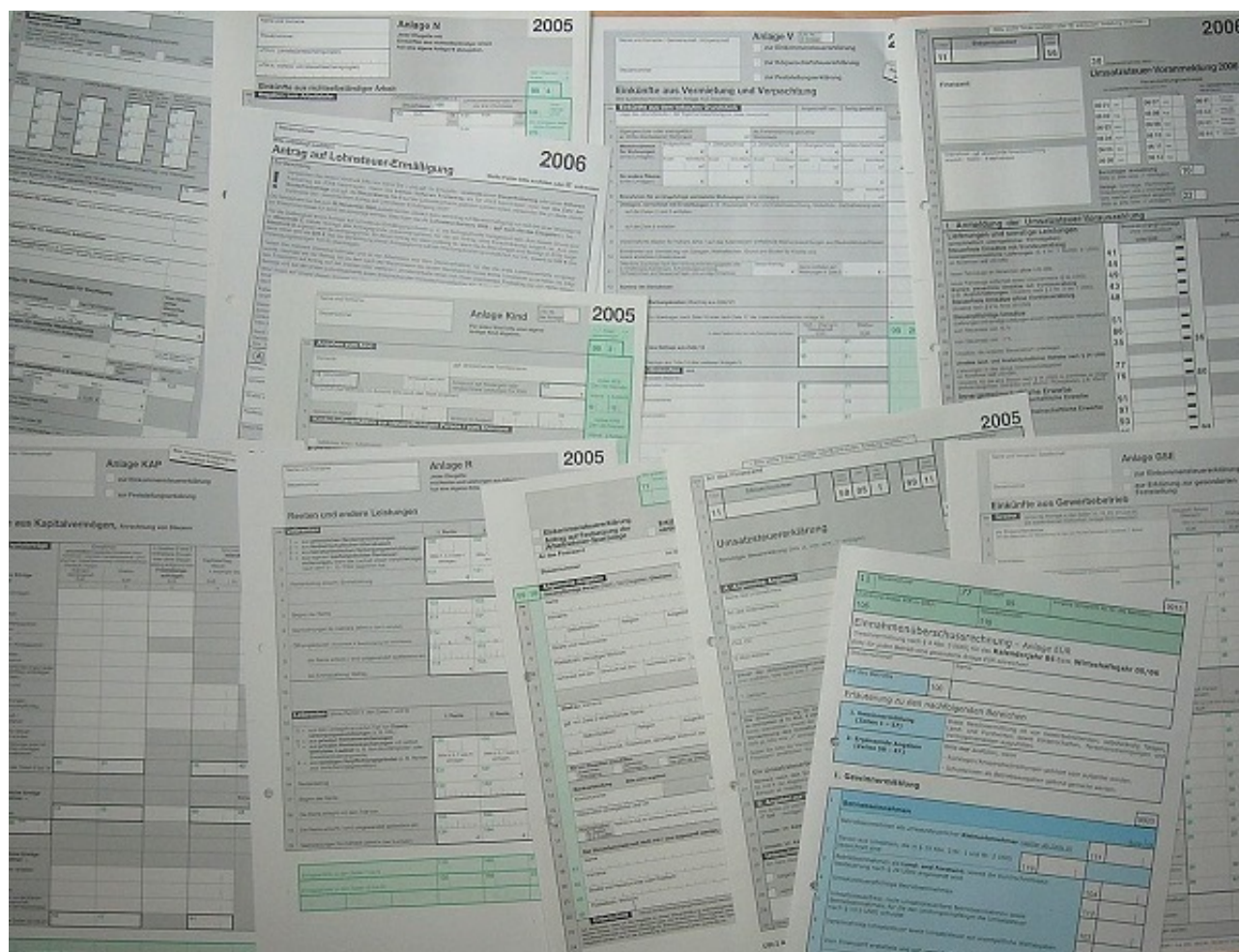


În cadrul acestei lecții, ne vom familiariza cu [formularele](#) HTML și vom face o comparație între formularele offline și cele online.

