Cursul 1 - Începuturile Web-ului

Unul dintre cele mai importante și de succes servicii ale Internetului, **World-Wide-Web-ul**, pe scurt **Web** sau **spațiul WWW**, a fost instituit de CERN (European Organization for Nuclear Research) în anul 1989.

În cazul Web-ului, două mari direcții precursoare trebuie menționate:

- dezvoltarea hipertextului (sau a procesării computerizate a documentelor electronice complexe);
- dezvoltarea protocoalelor Internet care au făcut posibilă comunicarea globală dintre rețelele de calculatoare.

Web reprezintă un sistem de distribuție locală sau globală a informațiilor hipermedia.

Funcționarea rețelelor client / server:

Din punct de vedere tehnic, Web-ul pune la dispoziție un sistem global standardizat de comunicare multimedia, informațiile fiind organizate asociativ și distribuite în funcție de cererile utilizatorului, funcționând conform modelului:

a) Clientul solicită un fișier;

b) **[Lipsește text]**

Clienți și serverele Web:

În prezent, pentru a avea acces la spațiul WWW, utilizatorii trebuie să folosească un program client sub numele de **navigator** sau **browser** și să cunoască localizarea documentelor dorite. Rolul unui browser este de a permite vizualizarea oricărei resurse disponibile în Web. Identificarea acestor resurse se face pe baza unei adrese unice numite **URL** (Uniform Resource Locator).

Structura unui URL este următoarea:

nume_protocol://nume_calculator_gazda/cale_fisier

Un browser nu face decât să contacteze în orice moment un server. El stabilește o conexiune, recuperează o pagină de informație, închide conexiunea și afișează informația pentru utilizatori. Serverele trebuie să poată fi în permanență contactate și trebuie să fie de acord să furnizeze informația oricărui client care o solicită, astfel utilizatorul poate naviga prin WWW fără să știe de unde se găsește informația.

Exemple de navigatoare:

- pentru **Windows** cele mai cunoscute sunt: Internet Explorer, Microsoft Edge, Opera, Fire Fox, etc.
- pentru **Linux** cele mai cunoscute sunt: Opera, Lynx sau Links (doar mod text).

Modul de dialogare se realizează prin prisma unei mulțimi de reguli de comunicare prin rețea denumită protocol. În cazul Web-ului, acest protocol este HTTP și reprezintă un protocol bazat pe stivă de protocoale TCP / IP.

Există două tipuri majore de site-uri:
a) site-urile Web – site-uri de pagini Web corelate și resursele la care se referă acestea.
b) site-urile FTP – servicii configurabile care furnizează acces la fișiere publice (documentații, programe, informații) prin intermediul protocolului FTP (File Transfer Protocol).

Tehnologia HTML

Un document Web pe care dorim să-l accesăm trebuie să fie în prealabil formatat cu ajutorul unui limbaj special denumit **HTML** (HyperText Markup Language).

Utilizarea HTML-ului propune scrierea textului, tabelelor și referințelor la imaginile încadrate în pagina Web, iar apoi se adaugă tag-urile HTML pentru a descrie amplasarea obiectelor în pagină. HTML-ul dispune de un set predefinit de tag-uri, deci nu se pot crea tag-uri proprii.

Exemple:

```
1. < b > ... < / b> -text boldat.
2. < u > ... < / u> -text subliniat.
3. < i > ... < / i> -text italic.
4. < p > ... < / p> -paragraf nou.
5. < table > ... < / table> -introducerea unui tabel.
```

Pentru editarea documentelor HTML se pot folosi procesoare de text precum:
- NotePad (NotePad++);
- Word Pad;
- Microsoft Word Document;
- WWW3.

Structura unui document HTML:

Un document HTML are un antet (head) și un corp (body). Exemplu:

```
<html>
	<head>
		<title> ... </title>
		<base> ... </base>
		<link> ... </link>
		<meta> ... </meta>
		<script> ... </script>
		<style> ... </style>
	</head>

	<body>
		<script> ... </script>
	</body>
</html>
```

Cursul 2

Marcajul de specificare a documentului de tip HiperText: *<HTML> … </HTML>.*

Pentru a specifica editorului de text în care lucrăm că dorim o pagină WEB folosindu-ne de limbajul HTML, vom cuprinde tot textul pe care urmează să-l scriem în marcajul *<HTML> … </HTML>*.

Documentul HTML se va secționa apoi în două părți:

1. **Marcajul de specificare al antetului documentului de tip HiperText**: <HEAD> … </HEAD>. Acesta delimitează informațiile referitoare la crearea paginii WEB, informații ce nu vor apărea la vizualizarea paginii respective cu un program de navigare (Internet Explorer).

Marcajul *<TITLE> … </TITLE>* delimitează un nume pe care noi îl dorim să-l dăm paginii ce urmează să o realizăm și care va apărea pe bara de stare a programului de navigare. Acest nume se dă de obicei pentru o claritate mai mare în ceea ce privește conținutul paginii.

2. Marcajul de specificare a corpului documentului de tip HiperText: <BODY> … </BODY>. Acest marcaj cuprinde conținutul paginii WEB ce va apărea la încărcarea ei cu un program de navigare.

Parametrii marcajului <BODY> … </BODY>:

Parametrul **bgcolor**:

Culoarea fundalului paginii se poate stabili prin valoarea parametrului **bgcolor** specificat în marcajul <BODY> … </BODY>. Valoarea acestui parametru va fi numele în limba engleză a culorii, încadrat de ghilimele (de exemplu „RED” – roșu, „GREEN” – verde, „BLUE” – albastru etc.) sau codul culorii precedet de semnul **#** și încadrat, de asemenea, de ghilimele.

Exemplu:

```
<BODY bgcolor = ”blue”>
       Conținutul paginii
</BODY>

<BODY bgcolor = ”#ooooFF”>
       Conținutul paginii
</BODY>
```

O culoare se construiește prin combinarea a trei culori de bază: roșu, verde și albastru (modelul RGB); iar codul unei culori este format din 3 grupe de câte 2 caractere. Fiecare grup de două caractere reprezintă compoziția din fiecare culoare de bază, astfel #RRGGBB.

Valoarea poate fi 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F (sistemul de numerotație hexazecimal).

Parametrul **background**:

De asemenea se poate utiliza pentru fundalul unei pagini WEB o imagine a cărei adresă este specificată ca valoare a parametrului background specific marcajului *<BODY> .. </BODY>.*

Utilizarea marcajului background se face astfel: *<BODY background = URL> .. </BODY>,* unde URL este un șir de caractere ce reprezintă adresa imaginii ce va fi utilizată de navigator ca fundal pentru pagina WEB.

Observație:

Dacă fișierul se află în directorul curent atunci este suficient să dăm prin URL doar numele imagine.jpg, în caz contrar trebuie să specificăm toată calea de acces a acesteia (de exemplu: C:\...\imagine.jpg).

Parametrul **text**:

Utilizarea marcajului text se face astfel: *<BODY TEXT = culoare> … </BODY>.*

Stabilește culoarea textului din pagina WEB. Culoarea se poate specifica prin nume sau prin modelul RGB.

Parametrul **link**:

Utilizarea marcajului link se face astfel: *<BODY LINK = culoare> … </BODY>.*

Stabilește culoarea cu care vor fi scrise în text legăturile (link-urile) nevizitate, legături cu alte pagini WEB sau legăturile în alte porțiuni din pagina curentă. Culoarea se poate specifica prin nume sau prin modelul RGB.

Parametrul **vlink**:

Utilizarea marcajului vlink se face astfel: *<BODY VLINK = culoare> … </BODY>.*

Stabilește culoarea cu care vor fi scrise în text legăturile (link-urile) vizitate, legături cu alte pagini WEB sau legăturile în alte porțiuni din pagina curentă. Culoarea se poate specifica prin nume sau prin modelul RGB.

Parametrul **alink**:

Utilizarea marcajului alink se face astfel: *<BODY ALINK = culoare> … </BODY>.*

Stabilește culoarea cu care vor fi scrise în text legăturile (link-urile) în curs de vizitare (activă – cea asupra căreia este plasat cursorul mouse-ului), legătură cu alte pagini WEB sau legătură în alte porțiuni din pagina curentă. Culoarea se poate specifica prin nume sau prin modelul RGB.

Parametrii:

- **leftmargin** (valoare prestabilită: 10 pixeli);
- **rightmargin** (valoare prestabilită: 10 pixeli);
- **topmargin** (valoare prestabilită: 15 pixeli);
- **bottommargin** (valoare prestabilită: 0 pixeli).

Stabilesc marginea, reprezintă distanța în pixeli între conținut și marginea ferestrei browserului. Aceste atribuite sunt pentru Internet Explorer. În Netscape Navigator, avem doar două atribute de margine: **marginwidth** și **marginheight**.

Trecerea la linie nouă:

Trecerea pe un rând nou se poate face cu *
 (BREAK)*. El se poate folosi și în combinație cu parametrul CLEAR = opțiune, opțiune ce forțează trecerea la linie nouă, față de marginea specificată. Valorile posibile pentru opțiune sunt **left** (stânga), **right** (dreapta) sau **all** (toate).

Exemplu:

```
<body>
       <br clear = ”all”> … </br>
       <br clear = ”left”> … </br>
       <br clear = ”right”> … </br>
</body>
```

Specificarea unui paragraf:

Într-un document HTML, se face folosind pentru text delimitatorul de marcaj * … *. Acesta este un stil logic de text ce este utilizat pentru a preciza o destinație specifică pentru un anumit bloc de text.

Alinierea unui paragraf:

Pentru alinierea unui paragraf la stânga, la dreapta sau centrat putem folosi la marcajul * … *, opțiunea **align**, opțiune ce poate primi ca valoare unul din cuvintele: **left**, **right**, **center**. Valoarea trebuie încadrată între ghilimele.

Exemplu:

```
<p align = ”center”> Salut! </p>
<p align = ”left”> Salut! </p>
<p align = ”right”> Salut! </p>
```

Preformatarea unui text:

Într-un document HTML, se face folosind pentru text delimitatorul de marcaj *<pre> … </pre>*. Această etichetă păstrează formatul din documentul sursă.

Elementul *<blockquote> … </blockquote>* este destinat **introducerii unor citate lungi** într-o pagină Web. Acest element este afișat începând de pe un rând nou, cu spațiu alb suplimentar înainte, după, la stânga și la dreapta elementului. Pentru HTML 4.01 se recomandă un atribuit numit *<cite> … </cite>.*

Formatarea fontului:

Un font reprezintă un set de caractere (în general 256) cu un aspect grafic unitar. Exemple de fonturi: Arial, Courier, Times New Romans, etc.

Stiluri de afișare:

Fiecare font poate fi afișat pe ecran în mai multe stiluri cum ar fi normal, **îngroșat**, *înclinat* sau subliniat. Orice modificare făcută în cadrul unui document HTML trebuie inclusă între marcaje. Alte elemente folosite pentru formatul textului sunt:
- *Accentuare* (Emphasis) - Browser-ul de obicei arată acest element ca italic.
- **Strong** - Browser-ul de obicei arată acest element ca bold.
- <tt> Teletype </tt> - Ca și tag-ul PRE, browser-ul prezintă textul acesta într-un singur tip de font, indiferent de fontul ales cu atributul FACE, în interioru elementului font.
- <cite> Citație </cite> - Reprezintă un citat din document.

Centrarea textului se poate face și cu marcajul: *<CENTER> … </CENTER>.*

Elemetnul *<q> … </q>* este destinat introducerii citatelor **inline** (în interiorul rândului), adică la afișare, elementul *<q>* nu începe pe rând nou.

Nivele de titlu:

Un alt mod de a pune în evidență o porțiune dintr-un text, dar în mod special nivele de subtitlu al unui document sunt marcajele *<H**n**> … </H**n**>*, unde **n** poate lua valori de la 1 la 6. Cel mai înalt nivel este 1, apoi nivelul descrește până la nivelul 6, fiind cel mai mic nivel.

Alegerea tipului de font:

Pentru alegerea tipului de font cu care dorim să fie afișat textul folosim marcajul … împreună cu opțiunea **face** ce va primi ca valoare numele fontului ce intenționăm să-l folosim încadrat de ghilimele și anume: *<FONT face = ”Courier, Arial”> … </FONT>*. Textul va fi scris folosind fontul Courier dacă acesta este disponibil, iar în caz contrar va fi folosit fontul Arial.

Alegerea dimensiunii caracterelor:

Caracterele unei porțiuni de text pot fi mai mari sau mai mici, în funcție de contextul în care apar. În documentele de tip HTML schimbarea dimensiunii fonturilor se face adăugând la marcajul *<FONT> … </FONT>* opțiunea **size**, opțiune care va primi ca valoare un număr încadrat între ghilimele.

Exemplu:

```
<FONT face = ”Courier, Arial” size = ”4”> … </FONT>
```

Valoarea primită de **size** poate fi:
- un număr natural de la 1 la 7 ce va reprezenta dimensiunea fontului.
- relativ la dimensiunea curentă a caracterelor, size va primi ca valoare +n dacă dorim caracterele să fie cu n mărimi mai mari decât mărimea implicită sau valoarea -n dacă dorim caracterele să fie cu n mărimi mai mici.

Observație:

Dacă fontul este ales cu valoare mai mare ca numărul 7 el este scris cu dimensiunea 7 (dimensiunea fontului maxim).

Alegerea culorii fontului:

Culoarea de afișare a unei porțiuni de text se poate alege folosind opțiunea **color** ce este specifică tot marcajului *<FONT> … </FONT>.* Valoarea acestei opțiuni va fi numele în limba engleză al culorii încadrat de ghilimele sau codul culorii precedat de semnul **#** și încadrat de asemenea între ghilimele.

Exemple ce utilizează parametrii marcajului * … :*

```
<html>
	<head> <title> Citat </title> </head>
	<body>
		<font face = „Times New Romans, Arial” color = „red”> Constantin Brâncuși </font> used to say: <font size = „+5”> <q> I give you pure joy” </q> </font>
	</body>
</html>
```

```
<html>
	<head>
		<title> Titlu pagina </title>
	</head>

	<body>
		<font face = "Courier, Arial" size = "4" color = "red"> Text 1 scris cu rosu </font>
		sau
		<font face = "Courier, Arial" size = "+2" color = "#00FFFF> Text 2 scris cu verde marin </font>
	</body>
</html>
```

În limbajul HTML există două tipuri de elemente:

a) Elemente de nivel bloc, care sunt afișate cu începere de pe un rând nou (exemplu: <p>, <hr>,
).

b) Elemente de nivel inline, care sunt afișate pe același rând (exemplu: <q>, <b>).

Elemente de grupare (stil logic de text):

Eticheta * * este elementul de nivel bloc generic și poate fi utilizat într-o gamă largă de moduri. Aceasta are ca și atribuit align:

- * * aliniere la stânga a blocului * * (valoare prestabilită dacă nu se specifică alinierea.
- * * aliniere centrată a blocului * *
- * * aliniere la dreapta a blcoului * *
- * * aliniere la stânga și la dreapta a blocului * *

Eticheta ** este elementul inline generic și poate fi utilizat într-o gamă largă de situații.

Observație: Diferența între cele două elementul este că elementul * * este de tip bloc, iar ** este de tip inline. Elementul ** nu acceptă atributul **align**, deoarece este un atribuit de interior de rând.

Stiluri fizice de text:

Se poate modifica stilul de text și utilizând următoarele etichete:

Etichetă	**Descriere**	**Engleză**	
tt	Teleimprimator	Teletype	Tele Type
i	Cursiv	Italic	*Italic*
b	Aldin	Bold	**Bold**
Big	Font mai mare decât cel curent	Bigger Text	Bigger
Small	Font mai mic decât cel curent	Smaller Text	Small
u	Subliniat	Underlined	Subliniat
sub	La indice	Subscript	La$_{indice}$
sup	La exponent	Superscript	La^{exponent}
strike.s	Stil de text tăiat cu o linie	Strike Text	~~Big~~ sau ~~s~~

Exemplu:

```
<html>
	<head>
		<title> Titlu document </title>
	</head>

	<body>
		<p> <font size = "+1"> O singura marire </font> - normal
		<font size = "-1"> O singura micsorare </font> <br>
			<b> Bold </b>
			<i> Italic </i>
			<u> Subliniat </u>
		<font color = "FF0000"> Colorat </font> <br>
			<em> Accentuat </em> - <strong> Strong </strong>
			<tt> Tele Type </tt> </p>
			<cite> Citatie </cite> </p>
		<p> <strike> Text taiat </strike> </br>
			<big> Prezinta textul intr-un font mare </big> </br>
			<small> Prezinta textul intr-un font mai mic </small> </br>
			<sub> Aseaza textul in pozitia subscript </sub> normal
			</sup> Aseaza textul in pozitia superscript </sup> </br>
		</p>
	</body>
</html>
```

Linii de delimitare:

Un control **HTML** foarte util pentru organizarea unui **document Web** este marcajul de linie ***<HR>.*** Plasat oriunde, într-un document, acest marcaj are ca efect trasarea unei linii horizontale. Se poate folosi în combinație cu parametrii *SIZE* = opțiune, *WIDTH* = opțiune, *ALIGN* = opțiune, *COLOR* = opțiune și *NOSHADE*.

