

Utilizare:

document.anchors.length
document.anchors[0].innerHTML
document.anchors[0].atribut


document.write("text")
document.writeln("text")    ← adauga un <br>
document.createElement()
document.createTextNode()
document.removeChild()
document.appendChild()
```

```
document.insertBefore(element,child)
document.replaceChild(element,child)
```

OBS: "document" poate fi orice alt element din structura arborescenta

```
document.getElementById()

document.getElementById(id).innerHTML
document.getElementById(id).atribut
document.getElementById(id).style.atribut
```

atribut poate sa fie: color, src, width, onclick etc
(cele din cadrul style pot sa aiba alte denumiri)

Exemplu:

```
<!DOCTYPE html>
<html>
<body>

<div id="main">
<div>xxx</div>
<div><p>yyy</p></div>
</div>

<script>
var x=document.getElementById("main");
var y=x.getElementsByTagName("div");
alert("Numar total: "+y.length);
alert("Continut: "+y[1].innerHTML);
</script>

</body>
</html>
```

Se va afisa : Continut: yyy

LISTA ELEMENTELOR HTML

ELEMENTE (TAGURI)

OBS: Elementele care au inchidere (de exemplu cum e …) trebuie inchise, pentru ca altfel duc la erori in cadrul structurii arborescente a paginii (in cel mai fericit caz sunt ignorate).

```
<body>
<iframe>
<a>
<div>
<div style="width:199px; height:99px; border:1px solid">
<table>
<form>
<input>
<button>   ← de evitat ; se foloseste <input> in schimb
```

OBS: Elementul <button> e bine sa fie evitat (mai ales in formulare) deoarece browserele reactioneaza diferit la interactiunea cu el. E de preferat sa se foloseasca <input> in schimb. Singura aplicatie practica a elementului <button> e cea de infrumusetare a designului, prin faptul ca da aspectul de buton. In acest scop se foloseste fara nici un atribut: <button> text, imagine etc </button>. Atributul onclick (sau altul din aceeasi familie) se va plasa asupra textului, imaginii etc. De exemplu: <button>text</button>

OBS: Redirectionarea catre un punct al unei pagini (eventual cea curenta) se face astfel: <a href="…/pagina.html#id"> respectiv <a href="#id"> , iar in cadrul paginii destinatie trebuie sa existe un element cu acel id (de exemplu …)

Maparea unei imagini

```
<img src ="planets.gif" width ="145" height ="126"
usemap="#planetmap">

<map name="planetmap">
<area shape ="rect" coords ="0,0,82,126"
onmouseover="myFunction1()" href ="sun.htm">

<area shape ="circle" coords ="90,58,3"
onmouseover="myFunction2()" href ="mercur.htm">
</map>
```

Realizarea unui IFRAME

```
<!DOCTYPE html>
<html>
<body>

<iframe id="myframe" name="iframe_a" src="demo_iframe.asp"
```

```
width="250" height="200">
<p>Your browser does not support iframes.</p>
</iframe>

<p>Click the button to change the background color of the
document contained in the iframe.</p>

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

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
var x = document.getElementById("myframe");
var y = (x.contentWindow || x.contentDocument);
if (y.document) y = y.document;
{ y.body.style.backgroundColor="red";
y.body.style.color="blue"; } }
</script>

</body>
</html>
```

ATRIBUTE DIRECTE

href

document.getElementById(anchor_id).search

returneaza sirul de parametri (?x1=val&x2=val) al unui link (numit querystring)

document.getElementById(anchor_id).hash

returneaza referinta din cadrul paginii (#eticheta)

onload ← la incarcarea paginii (sau a unei imagini)
onunload ← la parasirea paginii

onresize

onfocus ← la intrarea in camp
onblur ← la parasirea campului
onchange ← la parasirea campului

onmousemove

onmouseover
onmouseout
onmousedown
onmouseup

OBS: event.button ,
event.clientX , event.clientY , event.screenX , event.screenY

onkeydown ← la apasarea unei taste
onkeypress ← la apasarea unei taste
onkeyup ← la eliberarea tastei

OBS: event.keyCode , event.shiftKey

OBS: event e un cuvant rezervat , asa cum e si this
event.type returneaza tipul de eveniment ce a avut loc

onselect ← la selectarea unui fragment de text

onsubmit
onreset
ondblclick
onclick

OBS: in cadrul HTML are sintaxa onclick="myFunction()" sau "direct comenzi", iar in cadrul JavaScript e accesat prin onclick=functie

Exemplu:
document.getElementById(id).onclick =
function(){alert("mesaj");}

OBS: chiar daca elementul id nu are specificat explicit onclick, acesta este adaugat automat ; mai mult decat atat, onclick poate exista nu doar pentru butoane, ci pentru orice element (img, div etc)

Exemplu:

```
<!DOCTYPE html>
<html>
<head>
<script>
function whichElement(e)
{
var targ;
if (!e)
  {
  var e=window.event;
  }
```

```
if (e.target)
  {
  targ=e.target;
  }
else if (e.srcElement)
  {
  targ=e.srcElement;
  }
var tname;
tname=targ.tagName;
alert("You clicked on a " + tname + " element.");
}
</script>
</head>
<body onmousedown="whichElement(event)">

<p>Click somewhere in the document. An alert box will alert
the name of the element you clicked on.</p>
<h3>This is a heading</h3>
<img border="0" src="smiley.gif" alt="Smiley" width="32"
height="32">
<p>This is a paragraph.</p>

</body>
</html>
```

Exemplu:

```
<!DOCTYPE html>
<html>
<head>
<script>
function WhichButton(event)
{
alert("You pressed button: " + event.button)
}
</script>
</head>
<body>

<div onmousedown="WhichButton(event);">Click this text
(with one of your mouse-buttons)
<p>
0    Specifies the left mouse-button<br>
1    Specifies the middle mouse-button<br>
2    Specifies the right mouse-button</p>
```

```
<p><strong>Note:</strong> Internet Explorer 8, and earlier,
returns another result:<br>
1    Specifies the left mouse-button<br>
4    Specifies the middle mouse-button<br>
2    Specifies the right mouse-button</p>

</div>
</body>
</html>
```

Exemplu:

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction(e)
{
x=e.clientX;
y=e.clientY;
coor="Coordinates: (" + x + "," + y + ")";
document.getElementById("demo").innerHTML=coor
}

function clearCoor()
{
document.getElementById("demo").innerHTML="";
}
</script>
</head>
<body style="margin:0px;">

<div id="coordiv" style="width:199px;height:99px;border:1px
solid" onmousemove="myFunction(event)"
onmouseout="clearCoor()"></div>

<p>Mouse over the rectangle above, and get the coordinates
of your mouse pointer.</p>

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

</body>
</html>
```

ATRIBUTE PRIN CSS STYLE

```
style.color
style.backgroundColor
style.background
style.fontFamily
style.fontSize
style.visibility = "hidden|visible"
style.display = "none|block|inline|inherit"
style.cursor = "default|help|move|pointer|progress|url()…"
```

TABELE

```
<table style="width:300px; height:100px">
<table style="border:1px solid black; padding:5px">
```

```
document.getElementById(table_id).rows[0].innerHTML

returneaza continutul primului rand

document.getElementById(table_id).rows[0].cells[0].innerHTML

returneaza continutul primei casute de pe primul rand
```

```
var x = document.getElementById(table_id).rows  ← e un vector

var y = x[0].cells                              ← e un vector
```

ACTIUNI:

```
createCaption()  ← are innerHTML
deleteRow()
insertRow()      ← are innerHTML
insertCell()     ← are innerHTML
```

Exemplu:

```
<!DOCTYPE html>
<html>
<head>
<script>
function deleteRow(r)
{
var i=r.parentNode.parentNode.rowIndex;
document.getElementById('myTable').deleteRow(i);
}
```

```
</script>
</head>
<body>

<table id="myTable" border="1">
<tr>
  <td>Row 1</td>
  <td><input type="button" value="Delete"
onclick="deleteRow(this)"></td>
</tr>
<tr>
  <td>Row 2</td>
  <td><input type="button" value="Delete"
onclick="deleteRow(this)"></td>
</tr>
<tr>
  <td>Row 3</td>
  <td><input type="button" value="Delete"
onclick="deleteRow(this)"></td>
</tr>
</table>

</body>
</html>
```

FORMULARE

```
<form accept-charset="UTF-8"></form> ← are length si elements
<input>
<button></button>
<select><option></option></select>  ← are length si options
                                      (iar optiunile au text)

<option value="xxx">text</option>


document.getElementById(camp_id).disabled = true / false

dezactiveaza un camp

document.getElementById(select_id).multiple = true / false

permite selectare multipla a optiunilor
```

document.getElementById(select_id). selectedIndex

returneaza numarul optiunii selectate

document.getElementById(select_id).remove(index)

elimina o optiune

document.getElementById(camp_id).form.atribut

acceseaza formularul de care apartine un camp

document.getElementById(form_id).length

returneaza numarul de campuri al unui formular

document.getElementById(form_id).submit()

proceseaza un formular

document.getElementById(form_id).reset()

reseteaza campurile unui formular

Exemplu:

```
<!DOCTYPE html>
<html>
<body>

Nume: <input type="text" onchange="literemari(this)"
onfocus="fondgalben(this)">

Prenume: <input type="text" onchange="literemari(this)"
onfocus="fondgalben(this)">

<script>
function literemari(obj)
{obj.value=obj.value.toUpperCase();}

function fondgalben(obj)
{obj.style.background="yellow";}
</script>

</body>
```

```
</html>
```