Noțiunea de formulare HTML

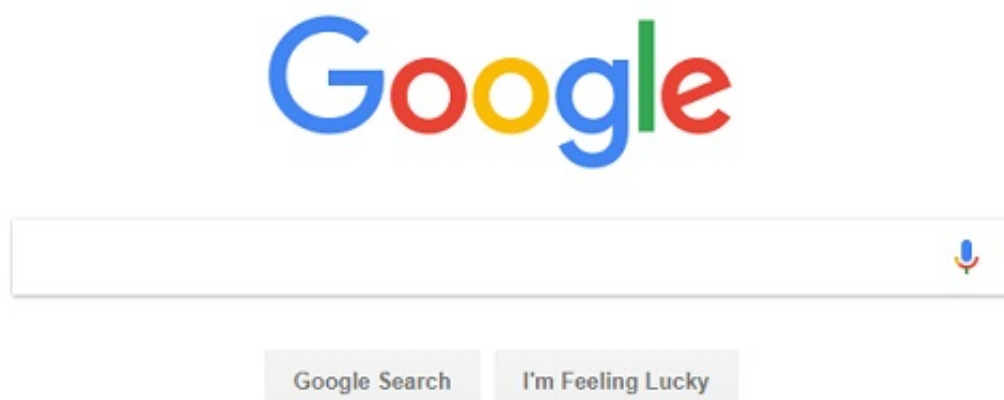
Formularele (engl. *forms*) reprezintă un instrument excelent pentru colectarea informațiilor de la vizitatorii unui site web. Formularele permit utilizatorilor să trimită comentarii și întrebări, să caute o anumită informație, să se înscrie pentru newsletter, să completeze aplicația online sau să introducă informații care înlesnesc cumpărarea unui produs. În cadrul acestei lecții, ne vom familiariza cu formularele și cu modul de introducere a acestora în pagină.

Inițial, termenul de formular era utilizat pentru documentul tipărit, care conținea câmpuri (spații goale) pentru introducerea datelor. Internetul a preluat acest concept și a modificat formularele ca să funcționeze în mod digital.



Imaginea 9.1. Exemplu de formulare clasice¹

Probabil cel mai cunoscut formular de pe web este cel de pe pagina de start a lui Google:

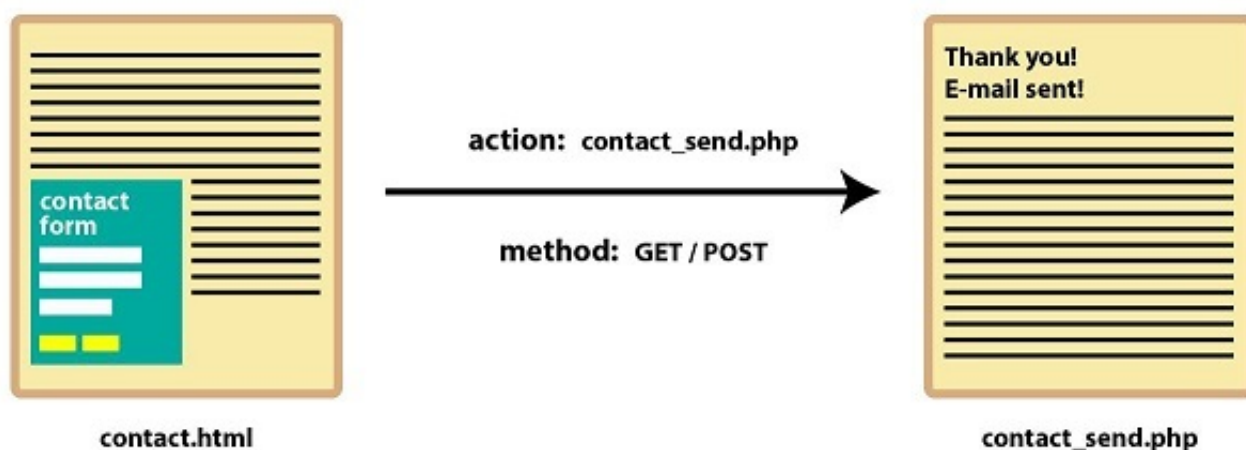


Imaginea 9.2. Formularul de pe pagina de start a lui Google

Formularul din imagine este cunoscut tuturor, deși este foarte simplu. Cu ajutorul acestui formular, introducem textul (cuvintele de căutare) într-un câmp text input, iar apoi, dând clic pe butonul submit (etichetat ca *Google Search*), transmitem datele.

Bazele formularelor

Formularele în sine nu au nicio funcționalitate, ci reprezintă doar un tip de șablon pentru colectarea informațiilor. Când utilizatorul completează un formular de pe un site, datele colectate sunt transmise până la script-urile de pe o altă pagină, care va finaliza activitatea de colectare a datelor.



Imaginea 9.3. Conceptul de funcționare a formularului

Să exemplificăm toate acestea. Pagina `contact.html` conține un formular în care utilizatorul introduce datele și un mesaj. După ce utilizatorul trimite formularul, acesta apelează pagina `contact_send.php`, pe care se află un script, care procesează datele colectate și le trimite prin e-mail la adresa indicată. Acesta este doar un exemplu, script-ul poate trimite e-mailuri, poate scrie/citi datele din bază sau poate face alte lucruri definite.

