Evenimentele JavaScript

Utilizând JavaScript, putem crea pagini web dinamice. Evenimentele sunt acţiuni ce pot fi detectate de JavaScript. Fiecare element dintr-o pagină web are un anumit număr de evenimente care pot declanşa un script. Spre exemplu, putem utiliza evenimentul onClick al unui buton pentru a indica ce funcţie va fi executată dacă utilizatorul acţionează butonul respectiv. Evenimentele sunt definite în tagurile HTML.

Exemple de evenimente

- Un click de mouse
- Încărcarea unei pagini web sau a unei imagini
- Mişcarea mouse-ului peste o anumită zonă din pagina web
- Selectarea unui câmp de intrare dintr-un formular HTML
- Submiterea unui formular HTML
- Apăsarea unei taste

Obs: Evenimentele sunt în mod normal asociate cu funcţii, care nu vor fi executate înainte de a avea loc evenimentul.

Evenimentele onLoad şi onUnload

Aceste evenimente sunt declanşate când utilizatorul intră într-o pagină web, respectiv când părăseşte pagina.

Evenimentul onLoad este folosit în mod frecvent pentru a detecta tipul şi versiunea browserului utilizatorului şi a încărca varianta de pagină potrivită cu aceste informaţii.

Ambele evenimente sunt folosite frecvent pentru a stabili ce cookies vor fi setate când utilizatorul intră în sau părăseşte pagina. Spre exemplu, puteţi întreba care este numele utilizatorului când acesta vizitează prima dată pagina.

Numele oferit de utilizator este memorat într-un cookie.

Evenimentele onFocus, onBlur şi onChange

Aceste evenimente sunt utilizate frecvent împreună cu validarea câmpurilor unui formular.

Exemplul următor ilustrează utilizarea evenimentului onChange. Funcţia verificaEmail() va fi apelată ori de câte ori utilizatorul modifică conţinutul unui câmp:

<input type="text" size="30" id="email" onchange="verificaEmail()">

Evenimentul onSubmit

Acest eveniment este utilizat pentru a valida toate câmpurile unui formular înainte de trimiterea lui către server.

Exemplul următor ilustrează utilizarea evenimentului onSubmit. Funcţia verificaFormular() va fi apelată atunci când utilizatorul acţionează butonul submit din formular. Dacă valorile introduse în câmpuri nu sunt valide, trimiterea formularului va fi anulată. Funcţia verificaFormular() returnează true sau false. Dacă funcţia va returna valoarea false, trimiterea formularului va fi anulată:

<form method="post" action="aaa.htm" onsubmit="return verificaFormular()">

Evenimentele onMouseOver şi onMouseOut

Aceste evenimente sunt utilizate frecvent pentru a crea butoane „animate”.

În exemplul următor va fi afişată o casetă de alertă când este detectat un eveniment onMouseOver:

```
<a href=http://www.google.com onmouseover="alert('Un eveniment onMouseOver detectat');return false">
<img src="flowers.gif" alt="Flowers" />
</a>
```

Instrucţiunea try...catch

Când navigăm prin paginile web de pe internet, pot să apară mesaje de eroare la încărcarea unei pagini. În acest caz, uzual, apare o casetă de alertare JavaScript care ne anunţă că s-a detectat o eroare de execuţie (runtime error). Aceste mesaje sunt utile pentru proiectanţii paginilor web, nu şi pentru vizitatori care, de obicei, părăsesc pagina respectivă.

Instrucţiunea try...catch vă permite să testaţi blocurile de cod pentru a depista erorile. Blocul try conţine codul ce trebuie executat, iar blocul catch conţine codul ce va fi executat dacă apare o eroare.

