UNIVERSITATEA „POLITEHNICA” DIN BUCUREȘTI
FACULTATEA TRANSPORTURI

Departamentul Telecomenzi și Electronică în Transporturi

Tehnici de Programare în Internet

Laborator

Cuprins

Capitolul 1. Programul CoffeeCup

CoffeeCup este o suită ce cuprinde aplicații software pentru crearea, proiectarea și editarea paginilor de Internet precum și o serie de unelte pentru servicii online necesare dezvoltatorilor. În cadrul acestei lucrări se va utiliza programul de editare a paginilor de Internet în format HTML (*Hyper Text Markup Language*). Varianta gratuită a acestuia se poate descărca de pe pagina www.coffeecup.com.

După instalare se deschide programul și se creează o pagină nouă prin selectarea opțiunii *New Blank Page* în fereastra *Welcome* (Figura 1), sau din meniul *File -> New From Quick Start…* .

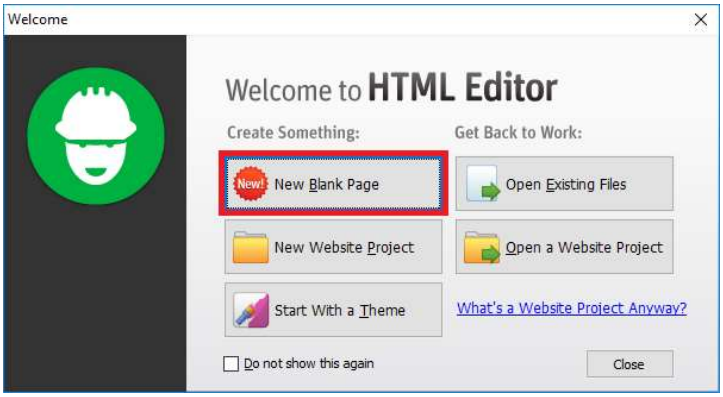

Figura 1. Crearea unei pagini noi

În fereastra *Quick Start* (Figura 2) se dă un nume paginii, în câmpul *Page Title*, și se selectează tipul de document, în câmpul *DOCTYPE*, ca fiind de tipul *HTML 4.01 Strict*. Restul câmpurilor sunt opționale.

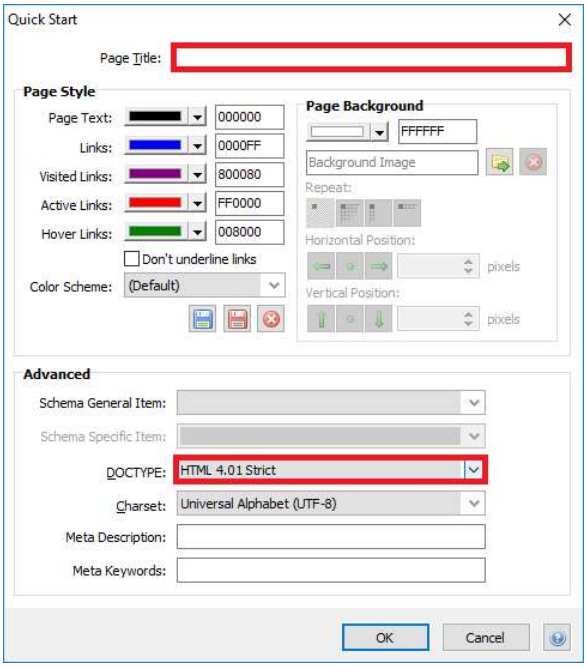

Figura 2. Stabilirea caracteristicilor principale ale unei pagini noi

Ca rezultat al caracteristicilor selectate în Figura 2, secvența de cod pentru pagina nou creată va arăta astfel:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
    <meta name="generator" content="CoffeeCup HTML Editor
(www.coffeecup.com)">
    <meta name="created" content="dum., 08 oct. 2017 18:37:56 GMT">
    <meta name="description" content="">
    <meta name="keywords" content="">
    <title>Laborator 1</title>

    <style type="text/css">
    <!--
    body {
      color:#000000;
      background-color:#FFFFFF;
      background-image:url('Background Image');
      background-repeat:no-repeat;
    }
    a  { color:#0000FF; }
    a:visited { color:#800080; }
    a:hover { color:#008000; }
    a:active { color:#FF0000; }
    -->
    </style>
  </head>
  <body>

  </body>
</html>
```

Structura paginii nou create este următoarea:

- Declarația `DOCTYPE`. Spune browser-ului ce versiune de HTML este folosită în document și trebuie să apară la începutul acestuia.
- Marcajul `<html>` ce definește un fișier în format HTML.
- Marcajul `<head>` ce definește antetul documentului.
- Marcajul `<body>` ce definește conținutul paginii.

Semnificația principalelor linii de cod din secvența de mai sus este următoarea:

- Marcajul de tip meta `http-equiv` specifică setul de caractere utilizat în documentul HTML, necesară pentru afișarea corectă a caracterelor speciale sau a diacriticilor.

- Marcajele de tip meta `description` și `keywords` definesc descrierea paginii (ce apare de exemplu în rezultatele motoarelor de căutare) și cuvintele cheie (folosite de asemenea de motoarele de căutare).
- Marcajul `<title>` definește titlul paginii care apare în browser.
- Marcajul `<style>` definește caracteristicile de stil ale paginii (fundal, culoare text).

În capitolele următoare se va lucra cu conținutul paginii.

Afișarea conținutului paginii construite se poate face fie prin pre-vizualizare direct în program cu ajutorul butonului  *Toggle External Preview* (sau apăsând tasta F11), fie prin accesarea meniului *View* și a opțiunii *Split-Screen Preview* (sau apăsând tasta F12), fie prin vizualizare în browser cu ajutorul butonului  *Preview -> Local preview-> Test With Default Browser* (sau apăsând tastele CTRL + F9).

Capitolul 2. Marcaje și elemente HTML

Limbajul HTML (*Hyper Text Markup Language*) este utilizat pentru crearea de pagini Web. Elementele utilizate în construcția unei pagini Web sunt componente cu rol predefinit. Acestea sunt definite prin utilizarea unui marcaj de deschidere `<cod_marcaj>`, introducerea unui conținut și utilizarea unui marcaj de închidere `</cod_marcaj>` (ce nu e obligatoriu în unele cazuri). Programele de navigare folosesc marcajele pentru a reda conținutul și forma paginii Web.

Limbajul HTML este *case-insensitive*, adică nu face distincția între literele mari și cele mici, în cazul sintaxei instrucțiunilor.

Caracterele precum *spațiu* și *tab* care apar între marcaje sunt ignorate la afișarea paginii de către un browser.

Marcajul `<!-- conținut -->` indică faptul că, conținutul este un comentariu și va fi ignorat de către browser.

2.1 Afișarea titlurilor

Titlurile se realizează cu ajutorul elementelor de tip *heading*, fiind disponibile 6 tipuri (rezultatul afișării fiind cel din Figura 3).

```
<h1> Text ce trebuie formatat ca titlu </h1>
<h2> Text ce trebuie formatat ca titlu </h2>
<h3> Text ce trebuie formatat ca titlu </h3>
<h4> Text ce trebuie formatat ca titlu </h4>
<h5> Text ce trebuie formatat ca titlu </h5>
<h6> Text ce trebuie formatat ca titlu </h6>
```

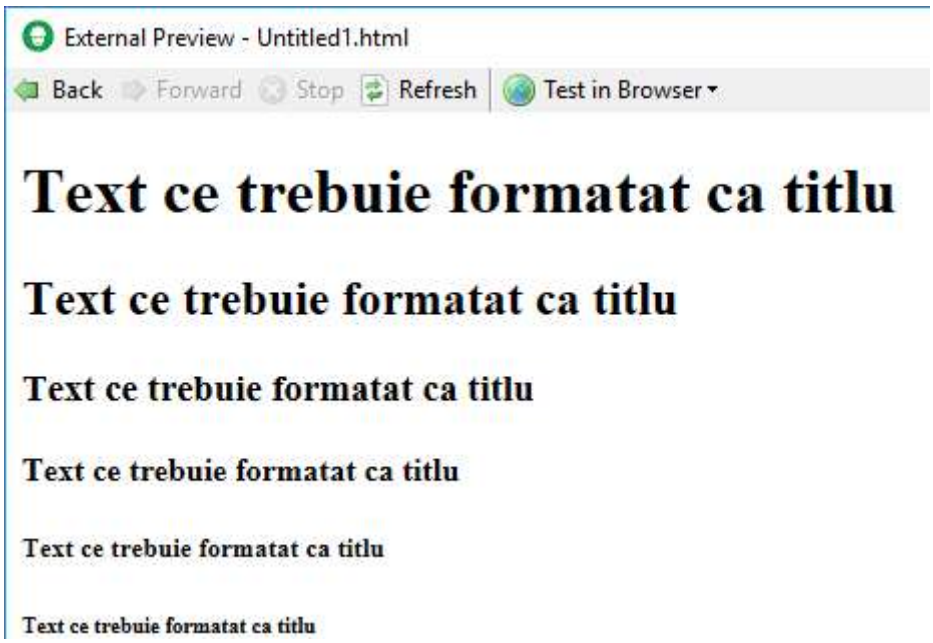

Figura 3. Tipuri de titluri

Să se afișeze în pagina HTML următoarele:

Capitolul 2. Marcaje pentru text (de tip Heading 1)
2.1 Afisarea titlurilor (de tip Heading 2)
2.2 Afisarea paragrafelor (de tip Heading 2)

2.2 Afișarea paragrafelor

Un paragraf obișnuit al unei pagini se identifică prin utilizarea marcajului `<p>`.

```
<p> Paragraf 1 </p>
<p> Paragraf 2 </p>
```

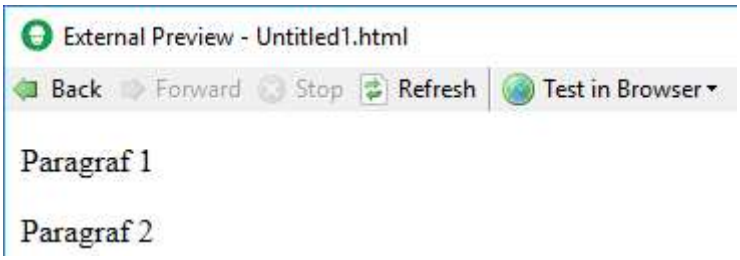

Figura 4. Afișarea paragrafelor

Două astfel de paragrafe vor fi însă afișate cu un rând gol între ele.

Pentru afișarea a două paragrafe unul sub altul se folosește un singur set de marcaje `<p> </p>` iar în interiorul acestora, între cele două paragrafe, se folosește marcajul `<br>`, ce nu are nevoie de un marcaj de închidere.

```
<p>
Paragraf 1
<br>
Paragraf 2
</p>
```

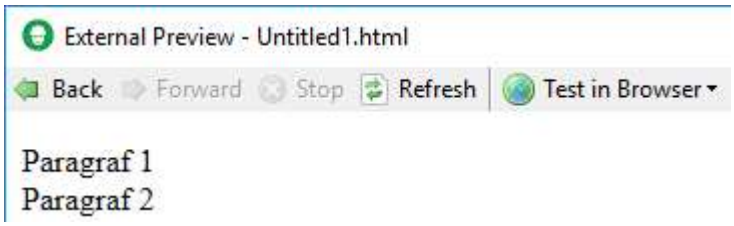

Figura 5. Afișarea paragrafelor fără spațiu între ele

Prin folosirea marcajului `<p>` paragraful va începe pe o linie nouă. Dacă se dorește ca un paragraf să fie afișat în continuarea sau înaintea unui alt element HTML se va folosi setul de marcaje `<span> </span>`.