În acest curs, nu ne vom ocupa cu script-uri concrete. În general, acestea se scriu și se inserează prin limbajul PHP, care nu intră în tematica acestui curs, ci în cea a programării web. Noi ne vom ocupa de partea de HTML și CSS, care ne vor ajuta să inserăm și să stilizăm formulare.

Pe de altă parte, chiar dacă nu cunoașteți limbajul PHP, pe internet se pot găsi diferite formulare PHP finalizate deja, care trebuie doar încorporate în HTML.

Tag-ul *form*

Baza formularului web este tag-ul *form*. Acesta este doar un simplu element HTML cu attribute și subelemente caracteristice. Astfel, în el introducem tot ceea ce vrem să trimitem serverului prin formularul respectiv. Fiecare formular (tag *form*) recunoaște evenimentul **submit**. Acesta activează formularul și transferă valorile acestuia către server.

Să o luăm de la început. Tag-ul se scrie foarte simplu:

```
<form></form>
```

Totuși, acest lucru nu este suficient pentru început. Fiecare formular trebuie să aibă următoarele attribute:

- action,
- method,
- ID.

De asemenea, putem adăuga name, enctype și target. Deși attributele menționate nu sunt obligatorii în HTML5 (Action este obligatoriu în versiunile anterioare), fără acestea formularul nu ar fi avut sens.

Action

Fiecare formular conține atributul **action** care indică script-urile la care vor fi transferate datele. În exemplul de mai sus, era vorba de pagina *contact_send.php*. Observăm că pagina-țintă este de tip .php. Acest lucru este frecvent (.php, .asp, .jsp) deoarece procesarea parametrilor de server se poate efectua doar din script-urile de pe server. În cazul în care nu vrem să transmitem parametrii către server, nu avem nevoie de formular.

Dacă omitem acest atribut, browserul va subînțelege că aceeași pagină care conține formularul va procesa și datele.

Method

Atributul **method** definește modul în care vor fi transmise datele. Există două opțiuni:

- **metoda GET** - Datele din formular sunt trimise serverului printr-un URL. Informațiile transmise în acest mod sunt transparente și supuse atacurilor din partea hackerilor. Deoarece un URL poate avea cel mult 8 192 de caractere, această metodă nu este adecvată pentru formulare mai lungi. De asemenea, putem ajunge la transliterație sau transcripție, iar unele caractere se pot modifica sau pierde pe parcurs.
- **metoda POST** - Această metodă împachetează datele formularului în cadrul cererii HTTP. Datele nu sunt codificate și de aceea (deși sunt mai sigure decât în cazul metodei GET) sunt supuse atacurilor din partea hackerilor. Așadar, în cazul în care colectăm informații personale, precum nume de utilizator, parole sau numere de carduri de credit, trebuie să asigurăm o conexiune mai sigură până la un server sigur.

Name și ID

ID-ul se folosește pentru determinarea unică a elementului HTML pe o pagină în *Document Object Model* (prin JavaScript sau pentru stilizare prin CSS). ID-ul trebuie să fie unic.

Name determină numele formularului. Este transmis serverului.

Ceea ce este important este că recomandarea va introduce, încă de la început, ambele atribute. Acestea pot avea și aceeași valoare, ceea ce se și recomandă.

Enctype

Atributul *enctype* determină cum vor fi codate (*encoded*) datele în momentul trimerii către server.

Valoarea implicită a acestui atribut este **application/x-www-form-urlencoded**. Nu trebuie să-l scriem sau să avem grijă de el. Ceea ce ne interesează este ca, în cazul în care vrem să [încărcăm fișiere](#), *enctype* trebuie să aibă valoarea: **multipart/form-data**.

Target

Acest atribut este același ca și în cazul linkurilor. Am analizat detaliile în lecția 6. Totul este identic, cu excepția faptului că în cazul linkurilor se deschide o pagină, iar la formular se deschide pagina/script-ul din atributul *action*.

Dacă îl folosim pe formular, de obicei utilizăm valoarea *_blank*, deoarece astfel îi transmitem formularului că trebuie să deschidă script-ul din atributul *action* într-un nou tab sau într-o nouă fereastră (în funcție de browser).

Inserarea unui formular

Având în vedere toate cele menționate mai sus, codul formularului nostru poate avea aspectul următor:

```
<form action="script" method="post" name="formular_exemplu" id="formular_exemplu">  
  
</form>
```

Astfel, am completat elementul de bază *form*. Dacă vrem să încărcăm fișiere prin formular și/sau să deschidem formularul trimis într-un nou tab, vom introduce și celelalte două atribute:

```
<form action="script" method="post" name="formular_exemplu"  
id="formular_exemplu" enctype="multipart/form-data" target="_blank">
```

</form>

Deși am introdus toate aceste elemente, avem doar cadrul gol al formularului, care nu înseamnă nimic. Trebuie să introducem și controalele formularului, care vor colecta datele.