Comentarii în cadrul codului HTML:

Comentariile sunt texte încadrate între *<!--* și *-->*. Rolul comentariilor este de a face un document HTML mai ușor de înțeles prin plasarea unor explicații din loc în loc despre ceea ce realizează anumite secvențe din document. Ele nu vor fi afișate în cuprinsul paginii Web.

Caractere speciale:

Spațiul într-un document HTML se realizează cu ** **; (non breaking space). Pentru a scrie cu caractere românești putem folosi următoarele coduri:

- à
- á
- &acire;
- ã
- ä
- å

Observație: Dacă avem *é* folosim &egrave sau &Egrave pentru $\acute{E}$.

- pentru **Ș** avem *&#350* sau pentru **ș** avem *&#351*.
- pentru **Ț** avem *&#355* sau pentru **ț** avem *&#354*.

Liste:

HTML permite declararea de:

- Liste ordonate – elementele sunt automat numerotate.
- Liste neordonate – elementele sunt marcate printr-un semn special.
- Liste de definiții de termeni și definițiile lor asociate.

Listele sunt declarate printr-un marcaj *(<OL>, <UL>, <DL>*) în funcție de tipul listei, iar fiecare element al ei este marcat prin marcajul *<LI>.* (OL – Order List, UL – Unordered List, DL – Definition List, LI – List Item).

1. Liste ordonate:

Declararea unei liste se poate face cu ajutorul marcajului *<OL> … </OL>.* Modul de numerotare a elementelor din listă este determinat de parametrul **TYPE**, parametru ce poate lua valori:

1 - dacă dorim **numerotarea 1, 2, 3, …** (aceasta este numerotarea implicită (cifre arabe), adică dacă vom scrie doar *<OL> … </OL>*).

a – dacă dorim **numerotarea a, b, c, …**

A – dacă dorim **numerotarea A, B, C, …**
i – dacă dorim **numerotarea i, ii, iii, iv, …** (pentru numerotarea cu cifre romane).
I – dacă dorim **numerotarea I, II, III, …**

Prin intermediul parametrului **START** al marcajului *<OL> … </OL>* se poate specifica explicit valoarea de la care începe numerotarea. Dacă se dorește ca unui anumit element să i se schimbe numerotarea se poate face prin includerea parametrului **VALUE** în marcajul *<LI>* căruia i se asociază un număr ce reprezintă eticheta elementului din listă.

2. Liste neordonate:

Declararea unei liste neordonate se face cu marcajul *<UL>.* Schimbarea simbolului care precede elementele listei se face utilizând parametrul **TYPE** al marcajului *<UL>,* iar **TYPE** poate să ia diverse valori de genul: *„circle”, „disc”, „square”*.

3. Liste de definiții:

Declararea unei liste de definiții se face cu ajutorul marcajului *<DL> … </DL>* utilizat împreună cu marcajul ***<DT>*** pentru termenul de definit și marcajul ***<DD>*** pentru definiția propriu – zisă.

Exemplu:

```
<html>
	<head>
		<title> Liste de definitie </title>
	</head>

	<body>
		<h3> <b> LIste de definitie </b> </h3>
		<dl>
			<dt> Crearea paginilor WEB
			<dd> Ned Snell, Ed. Teora
			<dt> Totul despre HTML
			<dd> Rick Darnell, ed. Teora, Ed. Teora
		</dl>
	</body>
</html>
```

4. Liste în liste ordonate – liste îmbricate:

Declararea unei liste îmbricate se poate face cu ajutorul listelor ordonate și a celor neordonate.

Cursul 3 – Legături (link-urile)

Principala caracteristică a hipertextelor o constituie utilizarea legăturilor. Un **link** este o conexiune către o altă resursă Web (de exemplu, o imagine, o secvență video, un program), resursă care poate fi accesată din fereastra *browser-ului* printr-un simplu click.

Avem:

- **link-uri interne**;
- **link-uri externe**.

Marcajul principal pentru cele două referințe este: *<a> … ,* unde **a** vine de la *anchor* (ancoră). Acest marcaj trebuie să conțină parametrul *href = ”valoare”*, unde **valoare** este calea spre pagina curentă, spre o altă pagină Web sau spre un fișier cu o alta extensie.

Exemplu:

```
<a href = ”file:\\F:\Tehnologii web\laboratorul 5\Laborator 5.pdf> </a>
nume.doc
nume.jpg (gif, bmp)
```

Referințe interne:

Definirea țintelor (salturilor) într-un document HTML se poate face cu ajutorul ancorelor cu nume și sunt specificate în cadrul documentului HTML, prin utilizarea parametrului name.

Exemplu:

```
<a name = "valoare"> mesaj </a> unde valoare poate fi o secventa de caractere, numere
sau semne de punctuatie.
<a href = "nume.html#nume_legatura"> legatura la titlu 1 </a>
<a name = "nume_legatura"> titlu 1 </a>
```

Referințe către site-uri din Internet:

În acest caz parametrul **href** va conține adresa paginii de legătură.

Exemplu: `<a href = ”http://www.yahoo.com”> legătură la pagina principală a motorului de căutare YAHOO </a>`

Trimiterea unui mesaj e-mail:

Exemplu: `<a href = ”mailto:nume@yahoo.com”> legătură la adresa de mail </a>.`

Acesta va permite trimiterea unui mesaj de e-mail.

Exemplu de referințe interne: *fișierul link1.html*.

Imagini:

Pentru a include o imagine într-un document HTML, se folosește tag-ul (marcajul): *<img src = ”nume_imagine.extensie”>*, unde parametrul **src** ete folosit pentru referirea fișierului imagine. Extensiile imaginilor pot fi ***.gif, *.jpg, *.jpeg, *.bmp**.

O imagine poate fi redimensionată cu ajutorul parametrilor:

- *WIDTH = ”val”* (lățime);
- *HEIGHT = ”val”* (înălțime).

O imagine poate fi bordată cu o linie groasă dată de parametrul *BORDER = ”val”.*

Exemplu: *<img scr = ”Verona.jpg” width = ”300” height = ”100” border = ”2”>*

Dacă se dorește adăugarea unei alternative de text la imagine folosim parametrul **alt**.

Sintaxa: *<img scr = ”nume_imagine.extensie” alt = ”descriere”>*

Aranjarea textului față de o imagine:

Observații:

Pentru ca textul să nu fie în dreptul imaginii se folosește tag-ul *<br clear = all>*. Acest tag este folositor și în cazul unei linii orizontale situate sub imagine și nu în dreptul ei. Folosim parametrul *<br clear = ”val”>* pentru textul ”text nedorit în dreptul imaginii”. Folosim eticheta *<nobr> Text </nobr>* pentru a nu permite trecerea la rând nou, adică scrierea se face pe un singur rând.

Link-uri și imagini:

Alinierea unei imagini în raport cu paragraful se face astfel:

* <img scr = ”nume_imagine.extensie” alt = ”descriere”> .*
* <img scr = ”nume_imagine.extensie” alt = ”descriere”> .*
* <img scr = ”nume_imagine.extensie” alt = ”descriere”> .*

Pentru a face ca **o imagine să devină un link** la un fișier vom proceda astfel:

* Right <img scr = ”nume_imagine.extensie” alt = ”descriere”> .*

Alinierea unei imagini **în raport cu liniile de text** se face astfel:

<img scr = ”Verona.jpg” align = ”top”> Scrisul va apărea în partea de sus a imaginii.
<img scr = ”Verona.jpg” align = ”middle”> Scrisul va apărea în mijlocul imaginii.
<img scr = ”Verona.jpg” align = ”bottom”> Scrisul va apărea în partea de jos a imaginii.
<img scr = ”Verona.jpg” align = ”left”> Scrisul va apărea în partea stângă a imaginii.
<img scr = ”Verona.jpg” align = ”right”> Scrisul va apărea în partea dreaptă a imaginii.

Maparea imaginilor:

Operația prin care specificăm zone senzitive ale unei imagini se numește **mapare** (map însemnând hartă). Pentru a asocia unei imagini o hartă de imagini, se utilizează în cadrul marcajului **IMG** parametrul **USEMAP**.

Exemple:

<img scr = ”fisier.html” usemap = ”#numeharta”>;
<img scr = ”verona.html” usemap = ”#numeharta”>.

Pentru a defini harta unei imagini se utilizează marcajul: *<MAP name = ”numeharta”> Descriere zona 1 </MAP>*.

Descrierea unei zone senzitive a imaginii se realizează cu ajutorul marcajului ***<AREA>*** care are ca parametru *SHAPE = ”val”*, unde **val** poate fi:
1. **DEFAULT** - se selectează întreaga imagine.
2. **RECT** - trasarea unui dreptunghi care este urmat de *COORD = ”x, y, x_1, y_1”*, unde (x, y) reprezintă linia și coloana din punctul de sus al imaginii, iar (x_1, y_2) reprezintă colțul din dreapta jos.
3. **CIRCLE** - trasarea unui cerc care este urmat de *COORD = ”x, y, raza”,* unde (x, y) reprezintă linia și coloana centrului cercului, iar raza reprezintă raza cercului.
4. **POLY** - trasarea unui poligon care este urmat de *COORD = (x, y, x_1, y_1, x_2, y_1, x_3, y_3, …).*

Observație:
Toate coordonatele se pun relativ la imagine, dacă există diferență de la monitor la display-ul telefonului mobil în momentul vizualizării, atunci totul se pune la o anumită scară.

Tabele:

Un tabel este delimitat de tag-ul *<table> … </table>* și este utilizat atât pentru poziționarea altor elemente, cât și pentru organizarea informației pe linii și coloane. Acesta are o serie de atribute:
• *border = ”valoare”* – definește grosimea conturului tabelului.

Exemple:
<table> … </table> este echivalent cu <table border = ”0”> … </table>
<table border> … </table> (valoarea prestabilită este de 1)
<table border = ”3”> … </table>

• *align* – pentru a preciza poziția unui tabel relativ la marginile blocului părinte (documentul HTML). Valorile acestui parametru sunt: *left, right, center*.
• width = ”valoare” – lățimea tabelului.
• height = ”valoare – înălțiea tabelului.

Exemple:
<table width = ”300” height = ”30”> … </table>
<table width = ”%30”> … </table> (reprezintă 30% din lățimea ecranului).

• *cellpadding = ”valoare”* – distanța dintre conținutul celulei și marginile ei.

Exemplu: *<table cellpadding = ”30”> … </table>*

• cellspacing = ”valoare” – distanța dintre celule.

Exemplu*: <table cellspacing = ”30”> … </table>*

• *bgcolor = ”culoare”* – culoarea de fundal a tabelului. ”Culoare” poate fi o culoare sau codul specific culorii dorite.
• *frame* – permite să precizăm marginea dorită a tabelului din bordura exterioară. Valorile acestui parametru sunt: void (fără bordură), above (sus), below (jos), hsides (sus și jos), lhs (stânga), rhs (dreapta), vsides (stânga și dreapta), box sau border (toate marginile).
• *rules* – permite precizarea modului în care vor fi afișate bordurile pentru celulele tabelului. Valorile acestui parametru sunt: none sau groups (fără borduri), rows (borduri orizontale), cols (borduri verticale), all (toate).
• *hspace* (spațiu pe orizontală) și *vspace* (spațiu pe verticală) – permit un spațiu alb în jurul tabelului.

Exemplu: *<table hspace = ”100” vspace = ”50”> … </table>.*

Principalele tag-uri utilizate la crearea tabelelor sunt:

• Titlul tabelului *(<caption>*) – descrie conținutul tabelului și este opțional *<caption> … </caption>.*
• Capetele de tabel – denumesc coloanele / liniile tabelului și sunt opționale:

```
<tr> (table row)
       <th> … </th> (este afișat în stil aldin (îngroșat) și reprezintă antetul tabelului) (table heading)
        <td> … </td> (table data)
</tr>
```

Alinierea conținutului celulelor se face astfel:

Aliniere	Atribut	Tag (marcaj)
Orizontală:		
Stânga	align = ”left”	<tr>, <th> sau <td>
Dreapta	align = ”right”	<tr>, <th> sau <td>
Centru	align = ”center”	<tr>, <th> sau <td>
Verticală:		
Sus	valign = ”top”	<tr>, <th> sau <td>
Jos	valign = ”bottom”	<tr>, <th> sau <td>
Centru (mijloc)	valign = ”middle”	<tr>, <th> sau <td>

Alinierea titlului din tabel se face astfel:

Aliniere	Atribut	Tag (marcaj)
Orizontală:		
Stânga	align = ”left”	Caption
Dreapta	align = ”right”	Caption

Centru	align = ”center”	Caption
Verticală: Deasupra tabelului Sub tabel Centru (mijloc)	 valign = ”top” valign = ”bottom” valign = ”middle”	 Caption Caption Caption

Structura unui tabel:

Conținutul unui tabel poate fi împărțit în următoarele secțiuni:

- *<thead> … </thead>* - definește secțiunea antet.
- *<tfoot> … </tfoot>* - definește secțiunea de final a tabelului.
- *<tbody> … </tbody>* - definește corpul tabelului.

Cu ajutorul a două atribute ale etichetelor *<pd>* și *<ph>*, o celulă se poate extinde peste celulele vecine astfel extinderea unei celule peste celulele din dreapta ei se va face cu ajutorul atributului **colspan**, a cărui valoare determină numărul de celule care se unifică, iar extinderea unei celule peste celulele de dedesubt se va face cu ajutorul atributului **rowspan**, a cărei valoare determină numărul celulelor care se unifică.

Cursurile 4 - 5

Formulare:

Formularele sunt cele mai frecvent utilizate pentru a colecta informații de la persoanele care vizitează site-ul. La activarea butonului de submit dintr-un formular informațiile sunt transmise serverului web pentru a fi prelucrate, iar activarea butonului de reset permite ștergerea tuturor informațiilor completate.

Utilizarea formularelor:

Atributele tag-ului de intrare *<form>* sunt:

a) Atributul **action** unde furnizăm URL-ul scriptului CGI care va prelucra datele din formular:

```
<form action = ”http://alpha.com/program.cgi”>
```

Există două metode de transmitere a datelor conținute în formular:

1. Sub forma unui set de date.
2. Concatenate cu URL-ului **scriptului CGI** (Common GateWay Interface).

b) Atributul **method** apreciază metoda de transmitere a datelor care poate lua ca valoare **post** pentru metoda 1 și **get** pentru metoda 2.

Exemple:

`<form action = http://alpha.com/program.cgi method = post”>` (transmiterea datelor către server).
`<form action = http://alpha.com/program.cgi method = get”>` (pentru atașarea datelor la șirul URL).

Un formular este delimitat de tag-ul *<form>*, iar între aceste tag-uri sunt inserate controalele formularului:

Control	Acțiunea utilizatorului	Atribute suplimentare	Tag
Text	Introducerea unui text de volum redus.	- disabled (folosit pentru dezactivarea câmpului de text). - maxlength (numărul maxim de caractere care pot fi introduse). - readonly (nu se poate efectua modificarea conținutului din exterior). - size (definește lungimea câmpului de text). - value (textul inițial dat controlului).	Input
Password	Introducerea unui text de volum redus, care apare pe ecran mascat cu ”**”.	- value (utilizarea este obligatorie).	Input
Hidden	Element special introdus în codul paginii, dar nu este afișat de brwoser. El este folosit pentru trecerea între pagini Web și pentru a adăuga un identificator formularului căruia îi aparține.	- name (utilizarea este obligatorie). - value (utilizarea este obligatorie).	Input
Submit	La activarea lui se transmit informațiile completate de utilizator.	- disabled. - nume. - value.	Input
Reset	La activarea lui se restabilesc valorile inițiale ale tuturor controalelor formuarlului.	- disabled. - nume. - value.	Input
Checkbox	Bifare.	- name. - value. - checked (pentru cazul în care se dorește selectarea căsuței cu stare inițială).	Input
Radio	Bifare.	- name. - value.	Input

		- checked.	
Button	Activare (buton generic, fără comportament prestabilit).	- disabled. - name. - value.	Input
Image	Activare (buton imagine) – pentru a transmite informațiile completate de utilizator.	- src (precizează adresa URL a imaginii). - align (aliniază imaginea relativ la alte elemente). - disabled. - name.	Input
Text - Area	Introduce un text de volum mare.	- disabled. - readonly. - cols. - rows. - wrap (determină cum sunt tratate caracterele CR / LF, adică întreruperile de rând). - off (doar utilizatorul poate insera caracterele CR / LF). - soft sau virtual (textul este trecut vizual de pe un rând pe altul, dar nu sunt transmise decât caracterele CR / LF introduse de utilizator). - hard sau physical (textul este trecut vizual de pe un rând pe altul și sunt inserate automat secvențe CR / LF).	Text Area
Reset, Submit	Activare (pentru a transmite informațiile completate de utilizator).	- disabled. - name. - value. - type.	Button
Menu	Alegerea unei opțiuni dintr-un meniu.	- disabled. - size. - multiple. - type.	Select, Option.