Exemplu:

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction(obj)
{obj.value=obj.value.toUpperCase();}
</script>
</head>
<body>

Enter your name: <input type="text"
onkeyup="myFunction(this)">

</body>
</html>
```

Exemplu:

```
<!DOCTYPE html>
<html>
<body>

Introduceti un numar:
<input id="id1" type="text">
<input type="button" onclick="verificare()" value="OK">

<script>
function verificare() {
var x=document.getElementById("id1").value;
if ((x=="")||isNaN(x)||(x.charAt(0)=="0"))
{ alert("Nu ati introdus un numar.");
document.getElementById("id1").value=""; }
else alert("OK"); }
</script>

</body>
</html>
```

Exemplu:

```
<!DOCTYPE html>
<html>
<body>

<form id="formular" action="action.aspx">
Nume:
<input type="text" name="nume" value="Donald"><br>
Prenume:
<input type="text" name="prenume" value="Duck"><br>
<input type="submit" value="Executa aplicatia">
</form>

<input type="button" value=" Numele introdus "
onclick="alert(procesare());">

<script>
function procesare() {
var x=document.getElementById("formular");
var txt="";
for (var i=0; i<x.length-1; i++)
txt=txt+x.elements[i].value+" ";
return txt;}
</script>


</body>
</html>
```

Exemplu:

```
<!DOCTYPE html>
<html>
<head>

<script>
function validateForm() {
var x=document.forms["myForm"]["fname"].value;
if (x==null || x=="")
  { alert("First name must be filled out");
    return false; }}
</script>

</head>
<body>
```

```
<form name="myForm" action="demo_form.asp"
onsubmit="return validateForm()" method="post">
First name:
<input type="text" name="fname">
<input type="submit" value="Submit">
</form>

</body>
</html>
```

OBS: document.forms["myForm"]["fname"].value este echivalent cu document.forms["myForm"].fname.value
De asemenea: fname poate fi definit atat prin name, cat si prin id

Exemplu:

```
<!DOCTYPE html>
<html>
<head>
<script>
function disable()
{document.getElementById("mySelect").disabled=true;}

function enable()
{document.getElementById("mySelect").disabled=false;}
</script>
</head>
<body>

<form id="myForm">
<select id="mySelect">
  <option>Apple</option>
  <option>Pear</option>
  <option>Banana</option>
  <option>Orange</option>
</select>
<br><br>
<input type="button" onclick="disable()" value="Disable
list">
<input type="button" onclick="enable()" value="Enable
list">
</form>

</body>
</html>
```

VERIFICAREA PREALABILA A DATELOR INTRODUSE INTR-UN FORMULAR PRIN METODA FALSULUI SUBMIT

PAG1.PHP

```
<!DOCTYPE html>
<html>
<body>


<form id="f1" method="post" action="pag2.php">
<input id="alfa" type="text" name="alfa">
<input id="fake" type="button" value="OK"
onclick="verify()">
</form>



<script>
function verify() {
var alfa=document.getElementById("alfa").value;
alert(alfa);
if ((alfa=="")||(alfa==null)) { alert("retry"); }
else { alert("ok");

var elem = document.getElementById("fake");
    elem.parentNode.removeChild(elem);

document.getElementById("f1").submit();

 }
}
</script>


</body>
</html>
```

PAG2.PHP

```
<?php

$alfa=$_POST["alfa"];

echo $alfa;
```

```
?>
```

OBS: Similar se pot transmite variabile prin $_POST printr-un formular ascuns si cu valori implicite.

CSS

```
a:link {color:red]}
a:not([href]) {color:red;}
```

PHP

FILOZOFIA INTERACTIUNII PHP – JavaScript

PAS 1 : PHP genereaza pachetul HTML – JavaScript pe baza parametrilor de intrare (cei furnizati in adresa URL a paginii) ; dupa ce pagina s-a terminat de incarcat, PHP-ul nu mai exista (el e viu doar in clipa incarcarii paginii)

PAS 2 : userul interactioneaza cu pagina HTML prin intermediul FORMULARELOR Java-Script (Java-Script e mort doar in clipa refresh-ului , in rest e permanent activ)

PAS 3 : RELOAD

Varianta 1 : din cadrul Java-Script se reincarca pagina (sau o alta pagina) cu noii parametri de intrare furnizati de user (adaugati in adresa linkului prin intermediul operatorilor ? si &) si se reia pasul 1 (PHP-ul reinviaza in momentul refresh-ului si preia datele din adresa linkului prin intermediul variabilei $_GET)

Varianta 2 : orice <FORM> are un atribut ACTION care specifica ce pagina sa se incarce si cu ce parametri (transmisi prin intermediul variabilei $_REQUEST), in momentul in care se apasa SUBMIT

www.php.net

http://www.microsoft.com/web/webmatrix/

http://stackoverflow.com/questions/4679756/show-a-pdf-files-in-users-browser-via-php-perl

http://stackoverflow.com/questions/19737051/filling-out-a-pdf-form-and-submitting-to-the-server-via-php?rq=1

OBS: JavaScript e caseSensitive , pe cand in PHP doar variabilele sunt

```
<?php   …    ?>

<?php include "xxx.php"; ?>

<?php require "xxx.php"; ?>
```

Variabilele sunt marcate prin $
Ca sa fie vazuta din cadrul HTML si JS , o variabila trebuie introdusa intre acolade sau prin intermediul json

OBS: Afisarea se poate face cu echo sau print. Pentru a "pasiva" / "ignora" elementele HTML in afisare, se foloseste functia htmlspecialchars()

Concatenarea e reprezentata prin punct , nu prin + ca in JS

```
$color="red";
echo "red";
echo $color;
echo "{$color}";
echo "My car is {$color} <br>";
echo "My car is " . $color . "<br>";
echo "My car is " , $color , "<br>";
```

OBS: A nu se confunda ultimele doua comenzi! In primul caz, functia echo are un singur parametru, obtinut prin concatenare. In al doilea caz, nu mai e vorba de nici o concatenare, ci pur si simplu functia echo are trei parametri (afisati succesiv, astfel incat se lasa impresia de concatenare).

Definirea unei functii:

```
function sum($x,$y) {
$z=$x+$y;
return $z; }
```

OBS:O functie poate avea parametri cu valori implicite (in caz ca e apelata fara parametri):

```
function myFunction($arg=10) { … }
myFunction(0);   → $arg=0
myFunction();    → $arg=10 (implicit)
```

OBS: O variabila declarata in afara oricarei functii se numeste globala si este vizibila doar in afara oricarei functii, iar una declarata in cadrul unei functii e vizibila doar in cadrul acelei functii. Astfel, in cadrul unor functii diferite se pot folosi variabile cu acelasi nume (deoarece nu se suprapun una peste alta). Pentru ca o variabila globala sa fie vizibila si in cadrul unei functii, ea trebuie preluata prin intermediul directivei global

```
<?php
$x=5; $y=7;
function myFunction() { global $x,$y; $z=$x+$y; … }
myFunction();
?>
```

OBS: PHP stocheaza toate variabilele globale in vectorul $GLOBALS
Prin intermediul $GLOBALS['x'] , unde $x e o variabila globala , se poate accesa / modifica valoarea lui $x in interiorul oricarei functii.

OBS: Variabilele definite prin directiva static (in interiorul unei functii) sunt pastrate (in interiorul acelei functii) chiar si dupa terminarea functiei. Mai mult decat atat, sunt pastrate cu ultima lor valoare, nu cu cea de initializare. Ele nu vor mai fi niciodata reinitializate (la urmatoarea apelare a functiei, ele vor porni cu valoarea de iesire a executiei anterioare).

