

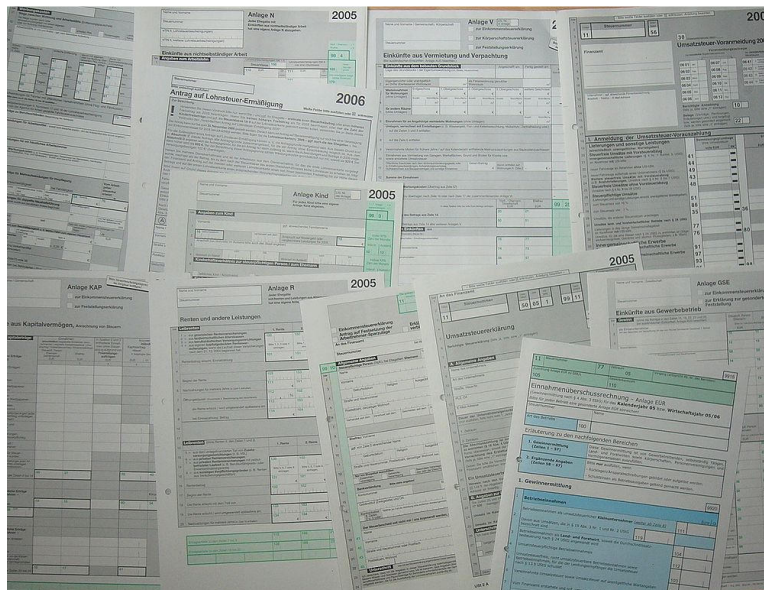
# Forme

U ovoj jedinici ćemo upoznati HTML forme i napraviti paralelu između offline formi (formulara) i online varijanti.

## Pojam HTML formi

Formulari, odnosno forme, predstavljaju odličan alat za sakupljanje informacija od posetilaca web sajta. Formulari dozvoljavaju korisnicima da pošalju komentare i pitanja, zatraže neku informaciju, prijave se za Newsletter, popune Online aplikaciju ili unesu informacije za plaćanje kako bi kupili neki proizvod. U ovoj lekciji ćemo se upoznati sa formularima i načinom unosa.

U originalu, termin forma, formular korišćen je za štampani dokument koji sadrži polja (prazan prostor) za upis podataka. Web je preuzeo taj koncept i prilagodio forme digitalnom funkcionisanju.



*Primer klasičnih formulara.*

Izvor: <http://en.wikipedia.org/wiki/File:Mantelb%C3%B6gen.JPG>

Verovatno najpoznatija forma na webu je na početnoj stranici Googlea:



Google Search

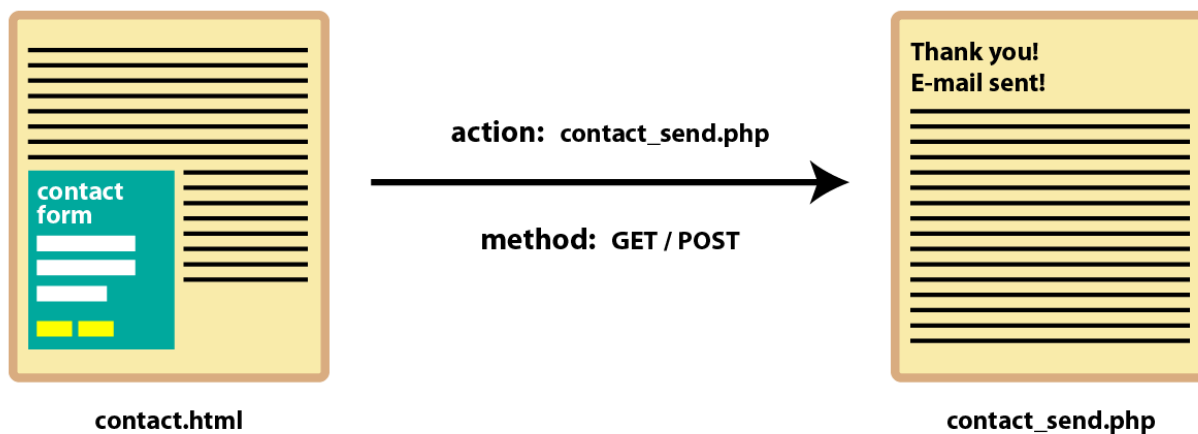
I'm Feeling Lucky

*Forma na početnoj strani Googla*

Ova forma na slici je svima dobro poznata, iako je vrlo jednostavna. Pomoću nje unosimo tekst (reči za pretragu) u jedno input tekstualno polje i zatim pomoću submit dugmeta (koje je označeno sa Google Search) prosleđujemo podatke.