Controlul Button Image:

Permitea afișarea unui buton care nu are un comportament deja prestabilit. Pentru a utiliza astfel de butoane vom folosi JavaScript, iar sintaxa *nume_element* reprezintă valoarea atributului *name*, atribuită elementului *image*. Coordonata X și coordonata Y reprezintă coordonatele punctului de pe imagine, unde utilizatorul a făcut click. Butonul imagine este o imagine hartă pe partea de server, iar serverul trebuie să aibă o aplicație care să o citească și cu care să prelucreze datele citite.

Structură:

```
nume_element . x = coordonată – x & nume_element . y = coordonata – y.
```

Observație:

În exemplul de mai jos, diferența dintre cele două butoane de transmitere este că la activarea butonului *imagine* sunt transmise coordonatele punctului imaginii în care s-a efectuat click relativ la colțul stânga sus al imaginii și numele imaginii, dar fără transmiterea datelor despre butonul *submit*. Dacă se dă click pe butonul *submit* vor fi transmise datele, dar fără date despre imagine.

Exemplu:

```
<html>
	<head>
		<title> Formular </title>
	</head>

	<body>
		<form>
			This is a text field:
			<input type = "text" name = "text" value = "valoare initiala">
			<input type = "submit" name = "submit" value = "Trimite">
			<input type = "image src = "pb.jpg" name = "imagime">
		</form>
	</body>
</html>
```

Controlul File:

Tag-ul de input permite atașarea de fișiere din sistemul local pe care le transmite împreună cu datele din câmpurile formularului, acest proces numindu-se încărcare de la client la server. Fișierul pe care îl folosim poate fi de orice tip. Folosind elementul de *<form>* putem face distincția între fișiere, adăugând următoarele valori pentru parametrii din *<form>*.

Structură:

```
<form method = ”post” enctype = ”multiform/form-data” = action = ”URL”>
```

Adresa URL este adresa aplicației de server care va prelua și prelucra fișierele.

Exemplu:

```
<html>
	<head>
		<title> Formular </title>
	</head>

	<body>
		<form name = "f2" method = "post" action = "http>//.../upload.asp"
enctype = "multiform/form-data">
```

```
                    This is a text field:
                    <input type = "text" name = "text" value = "Valoare initiala">
                    <input type = "file" name = "fisier" value = "Introduceti calea
fisierului">
                    Click pentru a trimite valoarea <input type = "submit" name =
"button" value = "Trimite">
             </form>
      </body>
</html>
```

Controlul Text-Area:

Permite introducerea unui volum mare de text.

Atributul **name** este folosit pentru a atribui un nume pentru fiecare control creat. Pentru afișarea unui text inițial se va introduce între cele două tag-uri textul:

```
<form>
      <textarea name = ”textarea1”> Text initial </textarea>
</form>
```

Atributele **rows** / **cols** se referă la numărul de linii și de coloane pe care le are controlul.

```
<form>
      <textarea name = ”textarea2” rows = ”3” cols = ”4”> Text inițial </textarea>
</form>
```

Exemplu:

```
<html>
      <head>
            <title> Formular </title>
      </head>
      <body>
            <form>
                  <textarea name = "def"> default textarea </textarea>
                  <textarea name = "dis" disabled> disabled textarea </textarea>
                  <textarea name = "rea" readonly> readonly textarea </textarea>
                  <textarea  name  =  "rc"  rows  =  "3"  colls  =  "10">  customized
textarea </textarea <br>
            </form>
      </body>
</html>
```

Controlul Select (Option):

Cu ajutorul căruia creăm o listă derulană din care utilizatorul poate alege unul sau mai multe elemente. Crearea controlului **meniu** se face folosind tag-ul *select* care delimitează secvența cu care se realizează controlul meniu, iar tag-ul *option* delimitează fiecare element din meniu.

Structură:

```
<form>
	<select name = ”alege” size = ”2> …
</select>
```

Atributul **name** este folosit pentru fiecare control creat, iar atributul **size** este folosit atunci când utilizatorul dorește derularea listei.

Atributul **multiple** dă utilizatorului posibilitatea să își aleagă mai multe opțiuni din listă. Selectarea se va face cu ajutorul tastei Control (CTRL) pe toată durata selectării.

Structură:

```
<form>
	<select name = ”alege” size = ”3” multiple> …
</select>
```

Atributul **value** este folosit pentru *option* și care oferă posibilitatea script-ului să primească un cod cu ce s-a selectat de către utilizator.

Structură:

```
<form>
	<select name = ”alege” size = ”2” multiple>
	<option value = ”1”> Primăvara </option>
	<option value = ”2”> Vara </option>
	<option value = ”3”> Toamna </option>
	<option value = ”4”> Iarna </option>
</select>
```

Cursurile 6 – 8 - Introducere în CSS (Cascading Style Sheets – foi de stil în cascadă)

CSS înseamnă **Cascading Style Sheets** (foi de stil în cascadă). Separă conținutul de formă a paginilor Web. Un stil este o definire de fonturi, culori, etc. Fiecare stil are un nume unic – un selector. Prin CSS se dorește controlul asupra designului unei pagini în timp ce

limbajele de marcaje controlează structura acestora. Prin folosirea acestor standarde se reduc considerabil costurile de întreținere a unei pagini Web prin comparație cu vechiile tehnici în care prezentarea conținutului se făcea în stil HTML pur. Cu ajutorul unui stil definesc fontul, culori, etc, fiecare stil având un nume unic. Selectorii și stilurile lor sunt definite într-un singur loc.

Avantaje și dezavantaje:

- definirea „look-ului" paginii într-un singur loc.
- modificarea foarte ușoară a "look-ului".
- poziționarea conținutului paginii cu precizie de pixeli.
- redefinirea tag-urilor HTML.
- definirea de stiluri customizabile.
- definirea de layere care pot fi poziționate chiar unul peste altul.
- paginile se vor încărca mai rapid, dar nu toate browserele le suportă.

Unde punem CSS-ul?

Vom aborda metoda stilurilor interne, adică style-ul va fi pus în cadrul tag-ului:

```
<style = "definirea_stilului: atributele_stilului;">
       Aici se definesc stilurile CSS
</style>

Aici <b style = "..."> se </b> modifică.
```

În cadrul paginii (această metoda înseamnă că vom plasa codul CSS în interiorul fiecărei pagini HTML pe care doriți să folosiți stilurile respective între tag-urile de **head**):

```
<head>
       <title> Titlu pagina </title>
       <style type = "text/css"> Aici se definesc stilurile CSS. </style>
</head>
```

Folosind această metodă fiecare fișier HTML va conține codul CSS folosit la stilizare, însemnând că atunci când dorim să modificăm un stil va trebui să operăm modificarea în toate paginile ce conțin acest stil. Metoda descrisă aici este bună atunci când avem un număr mic de pagini, dar neplăcută în cazul aplicațiilor cu mai multe pagini.

Unde punem CSS-ul?

CSS-ul se pune într-un fișier extern cu extensia **.css**.

Exemplu de inserare a unui fișier extern .css într-o pagină HTML:

```
<link trl = "stylesheet" type = "text/css" href = "Calea catre fisierul .css">

<head>
       <title> Titlu pagina </head>
       <link rel = "stylesheet" type = "text/css" href = "sil.css">
```

```
</head>
```

Acest fișier extern poate fi editat cu ajutorul unui WordPad, NotePad, etc. Fișierul CSS nu conține sub nicio formă cod HTML, ci doar cod CSS. Inserarea fișierului extern în paginile HTML se face foarte ușor prin plasarea link-ului în secțiunea **head** a fiecărei pagini pe care dorim să folosim stilul respectiv.

Exemplu de inserare a unui fișier extern .css folosind metoda de import:

```
<style type = "text/css"> @import url (calea catre fisierul .css) </style>
sau

<head>
	<title> Titlu pagina </title
	<style type = "text/css"> @import url (calea catre fisierul .css)
	</style>
</head>
```

Declarații:

Pentru a defini o regulă este nevoie de un **selector** și de o **declarație**. Selectorul permite alegerea elementelor care vor lua parametrii sub forma:

```
selector {nume_proprietate: valoare_proprietate}
```

Unde *nume_proprietate* este un cuvânt rezervat care definește o *proprietate*, iar *valoare_proprietate* este o valoare validă pentru respectiva *proprietate*.

Exemplu:

- Utilizat pentru stabilirea unei familii de fonturi.
- Utilizat la stabilirea culorii negre pentru text.
- Utilizat la stabilirea culorii albe pentru fundal.

```
body {
	font-family: Arial, Helvetica, "Trebuchet MS";
	color: black;
	background> white;
}
```

După cum putem observa, proprietățile sunt separate cu ajutorul caracterului „ ; ” și respectiv un tag.

Sintaxa CSS este compusă doar din trei părți.

Selectorul este elementul HTML pe care îl dorim să îl stilizăm (se aplică pe tot conținutul body-ului). Proprietatea este chiar titlul sau numele proprietății respective, iar valoarea reprezintă stilul pe care îl aplicăm proprietății. Dacă dorim să modificăm layout-ul codului vom putea face foarte ușor acest lucru, iar modificările ar fi recomandate să fie efectuate pe rânduri separate pentru o mai ușoară citire a codului.

Moștenirea:

Exemplu, un font specificat pentru corp (body):

```
body {font-family: Verdana, serif;}
h1 {font-family: Georgia, sans-serif;}
p {font-family: Tahoma, serif;}
```

Există însă și cazuri când elementele plasate în interiorul altor elemente, nu moștenesc proprietățile acestora din urmă.

```
body {margin: 20px;}
```

Atunci când plasăm un element în interiorul altuia, elementul plasat va moșteni proprietățile elementului în care a fost plasat, asta doar în cazul în care nu se aplică aceleași proprietăți, însă cu valori diferite fiecărui element.

Blocuri de declarații:

Un **bloc de declarații** este o listă de declarații separate prin semnul „;”, inclusă într-un bloc {…}.

Exemplu:

```
{
font-family: Arial, Helvetica, sans-serif;
color: black;
background: white;
}
```

Foi de stil:

Mai multe reguli pot fi adăugate pentru a defini o foaie de stil.

Stiluri in-line:

```
<body>
	<h3 style = "font-szie: 24pt;">    Titlu scris cu 24pt </h3>
	<h3> Heading 3 </h3>
	<img src = "Yahoo.jpg" style = "font-style: italic: cursor: crosshair">
</body>
```

Folosirea unui fișier extern care să conțină comenzi CSS este foarte practică deoarece poate fi folosită în mai multe situații. Mai multe fișiere HTML pot folosi același fișier CSS eliminând timpul necesar introducerii codului corespunzător în fiecare pagină și totodată editarea lor într-un singur loc pentru mai multe fișiere. Se folosește, de regulă, atunci când se dorește modificarea formatării unui site întreg.

Foi de stil definite în interiorul unui document HTML:

Pentru a defini un stil pe care îl putem aplica mai multor elemente care aparțin unei pagini HTML se poate utiliza foaia de stil ce poate fi plasată în interiorul documentului folosind marcajele de style sau poate fi afișată într-un fișier extern. Aceste marcaje se găsesc în interiorul marcajului de *head*. Un atribut obligatoriu al elementului *style* este *type = "text/css"* și pentru CSS are valori prestabilite.

Exemplu:

```
<html>
	<head>
		<style type = "text/css">
		<! Aici se introduc regulile foii de stil ->
		p {font-size: 12pt; text-transform: uppercase;}
		ol {background: yellow;}
		li {font-style: italic;}
		a {text-decoration: none; color: green;};
		</style>
	</head>
	<body>
		<p> Acest paragrafa este scris cu majuscule si cu marimea de 12pt. </p>
		<ol>
			<li> Primul element.
			</li> Al doilea element.
		</ol>
		<a href = "http://stiinte.ulbsibiu.ro"> Facultatea de Stiinte </a>
	</body>
</html>
```

Tag-urile de comentarii:

Exemplu: `/* Acesta este un comentariu și nu va fi interpretat de browsere. */`

Se pot folosi inclusiv în fișierele care au extensia .css.

Tipuri de selectori:

Selectorii în CSS sunt tipare care dacă se potrivesc cu vreun element din document îi vor aplica reguli corespunzătoare. Tipuri de selectori:

• selector de tip html – rescrie forma de afișaj a tag-urilor html:

```
h1 {font-family: Georgia, sans-serif; font-size: 14px}
```

• selector de tip class – se pot aplica oricăror tag-uri:

```
.text {font-family: Georgia, sans-serif; font-size: 10px}
<p class = „text”> … </p>
```

• selector de tip ID – se aplică obiectelor care se identifică printr-un ID:

```
#test1 {font-family: Georgia, sans-serif; font-size: 10px}
<div id = “test1”> … </div>
```

Selectorul de tip HTML:

Se folosește pentru a redefini modul de afișare a conținutului etichetei HTML. Un selector HTML reprezintă partea etichetei HTML care indică navigatorului tipul de etichetă. Definirea unui selector HTML folosind CSS-ul are ca rezultat redefinirea etichetei HTML. Selectorul și eticheta, deși par identice, totuși nu sunt.

Exemple:

```
HTMLSelector {Property: Value;}

<style type = "text/css">

p {
	font-family: Georgia, "Times New Roman", Times, serif;
	font-size: large;
	color: #03F;
}

</style>
```

```
<html>
	<head>
		<title> Unu </title>
	</head>

	<body>
		<h1> Titlu principal </h1>
		<h2> Subtitlu </h2>
		<p> Continut Continut Continut </p>
		<p> Continut Continut Continut </p>
	</body>
</html>
<html>
	<head>
		<title> Doi </title>

		<style>


p {font-family: "Verdana"; font-size: 10px; color: blue; }
			h1 {font-family: "Arial"; font-size: 30px; color:red; }
			h2 {font-family: "Arial"; font-size: 20px; color: #FF7700; }
		</style>
```

```
	</head>

	<body>
		<h1> Titlu principal </h1>
		<h2> Subtitlu </h2>
		<p> Continut Continut Continut </p>
		<p> Continut Continut Continut </p>

	</body>
</html>
```

Observație: Nu contează ordinea în care sunt declarați selectorii în antet.

Selectorul de tip clasă:

Clasa este un obiect care poate fi aplicat oricărei etichete HTML. În momentul definirii, numele clasei trebuie să fie precedat de un **punct**. O clasa trebuie creată în interiorul etichetei HTML, acest lucru făcându-se prin specificarea cuvântului *class* și numele clasei.

```
.ClassSelector {Property:Value;}

.titlu {
	font-family: "Arial";
	font-size: 30px;
	color: red;
}

<h1 class = "titlu"> Titlu principal </h1>
```

Definirea claselor fără utilizarea lor:

```
<html>
	<head>
		<title> Trei </title>

		<style>
			.paragraf {font-family: "Verdana"; font-size: 10px; color:blue;}
			.titlu {font-family: "Arial"; font-size: 30px; color:red;}
			.h2 {font-family: "Arial"; font-size: 20px; color: #FF7700;}
		</style>
	</head>

	<body>
		<h1> Titlu principal </h1>
		<h2> Subtitlu </h2>

<p> Continut Continut Continut </p>
		<p> Continut Continut Continut </p>
	</body>
</html>
```

Utilizarea claselor:

```
<html>
	<head>
		<title> Trei </title>
```

```
		<style>
			.paragraf {font-family: "Verdana"; font-size: 10px; color:blue;}
			.titlu {font-family: "Arial"; font-size: 30px; color:red;}
			.h2 {font-family: "Arial"; font-size: 20px; color: #FF7700;}
		</style>
	</head>

	<body>
		<h1 class = "titlu"> Titlu principal </h1>
		<h2 class = "h2" > Subtitlu </h2>
		<p> Continut Continut Continut </p>
		<p class = "paragraf"> Continut Continut Continut </p>
	</body>
</html>
```

Observații:

- numele clasei la utilizare în cadrul tag-ului HTML poate să fie pus între ghilimele sau fără ghilimele;
- putem utiliza aceleași nume atât pentru clasă, cât și pentru tag;
- doar paragraful în care s-a specificat clasa este formatat corect, celălalt având formatările implicite ale browser-ului.