Controalele formularului

Controalele din formular sunt responsabile pentru colectarea și trimiterea datelor. Acestea pot fi butoane de tip text, radio sau de bifare etc. În continuarea lecției, ne vom familiariza cu acestea.

[Input \(text\)](#)

Tag-ul pe care îl întâlnim cel mai des în formulare este **<input>**. Acesta este un tag cu autoînchidere, deoarece toți parametrii lui sunt setați prin atribute. Același tag se folosește pentru mai multe controale diferite, însă este determinat de atributul **type** obligatoriu. Prin urmare, dacă vrem să introducem un câmp text simplu, vom introduce tag-ul *input* cu atributul *type="text"*:

```
<input type="text" name="color" id="color">
```

În exemplul codului, se poate vedea că am scris atributul *type*, precum și atributul *name*, care este important pentru script-ul care primește datele. Prin atributul *name*, script-ul va ști cum să clasifice data. De asemenea, am adăugat și o valoare ID, deși nu este obligatorie.

Mai putem adăuga și atributul *maxlength*, cu care determinăm numărul maxim de caractere. De exemplu, dacă vrem să limităm numărul posibil de caractere la 4, vom adăuga *maxlength="4"*. De asemenea, mai devreme s-a folosit și atributul *size*, cu care am

determinat lățimea câmpului. Acum, pentru asta folosim regulile CSS.

Codul introdus anterior ar fi afișat acest detaliu pe pagina browserului:

color

Imaginea 9.4. Câmpul Input text afișat în browser

Mai exact, codul care ar fi afișat câmpul de mai sus ar fi fost următorul:

```
<label for="color">color</label>  
<input type="text" name="color" id="color">
```

Fără tag-ul *label* adăugat, s-ar fi afișat doar câmpul fără textul din stânga lui. Tag-ul *label* nu trebuie să se afle imediat înainte de tag-ul *input*, însă, de cele mai multe ori îl găsim în această poziție. Important este faptul că *label* posedă atributul *for*, care îl leagă de un anumit *input*. **Atributul *for* al tag-ului *label* trebuie să fie același ca și atributul ID al tag-ului *input* (sau al unui alt tag).**

Ambele tag-uri, și *input*, și *label*, sunt, conform naturii lor, tag-uri *inline*.

Input (password)

Tag-ul *input* se folosește tot pentru introducerea parolei (*password*). În acest caz, atributul *type* este adăugat pe *password*. Toate celelalte attribute sunt la fel ca și în cazul textului *input*.

```
<input type="password" name="pass" id="pass">
```

În browser, s-ar fi afișat următoarele:



Imaginea 9.5. Câmpul pentru inserarea parolei

Observați că textul nu este afișat lângă control, deoarece nu am adăugat tag-ul label. Totuși, putem face acest lucru foarte simplu, dacă adăugăm tag-ul label cu atributul `for="pass"`.

Acest tip de câmp *input* ascunde caracterele introduse, însă nu garantează siguranța în timpul trimiterii datelor. Pentru aceasta, avem nevoie de un server configurat corect și de script-uri scrise corect.

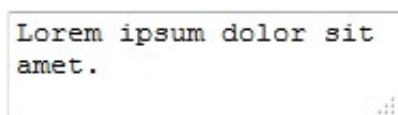
Textarea

Acest tag creează un câmp textual ceva mai mare, care poate susține și mai multe rânduri de text. Spre deosebire de câmpul *input*, *textarea* nu este un tag cu autoînchidere, așadar necesită un început și un sfârșit. Între acestea, se introduce textul care se va afișa în control după încărcarea paginii. Dacă utilizatorul nu îl șterge, acesta va fi transmis împreună cu datele.

Haideți să vedem un exemplu:

```
<textarea name="description" id="description">Lorem ipsum dolor sit amet.</textarea>
```

În browser, s-ar fi afișat:



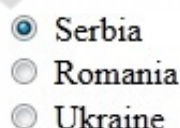
Imaginea 9.6. Câmpul Textarea în browser

În acest tip de control, nu introducem atributul `type`, deoarece *textarea* își stabilește singur rolul. În cod, vedem textul introdus între tag-ul inițial și cel final, care se afișează în câmp după încărcarea paginii.