## Osnovno o formularima

Sami po sebi formulari ne pružaju nikakvu funkcionalnost, već samo neku vrstu šablona za prikupljanje informacija. Kada korisnik popuni formular prikazan na sajtu, prikupljeni podaci se prosleđuju skripti na drugoj strani koja će obaviti sav posao.



*Koncept funkcionisanja formulara*

Videćemo na primeru. Strana kontakt.html sadrži formular na kome korisnik unosi svoje podatke i poruku. Pošto potvrdi formu, forma poziva kontakt\_slanje.php stranu, na kojoj se nalazi skripta i koja prikupljene podatke obrađuje i šalje e-mailom na podešenu adresu. Ovo je samo jedan od primera, skripta može poslati e-mail, upisati/pročitati podatke iz baze ili uraditi nešto treće što je definisano.

## Form tag

Osnova web formi je form tag. Ovo nije ništa drugo do običan HTML element, sa svojstvenim atributima i podelementima. Tako da sve što budemo želeli da bude prosleđeno serveru kroz tu formu stavljamo u njen sadržaj. Svaka forma (form tag) prepoznaje pojavu koja se naziva **submit**. Ova pojava/događaj dovodi do aktivacije forme i prosleđivanja njenih vrednosti serveru.

Krenućemo otpočetak. Form tag postavljamo jednostavno:

```
<form>    </form>
```

Ipak, to nije dovoljno za početak. Svaka forma mora imati attribute:

- Action
- Method
- Name
- ID

A možemo dodati i *enctype* i *target*.

### Action

Svaka forma sadrži **action** atribut koji definiše kome će se proslediti podaci, kojoj skripti. U našem primeru ranije je to bila strana *kontakt\_slanje.php*. Primećujemo da je ciljna strana tipa .php. To će obično biti slučaj (.php, .asp, .jsp), jer se obrada serverskih parametara može obraditi samo iz serverskog skripta. A u slučaju da ne želimo da prosleđujemo parametre serveru, sama forma nam nije ni potrebna.

Ukoliko se izostavi ovaj atribut, browser će podrazumevati da će ista strana na kojoj se nalazi forma obraditi podatke.

### Method

Atribut **method** je takođe obavezan i on definiše na koji način će podaci biti prosleđeni. Postoje dve opcije:

- **GET metod** - Podaci iz forme se ovom opcijom šalju na server kroz URL. Informacije koje se prosleđuju na ovaj način su transparentne i podložne napadima hakera. Pošto URL može da ima maksimum 8.192 karaktera, ovaj metod nije podesan za duže formulare. Takođe, može doći do transliteracije ili transkripcije i neki karakteri se mogu promeniti ili izgubiti.
- **POST metod** - Ovaj metod pakuje podatke formulara unutar HTTP zahteva. Podaci nisu šifrovani i stoga su (iako sigurniji nego u GET metodu) ipak podložni napadima hakera, tako da, ako sakupljamo personalne informacije kao što su korisnička imena, lozinke ili brojevi kreditnih kartica, moramo osigurati bezbednu konekciju do sigurnog servera.

## Name i ID

**ID** se koristi za unikatno određivanje HTML elementa na strani u *Document Object Modelu* (kroz Javascript ili za stilizaciju kroz CSS). ID mora biti jedinstven.

**Name** određuje ime forme. Prosleđuje se serveru.

Ono što je bitno je da je preporuka postavljati i jedan i drugi atribut u startu. Mogu imati istu vrednost, a to je i preporučeno.

## Enctype

Atribut enctype određuje kako će podaci biti enkodovani (encoded) prilikom slanja serveru. Podrazumevana vrednost ovog atributa je **application/x-www-form-urlencoded**. Ne moramo ga postavljati i brinuti o njemu. Ono što nas zanima je da u slučaju da želimo i upload fajlova, enctype mora imati vrednost: **multipart/form-data**.