Selectorul de tip identificator:

Obiectele de tip identificator sunt similare cu clasele și pot fi aplicate oricărei etichete HTML, dar spre deosebire de clase, numele unui identificator trebuie atribuit numai unei singure etichete HTML dintr-o pagină. Pentru altă etichetă se adaugă un ID cu un nume diferit. În interiorul codului CSS, identificatorul este definit prin adăugarea caracterului „#” înaintea numelui.

```
#IDSelector {Property:Value;}

#titlu {
	font-family: "Arial";
	font-size: 30px;
	color: red;
}

<h1 id = "titlu"> Titlu principal </h1>
<html>
	<head>
		<title> Patru </title>
		<style>
			#paragraf: {font-family: "Verdana"; font-size: 10px; color:blue;}
			#titlu {font-family: "Arial"; font-size: 30px; color: red;}
			#subtitlu {font-family: "Arial"; font-size: 20px; color:#FF7700;}
```

```
		</style>
	</head>
	<body>
		<h1 id = "titlu"> Titlu principal </h1>
		<h2 id = "subtitlu"> Subtitlu </h2>
		<p> Continut Continut Continut </p>
		<p id = "paragraf"> Continut Continut Continut </p>
	</body>
</html>
```

Tag-ul HTML Span

Majoritatea etichetelor HTML au unele proprietăți prestabilite. Acestea fie rămân așa cum sunt, fie pot fi redefinite. Există însă cazuri în care se dorește crearea unor etichete personalizate pornind de la zero. În acest caz se folosesc etichetele de *span* și *div*.

Eticheta ***span*** nu are proprietăți moștenite. Ea reprezintă doar o locație vidă care crează o etichetă în linie. Pentru a configura o etichetă *inline* trebuie definită o clasă sau un identificator care să poată fi aplicat apoi unei etichete span.

```
<SPAN> este un tag „dummy”, se folosește în combinație cu selectori de tip class.
<SPAN> este un „inline-tag” în HTML, adică nu se inserează salturi la rând nou înainte sau după.
```

```
<html>
	<head>
		<title> Cinci </title>
		<style>
			.paragraf {font-family: "Verdana"; font-size: 10px; color:blue;}
			.titlu {font-family: "Arial"; font-size: 30px; color:red;}
			.h2 {font-family: "Arial"; font-size: 20px; color: #FF7700;}
		</style>
	</head>
	<body>
		<span class = "titlu"> Titlu principal </span>
		<span class = "h2" > Subtitlu </span>
		<span> Continut Continut Continut </span>
		<span class = "paragraf"> Continut Continut Continut </span>
	</body>
</html>
```

Observăm că avem următoarele diferențe:

- nu avem trecerea la rând nou;
- o serie de formatări diferă, cum ar fi: titlul principal este bolduit, iar cel secundar nu este. Proprietatea de bold este automat setată, iar setările suplimentare din clasa .titlu sunt practic adăugate peste cele inițiale ale lui <h1>. În schimb, la utilizarea etichetei span nu avem nici un fel de setări inițiale.

Tag-ul HTML Div

În momentul în care se dorește confiurarea unui bloc separat de restul conținutului unui document HTML, soluția este folosirea etichetei **DIV**. Aceasta crează o zonă proprie în pagină cu o linie nouă atât deasupra, cât și dedesuptul său. Un exemplu simplu poate fi cel în care utilizăm un *div* care va cuprinde întreg site-ul. Pentru crearea etichetelor de tip *div* se procedează la fel ca și în cadrul etichetelor *span* prin definirea mai întâi a unor etichete de tip clasă sau identificator, urmată apoi de aplicarea ei asupra unor etichete *div*

```
<DIV> este un „block tag”, adică se inserează automat un rând nou (ca la <p> / <table>).
<DIV> sunt importante pentru poziționarea conținutului într-o pagină HTML.

<div> Conținutul site-ului este plasat aici. </div>
```

Exemplu:

```
<body>
	<div class = "titlu"> Titlu principal </div>
	<div class = "h2" > Subtitlu </div>
	<div> Continut Continut Continut </div>
	<div class = "paragraf"> Continut Continut Continut </div>
</body>
```

Observație: Similar cu tag-ul *span*, observăm diferențe față de utilizarea *<h1>, <h2>* și *<p>,* puterea div-urilor stă în faptul că pe lângă setări de formatare a textului și culorilor vor permite setări legate de dimensiune și de poziția în cadrul paginii.

Selectorii dependenți de context:

Este posibil să realizăm selectori care vor funcționa doar în anumite contexte. Un exemplu simplu arată că selectorii se aplică doar în cazul în care avem o ancoră sau un paragraf în cadrul unei liste, nu și în alte contexte.

```
<html>
	<head>
		<title> Titlu </title>

		<style>
			li a {font-size: 25px;}
			li p {font-family: "Verdana"; color:red;}
		</style>
	</head>

	<body>
		<a href = "http://stiinte.ulbsibiu.ro"> Aici nu se modifica </a>
		<p> Nemodificat </p>

		<li>
```

```
                    <a href = "http://stiinte.ulbsibiu.ro"> Aici se modifica </a>
                    <p> Modificat </p>
                </li>
        </body>
</html>
```

Combinarea selectorilor HTML:

Putem combina elementele selectorilor dacă dorim aceleași stiluri pentru mai multe elemente HTML. Numărul de selectori pe care îi putem grupa poate fi infinit. Elementele unui selector se pot combina astfel:

```
h1, h2, h3, h4, h5, h6 {
        color: #009900;
        font-family: "Georgia", sans-serif;
}
```

Tabel selectori:

Denumire	Descriere	Exemplu	Rezultat
Selector **universal**	Corespunde oricărui element (marcaj).	<style> *{color:red; font-style: italic;} </style>	
Selectorul **class**	Forma: class = ”valoare”, unde valoare trebuie să fie o listă de identificatori separați prin spații.	<head> <style> *.c1{color:red;} p.c2{font-style: italic;} </style> <head> <body> <p class = "c1"> Paragraf1 rosu <p class = "c2"> Paragraf2 inclinat <p class = "c1 c2"> Paragraf rosu si inclinat </body>	Paragraf1 roșu *Paragraf2 înclinat* *Paragraf roșu și înclinat*
Selectorul **id**	Valoarea atributului trebuie să fie unică pentru întregul document.	<head> <style> *# id1{color:red;} P #id2{font-style: italic;} </style> </head> <body> <p id = "id1"> Paragraf1 rosu <h1 id = "id2"> Paragraf inclinat </body>	Paragraf1 roșu *Paragraf2 înclinat*
Selectori descendenți	Corespund elementelor îmbricate din interiorul altor elemente.	<head> <style> h1 i{color:red;} p i b{font-style:italic;} </style> </head> <body> <h1> Paragraf rosu italic paragraf 1 </h1> <p> paragraf2 italic 1 bold	**Paragraf roșu *italic* paragraf 1** Paragraf2 *italic 1* ***bold*** *italic 2* paragraf3

		 italic 2 </i> paragraf3 </body>	
Selectorul **atribut**	Elementele care conțin anumite atribute.	<head> <style> p [id] {color:red;} {align = "center"} </style> </head> <body> Paragraf cu id Laragraf aliniat la centru </body>	Paragraf cu id Paragraf aliniat la centru
Pseudo – clasele legăturilor	Conține cinci subclase: link-legături nevizitate, visited (legături vizitate), hover (legătura pe care s-a plasat cursorul), active (pentru o legătură activată), focus (legătură selectată).	<head> <style> a:link {color:red;} a:visited{color:red;} </style> </head> <body> <a href = "doc.html"> </body>	
Pseudo – elementul first-line	Pentru a selecta primul rând dintr-un bloc text.	<head> <style> p:first-line {color:red;} center: first-line {color:blue;} </style> </head> <body> Primul paragraf Primul rand centrat </body>	Primul paragraf Primul rând centrat
Pseudo – elementul first-letter.	Pentru a selecta prima literă dintr-un bloc text.	<head> <style> p:first-lettter{color:red;} center: first-letter{color:blue; font-size: 30px;} </style> </head> <body> Primul paragraf Primul rand centrat </body>	Primul paragraf Primul rând centrat

Modificarea fontului:

Proprietatea font-family:

O listă de nume de fonturi care se vor utiliza dacă există pe mașina pe care rulează browser-ul în ordinea stabilită.

```
<html>
	<head>
		<style>
			.TNR{font-family: "Times New Roman", Times, serif;}
			.Arial{font-family: Arial, Helvetica, sans-serif;}
		</style>
	</head>

	<body>
		<h1> CSS font-family </h1>
```

```
		<p class = "TNR"> Acest paragraf este scris utilizand fontul Times New Roman. </p>
		<p class = "Arial"> Acest paragraf este scris utilizant fontul Arial. </p>
	</body>
</html>
```

Proprietatea font-size:

Setează dimensiunea fontului.

```
<html>
	<head>
		<style>
			h1 {font-size: 40px;}
			h2 {font-size: 30px;}
			p {font-size: 14px;}
		</style>
	</head>

	<body>
		<h1> Acest text este headin 1. </h1>
		<h2> Acest text este heading 2. </h2>
		<p> Acesta este un paragraf. </p>

		<p> <b> Note:</b> Acest exemplu nu e executat in Internet Explorer versiunea 9. </p>
	</body>
</html>
```

Proprietatea font-style:

Setează stilul fontului.

```
<html>
	<head>
		<style>
			.normal {font-style: normal;}
			.italic {font-style: italic;}
			.oblique {font-style: oblique;}
		</style>
	</head>

	<body>
		<p class = "normal"> Acesta este un paragraf normal. </p>
		<p class = "italic"> Acesta este un paragraf italic. </p>
		<p class = "oblique"> Acesta este un paragraf oblique. </p>
	</body>
</html>
```

Proprietatea font-variant:

Prin utilizarea fontului *small-caps*, toate literele mici sunt convertite în litere mari. Literele mici convertite vor avea totuși o dimensiune mai mică față de literele care inițial au fost mari.

```
<html>
	<head>
		<style>
			p.normal {font-variant:normal;}
			p.small {font-variant:small-caps;}
		</style>
	</head>

	<body>
		<p class = "normal"> Numele meu este Robert Leca. </p>
		<p class = "small"> Numele meu este Robert Leca. </p>
	</body>
</html>
```

Proprietatea font-weight:

Setează cât de subțire sau de gros să fie fontul.

```
<html>
	<head>
		<style>
			p.normal {font-weight:normal;}
			p.light {font-weight:lighter;}
			p.thick {font-weight: bold;}
			p.thicker {font-weight: 900;}
		</style>
	</head>

	<body>
		<p class = "normal"> Acesta este un paragraf. </p>
		<p class = "light"> Acesta este un paragraf. </p>
		<p class = "thick"> Acesta este un paragraf. </p>
		<p class = "thicker"> Acesta este un paragraf. </p>
	</body>
</html>
```

Proprietatea font:

Se setează toate proprietățile fontului într-o singură declarație. Proprietățile care pot fi setate sunt (în această ordine): *"font-style font-variant font-weight font-size/line-height font-family".*

```
<html>
	<head>
		<style>
```

```
                .stilul1 {font: 15px arial;}
                .stilul2 {font: italic bold 30px Verdana;}
        </style>
    </head>

    <body>
        <p class = "stilul1"> Acesta este un paragraf. </p>
        <p class = "stilul2"> Acesta este un paragraf. </p>
    </body>
</html>
```

Proprietatea letter-spacing:

Spațierea se referă la cantitatea de spațiu dintre literele unui cuvânt.

```
<html>
    <head>
        <style>
            h1 {letter-spacing: 2px;}
            h2 {letter-spacing: -3px;}
        </style>
    </head>

    <body>
        <h1> Acesta este heading 1. </h1>
        <h2> Acesta este heading 2. </h2>
    </body>
</html>
```

Proprietatea work-spacing:

Spațierea se referă la cantitatea de spațiu dintre cuvinte.

```
<html>
    <head>
        <style>
            p {word-spacing:30px;}
            h2 {word-spacing:-4px;}
        </style>
    </head>

    <body>
        <h2> Acesta este un titlu. </h2>
        <p> Acesta este un paragraf. </p>
    </body>
</html>
```

Proprietatea vertical-align:

Setează aliniamentul vertical al unui element.

```
<html>
	<head>
		<style>
			.sus {vertical-align:text-top;}
			.jos {vertical-align:text-bottom;}
		</style>
	</head>

	<body>
		<p> Aceasta <img src = "imag.jpg"> imagine cu aliniere implicita. </p>
		<p> Aceasta <img class = "sus" src = "imag.jpg"> imagine cu aliniere
sus. </p>
		<p> Aceasta <img class = "jos" src = "imag.jpg"> imagine cu aliniere
jos. </p>
	</body>
</html>
```

Proprietatea text-indent:

Indentează prima linie a paragrafului.

```
<html>
	<head>
		<style>
			p {text-indent:50px;}
		</style>
	</head>

	<body>
		<p> Acesta este un paragraf. Acesta este un paragraf. Acesta este un
paragraf. Acesta este un paragraf. Acesta este un paragraf. Acesta este un paragraf.
</p>
	</body>
</html>
```

Proprietatea text-decoration:

Aplicarea unor elemente decorative.

```
<html>
	<head>
		<style>
			h1 {text-decoration:overline;}
			h2 {text-decoration:line-through;}
			h3 {text-decoration:underline;}
		</style>
	</head>

	<body>
		<h1> Linie deasupra </h1>
		<h2> Linie peste </h2>
```

```
		<h3> Linie sub </h3>
	</body>
</html>
```

Proprietatea color:

Specifică culoarea unui text.

```
<html>
	<head>
		<style>
			body {color:red;}
			h1 {color:#00FF00;}
			.albastru {color: rgb(0, 0, 255);}
		</style>
	</head>

	<body>
		<h1> Acesta este un titlu </h1>
		<p> Acesta este un paragraf. Culoarea rosie este datorata definirii ei
in selectorul body. </p>
		<p class = "albastru"> Acesta este un paragraf cu clasa albastru. </p>
	</body>
</html>
```

Proprietatea background-color:

Setează culoarea de fundal a unui element.

```
<html>
	<head> <style>
			body {background-color:yellow;}
			h1 {background-color:#00FF00;}
			p {background-color:rgb(255,0,255);}
	</style> </head>

	<body> <h1> Acesta este un titlu. </h1>
		<p> Acesta este un paragraf. </p> </body>
</html>
```

Proprietatea background-image:

Setează imaginea de fundal a unui element.

```
<html>
	<head>
		<style>
			body{background-image:url('http://www.ulbsibiu.ro/image.jpg');}
		</style>
	</head>

	<body>
		<h1> Acesta este un titlu. </h1>
	</body>
```

```
</html>
```

Proprietatea background-repeat:

Setează cum se repetă imaginea de fundal.

Proprietatea background-attachment:

Setează dacă imaginea de fundal este fixă sau se deplasează odată cu pagina.

```
<html>
	<head>
		<style>
			background-image:url('www.ulbsibiu.ro/image.jpg');
			background-repeat:no-repeat;
			background-attachment:fixed;}
		</style>
	</head>

	<body>
		<p> Acesta este un paragraf. </p>
		<p> Acesta este un paragraf. </p>
		<p> Acesta este un paragraf. </p>
		<p> Acesta este un paragraf. </p>
		<p> Acesta este un paragraf. </p>
		<p> Acesta este un paragraf. </p>
		<p> Acesta este un paragraf. </p>
		<p> Acesta este un paragraf. </p>
		<p> Acesta este un paragraf. </p>
		<p> Acesta este un paragraf. </p>
	</body>
</html>
```

Proprietatea background-position:

Setează poziția de start a imaginii de fundal.

Structură: *body {background-position: center;}*

Proprietatea background:

Setează toate proprietățile de fundal într-o singură declarație.

Structură: *body {background: #00FF00;}*

Proprietatea list-style-type:

Stabilește modul de marcare a listei.

```
<html>
	<head>
		<style>
			.cerc {list-style-type: circle;}
```

```
                .patrat {list-style-type: square;}
                .roman {list-style-type: upper-roman;}
                .litere (list-style-type: lower-alpha;}
        </style>
    </head>

    <body>
        <p> Primele trei numere: </p>

        <ul class = "cerc"> <li> Unu </li> <li> Doi </li> <li> Trei </li> </ul>
        <ul  class  =  "patrat"> <li> Unu </li> <li> Doi </li> <li> Trei </li>
</ul>
        <ul class = "roman"> <li> Unu </li> <li> Doi </li> <li> Trei </li> </ul>
        <ul  class  =  "litere"> <li> Unu </li> <li> Doi </li> <li> Trei </li>
</ul>
    </body>
</html>
```

Proprietatea list-style-imagine:

Utilizează o imagine ca marcator al listei.

```
<html>
    <head>
        <style>
            ul {list-style-image:url('happy.png');}
        </style>
    </head>

    <body>
        <ul>
            <li> Unu </li>
            <li> Doi </li>
            <li> Trei </li>
        </ul>
    </body>
</html>
```

Proprietatea list-style-position:

Produce o extra-indentare a listei.

```
<html>
    <head>
        <style>
            .inauntru {list-style-position:inside;}
            .afara {list-style-position:outside;}
        </style>
    </head>

    <body>
        <p> Primele trei numere: </p>
        <ul class = "inauntru">
            <li> Unu </li>
            <li> Doi </li>
```

```
                <li> Trei </li>
            </ul>

            <p> Primele trei numere: </p>
            <ul class = "afara">
                <li> Unu </li>
                <li> Doi </li>
                <li> Trei </li>
            </ul>
        </body>
</html>
```

Proprietatea list-style:

Specifică toate proprietățile listei într-o singură declarație. Proprietățile care pot fi setate sunt (în această ordine): *list-style-type, list-style-position, list-style-image.*

Elemente speciale meta, embed, marquee, object

Codul HTML conține și anumite elemente speciale, unele folosite pentru optimizarea paginii în vederea unei cât mai bine indexări în motarele de căutare sau altele pentru adăugarea de aplicații audio și video ori alte elemente în pagină.