```
<span> Paragraf 1 </span>
<span> Paragraf 2 </span>
```

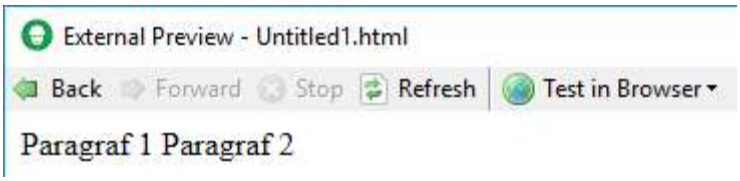

Figura 6. Afișarea paragrafelor unul după altul

Inserarea unei linii separatoare orizontale între două paragrafe se face cu marcajul `<hr>`, ce nu are nevoie de un marcaj de închidere.

```
<p> Paragraf 1 </p>
<hr>
<p> Paragraf 2 </p>
```

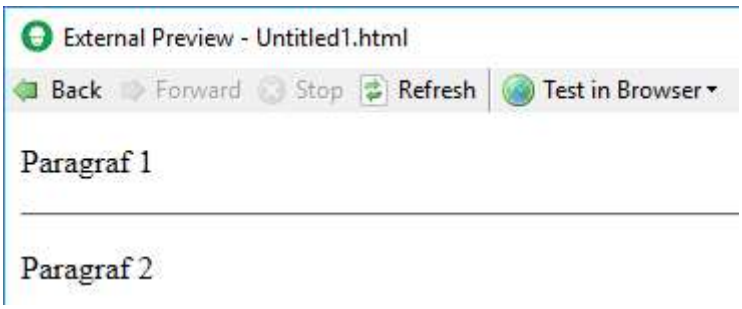

Figura 7. Afișarea unei linii separatoare între două paragrafe

Trecerea unui text pe linia următoare folosind tasta *Enter* (în cadrul unui set de marcaje `<p> </p>`) pentru a începe un paragraf nou va fi ignorată de browser. Un paragraf nou se marchează întotdeauna cu un nou set de marcaje `<p> </p>`.

Acest lucru nu se întâmplă la afișarea unui text exact așa cum este scris în secvența de cod cu ajutorul marcajului `<pre>`.

Alinierea unui text se face prin specificarea tipului de aliniere (stânga, dreapta, centrat sau stânga-dreapta) ca valoare a unui atribut, folosind sintaxa:

```
align="left sau right sau center sau justify"
```

Exemplu de utilizare a unui atribut pentru aliniere:

```
<p align="right">Text de afisat</p>
```

Să se afișeze în pagina HTML următoarele:

Capitolul 2. Marcaje pentru text (să se alinieze centrat)
2.1 Afisarea titlurilor
Titlurile se realizeaza cu ajutorul marcajelor de tip heading. (de tip paragraf)
(Inserarea unei linii separatoare orizontale)
2.2 Afisarea paragrafelor
Un paragraf obisnuit al unei pagini se identifica prin utilizarea marcajului p. (de tip paragraf)
Trecerea unui text pe linia urmatoare folosind tasta Enter intr-un paragraf va fi ignorata de browser. (de tip paragraf)
(Afișarea paragrafului următor imediat sub cel anterior, fără spațiu între ele)
Un paragraf nou se marcheaza intotdeauna cu un nou set de marcaje p. (de tip paragraf)

2.3 Formatarea textului

Pentru ca un text să fie afișat îngroșat (*bold*) se folosește setul de marcaje `<b> </b>`.

Pentru ca un text să fie afișat înclinat (*italic*) se folosește setul de marcaje `<i> </i>`.

Pentru ca un text să fie afișat subliniat (*underline*) se folosește setul de marcaje `<u> </u>`.

Alte tipuri de marcaje sunt: `<strong>`, `<big>`, `<em>`, `<code>`, `<sub>`, `<sup>`, `<del>`, etc.

Modificarea culorii, dimensiunii și tipului de font pentru un paragraf sau text se face folosind setul de marcaje `<font> </font>`. Acesta are nevoie și de valori pentru atribute conform sintaxei:

```
<font color="valoare" size="valoare" face="valoare">
```

Codarea culorilor se face începând cu simbolul # urmat de 6 cifre sau litere (ce reprezintă numere în formatul hexazecimal). Fiecare din perechile de două simboluri reprezintă intensitatea culorilor roșu, verde și albastru. Exemple: #000000 reprezintă culoarea negru, #FFFFFF reprezintă culoarea alb (a se accesa pagina https://html-color-codes.info/).

Dimensiunea fontului (atributul `size`) se stabilește printr-o valoare de la 1 la 7. Dimensiunea implicită utilizată de browser este 3.

Tipul fontului (atributul `face`) trebuie să cuprindă denumirea acestuia. Exemple: Georgia, serif / Times New Roman / Arial, Helvetica, sans-serif / Comic Sans MS etc.

Să se afișeze în pagina HTML următoarele:

Capitolul 2. Marcaje pentru text (acest rând se va scrie înclinat)
2.1 Afisarea titlurilor (acest rând se va scrie subliniat)
Titlurile se realizeaza cu ajutorul marcajelor de tip heading.
2.2 Afisarea paragrafelor (acest rând se va scrie subliniat)
Un paragraf obisnuit al unei pagini se identifica prin utilizarea marcajului p.
Trecerea unui text pe linia urmatoare folosind tasta Enter (cuvântul Enter se va scrie înclinat) intr-un paragraf va fi ignorata de browser.
Un paragraf nou se marcheaza intotdeauna cu un nou set de marcaje p. (acest rând se va scrie cu un font, culoare și dimensiune la alegere)

2.4 Inserarea de legături

Crearea de legături către alte pagini de Internet (*link*) se face cu setul de marcaje `<a> </a>` (prescurtare de la cuvântul *anchor*). Sintaxa este următoarea:

```
<a href="adresă site" target="destinație fereastră">text pentru link</a>
```

Atributul *target* definește unde se va deschide adresa specificată: `_blank` (în fereastră nouă), `_self` (în aceeași fereastră, valoare implicită), `_parent` (în fereastra părinte), `_top` (în fereastra principală) sau `framename` (în cadrul specificat).

Să se afișeze în pagina HTML următoarele:

Capitolul 2. Marcaje pentru text
2.1 Afisarea titlurilor
Titlurile se realizeaza cu ajutorul marcajelor de tip heading.
2.2 Afisarea paragrafelor
Un paragraf obisnuit al unei pagini se identifica prin utilizarea marcajului p.

Trecerea unui text pe linia urmatoare folosind tasta Enter intr-un paragraf va fi ignorata de browser.
Un paragraf nou se marcheaza intotdeauna cu un nou set de marcaje p.
Pentru mai multe detalii intrati aici. (cuvântul aici trebuie să fie o legătură către pagina http://tet.pub.ro cu destinația `_blank`)

2.5 Inserarea de imagini

Pentru inserarea unei imagini se folosește marcajul `img` cu atributul `src`, având următoarea sintaxă:

```
<img src="URL sau adresă">
```

Adresa imaginii poate fi relativă (se include directorul unde se află și pagina) sau absolută (o adresă din Internet).

Exemplu de utilizare a unui atribut pentru sursa imaginii:

```
<img src="https://upb.ro/image.png">
<img src="F:\images\image.png">
```

Alte atribute ce pot fi utilizate:

`alt="descriere"` – Oferă informații despre imagine ce pot fi afișate atunci când imaginea nu apare corect sau de către motoarele de căutare.

`width="dimensiune în pixeli" height="dimensiune în pixeli"` – Stabilește dimensiunile imaginii. Pentru a nu o deforma se utilizează numai unul din atribute.

`<a href="URL sau adresă"><img src="URL sau adresă"></a>` - Imaginea devine o legătură către o altă pagină, putându-se da clic oriunde pe suprafața ei.

Alinierea unei imagini se poate face încadrând-o într-un paragraf și folosind opțiunile de aliniere ale acestuia.

Să se afișeze în pagina HTML următoarele:

Se va descărca imaginea cu sigla UPB din pagina tet.pub.ro și se va salva în același director cu pagina HTML aflată în lucru.
Se va introduce imaginea în partea de sus a paginii, având descrierea Universitatea Politehnica din București, una din dimensiuni de 100 pixeli și legătură către pagina http://www.upb.ro.
Se va alinia centrat imaginea în cadrul paginii.

2.6 Inserarea de liste

Se pot crea două tipuri de liste: neordonate, marcate cu buline, sau ordonate, marcate cu numere. Pentru fiecare tip de listă se folosește un set de marcaje: `<ul> </ul>`, respectiv `<ol> </ol>`. Fiecare element al listei este cuprins între marcajele `<li> </li>`.

Exemple de sintaxă:

```
<ul>
    <li>Element 1</li>
    <li>Element 2</li>
    <li>Element 3</li>
</ul>
```

```
<ol>
    <li>Element 1</li>
    <li>Element 2</li>
    <li>Element 3</li>
</ol>
```

Figura 8. Afișarea listelor

Să se afișeze în pagina HTML următoarele:

(Siglă UPB)
Capitolul 2. Marcaje pentru text
2.1 Afisarea titlurilor
Titlurile se realizeaza cu ajutorul marcajelor de tip heading.
Tipuri de heading: (de tip paragraf)
(se va crea o listă la alegere cu următoarele elemente)
Heading 1
Heading 2
Heading 3
Heading 4
Heading 5
Heading 6
2.2 Afisarea paragrafelor
Un paragraf obisnuit al unei pagini se identifica prin utilizarea marcajului p.
Trecerea unui text pe linia urmatoare folosind tasta Enter intr-un paragraf va fi ignorata de browser.
Un paragraf nou se marcheaza intotdeauna cu un nou set de marcaje p.
Pentru mai multe detalii intrati aici.

2.7 Inserarea de tabele

Definirea unui tabel presupune utilizarea mai multor tipuri de marcaje.

Definirea tabelului se face cu setul de marcaje `<table> </table>`. Atributul `border` specifică dacă se afișează chenarul (valoarea `"1"`) sau nu (valoarea `"0"`).

Fiecare rând se definește folosind `<tr> </tr>`.

Celulele, dacă sunt de tip antet (*header*), se marchează cu setul `<th> </th>`.

Celulele obișnuite ce conțin date se definesc folosind setul de marcaje `<td> </td>`.

Un exemplu de sintaxă este următorul:

```
<table border="1">
    <tr>
        <th> Celula 1 </th>
        <th> Celula 2 </th>
    </tr>
    <tr>
        <td> Celula 3 </td>
        <td> Celula 4 </td>
    </tr>
    <tr>
        <td> Celula 5 </td>
        <td> Celula 6 </td>
    </tr>
</table>
```

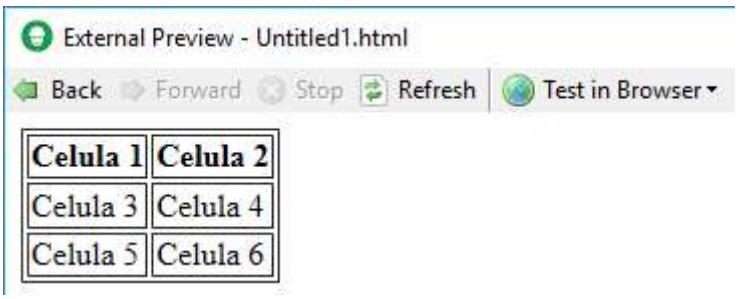

Figura 9. Afișarea unui tabel

Să se afișeze în pagina HTML următoarele:

(Siglă UPB)
Capitolul 2. Marcaje pentru text
2.1 Afisarea titlurilor
Titlurile se realizeaza cu ajutorul marcajelor de tip heading.
Tipuri de heading: (de tip paragraf)
(se va crea o listă la alegere cu următoarele elemente)

Heading 1
Heading 2
Heading 3
Heading 4
Heading 5
Heading 6
Să se realizeze un tabel care să conțină 3 coloane și 2 rânduri, primul rând fiind de tip antet, iar celulele să conțină denumirile celor 6 tipuri de titlu (heading).
2.2 Afisarea paragrafelor
Un paragraf obisnuit al unei pagini se identifica prin utilizarea marcajului p.
Trecerea unui text pe linia urmatoare folosind tasta Enter intr-un paragraf va fi ignorata de browser.
Un paragraf nou se marcheaza intotdeauna cu un nou set de marcaje p.

2.8 Inserarea de formulare

Utilizarea de formulare permite comunicarea prin intermediul paginilor web prin completarea de informații în câmpuri pentru text, realizarea de selecții folosind liste derulante, apăsarea de butoane, etc. Crearea listelor nu este suficientă, pentru ca acestea să realizeze o acțiune este necesar un limbaj de programare precum *JavaScript* sau *PHP*.

Toate elementele unui formular trebuie incluse între marcajele `<form>` și `</form>`. Marcajul `<form>` trebuie să conțină obligatoriu și atributul `action`. Acesta specifică ce se va întâmpla în momentul trimiterii formularului, adică este necesară utilizarea *JavaScript* sau *PHP*. Se va utiliza în acest moment un atribut nul: `<form action = "">`.

O grupare vizuală într-un chenar a elementelor ce formează un formular se poate realiza cu setul de marcaje `<fieldset> </fieldset>`. Împreună cu acestea se poate folosi și setul de marcaje `<legend> </legend>` ce permite afișarea împreună cu chenarul a unui text descriptiv pentru respectivul formular.