## Target

Ovaj atribut je isti kao kod linkova. Njegove detalje smo razmatrali u 6. jedinici. Sve je istovetno, osim što se kod linkova otvara stranica, kod formi se otvara, tj. pokreće strana/skripta iz action atributa.

Ukoliko se koristi na formi, najčešće se koristi vrednost *\_blank* jer tako ukazujemo formi da otvori skriptu iz action atributa u novom tabu ili prozoru (zavisno od browsera).

## Postavljanje forme

Uzevši u obzir sve prethodno navedeno, kôd naše forme može izgledati:

```
<form action="skripta" method="post" name="forma_primer"
id="forma_primer">

</form>
```

Ovako smo kompletirali osnovni form element. Ukoliko želimo da uploadujemo fajlove kroz formu i/ili da se forma posle submita otvara u novom tabu, unećemo i preostala dva atributa:

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

</form>
```

Iako smo sve ovo uneli, ovo je samo prazan okvir forme, koji sam po sebi ne znači ništa. Moramo uneti form kontrole koje će prikupiti podatke.

## Kontrole forme

Kontrole na formi su zaduženi za prikupljanje i prosleđivanje podataka. Mogu biti tekstualnog tipa, radio ili check buttoni i tako dalje. Upoznaćemo se sa njima u nastavku.

### Input (text)

Tag koji možda najčešće srećemo u formama je **<input>**. Ovo je samozatvarajući tag jer se svi njegovi parametri podešavaju kroz atribute. Taj isti tag se koristi za više različitih kontrola, ali određuje ga obavezni atribut **type**. Dakle, ukoliko želimo da unesemo obično tekstualno polje unesemo *input* tag sa *type="text"* atributom:

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

U primeru kôda vidimo da smo postavili atribut *type*, ali i atribut *name* koji je bitan skripti koja podatke prima. Preko atributa *name* skripta će znati kako da klasifikuje podatak. Takođe, dodali smo ID vrednost, mada ona nije obavezna.

Ono što još možemo dodati od atributa je *maxlength* atribut kojim određujemo maksimalni broj karaktera. Na primer, ako želimo da ograničimo broj mogućih karaktera na 4, dodaćemo *maxlength="4"*. Pored toga, ranije se koristio i atribut *size* kojim se određivala širina polja. Sada za to koristimo CSS opise.

Prethodno unet kod bi prikazao ovaj detalj na stranici u browseru:

color

*Input tekst polje u browseru*

Tačnije, kôd koji bi to prikazao je sledeći:

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

Bez *label* taga koji je dodat, prikazalo bi se samo polje, bez tekstualnog dela levo od njega. Sam *label* tag ne mora biti neposredno pre *input* taga, ali se često tako nalazi. Ono što je bitno je da *label* poseduje *for* atribut kojim je vezan za određeni *input*. **For atribut input taga treba biti isti kao name atribut input (ili drugog) taga.**

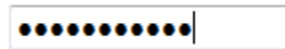
Oba taga, i *input* i *label*, su inline po prirodi.

### Input (password)

Input tag se takođe koristi za unos šifri (password). U tom slučaju atribut *type* postavljamo na *password*. Svi ostali atributi su isti kao za input text.

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

U browseru bi se prikazalo sledeće



*Polje za šifru*

Obratite pažnju na to da nije prikazan tekst pored kontrole jer nismo uneli *label* tag. Ipak, to možemo jednostavno učiniti ako dodamo *label* tag sa atributom *for="pass"*.

Ovo tip input polja sakriva karaktere koji se unose, ali ne garantuje sam po sebi sigurnost prilikom slanja podataka. Za to je potrebno pravilno konfigurisan server i ispravno pisanje skripti.

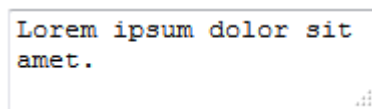
## Textarea

Ovaj tag kreira nešto veće polje za tekst koje pritom podržava više redova teksta. Za razliku od input polja, textarea nije samozatvarajući tag i zahteva početak i kraj. Između se unosi tekst koji će se prikazati u kontroli prilikom učitavanja strane. Ukoliko ga korisnik ne obriše, biće prosleđen zajedno sa podacima.

Pogledajmo primer:

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

U browseru bi se prikazalo:



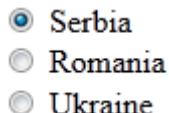
*Textarea polje u browseru*

U ovom tipu kontrole ne unosimo *type* atribut jer sam po sebi tag *textarea* određuje namenu. U kôdu vidimo tekst unet između početnog i krajnjeg taga koji se prikazuje u polju pošto se strana učitava.

Možda ćete na nekim starim stranama primetiti attribute *cols* i *rows* u kojima su unete brojčane vrednosti bez jedinica. Oni su se ranije koristili da odrede širinu (*cols*) i visinu (*rows*) ove kontrole. Kao i kod inputa, danas na to utičemo pomoću CSS-a i nema potrebe za njima.

## Radio dugmići

Radio button je kontrola koja je vizuelno prilično drugačija od ranije pomenutih, ali se i ona bazira na *input* tagu. Donekle je komplikovanija za postavljanje i pogledajmo prvo primer u browseru:



*Radio kontrole u browseru*

Radio kontrole ste sigurno susretali u formama i ono što je specifično je da ne unosimo vrednosti već samo pravimo izbor od ponuđenih opcija (samo jedna u grupi može biti označena).

Za svaki izbor moramo kreirati **poseban** *input* tag. Da bi browser znao da je više različitih *input* tagova (radio tipa) grupisano zajedno, moramo im dati u **name** atributu **isto ime**. Zbog toga, neće biti dozvoljeno da korisnik označi više izbora. Ukoliko selektuje jedno, sva ostala se isključuju u toj grupi.

U radio kontroli od ranije poznati *type* atribut mora biti postavljen na vrednost *radio*.

ID vrednost i dalje mora biti različita za svaki *input*, bez obzira što je *name* atribut isti.

**Value** atribut je novina i pomoću njega prosleđujemo vrednost koju je korisnik izabrao. U ranijim primerima prosleđivali smo sam tekst, odnosno šifru, a ovde prosleđujemo sadržaj *value* atributa izabranog radio polja.

Kôd za ovaj primer bi izgledao ovako:

```
<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 />

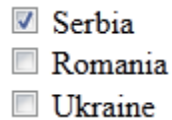
<input name="country" type="radio" id="Ukraine" value="Ukraine" />
<label for="Ukraine">Ukraine</label>
```

Ukoliko želimo da neka vrednost bude po učitavanju stranice označena, postavićemo kod nje, na njenom *input* tagu atribut *checked="checked"*. Uvek je ista vrednost. Možemo postaviti ovaj atribut **samo na jednom** radio inputu u grupi.

Naravno, dodali smo i *label* tagove radi razlikovanja radio polja. Tu je i par *br* tagova koji odvajaju redove jer su svi elementi ovde *inline*.

## Checkbox polja

Ova kontrola je vrlo slična *radio* kontroli, ali se razlikuje po tome što korisnik može označiti više vrednosti u istoj grupi. Pogledajmo primer u browseru:



*Checkbox kontrole u browseru*

Dok bi kôd bio:

```
<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>

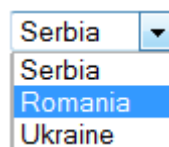
<br />

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

Kao što vidimo, sve je potpuno isto, samo je *type* atribut promenjen. Tako smo omogućili višestruki izbor, gde se vrednosti međusobno ne isključuju. Sve ostalo je isto kao za radio.

## Select (drop down)

Ova kontrola omogućuje korisniku da izabere jednu od vrednosti iz padajućeg menija (drop down).



*Aktivirana drop down kontrola u browseru*

Bazira se na ***select*** tagu, koji sadrži sada već dobro poznate name i id attribute. Unutar pomenutog *select* taga, nalazi se više ***option*** tagova u kojima postavljamo moguće vrednosti:

```
<select name="country" id="country">
```



```

<option value="Serbia">Serbia</option>
<option value="Romania" selected="selected">Romania</option>
<option value="Ukraine">Ukraine</option>

</select>

```

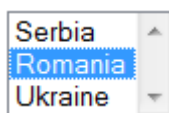
Svaki *option* je jedna vrednost i mora sadržati *value* atribut, jer je to ono što prosleđujemo serveru, odnosno skripti.

Poput nekih ranijih kontrola, možemo izabrati vrednost koja treba biti selektovana prilikom učitavanja stranice. U tom slučaju, možemo postaviti atribut i vrednost *selected="selected"* na željenu opciju (u primeru je postavljeno na drugom – "Romania").