Sintaxă:

```
try
{
```

```
codul ce trebuie executat
}
catch(err)
{
gestionarea erorilor
}
```

Exemple

În exemplul următor ar trebui afişată o casetă de alertare cu mesajul "Bine ati venit!" când butonul este acţionat. Totuşi, în corpul funcţiei mesaj() există o eroare, cuvântul rezervat alert este scris greşit. Această eroare va fi detectată de JS. Blocul catch sesizează eroarea şi execută un cod special pentru a o rezolva. Acest cod afişează un mesaj de eroare pentru a informa utilizatorul ce se întâmplă. Dacă utilizatorul apasă butonul OK, încărcarea paginii va continua fără probleme:

```
<html>
<head>
<script type="text/javascript">
var txt="";
function mesaj()
{
try
{
adddlert("Bine ati venit!");
}
catch(err)
{
text="In aceasta pagina este o eroare.\n\n";
text+="Descrierea erorii: " + err.description +"\n\n";
text+="Pentru a continua apasati OK.\n\n";
alert(text);
}
}
</script>
</head>
<body>
<h3>Utilizarea instructiunii try..catch pentru sesizarea erorilor</h3> <hr/>
<input type="button" value="Vedeti mesajul" onclick="mesaj()" />
</body>
</html>
```

În exemplul următor alert este de asemenea scris greşit. Blocul catch utilizează o casetă de confirmare pentru a afişa un mesaj care informează utilizatorii că pot apăsa OK pentru a continua să viziteze pagina în care a fost depistată eroarea sau pot apăsa Cancel dacă doresc să se întoarcă

la pagina principală (homepage). Dacă metoda confirm returnează false (utilizatorul a acţionat butonul Cancel), atunci utilizatorul este redirectat. Dacă confirm returnează true, codul din blocul catch nu are nici-un efect:

```
<html>
<head>
<script type="text/javascript">
var txt="";
function mesaj()
{
try
{
adddlert("Bine ati venit!");
}
catch(err)
{
text="In aceasta pagina este o eroare.\n\n";
text+="Apasati OK daca doriti sa continuati vizualizarea paginii,\n";
text+="sau Cancel pentru a va intoarce la pagina principala.\n\n";
if(!confirm(text))
{
document.location.href="http://carpenmanuela.wik.is/";
}
}
}
</script>
</head>
<body>
<h3>Un alt exemplu de utilizare a instructiunii  try..catch</h3> <hr/>
<input type="button" value="Vedeti mesajul"
onclick="mesaj()" />
</body>
</html>
```

Instrucţiunea throw

Această instrucţiune vă permite să creaţi o excepţie. Dacă o utilizaţi împreună cu instrucţiunea try...catch, puteţi controla execuţia programului şi afişa mesaje de eroare adecvate.

Sintaxă:

throw(exceptie)

Argumentul exceptie poate fi un şir de caractere, un număr întreg, o valoare booleană sau un obiect.

Exemplu:

Exemplul următor testează valoarea variabilei x. Dacă valoarea este mai mare decât 10, mai mică decât 10 sau nu este un număr, blocul throw aruncă o eroare. Această eroare este prinsă de blocul catch care afişează un mesaj corespunzător:

```
<html>
<body>
<h3>Utilizarea instructiunii throw pentru tratarea corespunzatoare a erorilor</h3> <hr/>
<script type="text/javascript">
var x=prompt("Introduceti un numar cuprins intre 0 si 10:","");
try
{
if(x>10)
{
throw "Err1";
}
else if(x<0)
{
throw "Err2";
}
else if(isNaN(x))
{
throw "Err3";
}
}
catch(er)
{
if(er=="Err1")
{
alert("Eroare! Valoarea este prea mare");
}
if(er=="Err2")
{
alert("Eroare! Valoarea este prea mică");
}
if(er=="Err3")
{
alert("Eroare! Valoarea nu este un numar");
}
}
</script>
</body>
</html>
```

Exemplu:

Exemplul următor ilustrează utilizarea evenimentului onerror.

```
<html>
<head>
<script type="text/javascript">
onerror=handleErr;
var txt="";
function handleErr(msg,url,l)
{
txt="In aceasta pagina este o eroare.\n\n";
txt+="Eroare: " + msg + "\n";
txt+="URL: " + url + "\n";
txt+="Linie: " + l + "\n\n";
txt+="Pentru a continua apasati OK.\n\n";
alert(txt);
return true;
}
function message()
{
adddlert("Bine ai venit!");
}
</script>
</head>
<body>
<h3>Exemplu de utilizare a evenimentului onerror</h3>
<hr/>
<input type="button" value="Afiseaza mesajul" onclick="message()" />
</body>
</html>
```

Obiectele JavaScript

JavaScript este un limbaj de programare orientat pe obiecte (POO). Un limbaj POO vă permite să vă definiţi propriile obiecte şi propriile tipuri de variabile.