Elemente pentru optimizarea indexării paginii:

Aceste elemente de optimizare se adaugă în secțiunea *HEAD* a documentului HTML. Cel mai important este tag-ul *<title> </title>.* Alte elemente importante pentru motoarele de căutare sunt cele ***META***, numite și meta tag-uri, acestea având 2 atribute: **name** (care determină tipul meta tag-ului) și **content** (care determină conținului meta tag-ului).

În continuare sunt prezentate exemple de meta tag-uri care sunt indicate să fie adăugate în fiecare pagină HTML:

<meta content = ”text/html; charset = UTF-8” http-equiv = Content-Type>:
- indică browser-ului că este un fișier HTML.
- charset stabilește tipul de caractere (în general se folosește ISO-8859-1 sau UTF-8), dar există și altele.

<meta name = ”description” content = ”O frază care descrie pe scurt conținutul paginii”>
- în cazul unei căutări după cuvinte cheie, motorul de căutare ne dă o listă de pagini; pentru fiecare pagină (site) din lista apare titlul ei (dat cu marcajul *<title>*), urmat de fraza (cea de la *name = ”description” content = ”O frază care descrie pe scurt conținutul paginii”*). Dacă nu avem în marcajul *META* acea frază, motorul va indexa după prima frază din pagină. Tot ea va fi dată la o eventuală căutare.
- textul adăugat cu acest tag pentru descriere are prioritate înaintea frazelor din conținut (folosim maxim 40, 50 de caractere).

<meta name = ”keywords” content = ”lista de cuvinte, separate, prin virgula>

- motoarele de căutare țin cont de cuvintele din meta "*keywords*" în momentul indexării site-ului pentru atunci când sunt afișate rezultate de căutări (folosim maxim 15-20 de cuvinte).

<meta name = "author" content = "Numele dvs, e-mail, etc">
- acesta nu este neapărat necesar, dar nu strică să-l folosim. Arată autorul documentului.

Recomandare: Cuvintele din tag-ul *<title> … </title>* este bine să se regăsească și în meta tag-urile "*keywords*", "*description*", cât și în titurile din conținutul paginii.

Un alt meta tag care este câteodată necesar, dar nu are legătură cu motoarele de căutare este "Refresh", acesta are următoarea formă:
<meta http-equiv = "Refresh" content = "4; url = "www.nume_site/pagină">
- acest determină browser-ul să încarce o nouă pagină, cea care este adăugată la URL-ul din acest tag, după un anumit număr de secunde (în acest exemplu 4). Practic, face un redirect.

Adăugarea de sunet la o pagină HTML:

Pentru a adăuga sunet într-o pagină Web, putem folosi elementele *<embed>* sau *<bgsound>:*
1. *<bgsound> </bgsound>* - introduce un background (fundal) audio în pagină. Acesta are următoarele atribute:
• **src** – definește locația fișierului audio folosit (.midi, .au sau .wav);
• **loop** – definește de câte ori se va repeta sunetul.
• **delay** – definește timpul dintre repetări.
• **title** – textul care va descrie sunetul.

Exemplu: *<bgsound src = "sunet.midi" loop = "3" title = "titlul melodiei" delay = "5">*

2. *<embed> </embed>* - afișează o consolă pentru sunet. Are următoarele atribute:
• **src** – definește locația fișierului audio folosit (.midi, .au sau .wav);
• **controls** – oferă posibilitatea alegerii mai multor controale care includ: console, console mici, butoane de play și altele.
• **autostart** – când este TRUE, sunetul începe în timp ce sunetul este descărcat de browser.
• **hidden** – când este TRUE, va ascunde controalele. Standard este FALSE.
• **loop** – definește de câte ori se va repeta sunetul.
• **volume** – setează volumul sunetului (sonorul).
• **height** – înălțimea în pixeli a consolei.
• **width** – lungimea în pixeli a consolei.

Cursul 9

Adăugarea de imagini video și imagini flash la o pagină HTML:

Tag-ul *<embed>* poate fi folosit și pentru afișarea de **imagini video** care au extensiile *.mpeg*, .avi, *.muf*, recomandat fiind .muf. La *src* se scrie calea către fișierul video. Diferența este că nu trebuie folosit atributul *hidden*, iar pentru dimensiuni, acestea nu trebuie să fie mai mici decât dimensiunile în care este salvat fișierul video. Totuși, afișarea de imagini video necesită mai mult timp deoarece acestea au dimensiuni destul de mari.

Tag-ul *<embed>* se folosește și pentru adăugarea de **imagini flash** cu extensia *.swf*, acesta fiind încadrat în alt element *<object>*.

```
<object width = ”580” height = ”400”>
<param name = ”movie” value = ”fisier.swf”>
<embed src = ”fisier.swf” width = ”580” height = ”400”>
</embed>
</object>
```

Width și **height** sunt dimensiunile în care este afișată imaginea flash, iar la atributele *value* și *src* se scrie calea către fișierul *.swf*. Trebuie scrisă aceeași cale la ambele atribute. Afișarea în pagină a elementelor cu *<embed>* necesită folosirea unui **plugin** care, de cele mai multe ori, dacă acesta nu este deja instalat este cerut automat să fie încărcat de către browser.

Elemente de animație text:

HTML-ul are un tag special *<marquee> … </marquee>* prin care putem crea un efect de mișcare a unui text sau a unei imagini. Tag-ul *<marquee>* are următoarele elemente:

- **loop** – definește de câte ori se va afișa textul;
- **height** – definește înălțimea în pixeli a zonei în care se mișcă textul;
- **width** – definește lungimea în pixeli a zonei în care se mișcă textul;
- **bgcolor** – definește culoarea zonei în care se mișcă textul;
- **direction** – definește direcția de mișcare: *left, right, up* și *down*;
- **behavior** – definește cum se va mișca textul cu ajutorul scroll-ului slide-urilor alternativ;
- **scrolldelay** – setează numărul în milisecunde între refresh-uri pe mișcare.

Exemplu de mișcare pe orizontală:

```
<marquee behavior = ”alternate” bgcolor = ”#00FFFF” direction = ”right”> Marquee Text
</marquee>
```

Exemplu de mișcare pe verticală:

```
<marquee behavior = ”alternate” bgcolor = ”#00FFFF” direction = ”up” width = ”100”
height = ”100”> Marquee Text </marquee>
```

În loc de text, poate fi folosită și o imagine, înlocuind textul, în cazul exemplelor de mai sus cu tag-ul *<img>* pentru imagine.

Includerea unei pagini HTML externe:

Înafară de modalitatea cu folosirea tag-ului *<iframe>* pentru afișarea în pagina Web HTML a conținutului din altă pagină HTML, avem modalitatea:

```
<iframe src = ”url_pagina.html” width = ”600” height = ”200” align = ”center” scrolling = ”yes”> </iframe>
```

Există și o altă soluție dată de tag-ul *<object>,* folosit cu atributele *data* și *type*. Următorul tag HTML este sintaxa generală de afișare într-o pagină Web a conținutului HTML aflat în altă pagină externă.

```
<object data = ”adresa_pagina_externa” type = ”text/html” width = ”600” height = ”400”> </object>
```

Avantajul utilizării uneia din aceste două metode pentru cei care fac site-uri doar cu HTML este faptul că același conținut HTML poate fi inclus în mai multe pagini din site, fiind scris o singură dată. De exemplu, în cadrul unui meniu ce trebuie afișat în toate paginile site-ului, codul acestuia poate fi scris într-un fișier special.

Div și Span

Tag-urile * * și ** nu au efecte importante dacă sunt folosite singure. Tag-ul * * crează secțiuni de blocuri în pagină a căror formă și grafică de conținut pot fi manipulate pentru fiecare separat. Tag-ul * * are doar un singur atribut HTML, respectiv *align*, pentru aliniere pe orizontală care poate avea următoarele valori: *left, right, center, justified.*

Tag-ul ** crează posibilitatea modificării separate a unei porțiuni dintr-un context, putând fi folosit asemenea CSS-ului. Singur, nu are nici un efect vizual propriu și nu folosește nici un atribut HTML special. Chiar dacă aceste două tag-uri sunt folosite singure, nu au efect major în combinație cu CSS-ul pot crea aspecte grafice importante. Pentru aceasta, ambele tag-uri pot folosi atributul *style* cu proprietăți CSS, ori atributele *align* sau *class* ca identificatori pentru stilurile CSS.

Tag-ul DIV: ...

Tag-ul * * este unul dintre cele mai folosite elemente HTML deoarece în combinație cu proprietăți CSS poate crea efecte grafice deosebite, iar în interiorul lui pot fi incluse oricare elemente HTML precum tabele, formulare, marcatori, linii sau alte tipuri. Cadrul în care acestea sunt adăugate poate avea propriul fundal background, lungime, înălțime și margini cu diferite linii.

Exemple:

```
<div style = ”width:250px”; background:#AAEE88; border:1px; solid blue;”>
	<form action = ”method = ”post”>
```

```
            Nume: <input type = ”text”> </input> <br>
            E-mail: <input type = ”text”> </input> <br>
            <input type = ”submit” value = ”Trimite> </input>
      </form>
</div>
```

```
<div style = ”width:180px; background:#88AAFE; border:5px; outset #888888;”>
      <ul>
            <li> Linie 1 </li>
            <li> Linie 2 </li>
            <li> Linie 3 </li>
      </ul>
</div>
```

În atributul *style* sunt specificate proprietățile CSS care stabilesc aspectul grafic al fiecărui tip, cum ar fi mărimea în pixeli, fundalul și respectiv bordura. Un alt aspect important este și faptul că putem poziționa conținutul din interiorul tag-ului oriunde în pagină folosind proprietăți CSS precum:

• **position** – care poate lua valorile: *static, relative*, *fixed* sau *absolute margin*.

• **margin** – care stabilește distanța dintre marginea cadrului secțiunii și elementele din jurul lui.

Tag-ul SPAN:

La tag-ul ** putem adăuga stiluri grafice unei anumite porțiuni dintr-un context. Pentru aceasta, trebuie folosit împreună cu proprietățile CSS care pot fi adăugate printr-un atribut *style* în interiorul etichetei ** sau în foi de stil. Pentru a înțelege mai bine, mai jos avem un exemplu la care se scoate în evidență anumite cuvinte dintr-un text. Pentru aceasta, încadrăm cuvintele respective într-un tag ** căruia îi adăugăm proprietățile dorite:

```
Aceasta este o lecție din <span style = ”background:#88FE88; font-weight:bold;”>
Cursul HTML </span> din cadrul Tehnologiilor Web.
```

Tag-ul ** poate fi folosit ca și o clasă pentru CSS, astfel proprietățile adăugate elementului ** într-o foaie de stil vor fi transferate tuturor elementelor din pagină, care sunt încadrate în tag-urile *.*

Exemplu:

```
<html>
      <head>
            <title> Titlul </title>
            <style type = "text/css">
                  span {border: 2px solid red; padding: 1px;}
            </style>
      </head>

      <body>
            <h4> Exemplu de <span>text cu eticheta SPAN</span> in interiorul frazei.
</h4>
```

```
            <ul>
                  <li> Linia 1 </li>
                  <li><span> Linie 2, in span </span></li>
                  <li> Linia 3 </li>
                  <li><span> Linie 4, in span </span></li>
                  <li> Linia 5 </li>
            </ul>
      </body>
</html>
```

Cursul 10

Diferenta dintre DIV si SPAM

Diferenta este faptul ca DIV-ul încadrează o sectiune din document sub forma unui bloc, iar SPAN-ul încadrează o porțiune din context sub forma de linii.

Exemplu: Înțelegem cum atribuim aceeași bordura grafica unui tag DIV si unui tag SPAN

```
<div style=”...”>
</div>
```

```
<span style=„.....;">
</span>
```

Din aceasta cauza este indicată folosirea tag-ului ... pentru conținutul din interiorul unei linii, iar cand sunt mai multe elemente *DIV* și *SPAN*, în pagina, este indicat folosirea atributului *id* sau *class*, acestora le acordăm proprietățile css, o singura data in head sau fișier css.

Este mai efecient decat sa scriem în *DIV* și *SPAN* cate un atribut style.

Introducere in JavaScript

Scurt istoric:

JavaScript a fost dezvoltat prima data de catre firma Netscape, cu numele de Live Script, un limbaj de script care extindea capacitatile HTML si oferea posibilitatea de a introduce programe in paginile web afisate de catre browser-ul Netscape Navigator.

Dupa lansarea limbajului Java, Netscape a inceput sa lucreze cu firma Sun, cu scopul de a crea un limbaj de script cu o sintaxa si semantica asemanatoare cu a limbajului Java, si din motive de marketing numele noului limbaj de script a fost schimbat in "JavaScript".

Daca toata logica este pe partea de server, intreaga prelucrare este facuta la server chiar si pentru lucruri simple, asa cum este validarea datelor.

JavaScript a permis scrierea aplicatiilor web moderne, aplicatii cu care se poate interactiona direct, fara a fi necesara reincarcarea paginii de fiecare data.

Limbajul HTML ofera autorilor de pagini web, o anumita flexibilitate, dar statica. Documentele HTML nu pot interactiona cu utilizatorul in alt mod, mai dinamic, decat pune la dispozitia acestuia legaturi la alte resurse (url-uri).

Crearea de CGI-uri ce sunt programe care ruleaza serverul web si care accepta informatii primite din pagina web si returneaza cod HTML adus la imbogatirea posibilitatilor de lucru.

Astfel, un pas important spre interactivizare a fost realizat de JavaScript care permite inserarea in paginile web a scripturilor care se executa in cadrul paginii web, mai exact in cadrul browser-urului utilizatorului, usurand astfel si traficul dintre server si client.

Exemplu: Intr-o pagina pentru colectarea de date de date de la utilizator se pot adauga script-uri JavaScript pentru a valida corectitudinea introducerii si apoi pentru a permite server-ului doar date corecte spre procesare.

JavaScript contine o lista destul de ampla de functii si comenzi menite sa ajute la operatii matematice, manipulari de siruri, sunete, imagini, obiecte si ferestre ale browser-ului, link-urile URL si verificari de introduceri ale datelor in formulare.

Codul necesar acestor actiuni poate fi inserat in pagina web si executat pe calculatorul vizitatorului.

JavaScript1.0 este aproximativ compatibil cu JavaScript1.1 care este recunoscut de catre Netscape Navigator. Totusi, versiunile ulterioare de JavaScript si diversele diferente specifice platformelor de operare au inceput sa dea destule probleme programatorilor web si astfel, Netscape si Microsoft si alti distribuitori au fost de acord sa predea limbajul unei organizatii internationale. Aceasta a finalizat o specificatie de limbaj cunoscuta ca ECMASCRIPT recunoscuta de toti distribuitori.

Desi standardul ECMA este util, atat Netscape cat si Microsoft au propriile lor implementari ale limbajului si continua sa extinda limbajul dincolo de standardul de baza.

Pe langa JavaScript, Microsoft a introdus un concurent pentru JavaScript numit VBScript realizat pentru a usura patrunderea pe web a programatorilor VB.

VBScript este un subset a limbajului Visual Basic cu toate acestea, JavaScript, cunoscut ca limbaj descriptiv standard pentru web.

In general se considera ca exista 10 aspecte fundamentale ale limbajului JavaScript pe care orice programator ar trebui sa le cunoasca.

Versiuni de JavaScript

In momentul in care utilizam JavaScript in paginile noastre este important sa stim pe ce browsere va trebui sa ruleze paginile. De exemplu, o pagina in care utilizam JavaScript cu clase nu va fi corect interpretata de Internet Explorer9 sau FireFox4.

An	Denumire oficiala	Descriere
1997	ECMAScript 1	Prima editie
1998	ECMAScript 2	Schimbari editoriale

1999	ECMAScript 3	Sunt adaugate expresiile regulare si try/catch
	ECMAScript 4	S-a sarit peste aceasta versiune
2009	ECMAScript 5	S-a adaugat „strict mode” si suport de Json
2011	ECMAScript 5.1	Schimbari editoriale
2015	ECMAScript 6 sau ECMAScript 2015	S-au adaugat clase si module
2016	ECMAScript 7 sau ECMAScript 2016	S-a adaugat operatorul exponential (**) si Array.prototype.includes.