```
function myTest() {
static $x=0;
echo $x;
$x++; }

myTest(); echo "<br>";
myTest(); echo "<br>";
```

Se obtine: 0 1 2

TIPURI DE DATE

OBS: Tipurile de date ale PHP sunt:
String, Integer, Floating point numbers, Boolean, Array, Object, NULL
Dar, spre deosebire de JS, aceste tipuri sunt doar principiale. Ele nu se definesc in mod explicit (nu se poate declara o variabila de un anume tip). Functia predefinita var_dump() returneaza tipul variabilei.

OBS: Variabilele pot fi "golite" prin NULL : $x=3; $x=null;

Definirea unei constante:

```
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
```

Variabila sir:

OBS: Sirurile pot fi cuprinse intre ghilimele duble sau simple (important e sa fie de acelasi tip).

strlen()
strpos() ← in caz ca nu gaseste, returneaza false

Variabila vector:

```
$cars = array("Volvo","BMW","Toyota");
$cars[0] = "Volvo";
count($cars) → returneaza numarul de elemente
```

OBS: In mod implicit, elementele unui vector sunt identificate prin pozitia lor (0,1,2,…), dar li se pot atribui si denumiri (numite keys):

```
$x = array("a" => "red", "b" => "green");
$x["a"] = "red";
```

OBS: Concatenarea vectorilor se face prin adunare +

Functiile de sortare a unui vector:

- pentru vectori cu elemente numarate: sort() , rsort()
- pentru vectori cu elemente denumite:
  - → dupa denumire: ksort() , krsort()
  - → dupa valoare : asort() , arsort()

Vectori de vectori (vectori multidimensionali, matrici):

```
$cars = array ( array("Volvo",100,96),
                array("BMW",60,59),
                array("Toyota",110,100) );

$cars[0][0] = "Volvo";
```

```
$families = array ( "Griffin" => array ("Peter","Lois","Megan"),
                    "Quagmire" => array ("Glenn"),
                      "Brown" => array ("Cleveland","Loretta") );

$families['Griffin'][2] = "Megan";
```

METODA DE A TRIMITE VECTORI PRIN FORMULAR

```
$postvalue=array("a","b","c");
foreach($postvalue as $value)
{
  echo '<input type="hidden" name="result[]" value="'. $value. '">';
}
```

and you will got `$_POST['result']` as array

```
print_r($_POST['result']);
```

Pentru a transmite si denumirile elementelor: result[$key]

SAU

```
<input type="hidden" name="result" value="<?php foreach($postvalue as
$value) echo $postvalue.","; ?>">
```

Either serialize:

```
$postvalue=array("a","b","c");
<input type="hidden" name="result" value="<?php echo
serialize($postvalue); ?>">
```

on receive: `unserialize($_POST['result'])`
or implode:

```
$postvalue=array("a","b","c");
<input type="hidden" name="result" value="<?php echo implode(',',
$postvalue); ?>">
```

on receive: `explode(',', $_POST['result'])`

De asemenea se poate folosi si $_SESSION

Variabila obiect:

Orice variabila obiect e definita de un constructor (clasa)
Inainte de a se defini obiectul, trebuie definita clasa.
Variabila obiect se mai numeste si instanta a acelei clase.

Orice obiect are parametri si functii. Apelarea / accesarea unui parametru sau functie din cadrul unui obiect se face cu operatorul ->

Exemplu: $myObj -> parametru
$myObj -> functie()

OBS: O clasa, pe langa proprii ei parametri si functii, poate sa preia si parametrii si functiile altei clase:

class myClass extends myInitialClass

```
<?php
class Car
{   var $color;
    function Car($color="green") { $this->color = $color; }
    function what_color() { return $this->color; } }

function print_vars($obj) {
   foreach (get_object_vars($obj) as $prop => $val)
   { echo "$prop = $val"; } }

$herbie = new Car("white");
echo "Properties of herbie:";
print_vars($herbie);
?>
```

Se afiseaza: color = white

INSTRUCTIUNI

Comanda IF

if (conditie) { … }

else { … }

OBS: Gruparea else if se scrie intotdeauna legat.

Comanda SWITCH

```
<?php
$favcolor="red";
switch ($favcolor) {
case "red"  : echo "Your favorite color is red!";   break;
case "blue" : echo "Your favorite color is blue!";  break;
case "green": echo "Your favorite color is green!"; break;
default:      echo "none"; }
?>
```

Comanda FOR

```
for ($x=0; $x<=10; $x++) { echo "The number is: $x <br>"; }
```

Pentru parcurgerea elementelor unui vector se foloseste:

```
<?php
$colors = array("red","green","blue","yellow");
foreach ($colors as $value) { echo "$value <br>"; }
?>
```

SAU (pentru vectori cu elemente denumite):

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");

foreach($age as $x=>$x_value)
{ echo "Key=" . $x . ", Value=" . $x_value . "<br>"; }
?>
```

Comanda WHILE

while (conditie) { … }

SAU

do { … } while (condition);

FUNCTII PREDEFINITE

date("Y.m.d")

FUNCTIA DE TRIMITERE EMAIL

mail(to,subject,message,headers,parameters)

```
<?php
$to = "someone@example.com";
$subject = "Test mail";
$message = "Hello! This is a simple email message.";
$from = "someonelse@example.com";
$headers = "From:" . $from;
mail($to,$subject,$message,$headers);
echo "Mail Sent.";
?>
```

FUNCTII DE FILTRARE / VERIFICARE A VARIABILELOR

filter_var()
filter_var_array()
filter_input
filter_input_array

```
<?php
$int = 123;

if(!filter_var($int, FILTER_VALIDATE_INT))
  {
  echo("Integer is not valid");
  }
else
  {
  echo("Integer is valid");
  }
?>
```

```
<?php
$var=300;
```

```
$int_options = array(
"options"=>array
  (
  "min_range"=>0,
  "max_range"=>256
  )
);

if(!filter_var($var, FILTER_VALIDATE_INT, $int_options))
  {
  echo("Integer is not valid");
  }
else
  {
  echo("Integer is valid");
  }
?>
```

```
<?php
if(!filter_has_var(INPUT_GET, "email"))
  {
  echo("Input type does not exist");
  }
else
  {
  if (!filter_input(INPUT_GET, "email", FILTER_VALIDATE_EMAIL))
    {
    echo "E-Mail is not valid";
    }
  else
    {
    echo "E-Mail is valid";
    }
  }
?>
```

```
<?php
if(!filter_has_var(INPUT_POST, "url"))
  {
  echo("Input type does not exist");
  }
else
  {
  $url = filter_input(INPUT_POST,
  "url", FILTER_SANITIZE_URL);
  }
?>
```

```php
<?php
$filters = array
  (
  "name" => array
    (
    "filter"=>FILTER_SANITIZE_STRING
    ),
  "age" => array
    (
    "filter"=>FILTER_VALIDATE_INT,
    "options"=>array
      (
      "min_range"=>1,
      "max_range"=>120
      )
    ),
  "email"=> FILTER_VALIDATE_EMAIL
  );

$result = filter_input_array(INPUT_GET, $filters);

if (!$result["age"])
  {
  echo("Age must be a number between 1 and 120.<br>");
  }
elseif(!$result["email"])
  {
  echo("E-Mail is not valid.<br>");
  }
else
  {
  echo("User input is valid");
  }
?>
```

Exemplu de varianta definita de user:

```php
<?php
function convertSpace($string)
{
return str_replace("_", " ", $string);
}

$string = "Peter_is_a_great_guy!";

echo filter_var($string, FILTER_CALLBACK,
array("options"=>"convertSpace"));
?>
```

VARIABILELE SUPERGLOBALE

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

```
$_SERVER['PHP_SELF']     → adresa relativa a paginii /x.php
$_SERVER['SCRIPT_NAME'] → adresa relativa a paginii /x.php
$_SERVER['SERVER_NAME'] → adresa principala www.xxx.com
$_SERVER['HTTP_HOST']    → adresa principala www.xxx.com
$_SERVER['HTTP_REFERER'] → adresa completa a paginii
$_SERVER['SCRIPT_FILENAME']

$_SERVER['SERVER_ADDR']   → IP
$_SERVER['REMOTE_ADDR']   → IP

$_SERVER['REQUEST_METHOD'] → GET sau POST
$_SERVER['QUERY_STRING']
```

$_REQUEST, $_POST, $_GET are used to collect data after submitting an HTML form.
$_POST is also widely used to pass variables.
$_GET can also collect data sent in the URL (querystring).

`$_REQUEST`, by default, contains the contents of `$_GET`, `$_POST` and `$_COOKIE`.

```php
$temp = $_REQUEST['s'];
```

OR

```php
if (isset($_GET['s'])) {
  $temp = $_GET['s'];
}
else {
  $temp = $_POST['s'];
}
```

EXEMPLUL 1

```
<!DOCTYPE html>
<html>
<body>

<form method="post" action="<?php echo
$_SERVER['PHP_SELF'];?>">
Name: <input type="text" name="fname">
<input type="submit">
</form>

<?php
$name = $_REQUEST['fname'];
echo $name;
?>

</body>
</html>
```

EXEMPLUL 2

PHP Form Validation Example

* required field.