Takođe, label tag možemo dodati ako želimo i povezati *for* atributom preko imena (*name*).

## Select dodatak

Pomenutu *select* kontrolu možemo promeniti ukoliko samom *select* tagu dodamo atribut **size** i kao njegovu vrednost unesemo broj. Vrednost tog broja određuje visinu kontrole i uvek će toliko opcija biti prikazano. Da bi bilo jasnije, pogledajmo na primeru:



Select sa size atributom 3

U ovom primeru smo u kôdu dodali *size="3"*. Više se ne pojavljuje padajući meni, već samo polje sa prikazom više vrednosti. Ukoliko je broj *size* manji od dostupnih vrednosti, pojaviće se scroll bar sa desne strane (ovde je disabled). Pogledajmo i kôd:

```

<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>

```

Pored toga, možemo omogućiti i izbor više vrednosti dodavanjem atributa *multiple="multiple"*. Da bi višestruki izbor funkcionisao, potreban je i *size* atribut (ne može postojati višestruki izbor u padajućem meniju). Kod bi izgledao:

```

<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>

```

```
</select>
```

Ovako smo omogućili da se više opcija selektuje, ali problem je to što korisnik mora držati *Control* taster na PC (Ctrl), odnosno *Command* taster na Mac računaru (⌘) dok pravi višestruki izbor. Zbog toga mora postojati i napomena, uputstvo o tome, jer većina korisnika neće shvatiti o čemu se radi.

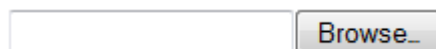
Pored ovoga, browseri ponekad imaju problema sa ovakvom *select* kontrolom. Obavezno testirajte u svim browserima.

## Upload fajla

Kontrola koja omogućava upload fajlova kroz formu je opet tipa *input*, ali sa tipom *file* (*type="file"*).

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

Dok bi u browseru bilo prikazano:



*Kontrola za upload fajlova*

Ukoliko koristimo ovu kontrolu, forma mora imati podešen *method* na POST, i pravilan *enctype* (pomenuto ranije).

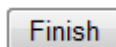
Ova kontrola je specifična po tome što ne dozvoljava stilizaciju. Ne možemo nikako stilizovati CSS opisima dugme, polje i ostalo, već ti detalji zavise od browsera koji formu prikazuje.

## Submit dugme

Submit dugme, odnosno submit kontrola kako je pravilnije, služi za prosleđivanje forme i unetih podataka serveru, gde će biti obrađeni. Opet se koristi dobro poznati *input* samo sa atributom *type* podešenim na (pogodili ste) ***submit***.

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

*Name* i *ID* nisu obavezni ovde. ***Value***, sa druge strane, definiše ono što će pisati na samom dugmetu. Ukoliko ga ne unesemo, na nekim browserima pišaće *Submit Query*.



*Submit dugme iz našeg primera*

Ukoliko želimo, možemo umesto običnog dugmeta koristiti sliku. U tom slučaju moramo postaviti atribut *type="image"*. Tada nad input atributom su dozvoljeni *src*, *width*, *height* i *alt* atributi, tj. ponaša se kao slika.

## Button

Nešto novija kontrola je *button*. Kod nje se postavlja tag `<button> ... </button>` koji može obuhvatiti više elemenata, na primer tekst i sliku.

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

## Hidden polje

Iako možda sama pomisao na sakriveno polje u formularu deluje kontradiktorno, ovakva kontrola je moguća i često je u upotrebi. Sakriveno polje, kao što mu ime kaže, nije vidljivo od strane korisnika, ali se može koristiti da sačuva privremeno neke podatke. Kada se kaže nije vidljivo, misli se na direktan prikaz u browseru, u prikazu koda stranice (*source*) je ipak dostupno, tako da ne treba postavljati osetljive podatke.

Pogledajmo primer u kodu:

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

Ova kontrola se najčešće koristi da se u nju prilikom učitavanja stranice upišu neki podaci, a zatim i oni proslede skripti. Na primer, zamislite formular za sugestije i komentare na nekom sajtu koji prodaje proizvode. Ukoliko je korisnik ulogovan prilikom pristupa formi, sajt može u sakriveno polje upisati njegov ID po kome se vodi u bazi podataka, a korisniku to nije bitno da bude vidljivo na strani. Kada ostavi poruku i submituje formu, proslediće se i taj ID, što može olakšati sortiranje, praćenje i uopšte dalju obradu.