```
<form action = "">
      <fieldset>
            <legend>Eticheta</legend>
            Conținutul formularului
      </fieldset>
</form>
```

Să se afișeze în pagina HTML, în continuarea paragrafelor, următorul formular:

Creați titlul: Exemplu de formular (de tip Heading 1)
Creați un formular încadrat într-un chenar, având eticheta Date personale.

2.8.1 *Realizarea unui câmp pentru text*

Sintaxa cea mai simplă pentru realizarea unui câmp în care se poate completa un text este următoarea:

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

Adițional se mai poate adăuga o etichetă descriptivă înaintea câmpului de text folosind setul de marcaje `<label> </label>`, o valoare ce va apărea completată implicit folosind atributul `value`, sau se poate dimensiona câmpul cu atributele `size` (numărul de caractere afișate) și `maxlength` (limitarea numărului de caractere permise).

```
<label>Eticheta</label>
<input type = "text" name= "nume" value = "text implicit" id = "identificator">
```

Figura 10. Afișarea unui câmp pentru text

Identificatorul definit prin atributul `id` va fi utilizat în limbajul *JavaScript* pentru a extrage date din elementul respectiv, iar numele definit prin atributul `name` în limbajul *PHP*.

Atunci când un formular are mai multe elemente, acestea vor fi afișate unul după altul. Pentru ca acestea să se poziționeze unul sub altul trebuie folosit elementul `<br>`.

Să se afișeze în pagina HTML următorul conținut al formularului:

Exemplu de formular
Formular încadrat într-un chenar, având eticheta Date personale
Să se insereze în formular următorul conținut:
Să se creeze un câmp pentru text având eticheta Nume, Prenume.
Să se creeze un câmp pentru text (sub cel anterior) având eticheta E-mail.

2.8.2 *Realizarea unui câmp pentru parolă*

Sintaxa pentru realizarea unui câmp în care se poate completa o parolă este următoarea:

```
<label>Eticheta</label>
<input type = "password" name= "nume" id = "identificator">
```

Caracterele parolei nu vor fi afișate, în locul lor apărând caractere de tip asterisc.

Figura 11. Afișarea unui câmp pentru parolă

Să se afișeze în pagina HTML următorul conținut al formularului:

Exemplu de formular
Formular încadrat într-un chenar, având eticheta Date personale
Câmp pentru text având eticheta Nume, Prenume.
Câmp pentru text având eticheta E-mail.
Să se creeze un câmp pentru parolă având eticheta Parola.

2.8.3 Realizarea unui câmp pentru linii multiple de text

Sintaxa include specificarea numărului de linii și coloane a câmpului și este următoarea:

```
<textarea name = "nume" id = "identificator" rows = "număr"  cols = "număr">
</textarea>
```

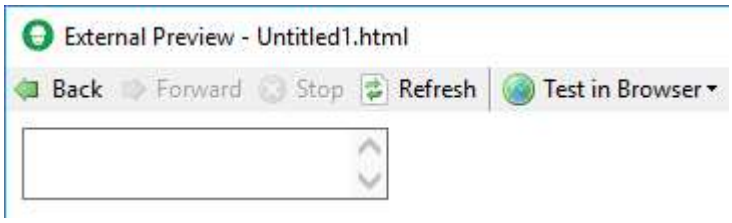

Figura 12. Afișarea unui câmp pentru linii multiple de text

Să se afișeze în pagina HTML următorul conținut al formularului:

Exemplu de formular
Formular încadrat într-un chenar, având eticheta Date personale
Câmp pentru text având eticheta Nume, Prenume.
Câmp pentru text având eticheta E-mail.
Câmp pentru parolă având eticheta Parola.
Să se creeze un câmp pentru linii multiple de text (sub cel anterior) având eticheta Alte detalii.

2.8.4 *Realizarea de liste derulante*

Permite utilizatorului alegerea unei opțiuni dintr-o listă predefinită. Lista este definită prin setul de marcaje `<select> </select >`, iar elementele listei prin `<option> </option>`.

Sintaxa pentru realizarea unei liste derulante este următoarea:

```
<select id = "identificator">
  <option value = "valoare">Text optiune</option>
  <option value = "valoare">Text optiune</option>
  <option value = "valoare">Text optiune</option>
</select>
```

Figura 13. Afișarea unei liste derulante

Să se creeze un nou formular încadrat într-un chenar, având eticheta Programare discipline.
Să se creeze în formular o listă derulantă având eticheta Alegere disciplina și opțiunile Limbaje de programare, Structuri de date si algoritmi, Tehnologii de programare in Internet.

2.8.5 *Realizarea de căsuțe pentru bifare*

Permite utilizatorului selectarea, sau nu, a unei opțiuni dintr-o listă predefinită. Toate opțiunile sunt afișate simultan și independente una față de alta.

Sintaxa pentru realizarea unei căsuțe pentru bifare este următoarea:

```
<input type = "checkbox" name= "nume" value = "valoare" id = "identificator">Text optiune
```

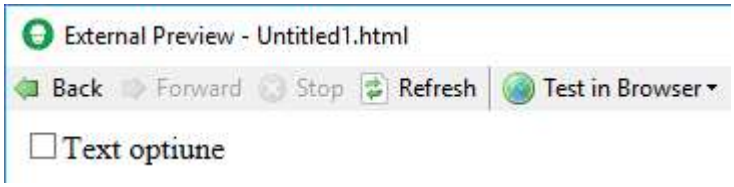

Figura 14. Afișarea unei căsuțe pentru bifare

Formular încadrat într-un chenar, având eticheta Programare discipline.
Listă derulantă având eticheta Alegere disciplina *și opțiunile* Limbaje de programare, Structuri de date si algoritmi, Tehnologii de programare in Internet.
Să se afișeze (sub lista derulantă) textul Alegere zile:
Să se creeze (unele sub altele):
- o căsuță de bifare cu textul Luni.
- o căsuță de bifare cu textul Miercuri.
- o căsuță de bifare cu textul Joi.

2.8.6 Realizarea de butoane radio

Permite utilizatorului selectarea unei singure opțiuni dintr-o listă predefinită. Toate opțiunile sunt afișate simultan.

Sintaxa pentru realizarea unui grup de butoane radio este următoarea:

```
<input type = "radio" name= "nume" value = "valoare" id = "identificator">Text optiune
```

Figura 15. Afișarea unui buton radio

Pentru ca mai multe butoane radio să se afle într-un grup toate trebuie să aibă atributul `name` identic. Dacă unul dintre ele se dorește să fie bifat implicit, se va adăuga acestuia atributul `checked`.

Formular încadrat într-un chenar, având eticheta Programare discipline.
Listă derulantă având eticheta Alegere disciplina *și opțiunile* Limbaje de programare, Structuri de date si algoritmi, Tehnologii de programare in Internet.
Alegere zile:
- *căsuță de bifare cu textul* Luni.
- *căsuță de bifare cu textul* Miercuri.
- *căsuță de bifare cu textul* Joi.

Să se afișeze (sub căsuțele de bifare) textul Alegere intervale orare:
Să se creeze un grup de butoane radio, astfel:
- un buton radio cu textul 10-12.
- un buton radio cu textul 12-14.
- un buton radio cu textul 14-16.

2.8.7 Realizarea de butoane

Butoanele permit declanșarea unei acțiuni. Există mai multe tipuri: butoane de intrare, de trimitere sau de resetare. Există chiar și un set de marcaje `<button> </button >` care oferă mai multă libertate în definirea unui buton.

Sintaxa pentru realizarea butoanelor este următoarea:

```
<input type = "button" value = "text buton">
<input type = "submit" value = "text buton Submit ">
<input type = "reset" value = "text buton Reset">
```

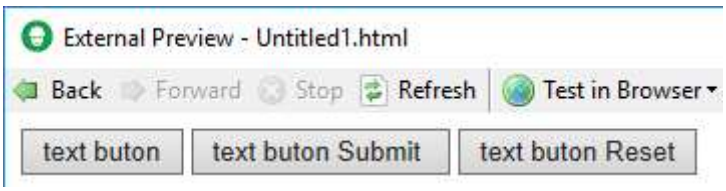

Figura 16. Afișarea de butoane

Formular încadrat într-un chenar, având eticheta Programare discipline.
Listă derulantă având eticheta Alegere disciplina *și opțiunile* Limbaje de programare, Structuri de date si algoritmi, Tehnologii de programare in Internet.
Alegere zile:
- *căsuță de bifare cu textul* Luni.
- *căsuță de bifare cu textul* Miercuri.
- *căsuță de bifare cu textul* Joi.

Alegere intervale orare:
- *buton radio cu textul* 10-12.
- *buton radio cu textul* 12-14.
- *buton radio cu textul* 14-16.

Să se creeze trei butoane, afișate unul după altul, astfel:
- un buton de tip intrare cu textul Anulare.
- un buton de tip trimitere cu textul Trimite.
- un buton de tip resetare cu textul Resetează.

Capitolul 3. Stilizarea paginilor cu CSS

3.1 Introducere

CSS (Cascading Style Sheets) este un limbaj care descrie stilul unui document HTML și cum sunt afișate elementele acestuia, totul într-un mod eficient și simplu care permite reutilizarea stilurilor în toate paginile unui site Web, folosind formatări stocate separat.

CSS permite definirea de stiluri pentru paginile Web, incluzând design-ul, modul de încadrare în pagină și varierea modului de afișare în funcție de dimensiunea ecranului.

CSS se poate adăuga într-un document HTML în 3 moduri:

- Inline (în linia de cod) - utilizând atributul „style" al elementelor HTML.

Intern - utilizând un marcaj `<style>` în secțiunea de antet (`<head>`).

- Extern - utilizând un fișier extern CSS.

Când unui element HTML îi sunt aplicate stiluri prin mai multe din modurile enumerate, alegerea stilului ce va fi aplicat se va face în "cascadă" în funcție de nivelul de prioritate, astfel:

- Modul Inline (are cea mai mare prioritate).
- Modurile intern sau extern.
- Modul implicit de afișare al programului de navigare.

Sintaxa pentru includerea fișierelor externe CSS este următoarea:

```
<head>
      <link rel="stylesheet" type="text/css" href="stil.css">
</head>
```

Un set de reguli CSS este compus dintr-un *selector*, care indică elementul HTML la care se aplică stilul, și un bloc ce conține descrierea *atributelor*, sub forma unei declarații de tipul `proprietate:valoare;` astfel:

```
Selector {proprietate1:valoare1; proprietate2:valoare2;...}
```

Blocul de descriere a atributelor conține declarațiile dorite, separate de marcajul „`;`" (inclusiv pentru ultima dintre ele).

Selectorii CSS sunt utilizați pentru găsirea elementelor HTML, pe baza numelui, ID-ului, clasei, sau atributului.

3.2 Selectori CSS

3.2.1 Selectorul pentru elemente

Se bazează pe numele elementului și se pot selecta și formata toate elementele de acel fel din pagina HTML.

De exemplu, sintaxa pentru formatarea paragrafelor este următoarea:

```
<html>
	<head>
		<style>
			p {
				text-align: center;
				color: red;
			}
		</style>
	</head>
<body>
	<p>Paragraf.</p>
</body>
```

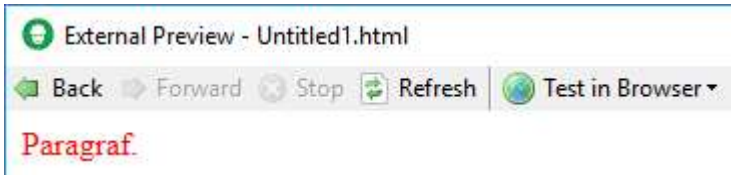

Figura 17. Formatarea unui paragraf folosind selectorul pentru elemente

3.2.2 Selectorul pentru ID

Se pot selecta numai acele elemente care au ID-ul identic pentru a se formata toate în același mod.

Sintaxa pentru definirea selectorului pentru ID este următoarea:

```
#paragraf_1 {color: red;}
```

ID-ul trebuie inclus apoi ca atribut în elementul HTML dorit:

```
<p id="paragraf_1">
```

Exemplu de formatare:

```
<html>
	<head>
		<style>
			#paragraf_1 {color: red;}
```

```
            </style>
      </head>
<body>
      <p id="paragraf_1">Paragraf cu font rosu.</p>
      <p>Paragraf care nu are modificari de stil.</p>
</body>
</html>
```

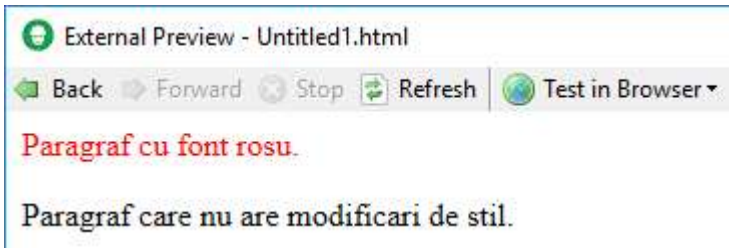

Figura 18. Formatarea unui paragraf folosind selectorul pentru ID

3.2.3 Selectorul pentru clase

Selectează elementele care fac parte dintr-o anumită clasă definită prin caracterul "." urmat de numele clasei.

Sintaxa pentru definirea selectorului pentru clase este următoarea:

```
.rosu {color: red;}
```

Clasa trebuie inclusă apoi ca atribut în elementul HTML dorit:

```
<p class="rosu">
```

Exemplu de formatare:

```
<html>
      <head>
            <style>
                  .rosu {color: red;}
            </style>
      </head>
      <body>
            <h1 class="rosu">Titlu afisat cu rosu.</h1>
            <p class="rosu">Paragraf afisat cu rosu.</p>
      </body>
</html>
```

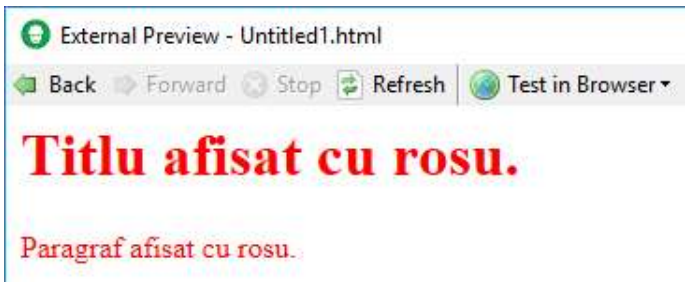

Figura 19. Formatarea elementelor folosind selectorul pentru clase

Se poate specifica și tipul de element care va fi afectat de formatare (element + "." + nume clasă):

```
<html>
	<head>
		<style>
			p.rosu {text-align: center; color: red;}
		</style>
	</head>
	<body>
		<h1 class="rosu">Titlu la care nu se aplica stilul.</h1>
		<p class="rosu">Paragraf la care se aplica stilul.</p>
	</body>
</html>
```

Figura 20. Formatarea elementelor folosind selectorul pentru clase cu specificarea elementului afectat

Elementele HTML pot face parte din mai multe clase:

```
<html>
	<head>
		<style>
			p.rosu {color: red;}
			p.marit {font-size: 300%;}
		</style>
	</head>
	<body>
		<h1 class="rosu">Titlu la care nu se aplica stilul.</h1>
		<p class="rosu">Paragraf cu font rosu.</p>
		<p class="rosu marit">Paragraf cu font rosu si marit.</p>
	</body>
</html>
```

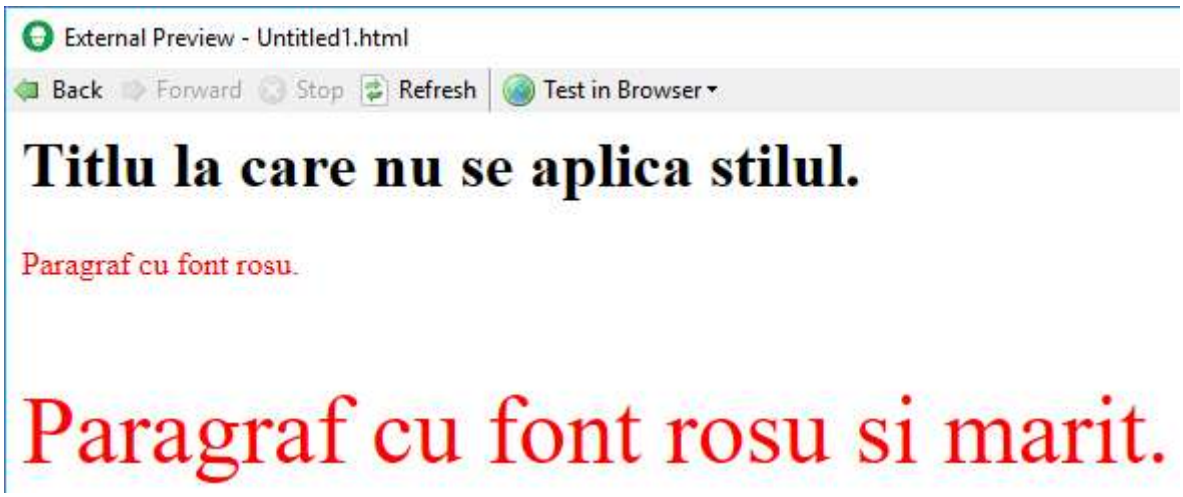

Figura 21. Formatarea elementelor prin utilizarea mai multor clase

3.2.4 Gruparea selectorilor

Pentru a minimiza dimensiunea ocupată de cod, atunci când sunt definiți, selectorii se pot grupa dacă au aceleași atribute.

De exemplu, în loc de:

```
h1 {
	text-align: center;
	color: red;
}
p {
	text-align: center;
	color: red;
}
```

se poate scrie:

```
h1, p {
	text-align: center;
	color: red;
}
```

3.2.5 Comentarii CSS

Cu ajutorul comentariilor se introduc explicații în cod. Sunt incluse între simbolurile „/*” și „*/” și pot cuprinde mai multe linii.

```
p {
	color: red;
	/* Comentariu pe un singur rând */
	text-align: center;
}
```

```
/* Comentariu
pe mai multe
rânduri */
```

Titlurile de tip Heading 1 să se formateze, folosind selectorul pentru elemente, astfel încât să fie afișate centrat și de culoare albastră.
Să se formateze primul paragraf din capitolele 2.1 și 2.2, folosind selectarea pe bază de Id (exemplu de Id: `paragraf_stil_1`), astfel încât textul să aibă culoarea roșie. Să se crească dimensiunea fontului la 120% folosind selectorul de clasă `.mare`.

Folosind aceeași clasă, să se crească dimensiunea fontului pentru antetul tabelului. În plus, să se definească o clasă `.verde` care să formateze numai elemente de tip th și care să schimbe culoarea textului în verde. Clasa `.verde` să se aplice și elementelor listei.
Celule de date ale tabelului să se formateze în același mod ca și titlurile de tip Heading 1, dar prin gruparea selectorilor.
Să se introducă un comentariu la alegere.

3.3 Atribute pentru culoare

Sunt utilizate pentru stabilirea culorii unui text, fundal pentru text sau chenar. Exemplele de mai jos utilizează modul de aplicare Inline.

```
<p style="color:lightblue;">Text de afișat</p>
<p style="background-color:pink;">Text de afișat</p>
<h1 style="border:3px solid red;">Titlu</h1>
```

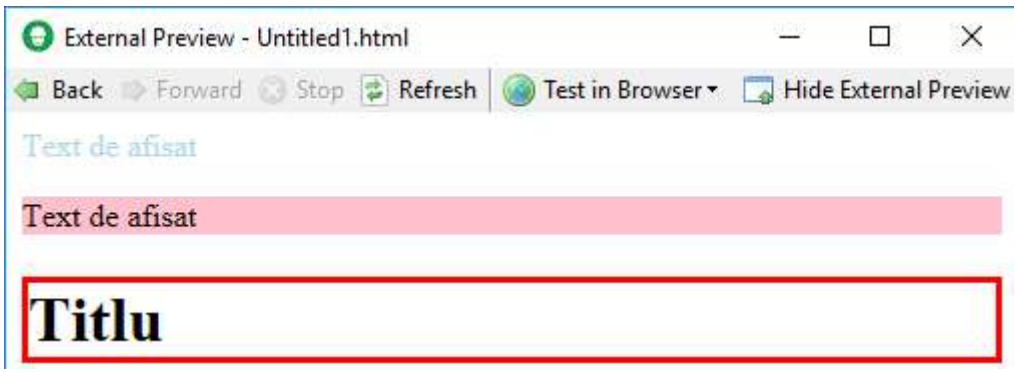

Figura 22. Utilizarea atributelor pentru culoare

Să se formateze (utilizând modul de aplicare Inline) primul titlu de tip Heading 1 astfel încât textul să aibă culoarea albă, pe fundal portocaliu și să fie încadrat de un chenar solid de culoare mov și grosime de 2 pixeli.

3.4 Atribute pentru fundal

Fundalul unei pagini HTML poate fi stabilit în două moduri, fie prin inserarea unei culori (utilizând proprietatea `background-color`), fie prin inserarea unei imagini (utilizând proprietatea `background-image`), conform următoarei sintaxe:

```
body {
    background-color: red;
}
```

sau

```
body {
    background-image: url("nume sau adresă imagine");
}
```

În mod implicit, atunci când se folosește ca fundal o imagine, aceasta va fi repetată pe orizontală și pe verticală, pentru a umple ecranul ce afișează pagina Web.

În cazul unor imagini, pentru un aspect plăcut al paginii este necesar imaginea de fundal să se repete doar pe o direcție, x sau y. Acest lucru se poate realiza utilizând proprietatea `background-repeat`, conform următoarei sintaxe:

```
body {
    background-image: url('nume sau adresă imagine');
    background-repeat: repeat-x sau repeat-y;
}
```

Pentru a afișa imaginea de fundal o singură dată, `background-repeat` trebuie să aibă valoarea `no-repeat`.

Pentru repeta imaginea de fundal pe ambele axe, `background-repeat` trebuie să aibă valoarea `repeat`.

Imaginea poate fi poziționată în cadrul paginii cu ajutorul proprietății `background-position` ce poate lua două din valorile `left`, `right`, `top`, `bottom`, `center`, sau `x% y%`, sau `x_pixeli y_pixeli`, conform sintaxei:

```
body {
    background-image: url('nume sau adresă imagine');
    background-repeat: no-repeat;
    background-position: left top;
}
```

Dacă se dorește păstrarea poziției fundalului în cazul derulării paginii sus-jos se va folosi proprietatea `background-attachment` conform sintaxei:

```
background-attachment: fixed;
```

Valorile pot fi următoarele:

- `scroll`: fundalul se mișcă împreună cu elementul (valoare implicită)
- `fixed`: fundalul este fix
- `local`: fundalul se mișcă împreună cu conținutul elementului
- `initial`: fundalul are proprietatea inițială
- `inherit`: fundalul are proprietatea moștenită

Să se transforme în comentariu linia de cod: `background-repeat: no-repeat;`.
Să se stabilească culoarea `lightblue` pentru fundal. După vizualizarea paginii se va transforma în comentariu linia de cod.
Să se utilizeze ca imagine de fundal *wallpaper_non_tileable.jpg* și să se vizualizeze pagina.
Să se utilizeze ca imagine de fundal *wallpaper_tileable_horizontal.jpg* și:

- Să se vizualizeze pagina.
- Să se realizeze afișarea imaginii o singură dată în poziția stânga-sus.
- Să se realizeze afișarea imaginii cu repetarea doar pe orizontală.
- Să se realizeze afișarea imaginii cu păstrarea poziției în cazul derulării.

Să se utilizeze ca imagine de fundal *wallpaper_tileable.jpg*, cu repetarea ei pe ambele axe.

3.5 Atribute pentru chenare

Proprietatea `border-style` permite stabilirea stilului, lățimii și culorii chenarului unui element și poate lua următoarele valori:

- `dotted` – Definește un chenar cu linie punctată
- `dashed` – Definește un chenar cu linie întreruptă
- `solid` – Definește un chenar cu linie continuă
- `double` – Definește un chenar cu linie dublă
- `groove` – Definește un chenar cu linie 3D grove
- `ridge` – Definește un chenar cu linie 3D ridge
- `inset` – Definește un chenar cu linie 3D inset
- `outset` – Definește un chenar cu linie 3D outset
- `none` – Nu definește nici un chenar
- `hidden` – Definește un chenar ascuns

Sintaxa este următoarea:

```
<p style="border-style: solid;">Text de afisat</p>
```

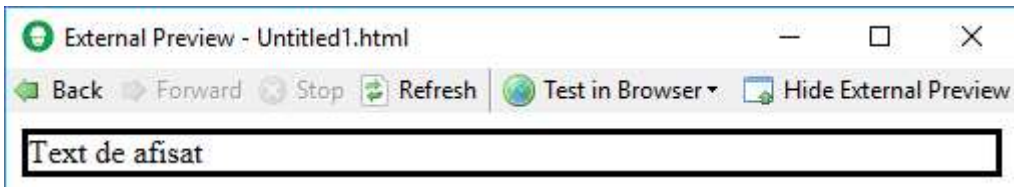

Figura 23. Utilizarea proprietății `border-style`

Proprietatea `border-style` poate lua până la patru valori, câte una pentru fiecare latură a chenarului.

```
<p style="border-style: double dashed dotted solid;">Text de afisat</p>
```

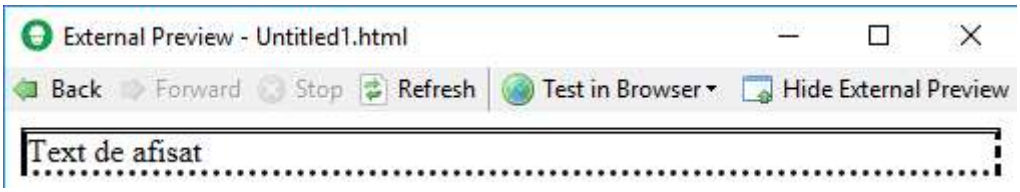

Figura 24. Utilizarea proprietății `border-style` cu valori individuale

Pentru ca proprietatea `border-style` să afecteze mai multe elemente simultan se pot defini clase, de exemplu:

```
p.linie_continua {border-style: solid;}
```

Stilul laturilor chenarului se poate defini și individual pentru fiecare latură în parte, astfel:

```
p {
    border-top-style: double;
    border-right-style: solid;
    border-bottom-style: double;
    border-left-style: solid;
}
```

sau, cu același rezultat:

```
p {
    border-style: double solid;
}
```

Dacă proprietatea are patru valori: `border-style: valoare1 valoare2 valoare3 valoare4;`, atunci:

- `top` are valoarea 1
- `right` are valoarea 2

- `bottom` are valoarea 3
- `left` are valoarea 4

Dacă proprietatea are trei valori: `border-style: valoare1 valoare2 valoare3;`, atunci:

- `top` are valoarea 1
- `right` și `left` au valoarea 2
- `bottom` are valoarea 3

Dacă proprietatea are două valori: `border-style: valoare1 valoare2;`, atunci:

- `top` și `bottom` au valoarea 1
- `right` și `left` au valoarea 2

Grosimea liniei chenarului se stabilește cu proprietatea `border-width` ce poate lua valori în pixeli, centimetri ș.a. sau o valoare predefinită: `thin`, sau `medium` sau `thick`, conform sintaxei:

```
border-width: 5px;
```

Culoarea liniei chenarului se stabilește cu proprietatea `border-color`, conform sintaxei:

```
border-color: red;
```
sau
```
border-color: red blue orange black;
```

Proprietatea poate lua până la patru valori, câte una pentru fiecare latură a chenarului.

Rotunjirea conturului chenarului se poate face cu proprietatea `border-radius` cu specificarea valorii în pixeli, conform sintaxei:

```
border-radius: 5px;
```

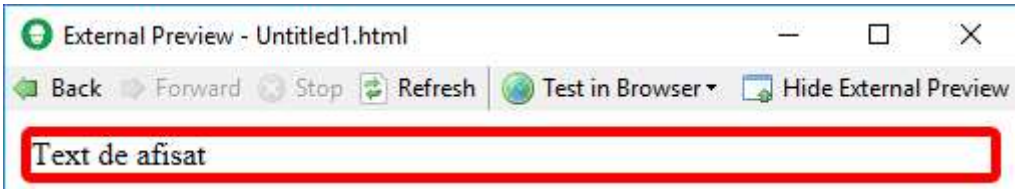

Figura 25. Utilizarea proprietăților `border-width`, `border-color` și `border-radius`

Să se definească un chenar cu linie continuă sus-jos și linie punctată stânga-dreapta care să afecteze toate titlurile de tip Heading 2. Să se stabilească grosimea liniei chenarului de 2 pixeli, culoarea `grey` și o rotunjire de 6 pixeli.

3.6 Atribute pentru margini

Marginea exterioară (*Margin*) reprezintă distanțele față de elementele vecine, calculate de la marginea exterioară a chenarului.

Marginea interioară (*Padding*) reprezintă distanțele între conținut și marginile interioare ale chenarului.

Figura 26. Modelul de încadrare al elementelor

3.6.1 Atribute pentru marginea exterioară

Sintaxa pentru stabilirea unei margini exterioare echidistante este următoarea:

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

Stabilirea marginii exterioare se poate face și individual pentru fiecare latură, conform exemplelor următoare, sau valorile pot fi grupate folosindu-se principiul prezentat la proprietatea `border-style`.

```
p {
    margin-top: 30px;
    margin-right: 20px;
    margin-bottom: 25px;
    margin-left: 15px;
}
```

sau, cu același rezultat:

```
p {
    margin: 30px 20px 25px 15px;
}
```

3.6.2 Atribute pentru marginea interioară

Sintaxa pentru stabilirea unei margini interioare echidistante este următoarea:

```
p {
    padding: 20px;
}
```

Stabilirea marginii interioare se poate face și individual pentru fiecare latură, conform exemplelor următoare, sau valorile pot fi grupate folosindu-se principiul prezentat la *border-style*.

```
p {
    padding-top: 30px;
    padding-right: 20px;
    padding-bottom: 25px;
    padding-left: 15px;
}
```

sau, cu același rezultat:

```
p {
    padding: 30px 20px 25px 15px;
}
```

Pentru titlul de tip Heading 1 să se stabilească o margine exterioară de 250 pixeli pentru stânga și dreapta, de 50 pixeli pentru sus și jos și o margine interioară de 20 pixeli pentru toate laturile.

Pentru titlurile de tip Heading 2 să se stabilească o margine exterioară în dreapta de 700 pixeli și o margine interioară de 10 pixeli sus și jos.

Capitolul 4. Introducerea de funcții pentru gestionarea/modificarea obiectelor dintr-o pagină Web – JavaScript

4.1 Introducere

JavaScript este un limbaj de programare de nivel înalt care permite realizarea de pagini Web interactive.

Liniile de cod JavaScript se inserează în pagina HTML între marcajele `<script>` și `</script>`. Pentru programele de navigare mai vechi se utilizează marcajul de deschidere `<script type="text/javascript">`.

Marcajele `<script>` pot fi inserate:

- în zona `<head>`
- în zona `<body>`. Este de preferat ca scripturile să fie amplasate la sfârșitul elementului `<body>` pentru încărcarea mai rapidă a paginii.
- în fișier extern (cu extensia „.js”) și declarația
  `<script src="Script.js"></script>`

Un program scris în JavaScript conține o serie de declarații ce trebuie executate de programul de navigare. O declarație poate cuprinde valori, operatori, expresii, cuvinte cheie sau comentarii și se termină, de obicei, cu caracterul “`;`”. Caracterul nu este obligatoriu dacă scriptul conține o singură declarație.

Declarațiile încep, de obicei, cu un cuvânt cheie (de ex. `var`, `function`, `if`, `for`). Acestea sunt cuvinte rezervate și nu pot fi folosite pentru variabile.

Când se dorește executarea mai multor declarații împreună se pot crea blocuri de cod care grupează declarațiile între caracterele `{ }`.

```
if (a<5) {
    x=12;
    y=20;
}
```

Aproape orice într-un cod JavaScript reprezintă un obiect, iar metodele sunt acțiuni care pot fi aplicate pe obiecte. Există metode predefinite dar și posibilitatea de a crea unele noi.

```
document.getElementById(id)
```

În limbajul JavaScript se face diferența între literele mari și cele mici, adică limbajul este *Case Sensitive*. De obicei, funcțiile sau variabilele se denumesc începând cu literă mică, iar constructorii cu literă mare. Caracterele *Enter* și *Space* sunt ignorate.

```
var a = getElementById('a');
var ziuaCurenta = new Data("2018", "Noiembrie", "2");
```

Comentariile se definesc cu ajutorul caracterelor `//` pentru un singur rând, sau `/*  */` pentru a cuprinde mai multe rânduri.

4.2 Obiecte JavaScript

Obiectele reprezintă baza limbajului JavaScript și aproape orice reprezintă un obiect.

Valorile logice (`Boolean`), numerele (`Number`) și șirurile (`String`) pot fi obiecte dacă sunt definite cu cuvântul cheie `new`.

Datele, elementele matematice, matricele, expresiile obișnuite, funcțiile, obiectele sunt întotdeauna obiecte.

Singurele care nu reprezintă obiecte sunt primitivele, adică valori ale datelor de tip șir de caractere (de ex. `Popescu`), număr (de ex. `2.1`), boolean (`true` sau `false`), nule sau nedefinite.

Adresarea obiectelor se face prin referință, nu prin valoare, prin urmare obiectele sunt mutabile. În exemplul următor, `persoana` nu este o copie a `utilizator` ci sunt același obiect, iar o schimbare a valorii lui `persoana` duce la schimbarea valorii și lui `utilizator`.