Poate veți observa pe unele pagini mai vechi attributele *cols* și *rows*, în care sunt introduse valori numerice fără unități. Acestea s-au utilizat în trecut pentru a determina lățimea (*cols*) și înălțimea (*rows*) acestui control. La fel ca și la *input*, astăzi efectuăm acest lucru utilizând CSS și nu mai avem nevoie de ele.

[Radio](#) buttons (Butoanele radio)

Butonul radio este un control destul de diferit din punct de vedere vizual față de controalele menționate mai devreme, însă și acesta se bazează pe tag-ul *input*. Este puțin mai complicat de scris, dar haideți să vedem un exemplu în browser:

- 
- ☒ Serbia
 - ☐ Romania
 - ☐ Ukraine

Imaginea 9.7. Controale radio în browser

Cu siguranță, ați întâlnit controale radio în formulare. Ceea ce le caracterizează este faptul că nu trebuie să introducem valori, ci doar să selectăm una dintre opțiunile oferite (putem selecta doar una singură).

Pentru fiecare selecție, trebuie să creăm un tag *input* **separat**. Pentru ca browserul să știe că există mai multe tag-uri *input* diferite (de tip radio) grupate împreună, trebuie să le acordăm **același nume** în **atributul *name***. De aceea, utilizatorului nu îi va fi permis să aleagă mai multe opțiuni. Dacă selectează una, toate celelalte se dezactivează.

Atributul *type* menționat mai devreme trebuie adăugat valorii *radio* în controlul radio.

Valoarea ID trebuie să fie diferită pentru fiecare *input*, indiferent dacă atributul *name* este identic.

Atributul **value** reprezintă o noutate și cu ajutorul lui transmitem valoarea pe care a selectat-o utilizatorul. În exemplele precedente, am trimis doar textul, respectiv parola, iar aici transmitem conținutul atributului *value* al butonului radio selectat.

Codul pentru acest exemplu ar arăta astfel:

```
<input name="country" type="radio" id="Serbia" value="Serbia" checked="checked">  
<label for="Serbia">Serbia</label>
```

```
<br />
```

```
<input name="country" type="radio" id="Romania" value="Romania">  
<label for="Romania">Romania</label>
```

```
<br />
```

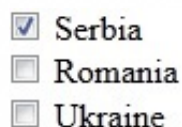
```
<input name="country" type="radio" id="Ukraine" value="Ukraine">  
<label for="Ukraine">Ukraine</label>
```

Dacă vrem ca, pe măsură ce pagina se încarcă, să fie indicată o valoare, vom adăuga atributul *checked="checked"* pe tag-ul ei input. Valoarea este întotdeauna aceeași. Putem adăuga acest atribut **doar pe un** input *radio* din grup.

Bineînțeles, am adăugat și tag-uri *label*, pentru a face deosebirea între butoanele radio. Aici se află și câteva tag-uri *br*, care separă rândurile, deoarece toate elementele de aici sunt *inline*.

Câmpurile [Checkbox](#)

Acest control este foarte similar cu controlul radio, însă se deosebește prin faptul că utilizatorul poate selecta mai multe valori în același grup. Să vedem exemplul din browser:



☒ Serbia
☒ Romania
☒ Ukraine

Imaginea 9.8. Controale Checkbox în browser

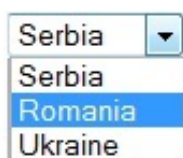
În timp ce codul ar fi următorul:

```
<input name="country" type="checkbox" id="Serbia" value="Serbia" checked="checked">  
<label for="Serbia">Serbia</label>  
<br />  
<input name="country" type="checkbox" id="Romania" value="Romania">  
<label for="Romania">Romania</label>  
<br />  
<input name="country" type="checkbox" id="Ukraine" value="Ukraine">  
<label for="Ukraine">Ukraine</label>
```

După cum putem vedea, totul este identic, doar atributul *type* este modificat. Astfel, am activat alegerea multiplă, unde valorile nu se anulează reciproc. Restul rămâne la fel ca în cazul butoanelor radio.

Select (drop down)

Acest control permite utilizatorului să selecteze o valoare din meniul *dropdown* (derulant).



Imaginea 9.9. Control dropdown activat în browser

Se bazează pe tag-ul ***select***, care conține deja bine cunoscutele attribute *name* și *id*. În cadrul tag-ului *select* menționat, se află mai multe tag-uri ***option***, în care introducem posibilele valori:

```
<select name="country" id="country">  
  <option value="Serbia">Serbia</option>  
  <option value="Romania" selected="selected">Romania</option>  
  <option value="Ukraine">Ukraine</option>  
</select>
```

Fiecare opțiune (*option*) este o valoare și ca atare trebuie să conțină atributul *value*, deoarece asta transmitem serverului, respectiv scripturilor.