Name: *

E-mail: *

Website:

Comment:

Gender: Female Male *

Submit

Your Input:

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST")
{

   if (empty($_POST["name"]))
```

```
    {$nameErr = "Name is required";}
  else
    {$name = test_input($_POST["name"]);}

  if (empty($_POST["email"]))
    {$emailErr = "Email is required";}
  else
    {$email = test_input($_POST["email"]);}

  if (empty($_POST["website"]))
    {$website = "";}
  else
    {$website = test_input($_POST["website"]);}

  if (empty($_POST["comment"]))
    {$comment = "";}
  else
    {$comment = test_input($_POST["comment"]);}

  if (empty($_POST["gender"]))
    {$genderErr = "Gender is required";}
  else
    {$gender = test_input($_POST["gender"]);}

}

function test_input($data)
{
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
   Name: <input type="text" name="name">
   <span class="error">* <?php echo $nameErr;?></span>
   <br><br>
   E-mail: <input type="text" name="email">
   <span class="error">* <?php echo $emailErr;?></span>
   <br><br>
   Website: <input type="text" name="website">
   <span class="error"><?php echo $websiteErr;?></span>
```

```
   <br><br>
   Comment: <textarea name="comment" rows="5"
cols="40"></textarea>
   <br><br>
   Gender:
   <input type="radio" name="gender" value="female">Female
   <input type="radio" name="gender" value="male">Male
   <span class="error">* <?php echo $genderErr;?></span>
   <br><br>
   <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

SAU (varianta cu verificari)

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST")
{
```

```
   if (empty($_POST["name"]))
     {$nameErr = "Name is required";}
   else
     {
     $name = test_input($_POST["name"]);
     // check if name only contains letters and whitespace
     if (!preg_match("/^[a-zA-Z ]*$/",$name))
       {
       $nameErr = "Only letters and white space allowed";
       }
     }

   if (empty($_POST["email"]))
     {$emailErr = "Email is required";}
   else
     {
     $email = test_input($_POST["email"]);
     // check if e-mail address syntax is valid
     if (!preg_match("/([\w\-]+\@[\w\-]+\.[\w\-
]+)/",$email))
       {
       $emailErr = "Invalid email format";
       }
     }

   if (empty($_POST["website"]))
     {$website = "";}
   else
     {
     $website = test_input($_POST["website"]);
     // check if URL address syntax is valid (this regular
expression also allows dashes in the URL)
     if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-
z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website))
       {
       $websiteErr = "Invalid URL";
       }
     }

   if (empty($_POST["comment"]))
     {$comment = "";}
   else
     {$comment = test_input($_POST["comment"]);}

   if (empty($_POST["gender"]))
     {$genderErr = "Gender is required";}
   else
```

```
     {$gender = test_input($_POST["gender"]);}
}

function test_input($data)
{
     $data = trim($data);
     $data = stripslashes($data);
     $data = htmlspecialchars($data);
     return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
   Name: <input type="text" name="name" value="<?php echo
$name;?>">
   <span class="error">* <?php echo $nameErr;?></span>
   <br><br>
   E-mail: <input type="text" name="email" value="<?php
echo $email;?>">
   <span class="error">* <?php echo $emailErr;?></span>
   <br><br>
   Website: <input type="text" name="website" value="<?php
echo $website;?>">
   <span class="error"><?php echo $websiteErr;?></span>
   <br><br>
   Comment: <textarea name="comment" rows="5"
cols="40"><?php echo $comment;?></textarea>
   <br><br>
   Gender:
   <input type="radio" name="gender" <?php if
(isset($gender) && $gender=="female") echo "checked";?>
value="female">Female
   <input type="radio" name="gender" <?php if
(isset($gender) && $gender=="male") echo "checked";?>
value="male">Male
   <span class="error">* <?php echo $genderErr;?></span>
   <br><br>
   <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
```

```
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

COOKIE HANDLING

Aceste functii se plaseaza inainte de <HTML> :

setcookie(name, value, expire, path, domain)

setrawcookie()

```
<?php
$days=365;
$expire=time()+60*60*24*$days;
setcookie("user", "Alex Porter", $expire);
?>

<html>
.....
```

```
<html>
<body>

<?php
if (isset($_COOKIE["user"]))
  echo "Welcome " . $_COOKIE["user"] . "!<br>";
else
  echo "Welcome guest!<br>";
?>

</body>
</html>
```

print_r($_COOKIE) → returneaza toate cookie-urile

O sesiune reprezinta un cookie "de moment". E folosita doar pentru a se pastra informatiile la trecerea de la o pagina la alta. La parasirea site-ului, sesiunea se sterge.

```
<?php session_start(); ?>

<html>
......
```

```
Pg1.php:
    <?php
    session_start();
    $_SESSION['variable_name'] = 'xxx';
    ?>

Pg2.php:
    <?php
    session_start();
    echo $_SESSION['variable_name'];
    ?>
```

```
<?php
session_start();
if(isset($_SESSION['views']))
$_SESSION['views']=$_SESSION['views']+1;
else
$_SESSION['views']=1;
echo "Pageviews=". $_SESSION['views'];
?>

<html>
......
```

Eliminarea unei variabile:

```
<?php
session_start();
if(isset($_SESSION['views']))
  unset($_SESSION['views']);
?>
```

Stergerea intregii sesiuni:

```
<?php session_destroy(); ?>
```

OBS: Singura deosebire intre SESSION si COOKIE e ca sesiunea se memoreaza pe server (si se sterge complet la parasirea site-ului) iar cookie-urile se memoreaza in browser-ul clientului (si re-inviaza la re-intrarea pe acel site).

FILE HANDLING

CREARE / GESTIONARE DE FISIERE

fopen() → returneaza 0 daca esueaza
fclose()

- pozitionare la inceput

r = read only r+ = read/write

- completare fisier prin pozitionare la sfarsit (daca fisierul nu exista, este creat)

a = append only a+ = read/append

- creare fisier nou (cu conditia sa nu existe deja)

x = write only x+ = read/write

- creare fisier nou (sau overwritting)

w = write only w+ = read/write

feof() → verifica daca s-a ajuns la sfarsitul fisierului
fgets() → citeste linie cu linie
fgetc() → citeste caracter cu caracter

die() → intreruperea executiei
exit()

```
$file = fopen("welcome.txt","r") or exit("Unable to open file!");
while(!feof($file)) { echo fgets($file). "<br>"; }
fclose($file);
```

```
<?php
if(!file_exists("welcome.txt")) { die("File not found"); }
else { $file=fopen("welcome.txt","r"); }
?>
```

Daca nu s-ar avea in vedere si eventualitatea unei erori, s-ar afisa:

Warning: fopen(welcome.txt) [function.fopen]: failed to open stream: No such file or directory in **C:\webfolder\test.php** on line **2**

Gestionarea erorilor se poate face si prin functii definite de user (o gestionare implicita a erorilor oricum exista in cadrul PHP, dar ea poate fi redefinita - doar pentru anumite erori, sau pentru toate):

```
myErrorFunction(error_level,error_message,
                error_file, error_line,   error_context)

set_error_handler("myErrorFunction", in caz de eroare x)
```

Daca nu se specifica eroarea x se considera implicit toate.

error_level poate fi:

```
   2 = E_WARNING
   8 = E_NOTICE
 256 = E_USER_ERROR
 512 = E_USER_WARNING
1024 = E_USER_NOTICE
4096 = E_RECOVERABLE_ERROR
8191 = E_ALL
```

```
<?php
function customError($errno, $errstr)
  { echo "<b>Error:</b> [$errno] $errstr<br>";
    echo "Ending Script";
    die(); }
set_error_handler("customError");
```

```
echo($test);
?>
```

Se va afisa:
Error: [8] Undefined variable: test

Afisarea unei erori se poate face si cu
trigger_error("mesaj", in caz de eroare x – implicit toate)

Tipuri de eroare:

E_USER_ERROR , E_USER_WARNING , E_USER_NOTICE

```
<?php
$test=2;
if ($test>1) { trigger_error("Value must be 1 or below"); }
?>
```

Se va afisa:
Notice: Value must be 1 or below
in **C:\webfolder\test.php** on line **6**

EXEMPLU COMBINAT:

```
<?php
//error handler function
function customError($errno, $errstr)
  {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Ending Script";
  die();
  }

//set error handler
set_error_handler("customError",E_USER_WARNING);

//trigger error
$test=2;
if ($test>1)
  {
  trigger_error("Value must be 1 or below",E_USER_WARNING);
  }
?>
```

Se va afisa:
Error: [512] Value must be 1 or below
Ending Script

Functia error_log() creeaza un fisier in care noteaza toate informatiile privind erorile aparute.

```
<?php
//error handler function
function customError($errno, $errstr)
  {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Webmaster has been notified";
  error_log("Error: [$errno] $errstr",1,
  "someone@example.com","From: webmaster@example.com");
  }

//set error handler
set_error_handler("customError",E_USER_WARNING);

//trigger error
$test=2;
if ($test>1)
  {
  trigger_error("Value must be 1 or below",E_USER_WARNING);
  }
?>
```

Se va afisa:

```
Error: [512] Value must be 1 or below
Webmaster has been notified
```

Iar la adresa de mail specificata se va transmite:

```
Error: [512] Value must be 1 or below
```

UPLOAD DE FISIERE

```
<html>
<body>

<form action="upload_file.php" method="post"
enctype="multipart/form-data">
<label for="file">Filename:</label>
<input type="file" name="file" id="file"><br>
```

```
<input type="submit" name="submit" value="Submit">
</form>

</body>
</html>
```

UPLOAD_FILE.PHP

```
<?php
if ($_FILES["file"]["error"] > 0)
  {
  echo "Error: " . $_FILES["file"]["error"] . "<br>";
  }
else
  {
  echo "Upload: " . $_FILES["file"]["name"] . "<br>";
  echo "Type: " . $_FILES["file"]["type"] . "<br>";
  echo "Size: " . ($_FILES["file"]["size"] / 1024) . " kB<br>";
  echo "Temp file: " . $_FILES["file"]["tmp_name"] . "<br>";


    if (file_exists("upload/" . $_FILES["file"]["name"]))
      {
      echo $_FILES["file"]["name"] . " already exists. ";
      }
    else
      {
      move_uploaded_file($_FILES["file"]["tmp_name"],
      "upload/" . $_FILES["file"]["name"]);
      echo "Stored in: " . "upload/" . $_FILES["file"]["name"];
      }



  }
?>
```