Proprietăţi

Proprietăţile (date membru) sunt valori asociate cu un obiect.

În exemplul următor, utilizăm proprietatea length a obiectului String (şir de caractere) pentru a determina numărul de caractere memorate într-un şir:

```
<script type="text/javascript">
var txt="Bine ati venit!";
document.write(txt.length);
</script>
```

Codul de mai sus va afişa valoarea: 15

Metode

Metodele sunt acţiuni ce pot fi realizate cu un obiect.

În exemplul următor, utilizăm metoda UpperCase() a obiectului String pentru a afişa un text cu litere mari:

```
<script type="text/javascript">
var txt="Bine ati venit!";
document.write(txt.toUpperCase());
</script>
```

Codul de mai sus va afişa şirul: BINE ATI VENIT!

Obiectul String

Obiectul String este folosit pentru a manipula secvenţe de caractere (text). Un obiect String este creat cu instrucţiunea new String().

Sintaxa:

var txt = new String(string);

sau mai simplu:

var txt = string;

Proprietăţile obiectului String

constructor

Returnează funcţia care a creat prototipul obiectului String

length

Returnează lungimea şirului

prototype

Permite adăugarea de proprietăţi şi metode unui obiect

Metodele obiectului String

charAt() Returnează caracterul cu indexul specificat

charCodeAt()

Returnează codul Unicode al caracterului cu indexul

specificat

concat()

Concatenează două sau mai multe şiruri şi returnează

şirul obţinut

fromCharCode()

Converteşte valori Unicode în caractere

indexOf()

Returnează poziţia primei apariţii a unui subşir într-un şir

lastIndexOf()

Returnează poziţia ultimei apariţii a unui subşir într-un şir

match()

Caută potrivirile dintre un subşir şi un string şi returnează subşirul sau null (dacă subşirul nu este găsit)

replace()

Caută toate apariţiile unui subşir într-un şir şi le înlocuieşte cu un nou subşir

search()

Caută potrivirea dintre un subşir şi un şir şi returnează poziţia în care apare potrivirea

slice()

Elimină o porţiune din şir şi returnează şirul extras

split()

Împarte un şir în subşiruri pe baza unui caracter separator

substr()

Extrage dintr-un şir secvenţa de caractere care începe într-o anumită poziţie şi are o anumită lungime

substring()

Extrage dintr-un şir caracterele situate între două poziţii

toLowerCase()

Converteşte un şir în litere mici

toUpperCase()

Converteşte un şir în litere mari

valueOf()

Returnează valoarea primară a unui obiect String

Metode împachetate în taguri HTML

Aceste metode returnează şirul împachetat în tagurile HTML potrivite.

anchor()

Creează o ancoră

big()

Afişează şirul cu font mare

blink()

Afişează un şir care clipeşte

bold()

Afişează şirul cu font bold

fixed()

Afişează şirul cu un font cu pas fix

fontcolor()

Afişează şirul folosind o anumită culoare

fontsize()

Afişează şirul cu o anumită dimensiune a fontului

italics()

Afişează şirul cu font italic

link()

Afişează şirul ca hiperlegătură

small()

Afişează şirul cu font mic

strike()

Afişează şirul ca tăiat

sub()

Afişează şirul ca subscript (indice)

sup()

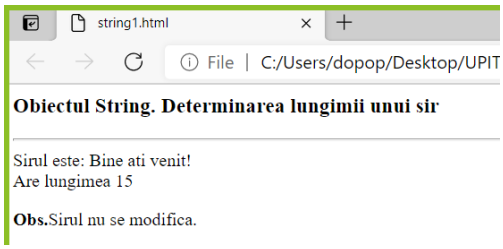
Afişează şirul ca superscript (exponent)

Exemplul 1

Ilustrează utilizarea proprietăţii length pentru a determina lungimea unui şir.

```
<html>
<body>
<h3>Obiectul String. Determinarea lungimii unui sir</h3> <hr/>
<script type="text/javascript">
var txt="Bine ati venit!";
document.write("Sirul este: "+txt+"<br/>");
```

```
document.write("Are lungimea "+txt.length);
</script>
<p><b>Obs.</b>Sirul nu se modifica.</p>
</body>
</html>
```