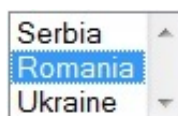
La fel ca și unele controale anterioare, putem alege valoarea care trebuie selectată în momentul încărcării paginii. În acest caz, putem adăuga atributul și valoarea *selected="selected"* la opțiunea dorită (în exemplu, a fost adăugată celei de-a doua opțiuni - "Romania").

De asemenea, putem adăuga și tag-ul *label* dacă vrem și îl putem lega de atributul *for* prin nume (*name*).

Elementul ***select***

Controlul *select* menționat poate fi schimbat dacă tag-ului *select* îi adăugăm atributul ***size***, iar ca valoare introducem un număr. Valoarea acestui număr determină înălțimea controlului și numărul de opțiuni care vor fi afișate. Pentru a înțelege mai bine, observăm exemplul de

mai jos:



Imaginea 9.10. Select cu atributul size 3

În acest exemplu, în cod am adăugat `size="3"`. Acum, nu mai este un meniu derulant, ci un câmp care afișează mai multe valori. Dacă numărul `size` este mai mic decât valorile disponibile, în partea dreaptă va apărea un *scroll bar*/bară de derulare (aici este dezactivată). Haideți să vedem următorul cod:

```
<select name="country" id="country" size="3">  
  <option value="Serbia">Serbia</option>  
  <option value="Romania" selected="selected">Romania</option>  
  <option value="Ukraine">Ukraine</option>  
</select>
```

De asemenea, putem să activăm și alegerea mai multor valori prin adăugarea atributului `multiple="multiple"`. Pentru ca selectarea multiplă să funcționeze, este necesar și un atribut `size` (într-un meniu derulant, nu poate exista selectare multiplă). Codul arată astfel:

```
<select name="country" id="country" size="3" multiple="multiple">  
  <option value="Serbia">Serbia</option>  
  <option value="Romania" selected="selected">Romania</option>  
  <option value="Ukraine">Ukraine</option>  
</select>
```

Astfel, am permis selectarea mai multor opțiuni, cu condiția ca

utilizatorul să țină apăsată tasta *Control* de pe tastatură (*Ctrl*), respectiv tasta *Command* (⌘) de pe tastatura calculatoarelor Mac, în timp ce efectuează selectarea multiplă. De aceea, trebuie să existe și o notă, o instrucțiune în acest sens, deoarece majoritatea utilizatorilor nu vor ști despre ce este vorba.

În plus, uneori browserele au dificultăți cu un astfel de control *select*. Testați-l obligatoriu în toate browserele.

Încărcarea fișierului

Controlul care asigură încărcarea fișierelor prin intermediul formularelor este din nou *input type*, însă cu tipul *file* (*type="file"*).

```
<input type="file" name="CV" id="CV">
```

În browser, se afișează următoarele:



Imaginea 9.11. Controlul pentru încărcarea fișierelor

Dacă folosim acest control, formularul trebuie să aibă metoda setată la POST, precum și un *enctype* corect (menționat mai devreme).

Acest control este specific prin faptul că nu permite stilizarea. Nu putem stiliza în niciun fel butonul, câmpul etc. cu regulile CSS, ci aceste detalii depind de browserul care afișează formularul.

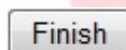
Butonul *submit*

Butonul submit, respectiv controlul submit, cum este mai corect, se folosește pentru a trimite formulare și date introduse unui server, unde

vor fi procesate. Din nou se folosește bine cunoscutul *input*, dar cu atributul *type* setat la **submit**.

```
<input type="submit" name="submit" id="submit" value="Finish">
```

Name și *ID* nu sunt obligatorii aici. **Value**, pe de altă parte, definește ceea ce va fi scris pe butonul respectiv. Dacă nu îl introducem, în unele browsere va apărea *Submit Query*.



Imaginea 9.12. Butonul submit din exemplul nostru

Dacă vrem, în loc de butonul obișnuit, putem utiliza o imagine. În acest caz, trebuie setat atributul *type="image"*. Atunci, pe atributul *input* sunt permise atributele *src*, *width*, *height* și *alt*, astfel încât să se comporte ca o imagine.

Controlul Button

Un control ceva mai nou este controlul *button*. La acest control, folosim tag-ul `<button> ... </button>`, care poate cuprinde mai multe elemente, de exemplu text și imagine.

```
<button>  
    
  Finish  
</button>
```

Câmpul Hidden

Deși ideea unui câmp ascuns în formular poate părea contradictorie, un astfel de control este posibil și se utilizează chiar foarte des. Câmpul ascuns, după cum îi spune numele, nu este vizibil utilizatorului, însă se poate utiliza pentru salvarea temporară a unor date. Când spunem că *nu este vizibil*, ne referim la faptul că nu este afișat direct în browser, însă este vizibil în codul-sursă (*source code*), așadar nu trebuie să postăm date sensibile.