SAU (varianta cu restrictii: doar gif,jpeg,png si sub 20kB)

```
<?php
$allowedExts = array("gif", "jpeg", "jpg", "png");
$temp = explode(".", $_FILES["file"]["name"]);
$extension = end($temp);
if ((($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] == "image/jpg")
|| ($_FILES["file"]["type"] == "image/pjpeg")
|| ($_FILES["file"]["type"] == "image/x-png")
|| ($_FILES["file"]["type"] == "image/png"))
&& ($_FILES["file"]["size"] < 20000)
```

```
&& in_array($extension, $allowedExts))
  {
  if ($_FILES["file"]["error"] > 0)
    {
    echo "Error: " . $_FILES["file"]["error"] . "<br>";
    }
  else
    {
    echo "Upload: " . $_FILES["file"]["name"] . "<br>";
    echo "Type: " . $_FILES["file"]["type"] . "<br>";
    echo "Size: " . ($_FILES["file"]["size"] / 1024) . " kB<br>";
    echo "Temp file: " . $_FILES["file"]["tmp_name"] . "<br>";


    if (file_exists("upload/" . $_FILES["file"]["name"]))
      {
      echo $_FILES["file"]["name"] . " already exists. ";
      }
    else
      {
      move_uploaded_file($_FILES["file"]["tmp_name"],
      "upload/" . $_FILES["file"]["name"]);
      echo "Stored in: " . "upload/" . $_FILES["file"]["name"];
      }


    }
  }
else
  {
  echo "Invalid file";
  }
?>
```

OBS: Cand se lucreaza cu diversele hostere existente pe net in loc de "upload/" se va pune "../" Oricum, locatia unde a fost salvat fisierul nu poate fi aflata (ci doar folosita). Acesta e un aspect practic ce variaza de la hoster la hoster (si e impus de masurile de securitate).

Temp file: /home/lovely.tk/tmp/phpvELcSE
Stored in: /home/lovely.tk/tmp/_smth.txt

Observatie: fisierele temporare sunt efemere , ele dispar cand pagina php s-a terminat de executat

Creare / Stergere fisier: fopen("xxx.txt","x+") , unlink()
Creare / Stergere folder: mkdir() , rmdir()

OBS: Cand se folosesc astfel de comenzi, trebuie tinut cont ca adresa reala a unui fisier este de fapt "/home/lovely.tk/public/xxx.php" , nu "http://lovely.tk/xxx.php"

EXEMPLU DE PROCESARE A UNUI FISIER PRELUAT PRIN <FORM>

```
<?php

if ($_FILES["file"]["error"] > 0)
  {
  echo "Error: " . $_FILES["file"]["error"] . "<br>";
  }
else
  {
  echo "Upload: " . $_FILES["file"]["name"] . "<br>";
  echo "Type: " . $_FILES["file"]["type"] . "<br>";
  echo "Size: " . ($_FILES["file"]["size"] / 1024) . "
kB<br>";
  echo "Temp file: " . $_FILES["file"]["tmp_name"] .
"<br>";

      move_uploaded_file($_FILES["file"]["tmp_name"],
      "/home/lovely.tk/tmp/" . $_FILES["file"]["name"]);
      echo "Stored in: " . "/home/lovely.tk/tmp/" .
$_FILES["file"]["name"];

$sursa=fopen("/home/lovely.tk/tmp/" .
$_FILES["file"]["name"],"r");


fclose($sursa);

unlink($sursa);

}


?>
```

COMENTARIU:

Cand se lucreaza cu fisiere (citire, modificare, creare, stergere) trebuie avute in vedere permisiunile (si eventual modificate corespunzator de pe server).

Orice fisier are OWNER si USER (format din GROUP si EVERYONE). Fisierele generate prin browser (prin fopen() sau <FORM>) au alt OWNER (intuitiv: in acest caz OWNER = browser , USER = server) decat cele uploadate direct pe hoster prin ftp (intuitiv: in acest caz OWNER = server , USER = browser). Astfel, nu ar fi de mirare ca nu poate fi modificat de pe server un fisier creat prin browser.

Permisiunile (RWX) au semnificatii diferite pentru FOLDERE si FISIERE.

Pentru FOLDERE:

→ permisiuni pentru USER (pentru OWNER sunt obligatoriu toate: RWX)

FARA X = folder blocat (indiferent de R/W)
  CU X = acces la fisierele existente
         (nu din punct de vedere R/W, ci doar ca entitati - mai departe, este posibil R/W daca fisierele la randul lor permit acest lucru)
               + W = creare fisiere noi in acel folder
               + R = vizualizare continut folder in browser

OBS: Daca un fisier poate fi citit prin fopen(), automat poate fi si downloadat / vizualizat in browser (si invers). De asemenea, citirea se poate face intotdeauna si prin "http:", nu e neaparat nevoie de "/home/", pe cand crearea si modificarea (in caz ca sunt permise) se pot face numai prin "home" si totodata numai din cadrul acelui hoster (server, site).

Pentru FISIERE:

→ permisiuni pentru USER (pentru OWNER sunt obligatoriu RW)

R = citire / vizualizare (in browser, respectiv interfata serverului - in functie de OWNER: browser sau server)

W = modificare (prin browser, respectiv interfata)

OBS: permisiunea X poate fi ignorata (ea are efect doar pentru fisierele executabile)

PRIORITATI:
Un fisier nu poate avea permisiuni mai mari decat folderul din care face parte (folderul este prioritar).

Si invers:
daca un folder are permisiuni, el nu poate "forta" fisierele din cadrul lui sa aiba aceleasi permisiuni (astfel, de exemplu, se poate seta ca folderul sa fie vizibil in browser, dar diverse fisiere din cadrul lui nu)

SFAT:

In practica, atunci cand se lucreaza cu diverse hostere, modul de gestionare a permisiunilor difera de la hoster la hoster. In acest caz, initial se poate face doar o simpla verificare a permisiunilor:

1) sa nu se afiseze in browser continutul folderelor
2) sa se poata crea (din browser) un fisier nou
3) sa se poata modifica un fisier deja existent

Pentru utilizari obisnuite, permisiunile (pentru USER) trebuie setate in felul urmator:
→ WX pentru foldere (fara R)
→ R pentru fisiere (fara W si X)
[daca se doreste ca prin browser sa se modifice continutul unui fisier uploadat prin server, atunci, pentru (doar pentru) acel fisier se va adauga si W]

OBS: Daca paginile php ale site-ului nu sunt protejate la scriere, oricine le poate modifica uploadand si apoi executand propriul fisier php uploadat prin intermediul unui <FORM> gasit pe acel site (presupunand ca exista un formular de upload de fisiere pe acel site). De asemenea, comenzi php se pot executa si prin orice simplu <FORM> (nu neaparat de upload fisiere), analizand continutul respectivei pagini php (care este din oficiu downloadabila) si gasind un punct slab.

OBS: Doar owner-ul isi poate sterge propriul fisier. (?)

MySQL

http://www.mysql.com
http://dev.mysql.com/doc/refman/5.1/en/charset.html
OBS: In MySQL se pot crea mai multe baze de date (db). Fiecare baza de date (db) poate avea una sau mai multe tabele. Fiecare tabel are un numar de campuri (coloane) si intrari (linii).

Comenzi SQL:

```
CREATE DATABASE
CREATE TABLE

INSERT INTO table_name (colname1, colname2, colname3,...)
                VALUES (  value1,   value2,   value3,...)
```

OBS: in coloanele unde nu s-a introdus nimic se va considera NULL (dar coloana primara, cu functie de contor, va fi automat incrementata - ea reprezinta pozitia/numarul inregistrarii)

```
UPDATE table_name SET colname1=value1, colname2=value2,...
                  WHERE colA=val AND|OR colB=val

DELETE FROM table_name WHERE colA=val AND|OR colB=val
```

OBS: daca se omite WHERE se sterg toate intrarile

```
SELECT [DISTINCT] nume_coloana FROM table_name
                              WHERE colA=val AND|OR colB=val
                              ORDER BY nume_coloana ASC|DESC
                              LIMIT numar_linii

SELECT * FROM table_name   → selecteaza toate coloanele
```

OBS: table_name poate aparea si intre paranteze patrate [table_name]

OBS: e recomandat sa se foloseasca ghilimele simple , nu duble , atunci cand e vorba de siruri in cadrul comenzilor SQL

OBS: atunci cand e vorba de mai multe comenzi SQL , trebuie separate prin ;

OBS: in cadrul WHERE , in loc de = poate fi orice operator: <> (echivalentul lui !=) , > , < , BETWEEN , [NOT] LIKE , IN

Exemple:
WHERE City LIKE 's%' → City sa inceapa cu "s"
WHERE City LIKE '%s' → City se termina cu "s"

OBS: variabilele SQL de tip session (adica statice) sunt precedate de @ , iar cele de tip procedure (adica locale) nu ; atribuirea de valori se poate face prin = sau := (nu e nici o diferenta)

```
SET @var := 1
SELECT @var2 := 2
```

Example of a stored procedure:

```
DELIMITER @@

CREATE PROCEDURE prc_test (var INT)
BEGIN
    DECLARE  var2 INT;
    SET var2 = 1;
    SELECT  var2;
END;
@@

DELIMITER ;
```