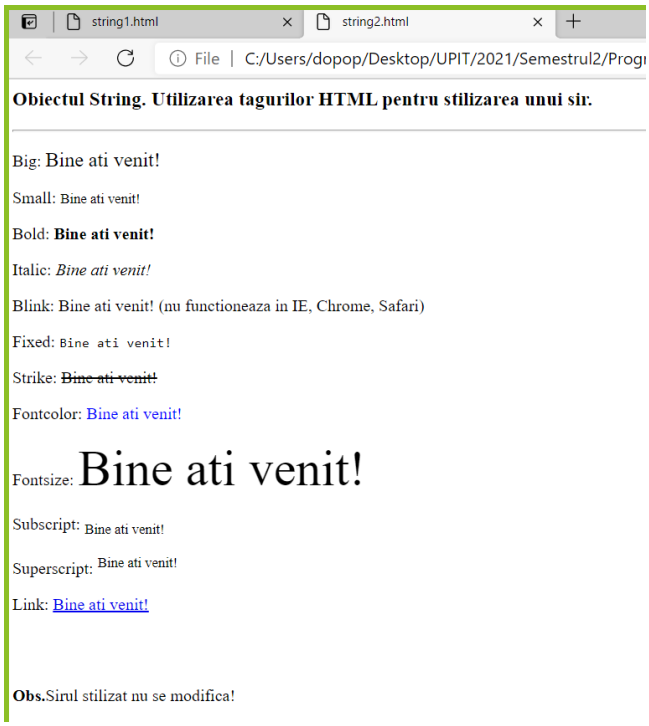


Exemplul 2

Ilustrează cum se utilizează tagurile HTML pentru a stiliza un şir.

```
<html>
<body>
<h3>Obiectul String. Utilizarea tagurilor HTML pentru stilizarea unui sir.</h3> <hr/>
<script type="text/javascript">
var txt="Bine ati venit!";
document.write("<p>Big: " + txt.big() + "</p>");
document.write("<p>Small: " + txt.small() + "</p>");
document.write("<p>Bold: " + txt.bold() + "</p>");
document.write("<p>Italic: " + txt.italics() + "</p>");
document.write("<p>Blink: " + txt.blink() + " (nu functioneaza in IE, Chrome, Safari)</p>");
document.write("<p>Fixed: " + txt.fixed() + "</p>");
document.write("<p>Strike: " + txt.strike() + "</p>");
document.write("<p>Fontcolor: " + txt.fontcolor("Blue") + "</p>");
document.write("<p>Fontsize: " + txt.fontsize(14) + "</p>");
```

```
document.write("<p>Subscript: " + txt.sub() + "</p>");

document.write("<p>Superscript: " + txt.sup() + "</p>");

document.write("<p>Link: " + txt.link("http://www.google.com") + "</p>");

</script>

<br/> <br/>

<p><b>Obs.</b>Sirul stilizat nu se modifica!</p>

</body>

</html>
```



Exemplul 3

Ilustrează cum se utilizează metoda concat() pentru a concatena şiruri.

Concatenarea a două şiruri:

```
<html>

<body>

<h3>Obiectul String. Concatenarea a doua siruri.</h3>
```

```
<hr/>
<script type="text/javascript">
var txt1 = "Buna ";
var txt2 = "ziua!";
document.write("Primul sir este: "+txt1+"<br/>");
document.write("Al doilea sir este: "+txt2+"<br/>");
document.write("Sirul concatenat este: "+txt1.concat(txt2)+"<br/>");
</script>
<p><b>Obs.</b>Sirurile concatenate nu se modifica. Rezultatul concatenarii poate fi pastrat intr-un nou sir.</p>
</body>
</html>
```

Concatenarea a trei şiruri:

```
<html>
<body>
<h3>Obiectul String. Concatenarea a trei siruri.</h3>
<hr/>
<script type="text/javascript">
var txt1="Buna ";
var txt2="ziua!";
var txt3=" Bine ati venit!";
document.write("Primul sir este: "+txt1+"<br/>");
document.write("Al doilea sir este: "+txt2+"<br/>");
```

```
document.write("Al treilea sir este: "+txt3+"<br/>");

document.write("Sirul concatenat este: "+txt1.concat(txt2,txt3)+"<br/>");

</script>

<p><b>Obs.</b>Sirurile concatenate nu se modifica. Rezultatul concatenarii poate fi pastrat intr-un nou sir.</p>

</body>

</html>
```

Exemplul 4

Ilustrează cum se utilizează metoda indexOf() pentru a determina poziţia primei apariţii a unei valori într-un şir.