Haideți să vedem exemplul din cod:

```
<input type="hidden" name="hiddenField" id="hiddenField" value="x">
```

Controlul se folosește, de obicei, pentru a scrie anumite date în el în timpul încărcării paginii și pentru a le trimite apoi script-urilor. De exemplu, imaginați-vă un formular pentru sugestii și comentarii de pe un site care vinde produse. Dacă utilizatorul este logat atunci când formularul este deschis, site-ul îi poate afișa ID-ul din baza de date în câmpul ascuns, iar utilizatorul nu trebuie să vadă acest lucru pe pagină. Când utilizatorul lasă un mesaj și trimite formularul, ID-ul este trimis și el, ceea ce poate facilita sortarea, monitorizarea și procesarea în continuare.

Fieldset și legend

Dacă vrem să încadrăm o parte din formular și să-i adăugăm un anumit titlu, putem face acest lucru cu ajutorul tag-ului `fieldset` cu care încadrăm toate controalele pe care le selectăm. Putem adăuga tag-ul `legend` imediat după ce deschidem tag-ul `fieldset`. În continuare, avem următorul exemplu:

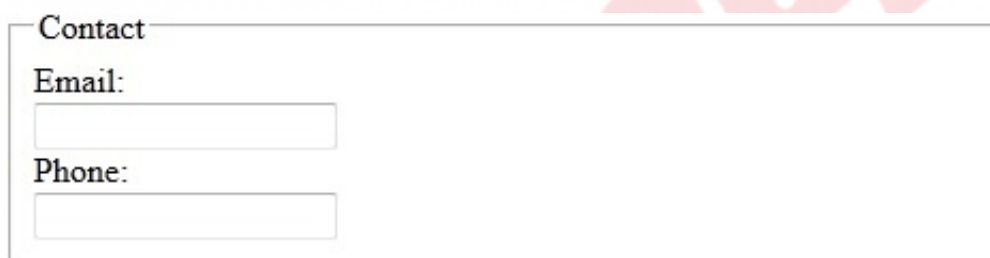
```
<fieldset>
<legend>Contact</legend>

  <label>Email:<br>
    <input type="text" name="email">
  </label><br>
```

```
<label>Phone:<br>  
  <input type="text" name="phone">  
</label>
```

```
</fieldset>
```

În browser, acesta ar arăta astfel:

A screenshot of a web browser displaying a form. The form is enclosed in a box with a title 'Contact' at the top left. Inside the box, there are two labels: 'Email:' and 'Phone:'. Each label is followed by a text input field. The 'Email:' label is positioned above the first input field, and the 'Phone:' label is positioned above the second input field. The input fields are empty.

Imaginea 9.13. Fieldset și legend într-un exemplu

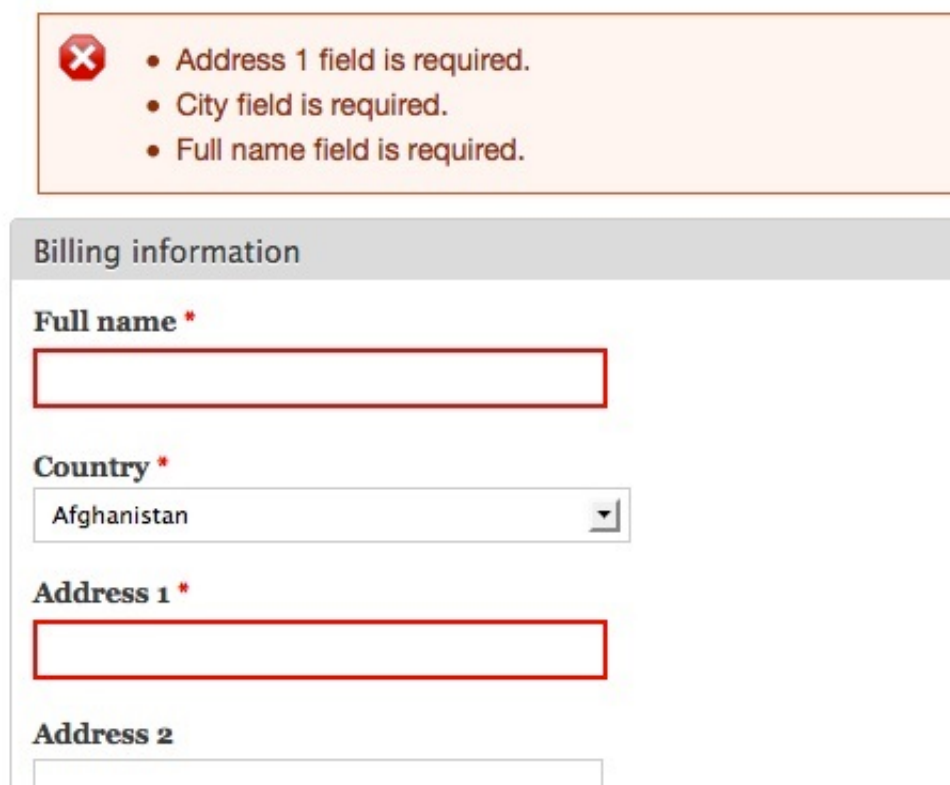
Fieldset este, în mod implicit, un element *block type*.