De fapt razboiul browserelor a inceput in anii ‘90 si continua si in ziua de azi si face uneori programarea web o adevarata aventura din cauza lipsei implementarii standardelor pe diferite browsere.

O lista completa a facilitatilor o gasim la adresa https:

Caracteristici ale JavaScript-ului

I. JavaScript-ul poate fi introdus in HTML:

De obicei, codul JavaScript este gazduit in documentele HTML si executat in interiorul lor prin intermediul unor etichete specifice: JavaScript foloseste HTML pentru a permite realizarea aplicatiilor web.

II. JavaScript este dependent de mediul pe care ruleaza:

JavaScript este un limbaj descripting software-ul care ruleaza de fapt programul este browser-ul web.

Este important sa luam in considerare aceasta dependenta de browser atunci cand utilizam aplicatii JavaScript.

III. JavaScript este un limbaj in totalitate interpretat, codul scriptului va fi interpretat de browser inainte de a fi executat.

JavaScript nu necesita compilari sau reprocesari ci ramane partea integranta a documentului HTML.

Comenzile JavaScript vor fi citite de navigatorul web si procesate atunci cand utilizatorul apeleaza la acele functii prin completare de formulare, apasare de butoane, etc.

Avantajul principal este faptul ca ruleaza local nu avem cereri la server.

IV. JavaScript este un limbaj flexibil in aceasta privinta, limbajul difera radical de c++ sau de Java.

In JavaScript putem declara o variabila desi nu-i cunoastem tipul specificat inainte de rulare.

V. JavaScript este bazat pe obiecte.

JavaScript nu este un limbaj de programare orientat obiect pe Java ci mai corect este bazat pe obiecte.

Modul de obiect, JavaScript, este bazat pe instanta si nu pe mostenire.

VI. JavaScript este condus de evenimente.

Mare parte a codului JavaScript raspunde la evenimente generate de utilizator sau sistem

Obiectele HTML cum ar fi butoanele sunt imbunatatite pentru a accepta handlere de evenimente.

VII. JavaScript evolueaza, limbajul evolueaza, fapt pozitiv care, insa, poate genera probleme, programatorii trebuie sa verifice permanent ce versiune sa foloseasca pentru ca aplicatiile sa poata fi disponibile unui numar mare de utilizator de browsere diferite.

VIII. JavaScript acopera contexe diferite

Programarea cu acest limbaj este indreptata mai ales catre partea de client, dar putem folosi JavaScript si pentru partea de server.

JavaScript este limbajul nativ pentru unele instrumente de dezvoltare web ca Borland IntraBuilder sau Macromedia Dreamweaver, dar este utilizat si pentru unele sisteme de gestiunea bazelor de date cum ar fi cogeDB.

Editoare JavaScript

Orice editor de texte in care putem scrie HTML va fi utilizat pentru scrierea de cod JavaScript.

Probabil cea mai buna solutie este utilizarea NotePad, iar pentru testare: incarcarea directa pe un browser.

Cu toate acestea, o serie de editoare sau medii de dezvoltare ofera facilitatile de dezvoltare.

Cateva exemple:

- ★ • Notepad++
- ★ • Netbeans
- ★ • Visual Studio
- ★ • Brackets

Unele aplicatii permit o previzualizare a paginii in timp real direct in mediu, altele pot folosi un browser pentru acest lucru.

O alta varianta ar fi utilizarea editoarelor on-line specializate, cum ar fi:

- ★ • https://js.do/
- ★ • https://jsbin.com/
- ★ • https://www.tutorialspoint.com/online_javascript_editor.php
- ★ • Tryit Editor v3.5 de pe www.w3schools.com

Exemple:

Editorul Brackets:

```
<html>
<head>
        <title>alert2</title>
</head>
```

```
<body>
    <p> test </p>
    <script type='text/javascript'>
        alert('Imi place Javascript!!');
    </script>
</body>
</html>
```

Editor w3schools:

```
<!DOCTYPE html>
<html>
<body>
<h2>My First JavaScript</h2>
<button type="button" onclick="document.getElementById('demo').innerHTML=
Date()">
Click me to display Date and Time.
</button>
<p id="demo"></p>
</body>
</html>
```

Cursul 11

Adaugarea codului JavaScript intr-o pagina HTML

JavaScript in-line

Pentru a insera JavaScript intr-un document HTML deja existent este necesara introducerea in fisier a etichetei <script>...</script>. Aceasta eticheta necesita atributul type care se pune in interiorul scriptului.

Type are urmatoarea sintaxa:

type="text/javascript"

Atributul type va spune browser-ului ca script-ul este scris in format line-text.

Exemplu:

```
<html>
<head>
```

```
    <title>Imi place Javascript</title>
</head>
<body>
    <p> Hello World!
</body>
</html>
```

La aceasta secventa putem pune comentariu, iar la script putem adauga secventa in orice zona dorim:

<script type="text/javascript"> //Aici scriem cod JavaScript </script>

De exemplu, il putem adauga in <head> sau in <body>.

Chiar daca nu exista limita in privinta regiunii din pagina in care scriem cod JavaScript, exista 2 reguli pe care ar fi bine sa le respectam:

I. Daca codul JavaScript genereaza continut dinamic in timpul incarcarii paginii. Vom scrie codul in locul in care dorim sa apara acel continut, de exemplu daca dorim afisarea mesajului "Hello World!" folosind JavaScript, acesta trebuie scris in sectiunea <body>.

II. Daca codul JavaScript genereaza cod dinamic in timpul incarcarii paginii, ar trebui sa punem codul JavaScript la finalul pagini HTML. Motivul este ca in momentul in care browser-ul intalneste un tag <script> se opreste, compileaza, executa script-ul si abia apoi continua cu generarea paginii. Acest fapt poate duce la cresterea timpului de incarcare a paginii, deci este mai bine ca browser-ul sa dea de codul JavaScript dupa ce a afisat pagina.

JavaScript extern:

Putem, de asemenea, sa introducem instructiunile JavaScript intr-un alt fisier extern care va avea extensia ".js".

Pentru editarea acestui fisier de nevoie, la fel de un editor simplu de texte, avantajul este ca putem folosi acelasi cod in mai multe pagini HTML si in cazul necesitatii unei modificari in codul JavaScript, modificam doar datele dintr-un singur fisier, cel cu extensia ".js".

Dezavantajul acestei metode este faptul ca intr-un fisier extern cu extensia ".js" nu putem folosi etichete HTML, ci numai instructiuni JavaScript.

In cazul in care codul JavaScript se afla intr-un fisier extern, eticheta script din pagina HTML va trebui sa contina atributul src a carei valoare determina locatia fisierului in care se afla codul JavaScript.

In fisierul extern, cu extensia ".js", nu trebuie sa scriem eticheta <script>, ci scriem direct instructiunile scriptului.

Exemplu:

```
<script type='text/javascript' src='doarUnComentariu.js'></script>
```

Ca recomandare este introducerea “<meta>” in portiunea de <head> a documentelor care folosesc JavaScript. Aceasta eticheta specifica limbajul prestabilit, folosit pentru toate script-urile JavaScript.

Exemplu:

```
<meta http-equiv="Content-Script-Type" content="text/javascript">
```

Totusi aceasta eticheta nu este obligatorie, script-urile functionand foarte bine si fara ea.

Ascunderea codului in browserele vechi

Unele browsere nu recunosc script-urile si le afiseaza in pagina web ca text. Pentru a evita acest lucru, putem folosi eticheta HTML pentru comentarii:<!-- text //→, delimitand cu aceasta instructiune restul instructiunilor JavaScript si evitam sa apsara scriptul in pagina web.

Exemplu:

```
<div id='sectiune'></div>
<script type='text/javascript'>
<!--
document.writeln("Hello world!");
//-->
</script>
```

Gestionarea intrarilor si iesirilor in JavaScript

Unul dintre primele lucruri care trebuie stapanite cand se invata un nou limbaj sunt intrarile si iesirile.

Exista 3 metode diferite de a comunica cu utilizatorul:

I. Output cu writeln:

Exemplu:

```
<script type='text/javascript'>
    document.writeln('Hello World!');
</script>
```

Aceasta comanda poate fi utilizata in timp ce se construieste pagina. Dupa ce pagina s-a terminat de incarcat, pe unele browsere o comanda writeln v-a duce la stergerea paginii, deci utilizam comanda doar in timpul incarcarii paginii.

In timpul incarcarii paginii, JavaScript, va intalni bucata de cod si o sa scrie mesajul exact in locul in care blocul script apare in pagina, obtinand rezultatul afisat.

II. Output cu alert:

Metoda 2 de afisare este utilizarea unei ferestre de alerta a browser-ului. Este utilizata cand invatam JavaScript sau cand dorim sa depanam un script, dar nu este recomandata ca mod de interactiune pe o pagina, deoarece intrerupe executarea script-ului pana cand se apasa “OK”, ceea ce devine frustrant pentru utilizator.

Exemple:

```
<script type='text/javascript'>
alert('Hello World!');
```

```
</script>

<script type="text/javascript">
document.writeln("Afisare cu writeln");
alert("Afisare cu alert");
</script>
```

III. Output cu getElementById:

Aceasta ultima metoda este cea mai puternica si complexa, ideea este ca orice bloc dintr-o pagina HTML poate sa fie etichetat de un identificator unic.

JavaScript permite gasirea si utilizarea acelor blocuri care au fost etichetate.

Exemplu:

```
<div id='sectiune'></div>
<script type='text/javascript'>
document.getElementById('sectiune').innerHTML = 'Hello World!';
</script>
```

Explicatii:

- Am creat un DIV si l-am etichetat cu numele “sectiune”, in acest moment, JavaScript va putea utiliza acest nume pentru a accesa sectiunea si pentru a face modificari asupra ei;
- Am explicat instructiunea pe bucati:
  - ➢ document= in pagina curenta;
  - ➢ getElementById('sectiune')= cauta un bloc denumit sectiune;
  - ➢ innerHTML = 'Hello World!’=schimba HTML din bloc in Hello World!

Continutul inner HTML poate fi orice cod HTML, asa cum poate fi observat in exemplul urmator, in acest caz inner HTML va procesa si redesena pagina cu noul continut.

Exemplu:

```
<div id='sectiune'></div>
<script type='text/javascript'>
document.getElementById('sectiune').innerHTML= 'Hello <font  style="color:blue"> blue
</font>world!';
</script>
```

Input pe evenimentul onClick

Asa cum fiecarui bloc putem sa-i asociem un identificator unic, intr-un mod similar putem asocia evenimente.

In exemplul urmator asociem pe DIV cu evenimentul onClick cu rularea functiei Goodbye.

Daca functiile vor fi prezentate pe larg, se intelege ideea generala ca o functie este un bloc caruia i-am dat un nume, nu o sa fie executata.

Exemplu:

```
<div id='sectiune' onClick='functie()'> </div>
<script type='text/javascript'>
```

```
        document.getElementById('sectiune').innerHTML = 'HelloWorld!';
            function functie() {
                document.getElementById('sectiune').innerHTML ='Goodbye World!';
            }
    </script>
```

Input de date de la utilizator

In momentul in care dorim ca utilizatorul sa introduca o serie de date pentru a fi prelucrate putem utiliza un input si un buton.

Acestea sunt elemente standard pentru formulare HTML, dar nu sunt in cadrul unui form pentru ca nu va urma sa trimitem informatia la server, in schimb la apasarea butonului apelam functia “prelucreaza”.

Exemplu:

```
<input id='intrare' size=20> <button onClick='prelucreaza()'>Trimite</button>
    <br>
        <div id='rezultat'></div>
        <script type='text/javascript'>
            function prelucreaza() {
                var v = document.getElementById('intrare').value;
                document.getElementById('rezultat').innerHTML = 'Ai introdus:'+v;
            }
        </script>
```

Exemplu:

```
<input id="v1" size=5 />
    +
    <input id="v2" size=5 />
    <input type="button" value="=" onclick="addNumbers()" />
    <input id="suma" size=5 />
    <script type="javascript">
            function addNumbers()
            {
                var val1 = parseInt(document.getElementById("v1").value);
                var val2 = parseInt(document.getElementById("v2").value);
                document.getElementById("suma").value = val1+val2;
            }
     </script>
```

Conventii de sintaxa

Regulide sintaxa a limbajului JavaScript:

In orice limbaj, scrierea are conventii si reguli de sintaxa, scrierea in limba romana are regulile ei de sintaxa (ex: o propozitie incepe cu litera mare, etc.).

La fel, orice limbaj de programare are regulile si sintaxa lor.

Regulile de sintaxa a limbajului JavaScript:

I. “case sensitive”= face diferenta intre litere mari si litere mici: “exemple”, “Exemple” vor fi tratate diferit.

II. “;”= toate declaratiile se termina cu el

III. spatii livere= JavaScript ignora spatiile libere, tab-urile si liniile libere care apar in instructiuni, acestea sunt utilizate pentru a face codul mai bine structurat si usor de citit, recunoaste doar spatii care apar in string

IV. ghilimele= cele simple si duble sunt folosite pentru a delimita sirurile de caractere

V. caractere speciale= cand scriem scripturi apare necesitatea de a folosii in cod sau datele de iesire un caracter special sau apasare de tasta: tab sau o linie noua (ex: \in fata codului escape):

Exemple:

- \b
- \f =indica o pagina noua;
- \n=indica o linie noua;
- \r=indica un retur de caractere;
- \t=indica o apasare a tastei tab;
- \\=indica un caracter \;
- \’=indica apostrof sau ghilimele simple;
- \”=indica apostrof sau ghilimele duble.

VI. Comentarii: “//”

Daca vrem sa scriem comentarii pe mai multe linii folosim: /* …*/

VII. numele variabilor si functiilor trebuie sa respecte regulile

1. Primul caracter trebuie sa fie o litera, un caracter de subliniere sau $
2. Primul caracter nu poate fi o cifra
3. Numele nu trebuie sa contina spatii libere
4. Nu se folosesc cuvinte rezervate care fac parte din limbajul JavaScript: exemplu: array, status, alert, deoarece interpretorul programului nu va face diferenta intre nume si comenzile JavaScript cu acelasi nume.

Cursul 12

JavaScript este un limbaj bazat pe evenimente.

Dupa cum am putut observa in primul curs de JavaScript, codul reactioneaza in functie de evenimentele setate de noi.

Exemplu: onClick pe button

Chiar daca vom revenii in cursul urmator, dam mai departe o lista simplificata a evenimentelor JavaScript si o scurta descriere a lor.

Eveniment	Descriere

onAbort	O imagine nu a reusit sa se incarce
onBeforeUnload	Utilizatorul navigheaza in afara paginii
onBlur	Un camp nu mai are focusul
onChange	S-a modificat continutul campului
onClick	Utilizatorul a facut click pe acest obiect
onDblClick	Utilizatorul a facut double-click pe acest obiect
onError	Eroare la incarcarea paginii
onFocus	Utilizatorul tocmai aintrat e acest obiect
onKeyDown	S-a apasat o tasta
onKeyPress	SimilarS-a apasat tasta si i s-a dat drumul
onKeyUp	O tasta a fost ridicata
onLoad	S-a terminat de incarcat obiectul
onMouseDown	S-a apasat un buton al mouse-ului
onMouseMove	S-a deplasat mouse-ul
onMouseOut	Pozitia mouse-ului s-a deplasat in afara obiectului
onMouseOver	POzitia mouse-ului este deasupra acestui element
onMouseUp	A fost eliberat butonul mouse-ului
onReset	S-a apasat butonul form rest
onResize	S-a redimensionat o fereastra
onSelect	S-a selectat un text
onSubmit	S-a apasat butonul Submit al unei forme
onUnload	Utilizatorul navigheaza in afara paginii

Exemplu:

```
<body>
    <input id='intrare' size=60> <button onClick='prelucreaza()' onblur="plecat()"
onfocus="intrat()">Trimite</button> <br>
    <div id='rezultat'></div>
    <div id='status'></div>
    <script type='text/javascript'>
```

```
        function prelucreaza() {
                var UI = document.getElementById('intrare').value;
                document.getElementById('rezultat').innerHTML = 'Ai introdus: '+ UI;
        }
        function intrat() {
                 document.getElementById('status').innerHTML = 'Ai intrat cu mouse-ul pe
buton';
        }
        function plecat() {
                document.getElementById('status').innerHTML = 'Ai iesit cu mouse-ul de pe
buton';
        }
    </script>
</body>
```

Exemplu:

```
<body onresize="Redimensionare()">
    <p>Redimensioneaza fereastra!</p>
    <p id="paragraf"></p>
    <script>
        function Redimensionare() {
            var I = window.outerWidth;
            var L = window.outerHeight;
            var text = "Inaltime fereastra:" + I + "<br> Latime fereastra:" + L;
             document.getElementById("paragraf").innerHTML =text;
        }
    </script>
</body>
```

Cometariile in JavaScript

Utilizate pentru adaugarea unor secvente de text care sunt ignorate de catre interpretor, dar care ofera explicatii suplimentare programatorului, comentariile in JavaScript sunt similare celor din C++.