```
var utilizator = "Popescu George";
var persoana = utilizator;
```

Obiectele pot să conțină mai multe valori, caz în care acestea sunt asociate unor proprietăți:

```
var utilizator = {nume: "Popescu", prenume: "George", grupa: "8200",
email: "george.popescu@upb.ro"};
```

În exemplul de mai sus *nume* este o proprietate iar *"Popescu"* este valoarea acesteia.

Cea mai utilizată modalitate de a accesa o proprietate a unui obiect este folosind sintaxa `numeObiect.proprietate`. De exemplu `utilizator.nume`.

O nouă proprietate se poate adăuga unui obiect doar prin menționarea ei și atribuirea unei valori, iar ștergerea uneia existente se face cu cuvântul cheie `delete`.

```
utilizator.varsta = 50;
delete utilizator.email;
```

Metodele sunt acțiuni care pot fi aplicate pe obiecte. Ca și definiție, metodele sunt funcții stocate ca proprietăți ale unui obiect. Metoda unui obiect poate fi accesată folosind sintaxa `numeObiect.numeMetoda()`.

```
var utilizator = {
    nume: "Popescu",
    prenume: "George",
    grupa: "8200",
    email: "george.popescu@upb.ro"
    numeComplet: function() {
        return this.nume + " " + this.prenume;
    }
};
```

`utilizator.numeComplet()` pentru apelare.

4.3 Operații asupra paginii Web

Rezultatul unui cod JavaScript poate fi afișat în pagina Web în mai multe feluri:

- Prin scriere într-un element HTML, cu ajutorul proprietății `innerHTML` atașată unei metode care să selecteze elementul sau elementele dorite.
- Prin scriere direct în pagina Web, utilizând metoda `document.write()`.
- Prin generarea unui mesaj într-o căsuță de alertare, utilizând metoda `window.alert()`.
- Prin scrierea în consola de depanare a programului de navigare, utilizând metoda `console.log()`.

4.3.1 Scrierea într-un element HTML

Pentru a accesa un element HTML se pot utiliza trei metode:

- Selectarea elementului pe baza id-ului asociat acestuia, folosind metoda `document.getElementById("id")`.
- Selectarea elementelor pe baza numelui asociat acestora, folosind metoda `document.getElementsByName("nume")`. Metoda nu returnează elementele ci o listă de noduri în ordinea acestora, numerotate începând cu 0.
- Selectarea elementelor pe baza clasei asociată acestora, folosind metoda `document.getElementsByClassName("clasă")`. Metoda nu returnează elementele ci o listă de noduri în ordinea acestora, numerotate începând cu 0.

Într-o pagină HTML nouă să se copieze liniile de cod următoare:

```
<h2>Pagina pentru testarea JavaScript</h2>
<p id="paragraf1">Text afisat cu HTML.</p>
<p id="paragraf2">Text afisat cu HTML.</p>
<p id="paragraf3" class="text">Text afisat cu HTML.</p>
<p id="paragraf4" class="text">Text afisat cu HTML.</p>
```

Să se introducă apoi următorul script:

```
<script>
    document.getElementById("paragraf1").innerHTML = "Text afisat
    cu JavaScript.";
</script>
```

Să se adauge în script următoarea linie de cod:

```
    document.getElementsByClassName("text")[0].innerHTML = "Text
    afisat cu JavaScript.";
```

Să se schimbe nodul [0] cu [1].

4.3.2 Scrierea direct în pagina Web

Să se adauge în script următoarea linie de cod:

```
document.write("© 2018");
```

Se recomandă folosirea numai pentru testare. Dacă se utilizează după ce pagina s-a încărcat metoda va șterge tot conținutul HTML din pagină.

4.3.3 Generarea unui mesaj într-o căsuță de alertare

Să se adauge în script următoarea linie de cod:

```
window.alert("Mesaj de alerta!");
```

4.3.4 Scrierea în consola de depanare a programului de navigare

Să se adauge în continuarea paragrafului 4 următoarele linii de cod:

```
<h3>Modul de depanare se activeaza cu tasta F12</h3>
<p>Se selecteaza apoi "Console" in meniu si se apasa pe
Refresh.</p>
```

Să se adauge în script următoarea linie de cod:

```
    console.log("Mesaj pentru depanare.");
```

4.4 Variabile

În JavaScript se pot evidenția patru tipuri de variabile:

- Numerice: `a = 4;`
- Caractere sau șiruri de caractere: `nume = "Popescu";` Se pot include atât între caracterele `" "` cât și între `' '`.
- Obiect: `utilizator = {nume: "Popescu", prenume: "George", grupa: "8200", email: "george.popescu@upb.ro"};` Accesarea unei proprietăți a unui obiect se face folosind sintaxa `numeObiect.proprietate`. De exemplu `utilizator.nume`.
- Vector: `facultate = ["Transporturi", "Electronică", "Automatică"];` În JavaScript vectorul sau matricea sunt considerate tot obiecte. Accesarea unei valori se face dintr-un vector sau matrice se face folosind sintaxa `numeVector[i]`, unde i reprezintă numărul poziției valorii, începând cu 0. De exemplu `facultate[0]` are valoarea `Transporturi`.

Variabilele se declară cu cuvintele cheie `var`, `let` sau `const` și pot conține valori de orice tip. Alocarea valorii se face utilizând semnul `=`. Variabilele pot fi locale, atunci când se declară în interiorul funcțiilor, sau globale când se declară în afara lor.

Pentru calcule se folosesc operatorii clasici `+ - * /` iar expresiile de calcul pot conține atât valori distincte cât și variabile.

```
var a = 4;
var b = 8;
var c = a + b;
```

sau

```
var a = 4, b = 8, c = a + b;
```

sau

```
var a = 4,
b = 8,
c = a + b;
```

Dacă variabila nu are o valoare inițială (de ex. `var a;`) atunci se consideră că aceasta este nedefinită (`undefined`).

În cadrul declarării unei variabile se pot utiliza operații matematice.

`var c = 4 + 8;` (adunare matematică a două numere)

`var nume = "Popescu" + " " + "George";` (concatenare de șiruri de caractere)

Atenție la adunarea numerelor cu șiruri de caractere! Numerele vor fi tratate ca șiruri de caractere și concatenate.

```
"4" + 5 + 2 va fi transformat în 452
4 + 5 + "2" va fi transformat în 92
```

Lungimea unui șir de caractere se poate afla folosind proprietatea `length`.

```
var nume = "Popescu";
```
`var lungime = nume.length;` va returna valoarea 7.

Să se adauge un nou paragraf cu id-ul `paragraf5` în continuarea paragrafului 4, care să conțină textul:
`Text afisat cu HTML.`
Să se creeze o variabilă de tip număr numită `x` care să ia valoarea `5`.
Să se scrie în elementul HTML cu id-ul `paragraf5` următorul text:

```
Acesta este paragraful cu numarul 5.
```

unde valoarea 5 să rezulte din citirea variabilei *x*.

Să se adauge un nou paragraf cu id-ul `paragraf6` în continuarea paragrafului 5, care să conțină textul:
`Text afisat cu HTML.`
Să se creeze o variabilă de tip obiect numită `limbaj` care să conțină proprietățile `varianta1` și `varianta2`, cu valorile de tip șir de caractere corespunzătoare: `HTML` și `JavaScript`.
Să se creeze o variabilă de tip șir de caractere numită `propozitie` care să conțină următorul text:

```
Textul scris in HTML a fost suprascris cu cel scris in
JavaScript.
```

unde cuvintele `HTML` și `JavaScript` să rezulte din citirea variabilei `limbaj`.
Să se scrie în elementul HTML cu id-ul `paragraf6` conținutul variabilei `propozitie`.

Să se adauge un nou paragraf, fără text inițial, cu id-ul `paragraf7` în continuarea paragrafului 6.
Să se creeze variabila `lungime` care să conțină lungimea șirului de caractere `propozitie`.
Să se scrie în elementul HTML cu id-ul `paragraf7` următorul text:

```
Paragraful anterior are x de caractere.
```

unde `x` să fie conținutul variabilei `lungime`.

4.5 Funcții

O funcție este o porțiune de cod executată numai atunci când este apelată, de exemplu în cazul unui eveniment precum apăsarea unui buton.

Sintaxa de definire a unei funcții este următoarea:

```
function numeFunctie(argument1, argument2, ...) {
    cod de executat;
    return valoare; opțional
}
```

Argumentele unei funcții sunt valori pe care le primește în momentul apelării acesteia și sunt considerate variabile locale în interiorul funcției.

Exemplu de funcție cu argumente:

```
function puterea2(valoare) {
    var rezultat = valoare * valoare;
    return rezultat;
}
document.getElementById("element").innerHTML = puterea2(10);
```

Exemplu de funcție fără argumente:

```
function schimbaText() {
    document.getElementById("id").innerHTML = "Text de introdus.";
}
```

Apelarea unei funcții se face utilizând sintaxa `numeFunctie(valoare1, valoare2, ...)`, de exemplu:

```
puterea2(10)
schimbaText()
```

Să se creeze funcția `schimbaText` care să conțină două argumente `id` și `text`. Aceasta va scrie în elementul HTML cu id-ul `id`, textul `text`.
Să se aplice funcția pentru a înlocui minim două din liniile de cod care au rolul de a scrie un text într-un paragraf selectat pe bază de id.

4.6 Instrucțiuni

4.6.1 Declarații condiționale

Declarațiile condiționale sunt acelea în urma cărora se execută o acțiune în funcție de îndeplinirea sau nu a unei condiții. În JavaScript se pot utiliza declarațiile `if`, `else` și `switch`.

Declarația `if` permite executarea unui bloc de cod numai dacă este îndeplinită condiția. Sintaxa este următoarea:

```
if (condiție) {
    bloc de cod executat daca este îndeplinită condiția
}
```

Declarația `else` se folosește împreună cu `if` pentru a executa un alt bloc de cod dacă nu este îndeplinită condiția menționată în declarația `if`. Sintaxa este următoarea:

```
if (condiție) {
    bloc de cod executat daca este îndeplinită condiția
} else {
    bloc de cod executat daca nu este îndeplinită condiția
}
```

Declarația `switch` conține mai multe blocuri de cod ce trebuie executate, fiecare dintre acestea fiind asociate unui caz. Declarația compară valoarea unei expresii cu valoarea fiecărui caz și dacă găsește unul în care cele două valori sunt identice (nu numai ca valoare ci și ca tip) va executa blocul de cod asociat acelui caz, după care execuția declarației se termină (prin inserarea cuvântului cheie `break`). `break` nu trebuie inclus în ultimul dintre cazuri deoarece declarația se va termina oricum.

Cuvântul cheie `default` se folosește pentru a indica ce bloc de cod se va executa dacă valoarea expresiei nu se regăsește în nici unul dintre cazuri.

Sintaxa este următoarea:

```
switch (expresie) {
    case a:
         bloc de cod executat daca expresie === a
         break;
    case b:
         bloc de cod executat daca expresie === b
         break;
    default:
         bloc de cod executat
}
```

4.6.2 Bucle

Buclele se utilizează pentru a executa blocuri de cod de un anumit număr de ori sau cât timp este îndeplinită o condiție. În JavaScript se pot utiliza declarațiile `for`, `for/in` și `while`.

Bucla `for` execută un bloc de cod de un anumit număr de ori. Sintaxa este următoarea:

```
for (declarația 1; declarația 2; declarația 3) {
    bloc de cod
}
```

Declarația 1 se execută o singură dată, înainte de începerea execuției blocului de cod. Aici se stabilește, de obicei, valoarea inițială a unei variabile. Se pot inițializa mai multe variabile, dar trebuie separate prin virgulă, sau se poate omite declarația cu totul, dacă inițializarea a avut loc deja în altă parte a codului JavaScript.

Declarația 2 reprezintă condiția care trebuie îndeplinită de variabilă pentru a se executa blocul de cod. Declarația 2 este și ea opțională, caz în care blocul de cod se va executa la nesfârșit. Pentru a evita acest lucru, în blocul de cod trebuie introdus cuvântul cheie `break`.

Declarația 3 se execută de fiecare dată, după execuția blocului de cod. Aici, de obicei, se modifică cu un pas valoarea variabilei din declarația 1. Declarația 3 este și ea opțională, dacă modificarea valorii variabilei se face în blocul de cod ce se va executa.

Bucla `for` / `in` parcurge proprietățile unui obiect. Sintaxa este următoarea:

```
var obiect = {proprietate1:valoare, proprietate2:valoare,
proprietate3:valoare}
var x;
for (x in obiect) {
    bloc de cod ce conține, de obicei, menționarea proprietăților prin
    sintaxa obiect[x]
}
```

Bucla `while` execută un bloc de cod atâta timp cât o condiție este îndeplinită. Sintaxa este următoarea:

```
while (condiție) {
    bloc de cod
}
```

Bucla `do / while` execută un bloc de cod atâta timp cât o condiție este îndeplinită. Se execută mai întâi bucla de cod și după aceea se verifică îndeplinirea condiției. Sintaxa este următoarea:

```
do{
    bloc de cod
}
while (condiție)
```

4.7 Evenimente

Evenimentele au loc atunci când programul de navigare sau utilizatorul acestuia realizează acțiuni asupra paginii HTML. Evenimentele pot reprezenta totul, de la interacțiunile de bază ale utilizatorilor până la notificările automate.

De exemplu, eveniment poate fi când utilizatorul dă clic pe un element al paginii, dacă apasă butonul de trimitere a unui formular, dacă apare o eroare, când o pagină se termină de încărcat, etc. Ca răspuns la aceste evenimente codurile JavaScript efectuează diferite acțiuni, de exemplu afișarea unui mesaj, schimbarea culorii unui fundal, închiderea paginii, validarea unor date, etc.

Sintaxa prin care se detectează un eveniment și se produce o acțiune este următoarea:

```
<elementHTML eveniment='cod JavaScript'>
```

sau

```
<elementHTML eveniment="cod JavaScript">
```

4.7.1 Evenimente pentru mouse

`onclick` – apare la efectuarea unui clic pe un element

`oncontextmenu` – apare la efectuarea unui clic dreapta pe un element

`onmouseover` – apare atunci când pointer-ul mouse-ului este mutat deasupra unui element

`onmouseleave` – apare atunci când pointer-ul mouse-ului este mutat de pe un element

Să se adauge titlului de tip Heading 2 următoarele atribute:

```
onmouseover="this.style.backgroundColor='orange'"
onmouseleave="this.style.backgroundColor='white'"
```

Să se adauge paragrafului 1 următorul atribut:

```
oncontextmenu="window.alert('Actiune interzisa!'); return false"
```

Să se adauge în continuarea paragrafului 7 următoarele linii de cod:

```
<p id="paragraf8" onclick="this.innerHTML = Date()">Clic pe acest text
pentru a vedea data si ora!</p>
<button onclick="document.getElementById('paragraf9').innerHTML =
Date()">Arata data si ora!</button>
<p id="paragraf9"></p>
```

Să se testeze efectul modificărilor.

4.7.2 *Evenimente pentru tastatură*

`onkeydown` – apare la apăsarea unui taste (tasta este jos)

`onkeypress` – apare la apăsarea unui taste (tasta a fost apăsată și eliberată)

`onkeyup` – apare la eliberarea unei taste ce a fost apăsată

Să se adauge în continuarea paragrafului 9 următoarele linii de cod:

```
<textarea maxlength="30" onKeyUp="numarare()" id="text"></textarea>
<label id="numarCaractere"></label>
```

Să se adauge în script următoarea funcție:

```
function numarare() {
   var contor = document.getElementById('text').value.length;
   if (contor == 1) {
      document.getElementById('numarCaractere').innerHTML = "Ati
      introdus 1 caracter";
   }
   else if (contor < 20) {
      document.getElementById('numarCaractere').innerHTML = "Ati
      introdus " + contor + " caractere.";
   }
   else if (contor < 30){
      document.getElementById('numarCaractere').innerHTML = "Ati
      introdus " + contor + " de caractere.";
   }
   else {
      document.getElementById('numarCaractere').innerHTML = "Ati atins
      limita maxima de 30 de caractere!";
   }
}
```

Să se testeze efectul modificărilor.

Să se adauge în continuarea câmpului pentru text următoarele linii de cod:

```
<form action = "">
   <label>Numar de telefon:</label>
   <input type="text" id="telefon" maxlength="10" onKeyPress="return
   doarCifre(event)">
   <label id="mesaj"></label>
</form>
```

Să se adauge în script următoarea funcție:

```
function doarCifre(event) {
   var codTasta = event.keyCode;
   if (codTasta < 48 && codTasta > 57) {
      document.getElementById("telefon").style.borderColor = "red";
      document.getElementById("mesaj").innerHTML = "Va rugam sa tastati
      doar cifre!";
      return false;
      }
   else {
      document.getElementById("telefon").style.borderColor = "";
      document.getElementById("mesaj").innerHTML = "";
      return true;
   }
}
```

Să se testeze efectul modificărilor.

4.7.3 Evenimente pentru formulare

`onfocus` – apare atunci când utilizatorul acționează asupra unui element

`onblur` – apare atunci când utilizatorul a terminat acțiunea asupra unui element

`onchange` – apare atunci când se schimbă starea unui element (pentru `<input>`, `<select>`, `<textarea>`)

`oninput` – apare atunci când utilizatorul introduce date într-un element

`onselect` – apare atunci când utilizatorul selectează un text dintr-un element (pentru `<input>` și `<textarea>`)

`onsearch` – apare atunci când utilizatorul scrie ceva într-un câmp de căutare (`<input="search">`)

`onreset` – apare la resetarea formularului

`onsubmit` – apare atunci când utilizatorul trimite formularul

Să se adauge la finalul formularului următorul buton:

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

Să se adauge formularului atributul:

```
onSubmit="return verificareFormular()"
```

Să se adauge în script următoarea funcție:

```
function verificareFormular() {
   if (document.getElementById('telefon').value == '') {
      window.alert("Numar de telefon necompletat!");
      return false;
   }
   if (document.getElementById('telefon').value.length < 10) {
      window.alert("Numar de telefon incomplet!");
      return false;
   }
}
```

Să se testeze efectul modificărilor.

Capitolul 5. Programarea răspunsului serverelor – PHP

5.1 Introducere

PHP (*Hypertext Pre-processor*) este un limbaj pentru crearea de scripturi ce pot fi încorporate într-o pagină HTML cu scopul de a crea pagini Web dinamice sau de a încorpora aplicații Web.

Interpretarea și executarea codului PHP are loc la nivelul serverului ce găzduiește pagina Web, rezultatul fiind încorporat în pagina HTML generată utilizatorului.

Fișierele care conțin cod PHP au extensia .php . Pe lângă codul PHP acestea pot conține și marcaje HTML sau scripturi JavaScript.

Dintre aplicațiile PHP se pot menționa deschiderea, citirea, scrierea sau crearea de fișiere, crearea, modificarea sau accesarea de fișiere tip *cookie*, lucrul cu baze de date, cu aplicații de email, criptarea datelor, etc.

Sintaxa pentru scrierea unui script PHP este următoarea:

```
<?php
    cod PHP
?>
```

O declarație PHP se termină, de obicei, cu caracterul "`;`". Caracterul nu este obligatoriu dacă scriptul conține o singură declarație.

În limbajul PHP, pentru cuvintele cheie (`echo`, `if`, `for`, etc.), clase sau funcții nu se face diferența între literele mari și cele mici (totuși se recomandă să se folosească numai litere mici), însă acest lucru contează în cazul numelor variabilelor.

Comentariile se definesc cu ajutorul caracterelor `//` pentru un singur rând, sau `/*  */` pentru a cuprinde mai multe rânduri.

Accesarea paginilor Web care conțin scripturi PHP se poate realiza cu orice program de navigare. Este necesară însă existența unui server Web, cu suport PHP, care să interpreteze și să execute scripturile. În acest sens se va utiliza o aplicație ce încorporează un sever Web și suport PHP pentru acesta precum Wamp (www.wampserver.com).

După instalare, pentru salvarea fișierului .php creat în această lucrare se va utiliza directorul C:\wamp64\www\TPI, iar accesarea paginii Web se va face cu orice program de navigare folosind adresa http://localhost/TPI/**nume_fisier**.php

5.2 Variabile

În PHP se pot evidenția mai multe tipuri de variabile:

- *Integer* – numere întregi, fără virgulă.
- *Double* – numere reale cu virgulă.
- *Boolean* – au două valori posibile, *true* sau *false*.
- *NULL* – un tip special de variabilă ce poate avea o singură valoare: NULL. Dacă unei variabile nu i se alocă o valoare inițială, i se va aloca în mod implicit valoarea NULL.
- *String* – Caractere sau șiruri de caractere ce se pot include atât între caracterele `" "` cât și între `' '`.
- Matrice – este utilizată pentru stocare mai multor valori într-o singură variabilă: de exemplu `$facultate = array("Transporturi", "Electronică", "Automatică")`. Accesarea unei valori se face dintr-o matrice se face folosind sintaxa `$numeMatrice[i]`, unde i reprezintă numărul poziției valorii, începând cu 0. De exemplu `$facultate[0]` are valoarea `Transporturi`.
- Obiect – este utilizat pentru stocarea datelor și a unor funcții necesare prelucrării acelor date. Obiectele se declară prin utilizarea cuvântului cheie `class`.
- Resursă – nu este un tip de date obișnuit ci stochează referințe către resurse externe cum ar fi o bază de date.

Tipul unei variabile nu trebuie declarat. În momentul alocării unei valori, PHP determină în mod automat de ce tip va fi aceasta.

Numele unei variabile începe întotdeauna cu caracterul `$`, iar acestea nu se declară cu o anumită comandă ci sunt create în momentul când sunt menționate pentru prima oară. Alocarea valorii se face utilizând semnul =.

```
$text = "Hello World!";
$a = 23;
$b = 55.44;
```

Pentru calcule matematice se folosesc operatorii clasici `+ - * /` iar expresiile de calcul pot conține atât valori distincte cât și variabile.

Concatenarea șirurilor de caractere se face folosind caracterul ".":

`"Hello " . "World!"` va fi echivalent cu `"Hello World!"`

Afișarea conținutului unei variabile într-o pagină Web se realizează cu declarația `echo`.

```
echo $text;
```

Variabilele super-globale sunt variabile predefinite și pot fi utilizate oriunde în cadrul unui script PHP. Acestea sunt:

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

Variabilele globale se declară și pot fi folosite numai în afara funcțiilor. Utilizarea unei variabile globale în interiorul unei funcții se poate face prin menționarea cuvântului cheie `global` înaintea variabilei sau prin utilizarea variabilei super-globală numită `$GLOBALS['index']`, unde index este numele variabilei globale.