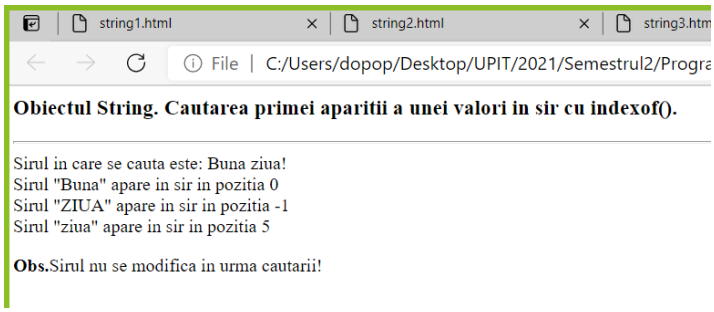
```
<html>

<body>

<h3>Obiectul String. Cautarea primei aparitii a unei valori in sir cu indexof().</h3> <hr/>

<script type="text/javascript">

var str="Buna ziua!";

document.write("Sirul in care se cauta este: "+str+"<br/>");

document.write("Sirul \"Buna\" apare in sir in pozitia "+str.indexOf("Buna") + "<br />");

document.write("Sirul \"ZIUA\" apare in sir in pozitia "+str.indexOf("ZIUA") + "<br />");

document.write("Sirul \"ziua\" apare in sir in pozitia "+str.indexOf("ziua"));

</script>

<p><b>Obs.</b>Sirul nu se modifica in urma cautarii!</p>
```

</body>

</html>

Valorile afişate sunt: 0 -1 5.

Obs. Dacă valoarea nu apare în şir, valoarea returnată este -1. Şirurile sunt indexate de la 0.



Exemplul 5

Ilustrează cum se utilizează metoda match() pentru a căuta un subşir într-un şir. Metoda returnează subşirul, dacă este găsit, sau valoarea null, dacă subşirul nu este găsit în şir.

<html>

<body>

<h3>Obiectul String. Cauta unui subsir intr-un sir cu match().</h3> <hr/>

<script type="text/javascript">

var str="Hello world!";

document.write("Sirul in care se cauta este: "+str+"
");

document.write("Sirul cautat: "+"world"+". ");

document.write("Valoarea returnata: "+str.match("world") + "
");

document.write("Sirul cautat: "+"World"+". ");

document.write("Valoarea returnata: "+str.match("World") + "
");

document.write("Sirul cautat: "+"worlld"+". ");

```
document.write("Valoarea returnata: "+str.match("worlld") + "<br />");

</script>

<p><b> Obs.</b>Sirul nu se modifica in urma cautarii. Rezultatul poate fi memorat intr-o variabila.</p>

</body>

</html>
```

Obiectul Date

Obiectul Date este utilizat pentru a lucra cu date calendaristice şi ore. Un obiect de tip Date este creat cu instrucţiunea new Date(). Sunt patru metode de a instanţia un obiect Date:

```
var d = new Date();

var d = new Date(milisecunde);

var d = new Date(dataString);

var d = new Date(an, luna, zi, ore, minute, secunde, milisecunde);
```

Setarea datei

Putem manevra uşor datele calendaristice folosind metodele obiectului Date.

În exemplul următor, data este setată la 18 aprilie 2016:

```
var myDate=new Date();

myDate.setFullYear(2016,4,18);
```

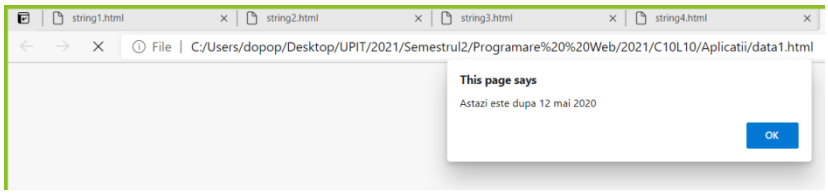
În exemplul următor, data este setată la şapte zile în viitor:

```
var myDate=new Date();

myDate.setDate(myDate.getDate()+7);
```