Functii PHP de interactiune cu MySQL:

mysqli_connect(host,username,password,dbname)
mysqli_close()
mysqli_query($con = db , $sql = cmd)

ACCESAREA UNEI DB:

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");

if (mysqli_connect_errno())
  { echo "Failed to connect to MySQL: " . mysqli_connect_error(); }
```

```
mysqli_close($con);
?>
```

CREAREA UNEI DB:

```
<?php
$con = mysqli_connect("example.com","peter","abc123");

if (mysqli_connect_errno())
  { echo "Failed to connect to MySQL: " . mysqli_connect_error(); }

$sql = "CREATE DATABASE my_db";
$cmd = mysqli_query($con,$sql);

if ($cmd) { echo "Database my_db created successfully"; }
    else  { echo "Error creating database: " . mysqli_error($con); }

mysqli_close($con);
?>
```

CREAREA UNEI TABELE:

```
<?php
$con = mysqli_connect("example.com","peter","abc123","my_db");

if (mysqli_connect_errno())
  { echo "Failed to connect to MySQL: " . mysqli_connect_error(); }

$sql = "CREATE TABLE Persons( PID INT NOT NULL AUTO_INCREMENT,
                              PRIMARY KEY(PID),
                              FirstName CHAR(15),
                              LastName CHAR(15),
                              Age INT )";

$cmd = mysqli_query($con,$sql);

if ($cmd) { echo "Table Persons created successfully"; }
    else  { echo "Error creating table: " . mysqli_error($con); }

mysqli_query($con,"INSERT INTO Persons (FirstName, LastName, Age)
                                VALUES ('Peter', 'Griffin',35)");

mysqli_close($con);
?>
```

CITIREA DATELOR DINTR-O TABELA

```php
<?php
$con = mysqli_connect("example.com","peter","abc123","my_db");

if (mysqli_connect_errno())
  { echo "Failed to connect to MySQL: " . mysqli_connect_error(); }

$result = mysqli_query($con,"SELECT * FROM Persons");

while ($row = mysqli_fetch_array($result))
  { echo $row['FirstName'] . " " . $row['LastName']; echo "<br>"; }

mysqli_close($con);
?>
```

Exemplu avansat:

```
$stmt = $dbh->prepare("INSERT INTO Customers
(CustomerName,Address,City)
VALUES (:nam, :add, :cit)");
$stmt->bindParam(':nam', $txtNam);
$stmt->bindParam(':val', $txtAdd);
$stmt->bindParam(':cit', $txtCit);
$stmt->execute();
```

OBSERVATIE:

Similar cu MySQL se poate folosi, pe exact acelasi principiu, orice baza de date (de exemplu Microsoft Excel).

Tot ce difera sunt functiile PHP de interactiune:

odbc_connect(data source name, username, password, optional cursor type)

odbc_exec()

odbc_fetch_row()

odbc_result()

odbc_close()

Exemplul 1

```
<html>
<body>

<?php
$conn=odbc_connect('northwind','','');
if (!$conn)
  {exit("Connection Failed: " . $conn);}
$sql="SELECT * FROM customers";
$rs=odbc_exec($conn,$sql);
if (!$rs)
  {exit("Error in SQL");}
echo "<table><tr>";
echo "<th>Companyname</th>";
echo "<th>Contactname</th></tr>";
while (odbc_fetch_row($rs))
  {
  $compname=odbc_result($rs,"CompanyName");
  $conname=odbc_result($rs,"ContactName");
  echo "<tr><td>$compname</td>";
  echo "<td>$conname</td></tr>";
  }
odbc_close($conn);
echo "</table>";
?>

</body>
</html>
```

Exemplul 2

```
// Basic ODBC connection to Excel
$excelFile = realpath('C:/ExcelData.xls');
$excelDir = dirname($excelFile);
$connection = odbc_connect("Driver={Microsoft Excel Driver
(*.xls)};DriverId=790;Dbq=$excelFile;DefaultDir=$excelDir"
, '', '');

// Basic selection from a sheet
$result = odbc_exec ($connection, "select * from
[Sheet1$]");

// Selection from an Excel sheet using a field name
$result = odbc_exec($connection, "select * from [employees]
where [first_name] = 'Joe'");


// Quick output to confirm data
odbc_result_all($result, "bgcolor='DDDDDD' cellpadding =
'1'");
```

```
// Loop to handle all results
while( $row = odbc_fetch_array($result) )
{
    // row data is accessible by field name
   $row['first_name'];

   // unnamed fields follow a F# pattern
   $row['F4'];

   // each key and value can be examined individually as
well
   foreach($row as $key => $value)
   {
      print "<br>Key: " . $key . " Value: " . $value;
   }
}
```

http://php.net/manual/en/function.odbc-connect.php
http://docstore.mik.ua/orelly/webprog/php/ch15_04.htm O'Reilly - Programming PHP
http://www.excelforum.com/excel-general/477932-excel-odbc-driver-get-column-names.html
http://code.goingasplannedby.us/2013/06/14/php-excel-with-odbc/

EXEMPLU PRACTIC : PROGRAMUL GESTIUNE AMENZI

OBS: orice site cu login trebuie sa aiba implementat sistem de verificare a loginului pe fiecare pagina, prin $_SESSION (astfel incat sa nu se poata "sari" direct in site fara a se trece prin login)

LOGIN.PHP

```
<!DOCTYPE html>
<html>
<body>
```

```
<div id="bloc"></div>

<div id="msg"></div>



<?php

if ($_SERVER["REQUEST_METHOD"] == "POST")

      { $user=$_POST["user"];
     $pass=$_POST["pass"];
     $con = mysqli_connect("mysql.3owl.com", $user, $pass,
$user);

     if (mysqli_connect_errno())

          { $cod='<form id="f1" method="post"
action="login.php">
                    <input id="user" type="text"
name="user">
                    <input id="pass" type="text"
name="pass">
                    <input type="submit" value="OK">
                    </form>';
                $json = json_encode($cod);
            echo "<script>

     document.getElementById('msg').innerHTML='Date
incorecte';

     document.getElementById('bloc').innerHTML={$json};
               </script>"; }

     else { mysqli_close($con);

            $cod='<form id="f1" method="post"
action="intro3.php">
               <input id="user" type="text" name="user"
value="'.$user.'">
                  <input id="pass" type="text" name="pass"
value="'.$pass.'">
                  </form>';
                $json = json_encode($cod);
            echo "<script>
```

```
document.getElementById('bloc').style.display='none';

document.getElementById('bloc').innerHTML={$json};
               document.getElementById('f1').submit();
               </script>"; } }

else { $cod='<form id="f1" method="post"
action="login.php">
          <input id="user" type="text" name="user">
          <input id="pass" type="text" name="pass">
          <input type="submit" value="OK">
          </form>';
       $json = json_encode($cod);
       echo "<script>

document.getElementById('bloc').innerHTML={$json};
          </script>"; }

?>


SAU (varianta estetica)


<!DOCTYPE html>
<html>

<head>

<title>Lux Love</title>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />


     <!-- Add jQuery library -->
     <script type="text/javascript"
src="http://luxlove.3owl.com/jquery-1.9.0.min.js"></script>

     <!-- Add mousewheel plugin (this is optional) -->
     <script type="text/javascript"
src="http://luxlove.3owl.com/jquery.mousewheel-
3.0.6.pack.js"></script>

     <!-- Add fancyBox main JS and CSS files -->
```

```
     <script type="text/javascript"
src="http://luxlove.3owl.com/jquery.fancybox.js?v=2.1.4"></
script>
     <link rel="stylesheet" type="text/css"
href="http://luxlove.3owl.com/jquery.fancybox.css?v=2.1.4"
media="screen" />

     <!-- Add Button helper (this is optional) -->
     <link rel="stylesheet" type="text/css"
href="http://luxlove.3owl.com/jquery.fancybox-
buttons.css?v=1.0.5" />
     <script type="text/javascript"
src="http://luxlove.3owl.com/jquery.fancybox-
buttons.js?v=1.0.5"></script>

     <!-- Add Thumbnail helper (this is optional) -->
     <link rel="stylesheet" type="text/css"
href="http://luxlove.3owl.com/jquery.fancybox-
thumbs.css?v=1.0.7" />
     <script type="text/javascript"
src="http://luxlove.3owl.com/jquery.fancybox-
thumbs.js?v=1.0.7"></script>

     <!-- Add Media helper (this is optional) -->
     <script type="text/javascript"
src="http://luxlove.3owl.com/jquery.fancybox-
media.js?v=1.0.5"></script>

     <script type="text/javascript">
          $(document).ready(function() {
               /*
                *  Simple image gallery. Uses default
settings
                */

               $('.fancybox').fancybox();

               /*
                *  Different effects
                */

               // Change title type, overlay closing speed
               $(".fancybox-effects-a").fancybox({
                    helpers: {
                         title : {
                              type : 'outside'
                         },
```

```
                    overlay : {
                        speedOut : 0
                    }
                }
            });

            // Disable opening and closing animations,
change title type
            $(".fancybox-effects-b").fancybox({
                openEffect  : 'none',
                closeEffect    : 'none',

                helpers : {
                    title : {
                        type : 'over'
                    }
                }
            });