Comentarii pe o singura linie

"//" ii va spune JavaScript-ului sa ignore tot pana la sfarsitul liniei.

Exemplu:

```
<body>
    <button onClick="Plus()"> + </button>
    <button onClick="Minus()"> - </button>
    <p id="paragraf"> Apasa un buton! </p>
```

```
    <script>
        function Plus() { // se executa la apasarea butonului +
            var text = "12+13=25"; // se pregateste textul de afisat
            document.getElementById("paragraf").innerHTML = text;
            // se modifica paragraful cu noul mesaj
        }
        function Minus() { // se executa la apasarea butonului -
            var text = "12-13=-1"; // se pregateste textul de afisat
            document.getElementById("paragraf").innerHTML = text;
            // se modifica paragraful cu noul mesaj
        }
    </script>
</body>
```

Blocul de comentarii

Acest bloc incepe cu “/*”, JavaScript va ignora tot pana la intalnirea secventei “*/” care este indicatorul de sfarsit de comentarii.

Exemplu:

```
<body>
    <button onClick="Plus()"> + </button>
    <button onClick="Minus()"> - </button>
    <p id="paragraf"> Apasa un buton! </p> <script>
        function Plus() {
/* functia se executa la apasarea butonului + in prim linie de cod se
pregateste textul de afisat iar in a doua se modifica paragraful cu
noul mesaj */
            var text = "12+13=25";
            document.getElementById("paragraf").innerHTML = text;
        }
        function Minus() {
/* functia se executa la apasarea butonului - in prim linie de cod se
pregateste textul de afisat iar in a doua se modifica paragraful cu
noul mesaj */
            var text = "12-13=-1";
            document.getElementById("paragraf").innerHTML = text;
        }
    </script>
</body>
```

Riscuri in utilizarea comentariilor JavaScript

Chiar daca recomandam utilizarea comentariile pentru a mentine claritatea codului, in JavaScript avem 2 situatii speciale care nu apar in alte limbaje (de exemplu: C++).

In primul rand, comentariile sunt transmise impreuna cu codul la fiecare reincarcare de pagina, ceea ce va duce la un timp de transmitere a datelor mai mare si implicit, un timp de incarcare a pagini crescute.

Pentru a rezolva aceasta problema merita sa tinem 2 versiuni ale paginilor noastre.

Cea cu comentarii: va fi utilizata in faza dezvoltarii aplicatiei web in momentul in care decidem ca am finalizat o versiune de productie, vom utiliza un software utilizat care va scoate comentariile din JavaScript si vom utiliza acea pagina, cea cu comentariile va ramane in continuare versiunea de baza pe care vom face , eventuale verificari.

Utilizarea altor optiuni vor reduce mai mult codul, dar il vor face mult mai greu de citit de programator, browserele nemaifacand diferenta in interpretarea codului transmis.

A doua problema legata de comentariile din JavaScript este legata de faptul ca browserele cauta tag-ul <script> si in momentul gasirii vor inceta procesarea JavaScript si vor trece la procesarea HTML.

Exemplu:

```
<body>
    <script>
        alert("</script>");
    </script>
</body>
Rezultat: “);
```

O solutie ar fi spargerea textului in ceva ce browserele ar recunoaste ca tag.

In cazul aparitiei tag-ului in comentarii vom avea exemplul urmator.

Exemplu:

```
 <body>
    <script>
        var x = 5;
/*
Browserul se opreste din interpretarea JAvaScript
cand da de tagul </script>
        Tot ce urmeaza dupa acest tag va fi tratat ca HTML.
        Ar fi fost bine sa se afiseze valoarea 10
*/
        alert(2*x);
    </script>
</body>
```

Daca stricam tag-ul din comentariu, pagina va functiona asa cum ne dorim.

Exemplu:

```
<body>
    <script>
        var x = 5;
/* Browserul se opreste din interpretarea JAvaScript cand da
de tagul </sc ript>
```

```
        Tot ce urmeaza dupa acest tag va fi tratat ca HTML.
        Ar fi fost bine sa se afiseze valoarea 10
*/
        alert(2*x);
        </script>
</body>
```

Variabile

Tipul variabilelor JavaScript:

In general, o variabila este un nume asociat unei locatii de memorie in care vom retine valori. Numele variabilei va permite accesul la valoarea acestei locatii de memorie pentru citire si modificare.

JavaScript nu este un limbaj proni type ca C++ sau Dober. Prin JavaScript nu ne intereseaza, de obicei, tipul datei pe care il retine o variabila. In JavaScript o variabila poate retine orice si nu trebuie sa fie declarata. Se poate defini o variabila si ii atribuim o valoare prin 2 metode:

I. Utilizarea cuvintelor cheie “var” pentru variabile locale
   Exemplu: var x=3;
II. Utilizarea directa a variabilei pentru variabile globale
   Exemplu: x=123;

Numele variabilei nu poate fi un cuvant rezervat si nu poate incepe ca niciun simbol special, mai putin cu linie de subliniere sau cu $.

Tipul variabilei este dat la atribuire si acesta poate sa isi schimbe tipul in timpul executiei codului.

Exemplu:

```
<body>
    <script>
        var x = 'Sir de caractere';
        document.writeln("variabila x este sir de caractere cu valoarea: " + x + "<br>");
        x = 123;
        document.writeln("variabila x este numar intreg cu valoarea: " + x + "<br>");
        x = 3.14;
        document.writeln("variabila x este numar real cu valoarea: " + x + "<br>");
    </script>
</body>
```

Variabila va avea 3 tipuri diferite la rularea scriptului, in C++ nu functioneaza.

Tipul string

Exemplu:

```
<body>
    <script>
        var suntUnString = 'Sunt un string';
```

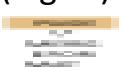
```
            document.writeln("variabila suntUnString este sir de caractere cu valoarea: " +
suntUnString + "<br>");
            var totUnString = "123";
            document.writeln("variabila totUnString este sir de caractere cu valoarea: " +
totUnString + "<br>");
            var suntNumarIntreg = 123;
            document.writeln("variabila suntNumarIntreg este numat intreg cu valoarea: " +
suntNumarIntreg + "<br>");
            var rezultat1 = totUnString + suntNumarIntreg;
            document.writeln("totUnString + suntNumarIntreg va da un string: " +  rezultat1 +
"<br>");
            var rezultat2 = suntNumarIntreg + suntNumarIntreg;
            document.writeln("totUnString + suntNumarIntreg va da un string: "+ rezultat2 +
"<br>");
        </script>
</body>
```

Tipuri numerice

Exemplu:

```
<body>
    <script>
        var suntIntreg = 123;
        document.writeln("variabila suntIntreg este un intreg cu valoarea: " + suntIntreg +
"<br>");
        var suntReal = 3.14;
        document.writeln("variabila suntReal este un real cu valoarea: " + suntReal +
"<br>");
        var adunareCuEroare = 0.05 + 0.01; // grija la adunarea numerelor reale
        document.writeln("variabila adunareCuEroare este un real cu valoarea: " +
adunareCuEroare + "<br>");
        var intregExponent = 1.23e+4; // scrierea sub forma aEb -> a * 10*b
        document.writeln("variabila intregExponent este un intreg cu valoarea: " +
intregExponent + "<br>");
        var intregHexadecimal = 0x10; // scrierea in haxazecimal e precedata de 0x
        document.writeln("variabila intregHexadecimal este un intreg cu valoarea: " +
intregHexadecimal + "<br>");
        var intregOctal = 010; // valorile in baza 8 incep cu 0
        document.writeln("variabila intregOctal este un intreg cu valoarea: " + intregOctal +
"<br>");
    </script>
</body>
```

Tipul boolean (logice)



Exemplu:

```
<body>
    <script>
        var adevarat = true;
        document.writeln("variabila adevarat este un boolean cu valoarea: " + adevarat +
"<br>");
        var fals = false;
        document.writeln("variabila fals este un boolean cu valoarea: " + fals + "<br>");
        var suntIntreg = 123;
        var suntString = "123";
        var egaleCaValoare = (suntIntreg == suntString);
        document.writeln("variabila egaleCaValoare este un boolean cu valoarea: " +
egaleCaValoare + "<br>");
        var egaleCaValoareSiTip = (suntIntreg === suntString);
        document.writeln("variabila egaleCaValoareSiTip este un boolean cu valoarea: " +
egaleCaValoareSiTip + "<br>");
    </script>
</body>
```

Varibile de tip vector, obiect sau functie

Exemplu:

```
<body>
    <script>
        var suntVector = [0, 'unu', 2, 3, 'patru', 5]; // vector de valori de tipuri diferite
        var patru = suntVector[4]; // string
        var obiect = { // obiect JavaScript
            'culoare': 'rosu',
            'copac': 'cires',
            'numere': [0, 1, 2, 3],
            'functie': f() {
                    alert('Salut!'); } }
        var culoare = obiect.culoare; // string-ul 'rosu'
        var oFunctie = f2() {
              return "Mesaj din functie"; }
        var altaFunctie = oFunctie; // lipsa () face sa nu avem o rulare a functiei, ci
atribuire de functii
        var mesaj = altaFunctie(); // acum functia va fi executata, si va fi atribuita valoarea
"Mesaj din functie"
    </script>
```

```
</body>
```

Durata de viata a variabilelor

O variabila declarata cu “var” in cadrul unei functii este o variabila locala, valoarea ei fiind recunoscuta numai in cadrul acelei functii.

La iesire din functie variabila este distrusa, astfel o alta functie poate declara si folosi o variabila cu acelasi nume. JavaScript trateaza cele 2 variabile distrincte.

Se pot declara si variabile in afara oricarei finctie care sunt vizibile globale si sunt variabilele care sunt variabile de la incarcarea pagini web pana la inchidere.

In orice script, de asemenea, variabilele declarate fara var, in functii vor fi globale.

Exemplu:

```
<body>
    <script>
        var a = 'Globala 1 - din afara functiilor <br>';
        function functie() {
            b = 'Globala 2 - din functie <br>';
            var c = 'Locala - definita in functie <br>';
            document.writeln("--------- In functie ---------- <br>");
            document.writeln(a);
            document.writeln(b);
            document.writeln(c); }
     document.writeln("---- Inainte de rularea functiei --- <br>");
     document.writeln(a);
     // document.writeln(b);// document.writeln(c);
     functie();
     document.writeln("---- Dupa rularea functiei -------- <br>");
     document.writeln(a);
     document.writeln(b);
    // document.writeln(c);
     </script>
</body>
```

Cuvinte cheie speciale

JavaScript ofera o serie de cuvinte speciale pe care le putem utiliza in conjunctie cu variabilele si expresiile distincte.

Cuvant cheie	**Descriere**
NaN (Not A Number)	Apare cand o expresie aritmetica returneaza un rezultat invalid. Ciudatenia lui Nan este ca nu este niciodata egal cu el insusi. Pentru a verifica daca o expresie este NaN vom utilzia functia

	isNaN(expresie). In expresii logice NaN se evalueaza la false.
Infinity	Apare cand o operatie depaseste precizia maxima oferita de JavaScript. Pentru a afla valoarea maxima si cea minima oferita de JavaScript putem utiliza Number.MAX_VALUE si Number.MIN_VALUE
null	Utilizat pentru a arata lipsa informatiei
undefined	Apare daca o variabila nu a fost inca declarata sau atribuita

Exemplu:

```
<body>
    <script>
        var a = Number.MAX_VALUE;
        var s = "cinci";
        var eroare = a * s;
        document.writeln(eroare + "<br>");
        var preaMare = a * a;
        document.writeln(preaMare + "<br>");
        var x;
        document.writeln(x + "<br>");
        var gol = null;
        document.writeln(gol + "<br>");
    </script>
</body>
```

Operatori

Cu ajutorul operatorilor manipulam, combinam si modificam datele dintr-un program sau script.

Acestia se pot clasifica dupa paritate:

I. operatori unari
   exemplu: +123;
II. operatori binari
   exemplu: 12+13;
III. operatori ternari
   exemplu: operator val?:

Vom prezenta tipurile uzuale de operatori din JavaScript clasificati dupa categoriile:

I. operatori de atribuire
II. operatori aritmetici
III. operatori logici si de comparare
IV. operatori pentru siruri

V. operatori convitionali

Operatori de atribuire

Forma:
variabila=expresie

In cazul acestui operator, JavaScript actioneaza mereu de la dreapta la stanga, se evalueaza expresia din dreapta, iar valoarea se atribuie variabila din stanga semnului “=”.

Operator	Exemplu	Similar cu
=	x = y	x = y
+=	x += y	x = x+y
-=	x -= y	x = x-y
*=	x *= y	x = x*y
/=	x /= y	x = x/y
%=	x %= y	x = x%y

Operatori aritmetici

Operatorii aritmetici, impreuna cu parantezele, ne permit sa realizam calcule pe numere intregi sau reale.

Operator	**Semnificatie**
+	Adunare
-	Scadere
*	Inmultire
/	Impartire
%	Modulo
++	Incrementare
--	Decrementare

++ sau --=prefix in fata variabilei cat si ca sufix sau valoare obtinuta

Exemple:

```
<body>
    <script>
        var a = 12;
```

```
            var b = 5;
            var S = a + b;
            document.writeln(a + "+" + b + "=" + S + "<br>");
            var D = a - b;
            document.writeln(a + "-" + b + "=" + D + "<br>");
            var P = a * b;
            document.writeln(a + "*" + b + "=" + P + "<br>");
            var R = a / b;
            document.writeln(a + "/" + b + "=" + R + "<br>");
            var M = a % b;
            document.writeln(a + "%" + b + "=" + M + "<br>");
        </script>
</body>

<body>
        <script>
            var a = 5;
            var b = ++a;
            document.writeln(a + " " + b + "<br>");
            var a = 5;
            var b = a++;
            document.writeln(a + " " + b + "<br>");
            var a = 5;
            var b = --a;
            document.writeln(a + " " + b + "<br>");
            var a = 5;
            var b = a--;
            document.writeln(a + " " + b + "<br>");
        </script>
</body>
```

Operatori logici si de comparare

Operator	Semnificatie
=	Atribuire
==	Egalitate de valoare
===	Egalitate de valoare si de tip
!=	Diferit ca valoare
!==	Diferit ca valoare sau tip

!	Negarea
&&	Si logic
||	Sau logic
>	Mai mare
<	Mai mic
>=	Mai mare sau egal
<=	Mai mic sau egal

Exemplu:

```
<body>
  <div style='font-family:Consolas'>
      <script>
      document.writeln("3 == 3 -> " + (3 == 3) + "<br>");
      document.writeln("3 == '3' -> " + (3 == '3') + "<br>");
      document.writeln("3 == 5 -> " + (3 == 5) + "<br>");
      document.writeln("---------------------------------<br>");
      document.writeln("3 === 3 -> " + (3 === 3) + "<br>");
      document.writeln("3 === '3' -> " + (3 === '3') + "<br>");
      document.writeln("3 === 5 -> " + (3 === 5) + "<br>");
      document.writeln("---------------------------------------<br>");
      document.writeln("3 != 3 -> " + (3 != 3) + "<br>");
      document.writeln("3 != '3' -> " + (3 != '3') + "<br>");
      document.writeln("3 != 5 -> " + (3 != 5) + "<br>");
      document.writeln("---------------------------------------<br>");
      document.writeln("3 !== 3 -> " + (3 !== 3) + "<br>");
      document.writeln("3 !== '3' -> " + (3 !== '3') + "<br>");
      document.writeln("3 !== 5 -> " + (3 !== 5) + "<br>");
      </script>
  </div>
</body>
```

Operatori relationali

Exemplu:

```
<body>
    <div style='font-family:Consolas'>
        <script>
            var x = 4;
            var rez = (3 < x && x < 8);
```

```
            document.writeln(x + " apartine intervalului [3,8] -> " + rez+"<br>");
            var x = 2;
            var rez = (3 < x && x < 8);
            document.writeln(x + " apartine intervalului [3,8] -> "+ rez+"<br>");
            var x = 25;
            var rez = (3 < x && x < 8);
            document.writeln(x + " apartine intervalului [3,8] -> "+ rez+"<br>");
            var x = 4;
            var rez = (x <= 3 || x >= 8);
            document.writeln(x + " nu apartine intervalului [3,8]"->"+rez+"<br>");
            var x = 2;
            var rez = (x <= 3 || x >= 8);
            document.writeln(x + " nu apartine intervalului [3,8] ->"+rez+"<br>");
            var x = 25;
            var rez = (x <= 3 || x >= 8);
            document.writeln(x + " nu apartine intervalului [3,8]-> "+rez+"<br>");
        </script>
    </div>
</body>
```

Relatiile lui De Morgan

Exemplu:

```
<body>
    <div style='font-family:Consolas'>
        <script>
        var x = 4;
        var rez1 = (3 < x && x < 8);
        document.writeln(x+"apartine intervalului [3,8] ->"+rez1+"<br>");
        var rez2 =!(3 < x && x < 8); // negare
        document.writeln(x+"nu apartine intervalului [3,8]->"+rez2+"<br>");
        var rez3 = !(3 < x) || !(x < 8); // De Morgan
        document.writeln(x + "nu apartine intervalului [3,8]->"+rez3+"<br>");
        var rez4 = (3 >= x || x >= 8); // De Morgan
        document.writeln(x + "nu apartine intervalului [3,8]->"+rez4+"<br>");
        </script>
    </div>
</body>
```

Operatori pentru siruri

Operator	Semnificatie
+	Concatenare
+=	Concatenare

Exemplu:

```
<body>
    <script>
        var S1 = "Imi place ";
        var S2 = S1 + "JavaScript";
        var S3 = "!";
        document.writeln(S1 + "<br>");
        document.writeln(S2 + "<br>");
        S2 += S3;
        document.writeln(S2 + "<br>");
    </script>
</body>
```

Operatorul conditional

Precedenta operatorilor

Operator	Nume operator
( ) [ ] .	de apelare, pt. structuri de date
! ++ --	de negare, incrementare
* / %	de inmultire, impartire
+ -	de adunare, scadere
< <= > >=	de comparatie
== !=	de egalitate
&&	SI logic
||	SAU logic
?:	conditionnal
= += -= *= /= %=	de atribuire

,	virgula

Cursul 13

Instrucţiuni condiţionale

Partea cea mai interesanta, dar si dificila in scrierea unui script este proiectarea lui asa incat sa ia decizii in timp ce este executat.