```
function nume_functie() {
    global $a, $b;
    cod ce utilizează variabilele $a și $b

    sau

    cod ce utilizează variabilele $GLOBALS['a'] și $GLOBALS['b']
}
```

Variabilele locale se declară într-o funcție și pot fi folosite numai în interiorul acelei funcții. Se pot defini mai multe variabile cu același nume, cu condiția ca acestea să fie locale și să se afle în funcții diferite.

Orice variabilă locală este ștearsă după execuția funcției. Dacă dorim să folosim variabila și la o altă execuție a funcției se va menționa cuvântul cheie `static` atunci când se declară variabila.

```
function nume_functie() {
    static $x, $y;
    cod ce utilizează/reutilizează variabilele $x și $y
}
```

5.3 Prelucrarea formularelor

Pentru prelucrarea datelor dintr-un formular sunt necesare două pagini HTML distincte: pagina care conține formularul și pagina care preia datele și le prelucrează.

În pagina care conține formularul (ce nu trebuie neapărat să conțină scripturi PHP), în cadrul marcajului `<form>` trebuie menționate două atribute:

- `action="`*`valoare`*`"` ce trebuie să conțină ca valoare numele paginii care va prelua și prelucra datele din formular (de exemplu `login.php`).
- `method="`*`valoare`*`"` ce trebuie să conțină ca valoare numele metodei prin care se preiau datele (`post` sau `get`).

De asemenea, marcajul fiecărui câmp de introducere sau selectare de date trebuie să conțină o valoare distinctă pentru atributul `name`.

Într-o pagină HTML nouă să se insereze următorul formular:

```
<form action="autentificare.php" method="post">
  Utilizator: <input type="text" name="utilizator"><br>
  Parola: <input type="password" name="parola"><br>
  <input type="submit">
</form>
```

Să se salveze pagina cu numele acasa.php în directorul C:\wamp64\www\TPI.

În pagina care prelucrează datele din formular se introduce un script care le preia cu ajutorul unei variabile super-globale: `$_GET` sau `$_POST`, în funcție de metoda aleasă.

Într-o pagină HTML nouă să se insereze următoarele scripturi:

```
Utilizator: <?php echo $_POST["utilizator"] ?> <br>
Parola: <?php echo $_POST["parola"] ?>
```

Să se salveze pagina cu numele autentificare.php în directorul C:\wamp64\www\TPI.
Să se completeze formularul și să se vizualizeze rezultatul apăsării butonului *Submit*.
Să se testeze și folosirea metodei GET.

`$_GET` va include datele completate în formular direct în adresa URL a paginii care urmează să le prelucreze (pentru exemplul de mai sus adresa este http://localhost/TPI/autentificare.php?utilizator=&parola=). Această metodă nu se va folosi niciodată pentru transmiterea de date importante deoarece ele sunt vizibile în înregistrările de pe server și rămân în istoricul programului de navigare. Metoda are totuși un avantaj, acela că adresa URL poate fi salvată și refolosită ulterior (de exemplu pentru interogarea unei baze de date).

`$_POST` va include datele completate în formular în corpul mesajului transmis printr-o cerere de tip POST caracteristică protocolului HTTP, prin urmare vor fi greu de accesat de o terță parte. Este metoda recomandată și cel mai des utilizată.

5.4 Lucrul cu fișiere de tip *cookie*

Cookie-urile sunt fișiere text de mici dimensiuni ce sunt salvate în calculatorul utilizatorului de către programul de navigare.

Sunt utilizate pentru a identifica utilizatorul atunci când acesta va accesa din nou pagina Web care a generat fișierul și a memora anumite informații, precum cele de autentificare, activitatea de navigare, memorarea preferințelor personalizate, etc.

Un fișier *cookie* poate fi citit numai de pagina Web care l-a generat. O dată setat toate cererile făcute de utilizator pentru a reaccesa pagina Web vor fi însoțite de numele fișierului și de valorile memorate în acesta.

Cu ajutorul PHP se pot crea sau extrage fișiere *cookie*.

Crearea unui fișier cookie se face cu ajutorul funcției `setcookie()`, iar sintaxa este următoarea (nu toate argumentele trebuie completate, cele mai importante fiind numele și valoarea):

```
setcookie(nume, valoare, expirare, cale, domeniu, securitate, doar_http);
```

- Valorile câmpurilor *nume* și *valoare* se scriu ca șiruri de caractere.
- Câmpul *expirare* conține timpul de valabilitate al fișierului exprimat cu sintaxa `time() + numar_secunde`.
- Câmpul *cale* stabilește accesul la subdirectoare. Utilizarea ca valoare a caracterului `"/"` permite accesul la întreg domeniul.
- Câmpul *domeniu* stabilește accesul la subdomenii.
- Câmpul *securitate* stabilește dacă fișierul este trimis prin http sau https.
- Câmpul *doar_http* limitează accesul la fișier limbajelor scriptice, cum ar fi JavaScript.

Atenție, funcția `setcookie()` trebuie menționată înainte de marcajul `<html>`!

Citirea unui fișier *cookie* se face cu ajutorul variabilei super-globale `$_COOKIE` cu sintaxa `$_COOKIE[nume]`.

Verificarea setării valorii unui fișier *cookie* se face cu funcția `isset()`. Aceasta returnează valoarea *true* dacă valoarea a fost setată, sau *false* dacă nu a fost setată.

Ștergerea valorii unui fișier *cookie* se realizează prin setarea unui timp de expirare negativ.

Să se adauge în pagina acasa.php scriptul următor ce creează un fișier *cookie* cu numele Utilizator, valoarea Popescu George, timpul de expirare 1 oră și acces la întreg domeniul:

```
<?php
   setcookie("Utilizator", "Popescu George", time() + 3600, "/");
?>
```

Să se adauge următorul script care să afișeze în pagină starea și valoarea fișierului *cookie*:

```
<?php
if(!isset($_COOKIE["Utilizator"])) {
   echo "Fisierul cookie cu numele Utilizator nu are valoare
   setata!";
} else {
   echo "Valoarea fisierului cookie cu numele Utilizator este: " .
   $_COOKIE["Utilizator"];
}
?>
```

Să se reîncarce pagina pentru a se seta fișierul *cookie*.
Pentru a vedea dacă programul de navigare Google Chrome a salvat fișierul *cookie* se va accesa adresa chrome://settings/siteData. Se caută în listă domeniul *localhost* și se verifică prezența și conținutul fișierului *cookie* cu numele Utilizator.

5.5 Lucrul cu fișiere externe

Limbajul PHP permite, cu ajutorul unor funcții predefinite, realizarea de operații asupra unui fișier precum deschidere, citire, scriere, închidere, creare sau încărcare.

Pentru încărcarea unui fișier cu ajutorul unui formular sunt necesare două pagini HTML distincte: pagina care conține formularul și pagina care preia fișierul (într-un director temporar și cu un nume temporar) și îl salvează apoi într-un director specificat de pe server.

În cadrul formularului se va folosi un tip special de buton (`file`) care trebuie să aibă setat atributul `name`, astfel:

```
<input type = "file" name="fisier">
```

În plus, marcajul `<form>` trebuie să cuprindă trei atribute:

- `action="pagina.php"` care trebuie să conțină numele paginii care va prelua fișierul;
- `method="post"` care specifică metoda de preluare a fișierului;

- `enctype="multipart/form-data"` care specifică modalitatea de codare a datelor transmise din formular.

Să se adauge în pagina acasa.php următorul formular:

```
<br>
<form action="incarcare.php" method="post" enctype="multipart/form-
data">
   Selectati fisierul pentru incarcare:<br>
   <input type="file" name="fisier"><br>
   <input type="submit" value="Trimite fisier">
</form>
```

Pagina HTML care preia fișierul și îl salvează într-un director specificat de pe server va conține un script PHP ce utilizează funcția `move_uploaded_file(`*`cale_fisier_temporar`*`, `*`cale_fisier_final`*`)`.

Calea către fișierul temporar se află cu ajutorul variabilei super-globale `$_FILES` astfel:

```
$_FILES["fisier"]["tmp_name"]
```

Exemplu: `tmp/phpD44F.tmp`

Valoarea `"fisier"` trebuie să fie identică cu valoarea atributului `name` specificată în formular pentru butonul de tip `file`.

Calea către fișierul final se stabilește tot cu ajutorul variabilei super-globale `$_FILES` prin concatenarea dintre numele directorului final (sau calea completă către acesta) și numele original al fișierului, astfel:

```
"director_final/".$_FILES["fisier"]["name"]
```

Exemplu: `fisiere/document.txt`

În directorul TPI să se creeze un director nou cu numele *fisiere*.
Să se creeze într-o locație separată din calculator un fișier de tip .txt care să aibă nume și un conținut la alegere.
Să se creeze o pagină HTML nouă în care să se introducă următorul script:

```
<?php
   $cale_fisier_temporar = $_FILES["fisier"]["tmp_name"];
   $cale_fisier_final = "fisiere/".$_FILES["fisier"]["name"];
```

```
    move_uploaded_file($cale_fisier_temporar, $cale_fisier_final);
    echo "Fisier incarcat cu succes."."<br>";
?>
```

Să se salveze pagina cu numele incarcare.php în directorul C:\wamp64\www\TPI.
Se va încărca fișierul .txt în formularul din pagina acasa.php și se va apăsa butonul *Trimite fișier*. Se verifică dacă fișierul a fost salvat în directorul C:\wamp64\www\TPI\fisiere.

Citirea unui fișier salvat pe server se poate face în mod direct utilizând funcția `readfile()` însă aceasta nu oferă alte opțiuni.

O metodă mai bună este utilizarea funcției `fread()`.

Pentru a citi un fișier acesta trebuie mai întâi deschis folosind funcția `fopen()`. Aceasta necesită două argumente: calea către fișier și modul în care va fi deschis fișierul: doar citire (`"r"`), doar scriere (`"w"` sau `"a"`) sau citire + scriere (`"r+"` sau `"w+"` sau `"a+"`), cu ștergerea conținutului sau cu păstrarea acestuia și scrierea de la începutul sau la sfârșitul fișierului, etc. Sintaxa pentru deschiderea fișierului doar pentru citire (de exemplu) de la începutul acestuia este următoarea:

```
$fisier_deschis = fopen(cale_fisier, "r")
```

După deschidere se utilizează funcția `fread()` ce returnează ca resursă fișierul deschis. Funcția necesită două argumente: primul este fișierul deschis, iar al doilea numărul de octeți care să fie citiți din fișier. Dacă se dorește citirea întregului fișier se folosește rezultatul aplicării funcției `filesize(cale_fisier)`.

```
fread($fisier_deschis, filesize(cale_fisier))
```

După citire fișierul trebuie închis utilizând funcția `fclose()` care necesită un singur argument, fișierul deschis.

```
fclose($fisier_deschis)
```

În pagina cu numele incarcare.php să se introducă în scriptul PHP existent următoarele linii de cod:

```
$fisier = fopen($cale_fisier_final, "r");
echo "Continutul fisierului este: ";
echo fread($fisier, filesize($cale_fisier_final));
fclose($fisier);
```

Se va încărca fișierul .txt în formularul din pagina acasa.php și se va apăsa butonul *Trimite fișier*. Se vizualizează rezultatul.

Pentru a scrie într-un fișier acesta trebuie mai întâi deschis folosind funcția `fopen()` și un mod de deschidere care să permită scrierea.

După deschidere se utilizează funcția `fwrite()`. Aceasta necesită două argumente: fișierul deschis și șirul de caractere care trebuie scris, astfel:

```
fwrite($fisier_deschis, "sir de caractere")
```

Sfat: pentru a introduce un șir de caractere pe un rând nou, acesta se poate concatena cu o constantă predefinită numită `PHP_EOL` (End Of Line) .

După scriere fișierul trebuie închis utilizând funcția `fclose()`.

Să se scrie un text la alegere în fișierul .txt creat anterior.

Capitolul 6. Utilizarea bazelor de date – MySQL

Capitolul 7. Realizarea unui site interactiv