            // Set custom style, close if clicked,
change title type and overlay color
            $(".fancybox-effects-c").fancybox({
                wrapCSS    : 'fancybox-custom',
                closeClick : true,

                openEffect : 'none',

                helpers : {
                    title : {
                        type : 'inside'
                    },
                    overlay : {
                        css : {
                            'background' :
'rgba(238,238,238,0.85)'
                        }
                    }
                }
            });

            // Remove padding, set opening and closing
animations, close if clicked and disable overlay
            $(".fancybox-effects-d").fancybox({
                padding: 0,

                openEffect : 'elastic',
                openSpeed  : 150,
```

```
                    closeEffect : 'elastic',
                    closeSpeed  : 150,

                    closeClick : true,

                    helpers : {
                         overlay : null
                    }
               });

               /*
                *  Button helper. Disable animations, hide
close button, change title type and content
                */

               $('.fancybox-buttons').fancybox({
                    openEffect  : 'none',
                    closeEffect : 'none',

                    prevEffect : 'none',
                    nextEffect : 'none',

                    closeBtn  : false,

                    helpers : {
                         title : {
                              type : 'inside'
                         },
                         buttons   : {}
                    },

                    afterLoad : function() {
                         this.title = 'Image ' +
(this.index + 1) + ' of ' + this.group.length + (this.title
? ' - ' + this.title : '');
                    }
               });


               /*
                *  Thumbnail helper. Disable animations,
hide close button, arrows and slide to next gallery item if
clicked
                */

               $('.fancybox-thumbs').fancybox({
```

```
                    prevEffect : 'none',
                    nextEffect : 'none',

                    closeBtn  : false,
                    arrows    : false,
                    nextClick : true,

                    helpers : {
                         thumbs : {
                              width  : 50,
                              height : 50
                         }
                    }
               });

               /*
                *  Media helper. Group items, disable
animations, hide arrows, enable media and button helpers.
               */
               $('.fancybox-media')
                    .attr('rel', 'media-gallery')
                    .fancybox({
                         openEffect : 'none',
                         closeEffect : 'none',
                         prevEffect : 'none',
                         nextEffect : 'none',

                         arrows : false,
                         helpers : {
                              media : {},
                              buttons : {}
                         }
                    });

               /*
                *  Open manually
                */

               $("#fancybox-manual-a").click(function() {
                    $.fancybox.open('1_b.jpg');
               });

               $("#fancybox-manual-b").click(function() {
                    $.fancybox.open({
                         href : 'iframe.html',
                         type : 'iframe',
                         padding : 5
```

```
                    });
               });

               $("#fancybox-manual-c").click(function() {
                    $.fancybox.open([
                         {
                              href : '1_b.jpg',
                              title : 'My title'
                         }, {
                              href : '2_b.jpg',
                              title : '2nd title'
                         }, {
                              href : '3_b.jpg'
                         }
                    ], {
                         helpers : {
                              thumbs : {
                                   width: 75,
                                   height: 50
                              }
                         }
                    });
               });


          });
     </script>


     <style type="text/css">
          .fancybox-custom .fancybox-skin {
               box-shadow: 0 0 50px #222;
          }
     </style>

</head>

<body bgcolor="black" style="background-
image:url('gradient.png');background-repeat:no-
repeat;background-attachment:fixed;background-position:0%
100%;background-size: 100% 80%;" >

<table width="100%">
<tr><td align="center">

<a class="fancybox" href="#select" style="text-
decoration:none; color:lime" title="LOGIN"><img id="log"
```

```
src="login.png"
style="position:fixed;width:25%;height:15%;left:38%;top:20%
;" border="0"></a>

<div id="select" style="display:none;" align="center">
<span style="width:100%">

<div id="bloc"></div>


</span>
</div>

</td></tr>
</table>



<?php

if ($_SERVER["REQUEST_METHOD"] == "POST")

      { $user=$_POST["user"];
     $pass=$_POST["pass"];
     $con = mysqli_connect("mysql.3owl.com", $user, $pass,
$user);

     if (mysqli_connect_errno())

          { $cod='<br><form id="f1" method="post"
action="login.php">
                    <input id="user" type="text"
name="user">
                    <input id="pass" type="password"
name="pass"><br><br>
                    <input type="submit" value="OK">
                    </form>';
                $json = json_encode($cod);
            echo "<script>

     document.getElementById('log').src='retry.png';

     document.getElementById('bloc').innerHTML={$json};
               </script>"; }

     else { mysqli_close($con);
```

```
            $cod='<form id="f1" method="post"
action="intro3.php">
                <input id="user" type="text" name="user"
value="'.$user.'">
                  <input id="pass" type="password"
name="pass" value="'.$pass.'">
                  </form>';
                $json = json_encode($cod);
            echo "<script>

document.getElementById('bloc').style.display='none';

document.getElementById('bloc').innerHTML={$json};
                document.getElementById('f1').submit();
                </script>"; } }

else { $cod='<br><form id="f1" method="post"
action="login.php">
          <input id="user" type="text" name="user">
          <input id="pass" type="password"
name="pass"><br><br>
          <input type="submit" value="OK">
          </form>';
       $json = json_encode($cod);
       echo "<script>

document.getElementById('bloc').innerHTML={$json};
          </script>"; }

?>

</body>
</html>
```

INTRO3.PHP

```
<html>
<head>

<style type="text/css">

@font-face {
 font-family: absolut;
 src: url("/absolut1.eot")
```

```
}

@font-face {
 font-family: absolut;
 font-weight: normal;
 src: url("/absolut1.ttf")
}

@font-face {
 font-family: calligraphia;
 src: url("/Calligraphia_One.eot")
}

@font-face {
 font-family: calligraphia;
 font-weight: normal;
 src: url("/Calligraphia_One.ttf")
}



</style>

</head>


<body bgcolor="lightblue">

<p align="center"><img src="gestiuneamenzi.gif"></p>

<form action="choose.php" method="post"
enctype="multipart/form-data">

<table width="100%" align="center">
<tr>
<td width="35%" style="font-size:200%;color:magenta;font-
family:verdana" align="right"><img src="nume.gif"
width="100em">  </td>
<td width="65%"><input type="text" name="nume" style="font-
size:200%;color:green;"></td>

<tr>
<td width="35%" style="font-size:200%;color:magenta;font-
family:verdana" align="right"><img src="prenume.gif"
width="150em">  </td>
<td width="65%"><input type="text" name="prenume"
style="font-size:200%;color:green;"></td>
```

```
<tr>
<td width="35%" style="font-size:200%;color:magenta;font-
family:verdana" align="right"><img src="cnp.gif"
width="100em">  </td>
<td width="65%"><input type="number" name="CNP"
style="font-size:200%;color:green;"></td>

<tr>
<td width="35%" style="font-size:200%;color:magenta;font-
family:verdana" align="right"><img src="pv.gif"
width="70em">  </td>
<td width="65%"><input type="text" name="PV" style="font-
size:200%;color:green;"></td>

<tr>
<td width="35%" style="font-size:200%;color:magenta;font-
family:verdana" align="right"><img src="pdf.gif"
width="100em">  </td>
<td width="65%"><input type="file" name="uploaded_file"
style="font-size:200%;color:blue;"></td>

</table>

<br><br>

<table width="100%" align="center">
<tr>
<td width="53%" align="right"><input type="Submit"
name="introdu" value="Introdu" style="font-
size:200%;color:blue;font-family:absolut;font-weight:700;"
align="center"></td>
<td width="47%" align="left">    <input
type="Submit" name="cauta" value="Cauta" style="font-
size:200%;color:blue;font-family:absolut;font-
weight:700;"></td>

</table>
</form>

</body>
```

CHOOSE . PHP

```
<?
```

```
$introdu=$_POST['introdu'];
$cauta=$_POST['cauta'];


if ($introdu == "Introdu") {

echo "<head>

<meta http-equiv='refresh' content='0;url=/intro3.php'>

</head> ";


$username="u709308397_x";
$password="...";
$database="u709308397_x";
$host = "mysql.3owl.com";


    // Check if a file has been uploaded
    if(isset($_FILES['uploaded_file'])) {
        // Make sure the file was sent without errors
        if($_FILES['uploaded_file']['error'] == 0) {
            // Connect to the database
            $dbLink = new mysqli($host, $username,
$password, $database);
            if(mysqli_connect_errno()) {
                die("MySQL connection failed: ".
mysqli_connect_error());
            }

            // Gather all required data

$nume=$_POST['nume'];
$prenume=$_POST['prenume'];
$CNP=$_POST['CNP'];
$PV=$_POST['PV'];



            $name = $dbLink-
>real_escape_string($_FILES['uploaded_file']['name']);
            $mime = $dbLink-
>real_escape_string($_FILES['uploaded_file']['type']);
```

```
            $data = $dbLink-
>real_escape_string(file_get_contents($_FILES
['uploaded_file']['tmp_name']));
            $size =
intval($_FILES['uploaded_file']['size']);

            // Create the SQL query
            $query = "
                INSERT INTO `amenzi` (
               `nume`, `prenume`, `CNP`, `PV`,   `name`,
`mime`, `size`, `data`, `created`
                )
                VALUES (
                '{$nume}', '{$prenume}', '{$CNP}', '{$PV}',
'{$name}', '{$mime}', {$size}, '{$data}', NOW()
                )";