Compararea a două date calendaristice

Exemplul următor compară data curentă cu 12 mai 2020:

```
var myDate=new Date();
myDate.setFullYear(2020,5,12);
var today = new Date();
if (myDate>today)
{
alert("Astazi este inainte de 12 mai 2020");
}
else
{
alert("Astazi este dupa 12 mai 2020");
}
```



Metodele obiectului **Date**

getDate() Returnează ziua din lună (între 1 şi 31)

getDay() Returnează ziua din săptămână (0-6)

getFullYear() Returnează anul (patru cifre)

getHours() Returnează ora (0-23)

getMilliseconds() Returnează milisecundele (0-999)

getMinutes() Returnează minutele (0-59)

getMonth() Returnează luna (0-11)

getSeconds() Returnează secundele (0-59)

getTime() Returnează numărul de milisecunde scurse de la 1.01.1970

getTimezoneOffset() Returnează diferenţa dintre GMT şi timpul local, în minute

parse() Analizează(parsează) o dată ca şir de caractere şi returnează numărul de milisecunde scurse de la 1.01.1970

setDate() Setează data din lună (1-31)

setFullYear() Setează anul (patru cifre)

setHours() Setează ora (0-23)

setMilliseconds() Setează milisecundele (0-999)

setMinutes() Setează minutele (0-59)

setMonth() Setează lunile (0-11)

setSeconds() Setează secundele (0-59)

setTime() Setează o dată şi o oră adunând sau scăzând un anumit număr de milisecunde la/din 1.01.1970

toDateString() Converteşte porţiunea corespunzătoare datei calendaristice dintr-un obiect Date intr-un şir de caractere

toString() Converteşte un obiect Date într-un şir de caractere

toTimeString() Converteşte porţiunea corespunzătoarea timpului dintr-un obiect Date într-un şir de caractere

valueOf() Returnează valoarea primară a unui obiect Date

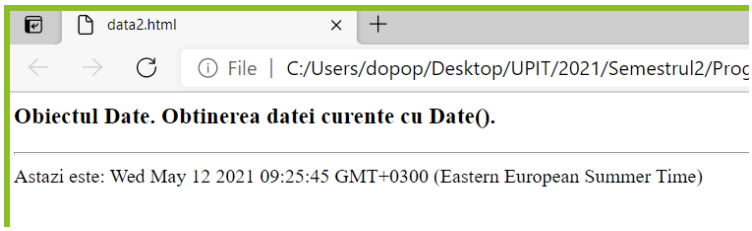
==========

Exemplul 1

Ilustrează utilizarea metodei Date() pentru a obţine data curentă.

```
<html>

<body>

<h3>Obiectul Date. Obtinerea datei curente cu Date().</h3> <hr/>

<script type="text/javascript">

document.write("Astazi este: "+Date());

</script>
```

```
</body>

</html>
```



Exemplul 2

Ilustrează utilizarea metodei getTime() pentru a calcula anii scurşi din 1970 până în prezent.

```
<html>

<body>

<h3>Obiectul Date. Utilizarea metodei getTime().</h3>

<hr/>

<script type="text/javascript">

var d=new Date();

document.write("Au trecut "+d.getTime() + " milisecunde din 01.01.1970 si pana acum.");

</script>

</body>

</html>
```

Exemplul 3

Ilustrează utilizarea metodei setFullYear() pentru a seta o dată specifică.

```
<html>

<body>

<h3>Obiectul Date. Setarea datei cu setFullYear().</h3> <hr/>

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

```
var d = new Date();
d.setFullYear(2010,1,19);
document.write("Data a fost setata la "+d);
</script>
</body>
</html>
```

Exemplul 4

Ilustrează utilizarea metodei toString() pentru a converti data curentă într-un şir de caractere.

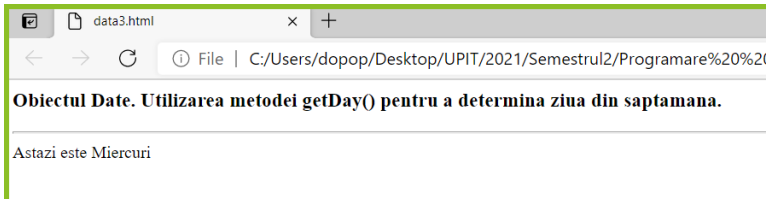
```
<html>
<body>
<script type="text/javascript">
var d=new Date();
document.write(d.toString());
</script>
</body>
</html>
```

Exemplul 5

Ilustrează utilizarea metodei getDay() şi a unui tablou pentru a scrie denumirea zilei din săptămână curente.