Dacă ați fi analizat cu atenție codul în acest exemplu, ați fi observat că tag-urile *label* sunt scrise **în jurul** controlului, în acest caz *input*. Acesta este un alt mod de a adăuga labels/etichete, deși vă recomandăm să le scrieți separat, utilizând atributul *for*.

Controalele HTML5

HTML5 și validarea

Cred că ați observat până acum că unele formulare de pe web includ validarea. Acesta este un principiu conform căruia formularul este verificat în sensul datelor introduse, a tipului de date etc. Acest lucru se realiza până acum cu ajutorul unui JavaScript applet, care se executa pe pagină. Astfel, se evita angajarea inutilă a serverului în cazul în care formularul nu era completat corect.



The image shows a web form titled "Billing information" with a light gray header. Below the header, there are four input fields: "Full name *", "Country *", "Address 1 *", and "Address 2". The "Full name *", "Address 1 *", and "Country *" fields are highlighted with a red border, indicating they are required. The "Country *" field is a dropdown menu with "Afghanistan" selected. Above the form, there is a red error message box with a red 'X' icon and the following text: "Address 1 field is required.", "City field is required.", and "Full name field is required."

Imaginea 9.14. Exemplu de formular cu validare²

Până acum, validarea era efectuată de JavaScript, însă, odată cu apariția limbajului HTML5, aceste opțiuni sunt încorporate în HTML.

```
<input type="password" name="password" required="required">
```

Prin linia anterioară de cod, am definit că acest câmp pentru inserarea parolei este parte obligatorie a formularului, ceea ce se obține cu atributul `required`.

Noi variante HTML5 de *input*

Versiunea HTML5 a adus și variante noi ale controlului *input*. Unele dintre acestea sunt:

- câmpul textual de căutare (search) – `type="search"`;

- câmpul pentru dată - `type="date"`;
- câmpul pentru URL - `type="url"`;
- câmpul pentru e-mail - `type="email"`.

Nu trebuie să le explicăm în detaliu, deoarece ele setează câmpul astfel încât acesta să fie pregătit pentru introducerea conținutului corespunzător. Afișarea și funcționalitatea le sunt influențate de capacitatea browserului de a le procesa. Folosiți-le cu atenție și testați-le.

HTML5 placeholder

O altă noutate este opțiunea `placeholder`, care asigură afișarea textului înainte de interacțiunea cu utilizatorul. De exemplu, în câmpul de căutare putem să introducem: „Search...” Acest cuvânt ar fi dispărut imediat după ce câmpul ar fi fost focalizat (când utilizatorul dă clic pe câmp și începe să introducă date).

Este suficient să adăugăm atributul *placeholder* cu cuvântul dorit sau cu o propoziție ca valoare. În continuare, avem și un exemplu:

```
<input type="search" name="search" placeholder="Search...">
```

Exercițiu:

Tema acestui exercițiu este crearea formularului prezentat în următoarea imagine:

Personal information:

Your full name:

E-mail Address:

You are:

Student ▾

Teacher

Student

Other

Reason you are contacting us:

☐ I would like detailed information

☐ I am interested in employment

☐ I would like to give you my feedback

☐ Other(see below)

Comments:

Send

Clear

Imaginea 9.15. Exemplu de exercițiu

Notă:

Prezentarea tabelului depinde de browserul pe care îl folosiți și de dimensiunea ferestrei, așadar, soluția voastră nu trebuie să fie identică.

Dacă aveți dificultăți în rezolvarea acestui exercițiu sau dacă vreți să verificați codul HTML, soluția exercițiului se poate găsi la următorul [link](#).

1. <http://en.wikipedia.org/wiki/File:Mantelb%C3%B6gen.JPG>
2. www.drupal.org

LINKgroup