Cu ajutorul instructiunilor conditionate putem face programele sa testeze diferite conditii dupa are sa decida modul de executare a datelor.

In JavaScript avem urmatoarele instructiuni conditionate:

I. if (executa comenzile dorite cand o conditie este adevarata)
II. if … else (executa comenzile dorite cand o conditie este adevarata, altfel comenzi cand este falsa)
III. switch (selecteaza care comanda va fi executata)

1. Instructiunea “if”

Se poate spune ca aceasta este una din cele mai des folosite.

Forma generala este:

```
if (conditie) {
    codul care va fi executat dacă este adevărată condiţia
}
```

conditie=orice expresie logica

Daca rezultatul expresiei este TRUE se executa codul dintre {}, in caz contrar returneaza FALSE si se trece peste acest cod.

Exemplu:

```
<script type="text/javascript">
 // daca ora > 11,
 // va scrie în fereastra Buna ziua!
 var d = new Date()
 var time = d.getHours()
 if (time>11) {
        document.write("<b>Buna ziua!</b>")
 }
</script>
```

Se foloseste obiectul date care determina data curenta.

Am definit variabila d a carei valoare este data curenta, apoi variabila time care preia numai ora din variabila d.

Conditia verifica daca ora este mai mare decat 11 si daca este A executa comanda dintre {} (afiseaza mesajul “Buna ziua!”).

2. Instructiunea “if … else”

Folosirea acestei instructiuni duce la stabilirea comenzii care sa fie executata cand conditia instructiunii “if” este F.

Forma generala:

```
if (condiţie) {
   codul care va fi executat dacă este adevărată condiţia }
else {
   codul care va fi executat dacă condiţia este falsă }
```

conditie=orice expresie logica

Daca rezultatul conditiei este A se executa codul din primele {} care apartin de if, in caz contrar sunt executate comenziile din a 2-a grupa de {} (dupa else).

Exemplu:

```
<script type="text/javascript">
 // dacă ora > 11, va scrie în fereastra Bună ziua! Alfel afişează"Este ora ..."
 var d = new Date()
 var time = d.getHours()
 if (time>11) {
       document.write("<b>Bună ziua!</b>")
 }
 else {
        document.write("<b>Este ora " +time+ "</b>")
 }
</script>
```

- ➔ Afiseaza “Buna ziua!” daca ora este >11.
- ➔ Altfel afiseaza “Ora este: …”
- ➔ m definit variabila d a carei valoare este data curenta, apoi variabila time care preia numai ora din variabila d
- ➔ Conditia verifica daca ora este mai mare decat 11 si daca este A executa comanda dintre {} (afiseaza mesajul “Buna ziua!”).
- ➔ Daca ora este <11, scriptul va executa comanda din grupa de {} de la else. Observam folosirea operatorului de concatenare

3. Instrucţiunea „switch”

Este folosita pentru a compara o valoare cu altele dintr-o lista.

Forma generala:

```
switch (expresie) {
      case valoare1:
            cod executat dacă expresie = valoare1
            break
      case valoare2:
            cod executat dacă expresie = valoare2
            break
      case valoare3:
            cod executat dacă expresie = valoare3
```

```
            break
        default : //in folosesc in cazul in care am mai mult de 3 valori
                cod executat dacă expresie e diferit de valoare1, valoare2 sau valoare3
}
```

- ➔ Este evaluata expresia scrisa intre “()”
- ➔ Valoarea expresiei este comparata pe rand cu fiecare valoare determinata de case
- ➔ Daca se gaseste o identitate se executa codul asociat acelui case
- ➔ Se iese din instructiunea “switch”
- ➔ Daca parcurgand fiecare case nu se gaseste o egalitate, se executa codul de dupa default
- ➔ Folosirea lui “break”: se opreste parcurgerea corpului instructiunii atunci cand s-a gasit valoare=expresie si se iese din switch.

Exemplu:

```
<script type="text/javascript">
  var d = new Date()
  var ziua = d.getDay()
 switch (ziua) {
      case 5:
          document.write("Astăzi e vineri");
          break
      case 6:
          document.write("Astăzi e sâmbătă");
         break
      case 0:
          document.write("Astăzi e duminică");
          break
      default:
           document.write("Mai e până sâmbătă”); }
</script>
```

Exemplu:

```
 <script type="text/javascript">
   var nume = "Marius";
  switch (nume) {
          case "Cristi":
              document.write("Coleg");
              break
          case "Marius":
               document.write("Frate");
               break
          case "Maria":
               document.write("Sora");
               break
          default:
               document.write("Altcineva"); }
```

</script>

Instrucţiuni ciclice (repetitive)

Instructiunile repetitive se folosesc atunci cand se doreste efectuarea unei comenzi de mai multe ori.

In JavaScript putem folosi urmatoarele instructiuni:

I. for (la fel ca la celelalte limbaje, executa codul de un numar precizat de ori)
II. for … in (executa un set de instructiuni pentru fiecare proprietate dintr-un obiect)
III. while (repeta codul atata timp cat o anumita conditie este A)
IV. do … while (executa o data codul, apoi il repeta atata timp cat o anumita conditie este A)

1. Instrucţiunea „for”

Sintaxa:

```
for (începere_nr; condiţie_nr; ciclu) {
    cod care va fi executat }
```

- începere_nr - este folosit pentru a da o valoare initiala numarului de repetari, de multe ori, prin aceasta se declara o variabila care poate fi folosita ca un contor al ciclului.
- condiţie_nr - verifica daca numarul de cicluri se incadreaza intr-o anumita valoare si daca rezultatul este TRUE se executa inca o data codul dintre {}
- ciclu -incrementeaza sau decrementeaza valoarea la care a ajuns contorul ciclului, apoi aceasta valoare este verificata din nou de conditie_n pana cand este falsa.

Aceste 3 expresii dintre () sunt optionale, dar daca este omisa una dintre ele caracterul “;” trebuie sa ramana pentru ca fiecare expresie sa ramana la locul ei.

In interiorul instructiunii for, ca si la if, pot fi introduse si alte instructiuni for sau alte instructiuni conditionate. Acest lucru se numeste imbricarea instructiunilor.

Exemplu:

```
<script type="text/javascript">
    for (x=1; x<5; x++) {
        document.write("<br /> x este "+x);
    }
</script>
```

- in acest exemplu se atribuie x=1, se verifica x<5 A=>se executa corpul de instructiuni dintre {} (afiseaza: x este 1)
- se incrementeaza valoarea lui x cu o unitate=>x=2, se verifica x<5 A=> executa iar corpul de instructiuni
- se incrementeaza x pana cand x ajunge sa aiba valoarea 5 care la verificare conditiei returneaza FALSE, moment in care se termina executia instructiunii “for”

2. Instrucţiunea „for ... in”

Pentru utilizarea acestei instructiuni trebuie sa avem cunostiinte despre obiectele JavaScript, cu for … in se executa cate un set de instructiuni pentru fiecare proprietate dintr-un obiect.

Acest ciclu se poate executa cu orice obiect JavaScript indiferent daca are sau nu proprietati. Pentru fiecare proprietate se executa cate o iteratie daca obiectul nu are nicio proprietate, nu se desfasoara niciun ciclu.

Sintaxa:

```
for (nume_proprietate în obiect)
{
    instrucţiuni
}
```

nume_proprietate=un literal de tip sir generat de JavaScript pentru fiecare repetare a executiei instructiunii, lui nume_proprietate i se atribuie in urmatorul nume de proprietate continutul in obiect pana cand sunt folosite toate.

3. Instrucţiunea „while”

Repeta un corp atata timp cat conditia este adevarata, comanda actioneaza similar cu instructiunea “for”, dar nu include functiile de initializare si incrementare a variabilelor.

Sintaxa:

```
while (condiţie) {
    codul care va fi executat
}
```

Exemplu:

```
<script type="text/javascript">
  var x = 1;
  while (x<5) {
        document.write("<br /> x este "+x);
        x++;
  }
</script>
```

- afiseaza la fel ca la cel de la “for”
- am declarat variabila x dandu-i valoarea 1
- instructiunea “while” verifica conditia x<5 care este A si permite executarea corpului functiei dintre {} care va afisa x este 1
- incrementeaza valoarea lui x cu o unitate
- x=2
- se verifica conditia care este A si se executa iar codul din {}
- si asa mai departe
- pana cand rezultatul conditiei este FALSE, moment in care se termina rularea instructiunii “while”

4. Instrucţiunea „do ... while”

Sintaxa:

```
do {
```

codul care va fi executat
}
while (condiţie)

Asemanatoare in mare parte cu instructiunea “while”, instructiunea “do … while”, intai executa codul din corpul instructiunii, dupa care verifica conditia, apoi il repeta pana cand conditia returneaza FALSE, astfel corpul functiei este executat cel putin o data chiar daca conditia este FALSE.

Exemplu:

```
<script type="text/javascript">
    var x = 8;
    do {
        document.write("<br /> x este "+x);
        x++;
        }
     while (x<5)
</script>
```

- aceasta functie din exemplu afiseaza x este 8, chiar daca am conditia FALSE (x<5)
- codul dintre {} este totusi executat o singura data

Instructiuni de implementare

Pe langa instructiunile “for” si “while”, avem alte instructiuni care pot fi executate cu ele:

1. Instrucţiunile „break” şi „continue”

I. break (intrerupe definitiv executia unui ciclu)
II. continue (sare peste instructiunile care au mai ramas in ciclu respectiv)

Exemplu:

```
<script type="text/javascript">
    for (x=1; x<8; x++) {
        if (x==3 || x==5) {
        continue;
        }
        document.write("<br /> X este "+x);
        }
</script>
```

2. Instrucţiunea „label” (permite iesirea dintr-un ciclu cu instructiuni ciclice imbricate la o locatie specificata a scriptului)

Exemplu:

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

```
      for (x=1; x<=5; x++) {
          for (y=3; y<8; y++) {
              document.write("X este "+x+" - Y este "+y+" --" );
              if (x==4) {
                  break loopX;
              }
            }
            document.write("<br />")
       }
</script>
```

3. Instrucţiunea „with” (se foloseste pentru a fi evitata specificarea repetata la referirea unui obiect cand ii accesam metodele sau proprietatile)

Sintaxa:

```
with (obiect) {
    instrucţiuni
}
```

Exemplu:

```
<script type="text/javascript">
   with(document) {
          write("Salut");
          write("Acum nu mai este necesara folosirea obiectului ca prefix al functiei");
    }
</script>
```

Ferestre „Alert”, „Prompt” şi „Confirm”

1. Fereastra „Alert”

Sintaxa:

```
window.alert("mesaj")
```

Exemplu:

```
<script type="text/javascript">
       window.alert("Bine ai venit");
</script>
```

2. Fereastra „Prompt”

Sintaxa:

```
 window.prompt("mesaj", "default")
```

Exemplu:

```
<script type="text/javascript">
        window.prompt("Scrieti numele", "Nume");
</script>
```

Exemplu:

```
<script type="text/javascript">
    var nume = window.prompt("Scrieti numele", "Nume");
    alert("Salut "+nume+"\n Bine ai venit.");
```

```
</script>
```

3. Fereastra „Confirm”

Sintaxa:

```
window.confirm(„întrebare")
```

Exemplu:

```
<script type="text/javascript">
       intrebare = window.confirm("Rezultatul lui 0+0 este 0?");
       if (intrebare) alert("Corect");
       else alert("Incorect");
</script>
```

I. Crearea (definirea) funcţiilor:

Sintaxe:

```
function nume_functie(argument1, argument, ...) {
       codul care va fi executat
}

function nume_functie() {
      codul care va fi executat
}
```

II. Instrucţiunea return

Exemplu:

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

III. Apelarea funcţiilor

Sintaxa:

```
nume_functie(argument1, argument, ...);

nume_functie();
```

IV. Exemple de scripturi cu funcţii

Exemplu 1:

```
<html>
  <head>
       <script type="text/javascript">
            function exemplu1() {
                  return document.write("Bine ati venit!")
             }
       </script>
   </head>
<body>
       <script type="text/javascript">
             exemplu1()
```

```
    </script>
</body>
</html>
```

Exemplu 2:

```
<html>
<head>
    <script type="text/javascript">
        function exemplu2(text) {
            alert(text);
        }
    </script>
</head>
<body>
    <form>
        <input type="button" onclick="exemplu2('Buna dimineata!')" value="dimineata" />
        <input type="button" onclick="exemplu2('Buna ziua!')" value="ziua"/>
    </form>
</body>
</html>
```

Exemplu 3:

```
<html>
<head>
    <script type="text/javascript">
        function exemplu3(x,y) {
            var z = 0;
            z = x+y;
            return alert("Suma lui "+x+" si "+y+" este: "+z)
        }
    </script>
</head>
<body>
    <form>
        <input type="button" onclick="exemplu3(7, 8)" value="Suma" />
    </form>
</body>
</html>
```

1. Modificarea numărului argumentelor

Exemplu:

```
<html>
<head>
      <script type="text/javascript">
          function mesaj(utilizator) {
              if (utilizator!=null) {
                    document.writeln("Salut "+utilizator);
              }
              else {
                  document.writeln("Bine ati venit pe site!");
             }
          }
       </script>
</head>
<body>
   <script type="text/javascript">
          document.writeln("Prima apelare a functiei mesaj() <br/>");
          mesaj("Media");
          document.writeln("<br />A doua apelare a functiei mesaj() <br />");
          mesaj();
   </script>
</body>
 </html>
```

Exemplu:

```
nrArg = mesaj.arguments.length
<html>
<head>
    <script type="text/javascript">
       function mesaj(utilizator) {
          if (utilizator!=null) {
               document.writeln("Salut "+utilizator); }
          else {
               document.writeln("Bine ati venit pe site"); }
           numarArgumente=mesaj.arguments.length;
           if(numarArgumente>1) {
                for (var i=1; i<numarArgumente; i++) {
                        document.writeln(mesaj.arguments[i]); } } }
     </script>
</head>
<body>
      <script type="text/javascript">
           var utilizator="Marius", extraMesaj="Bine ai revenit";
           var utilizator2=null;
           var extraMesaj1="Vrei sa devii membru ?";var extraMesaj2="Te poti inscrie online";
```

```
            mesaj(utilizator,extraMesaj);
            document.writeln("<hr>");
            mesaj(utilizator2,extraMesaj1,extraMesaj2);
      </script>
</body>
 </html>
```

2. Funcţii recursive

Exemplu:

```
<html>
<head>
     <script type="text/javascript">
          document.writeln("Calculeaza factorialul de 7 prin functie recursiva")
          function factorial(n) {
               var rezultat;
               if (n>0) {
                  rezultat = n*factorial(n-1); }
               else if (n==0) {
                  rezultat = 1; }
                     else {
                        rezultat = null; }
          return(rezultat) }
      </script>
</head>
<body>
    <form>
          <input type = "button" value = "Factorial 7" onclick="alert(factorial(7))" >
    </form>
</body>
</html>
```