```
<html>
<body>
<h3>Obiectul Date. Utilizarea metodei getDay() pentru a determina ziua din saptamana.</h3>
<hr/>
<script type="text/javascript">
```

```
var d=new Date();
var weekday=new Array(7);
weekday[0]="Duminica";
weekday[1]="Luni";
weekday[2]="Marti";
weekday[3]="Miercuri";
weekday[4]="Joi";
weekday[5]="Vineri";
weekday[6]="Sambata";
document.write("Astazi este " + weekday[d.getDay()]);
</script>
</body>
</html>
```

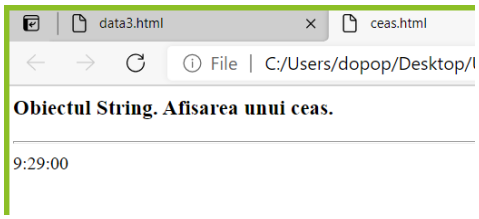


Exemplul 6

Ilustrează cum se poate afişa un ceas într-o pagină web.

```
<html>
<head>
<script type="text/javascript">
function ceas()
{
var today=new Date();
var h=today.getHours();
```

```
var m=today.getMinutes();
var s=today.getSeconds();
//functia urmatoare adauga un zero in fata
//numerelor<10
m=verifica(m);
s=verifica(s);
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
t=setTimeout('ceas()',500);
}
function verifica(i)
{
if (i<10)
{
i="0" + i;
}
return i;
}
</script>
</head>
<body onload="ceas()">
<h3>Obiectul String. Afisarea unui ceas.</h3> <hr/>
<div id="txt"></div>
</body>
</html>
```



Obiectul Array

Un tablou este o variabilă specială care poate păstra la un moment dat mai multe valori de un anumit tip. Dacă aveţi o listă de elemente, animale de companie de exemplu, aţi putea păstra valorile în variabile simple, ca în exemplul următor:

pet1="Caine";

pet2="Pisica";

pet3="Papagal";

Desigur, problema se complică dacă aveţi de memorat zeci, sau sute de valori. Cea mai bună soluţie este să folosiţi tablouri. Un tablou poate reţine toate valorile sub un singur nume şi puteţi accesa fiecare valoare stocată în tablou folosind numele tabloului şi indexul valorii.

Crearea unui tablou

Un tablou poate fi definit în trei moduri:

1:

var pets=new Array();

//tablou obisnuit

pets[0]="Caine";

pets[1]="Pisica";

pets[2]="Papagal";

2:

var pets=new Array("Caine","Pisica","Papagal");

//tablou condensat

3:

var pets=["Caine","Pisica","Papagal"];

//tablou literal

Obs: Dacă în tablou stocaţi valori numerice sau logice, tipul tabloului va fi Number sau Boolean, în loc de String.

Accesarea elementelor dintr-un tablou

Puteţi accesa un element dintr-un tablou precizând numele tabloului şi indexul elementului. Primul element din tablou are indexul 0.

Următoarea linie de cod

document.write(pets[0]);

va afişa şirul: Caine

Modificarea valorilor dintr-un tablou

Pentru a modifica o valoare dintr-un tablou, este suficient să atribuiţi o nouă valoare elementului respectiv, ca în exemplul următor:

pets[0]="Iguana";

Proprietăţile obiectului Array

constructor Returnează funcţia care a creat prototipul obiectului Array

length Setează sau returnează numărul elementelor stocate în tablou

prototype Permite adăugare de proprietăţi şi metode unui obiect

Metodele obiectului Array

concat() Concatenează două sau mai multe tablouri şi returnează tabloul obţinut

join() Concatenează toate elementele unui tablou într-un şir de caractere

pop() Înlătură ultimul element dintr-un tablou şi returnează respectivul element

push() Adaugă noi elemente la sfârşitul unui tablou şi returnează noua lungime a tabloului

reverse() Răstoarnă ordinea elementelor dintr-un tablou

shift() Înlătură primul element dintr-un tablou şi returnează respectivul element

slice() Selectează o parte dintr-un tablou şi returnează elementele selectate

sort() Sortează elementele unui tablou

splice() Adaugă/Înlătură elemente dintr-un tablou.

toString() Converteşte un tablou în şir de caractere şi returnează rezultatul

unshift() Adaugă noi elemente la începutul unui tablou şi returnează noua lungime a tabloului

valueOf() Returnează valoarea primară a unui tablou

Exemplul 1

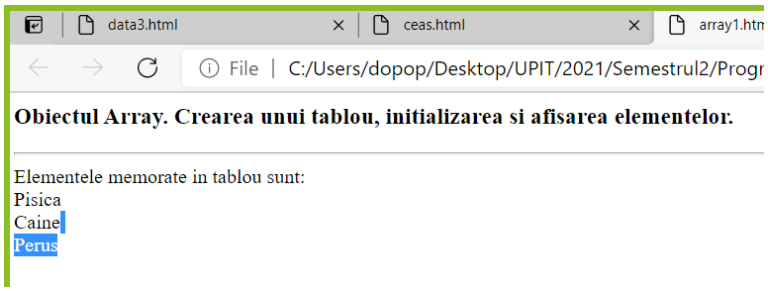
Ilustrează crearea unui tablou, atribuirea de valori şi afişarea elementelor tabloului.