            // Execute the query
            $result = $dbLink->query($query);

            // Check if it was successfull
            if($result) {
                echo 'Success! Your file was successfully
added!';
            }
            else {
                echo 'Error! Failed to insert the file'
                   . "<pre>{$dbLink->error}</pre>";
            }
        }
        else {
            echo 'An error accured while the file was being
uploaded. '
               . 'Error code: '.
intval($_FILES['uploaded_file']['error']);
        }

        // Close the mysql connection
        $dbLink->close();
    }
    else {
        echo 'Error! A file was not sent!';
    }

    // Echo a link back to the main page
    echo '<p>Click <a href="intro3.php">here</a> to go
back</p>';
```

```
 }


if ($cauta == "Cauta")  {


$username="u709308397_x";
$password="...";
$database="u709308397_x";
$host = "mysql.3owl.com";

mysql_connect($host,$username,$password);
@mysql_select_db($database) or die( "Unable to select
database");

$nume=$_POST['nume'];
$prenume=$_POST['prenume'];
$CNP=$_POST['CNP'];
$PV=$_POST['PV'];


$num=0;

if (!empty($_POST['nume']) && !empty($_POST['prenume']) &&
!empty($_POST['CNP']) && !empty($_POST['PV']) ) {

$query=" SELECT * FROM amenzi WHERE nume='{$nume}' AND
prenume='{$prenume}' AND CNP='{$CNP}' AND PV='{$PV}'  ";
$result=mysql_query($query);
$num=mysql_numrows($result); }

if (!empty($_POST['nume']) && !empty($_POST['prenume']) &&
!empty($_POST['CNP']) && empty($_POST['PV']) ) {

$query=" SELECT * FROM amenzi WHERE nume='{$nume}' AND
prenume='{$prenume}' AND CNP='{$CNP}' ";
$result=mysql_query($query);
$num=mysql_numrows($result); }

if (!empty($_POST['nume']) && empty($_POST['prenume']) &&
!empty($_POST['CNP']) && !empty($_POST['PV']) ) {

$query=" SELECT * FROM amenzi WHERE nume='{$nume}'  AND
CNP='{$CNP}' AND PV='{$PV}'  ";
$result=mysql_query($query);
```

```
$num=mysql_numrows($result); }

if (empty($_POST['nume']) && !empty($_POST['prenume']) &&
!empty($_POST['CNP']) && !empty($_POST['PV']) ) {

$query=" SELECT * FROM amenzi WHERE  prenume='{$prenume}'
AND CNP='{$CNP}' AND PV='{$PV}'  ";
$result=mysql_query($query);
$num=mysql_numrows($result); }

if (!empty($_POST['nume']) && !empty($_POST['prenume']) &&
empty($_POST['CNP']) && empty($_POST['PV']) ) {

$query=" SELECT * FROM amenzi WHERE nume='{$nume}' AND
prenume='{$prenume}'  ";
$result=mysql_query($query);
$num=mysql_numrows($result); }

if (!empty($_POST['nume']) && empty($_POST['prenume']) &&
!empty($_POST['CNP']) && empty($_POST['PV']) ) {

$query=" SELECT * FROM amenzi WHERE nume='{$nume}' AND
CNP='{$CNP}'  ";
$result=mysql_query($query);
$num=mysql_numrows($result); }

if (!empty($_POST['nume']) && empty($_POST['prenume']) &&
empty($_POST['CNP']) && !empty($_POST['PV']) ) {

$query=" SELECT * FROM amenzi WHERE nume='{$nume}' AND
PV='{$PV}'  ";
$result=mysql_query($query);
$num=mysql_numrows($result); }

if (empty($_POST['nume']) && !empty($_POST['prenume']) &&
!empty($_POST['CNP']) && empty($_POST['PV']) ) {

$query=" SELECT * FROM amenzi WHERE  prenume='{$prenume}'
AND CNP='{$CNP}'  ";
$result=mysql_query($query);
$num=mysql_numrows($result); }

if (empty($_POST['nume']) && !empty($_POST['prenume']) &&
empty($_POST['CNP']) && !empty($_POST['PV']) ) {

$query=" SELECT * FROM amenzi WHERE  prenume='{$prenume}'
AND PV='{$PV}'  ";
```

```
$result=mysql_query($query);
$num=mysql_numrows($result); }

if (empty($_POST['nume']) && empty($_POST['prenume']) &&
!empty($_POST['CNP']) && !empty($_POST['PV']) ) {

$query=" SELECT * FROM amenzi WHERE  CNP='{$CNP}' AND
PV='{$PV}'  ";
$result=mysql_query($query);
$num=mysql_numrows($result); }

if (!empty($_POST['nume']) && empty($_POST['prenume']) &&
empty($_POST['CNP']) && empty($_POST['PV']) ) {

$query=" SELECT * FROM amenzi WHERE nume='{$nume}' ";
$result=mysql_query($query);
$num=mysql_numrows($result); }

if (empty($_POST['nume']) && !empty($_POST['prenume']) &&
empty($_POST['CNP']) && empty($_POST['PV']) ) {

$query=" SELECT * FROM amenzi WHERE  prenume='{$prenume}'
";
$result=mysql_query($query);
$num=mysql_numrows($result); }


if (empty($_POST['nume']) && empty($_POST['prenume']) &&
!empty($_POST['CNP']) && empty($_POST['PV']) ) {

$query=" SELECT * FROM amenzi WHERE  CNP='{$CNP}' ";
$result=mysql_query($query);
$num=mysql_numrows($result); }

if (empty($_POST['nume']) && empty($_POST['prenume']) &&
empty($_POST['CNP']) && !empty($_POST['PV']) ) {

$query=" SELECT * FROM amenzi WHERE PV='{$PV}'  ";
$result=mysql_query($query);
$num=mysql_numrows($result); }



echo "
<html>
<head>
```

```
<style type='text/css'>

@font-face {
 font-family: absolut;
 src: url('/absolut1.eot')
}

@font-face {
 font-family: absolut;
 font-weight: normal;
 src: url('/absolut1.ttf')
}

@font-face {
 font-family: calligraphia;
 src: url('/Calligraphia_One.eot')
}

@font-face {
 font-family: calligraphia;
 font-weight: normal;
 src: url('/Calligraphia_One.ttf')
}



</style>

</head>


<body bgcolor='white'>";




if ( $num > 0) {


echo "

<br>
<b><center><font color='cyan' size='10em'
face='absolut'>Rezultatele
căutării</font></center></b><br><br>

<table align='center'>";
```

```
$i=0;
while ($i < $num) {

$id=mysql_result($result,$i,"id");

$nume=mysql_result($result,$i,"nume");
$prenume=mysql_result($result,$i,"prenume");
$CNP=mysql_result($result,$i,"CNP");
$PV=mysql_result($result,$i,"PV");

echo "
<tr>
<td style='font-size:2em;color:magenta;font-
family:absolut'>$nume $prenume      </td>
<td style='font-size:2em;color:magenta;font-
family:absolut'><font color='lime'>cnp<font
color='magenta'> $CNP       </td>
<td style='font-size:2em;color:magenta;font-
family:absolut'><font color='lime'>pv<font color='magenta'>
$PV       </td>
<td style='font-size:2em;color:magenta;font-
family:absolut;'><a href='download.php?id={$id}'
style='text-decoration:none;'>download</a></td>
</tr>";

$i++;
}

echo "</table>"; }

else { echo "

<br>
<b><center><font color='lime' size='10em' face='absolut'>Nu
există o astfel de situaţie.</font></center></b><br><br>

"; }




mysql_close();

echo "</body></html>";
```

```
 }


?>
```

DOWNLOAD . PHP

```
<?php
    // Make sure an ID was passed
    if(isset($_GET['id'])) {
    // Get the ID
        $id = intval($_GET['id']);

        // Make sure the ID is in fact a valid ID
        if($id <= 0) {
            die('The ID is invalid!');
        }
        else {
            // Connect to the database
            $dbLink = new mysqli('mysql.3owl.com',
'u709308397_x', '...', 'u709308397_x');
            if(mysqli_connect_errno()) {
                die("MySQL connection failed: ".
mysqli_connect_error());
            }

            // Fetch the file information
            $query = "
                SELECT `mime`, `name`, `size`, `data`
                FROM `amenzi`
                WHERE `id` = {$id}";
            $result = $dbLink->query($query);

            if($result) {
                // Make sure the result is valid
                if($result->num_rows == 1) {
                // Get the row
                    $row = mysqli_fetch_assoc($result);

                    // Print headers
                    header("Content-Type: ". $row['mime']);
                    header("Content-Length: ".
$row['size']);
```

```
                    header("Content-Disposition:
attachment; filename=". $row['name']);

                    // Print data
                    echo $row['data'];
                }
                else {
                    echo 'Error! No image exists with that
ID.';
                }

                // Free the mysqli resources
                @mysqli_free_result($result);
            }
            else {
                echo "Error! Query failed: <pre>{$dbLink-
>error}</pre>";
            }
            @mysqli_close($dbLink);
        }
    }
    else {
        echo 'Error! No ID was passed.';
    }
    ?>
```