```
<html>
<body>
<h3>Obiectul Array. Crearea unui tablou, initializarea si afisarea elementelor.</h3> <hr/>
<script type="text/javascript">
var pets = new Array();
pets[0] = "Pisica";
pets[1] = "Caine";
pets[2] = "Perus";
document.write("Elementele memorate in tablou sunt:"+"<br/>");
for (i=0;i<pets.length;i++)
{
document.write(pets[i] + "<br />");
}
```

</script>

</body>

</html>



Exemplul 2

Ilustrează utilizarea instrucţiunii for...in pentru a parcurge elementele unui tablou.

<html>

<body>

<h3>Obiectul Array. Afisarea elementelor unui tablou cu instructiunea for..in.</h3> <hr/>

<script type="text/javascript">

var x;

var pets = new Array();

pets[0] = "Pisica";

pets[1] = "Caine";

pets[2] = "Perus";

document.write("Elementele memorate in tablou sunt:"+"
");

for (x in pets)

{

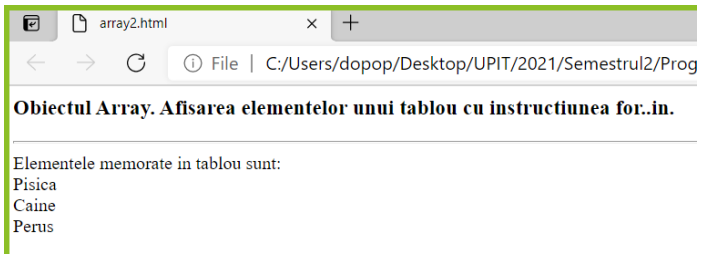
document.write(pets[x] + "
");

}

</script>

</body>

</html>



Exemplul 3

Ilustrează utilizarea metodei concat() pentru a concatena trei tablouri.

<html>

<body>

<h3>Obiectul Array. Concatenarea a trei tablouri cu concat().</h3> <hr/>

<script type="text/javascript">

var parinti = ["Maria", "George"];

var copii = ["Elena", "Mihai"];

var frati = ["Paul", "Dan"];

var familie = parinti.concat(copii,frati);

document.write("Parinti: "+parinti+"
");

document.write("Copii: "+copii+"
");

document.write("Frati: "+frati+"
");

document.write("Familia: "+familie);

</script>

Obs.Tablourile concatenate nu se modifica. Rezultatul concatenarii este un nou tablou.

</body>

</html>


array3.html
File | C:/Users/dopop/Desktop/UPIT/2021/Semestrul2/Prograr
Obiectul Array. Concatenarea a trei tablouri cu concat().
Parinti: Maria,George
Copii: Elena,Mihai
Frati: Paul,Dan
Familia: Maria,George,Elena,Mihai,Paul,Dan
Obs.Tablourile concatenate nu se modifica. Rezultatul concatenarii este un nou tablou.