## Fieldset i legend

Ukoliko želimo da deo formulara uokvirimo i da mu damo određeni naslov, to možemo učiniti pomoću *fieldset* taga kojim okružimo sve kontrole koje izaberemo. Pri tom, *legend* tag možemo postaviti odmah po otvaranju *fieldset* taga. Pogledajmo primer:

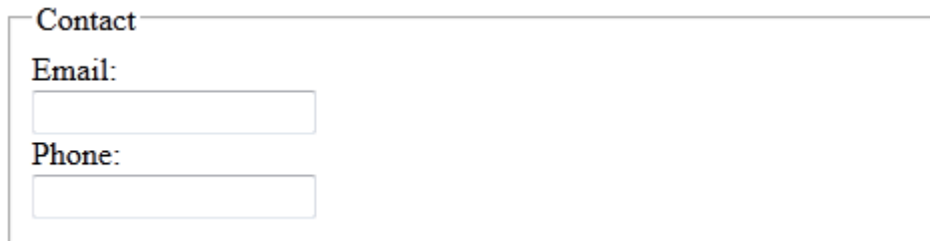
```
<fieldset>
<legend>Contact</legend>

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

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

`</fieldset>`

U browseru bi to izgledalo:



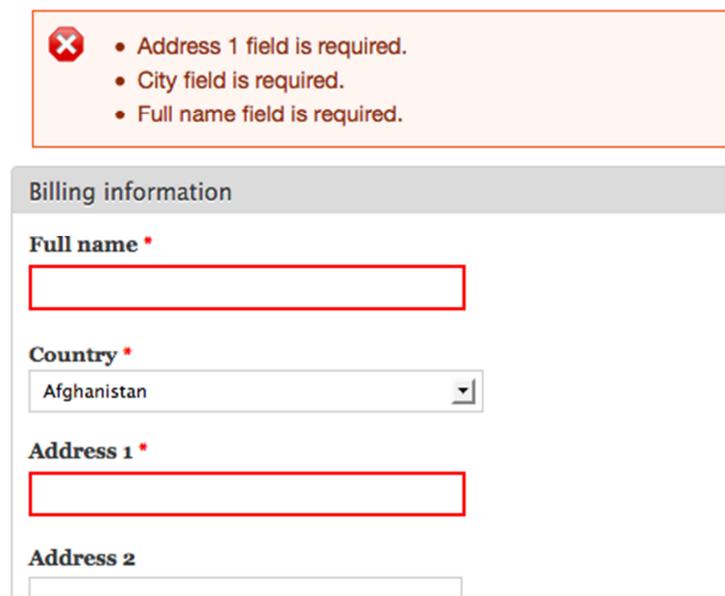
Fieldset je po default vrednostima *block* tip elementa.

Ukoliko ste pažljivije pogledali kôd u ovom primeru, primetili ste da su *label* tagovi postavljeni **oko** kontrola, u ovom slučaju *input*. To je drugi način da postavite labele, mada preporučujem da ih postavljate odvojeno i koristeći *for* atribut.

## HTML5 kontrole

### HTML5 i validacija

Sigurno ste do sada primetili da neke forme na webu sadrže validaciju. Validacija je princip u kome se forma proverava u smislu unetih podataka, tipova podataka i slično. Do sada se to vršilo pomoću nekog JavaScript apleta koji se izvršava na samoj strani. Time sprečavamo nepotrebno angažovanje servera ukoliko forma nije popunjena pravilno.



Primer forme sa validacijom. Izvor slike: [www.drupal.org](http://www.drupal.org)

Do sada je validacija uglavnom vršena JavaScriptom, ali sa pojavom HTML5 jezika, te mogućnosti su ugrađene u sam HTML.

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

U trenutku pisanja kursa, samo neki browseri podržavaju ovu opciju, ali će u budućnosti svakako biti zastupljenija.

Atribut možemo (po HTML5 pravilima) pisati i bez vrednosti, ali možda je bolje pisati dužu varijantu, kako je i primeru.

## Nove HTML5 varijante inputa

HTML verzije 5 je doneo i nove varijante *input* kontrole, a neke od njih su:

- Tekstualno polje za pretragu (search) - *type="search"*
- Polje za datum - *type="date"*
- Polje za URL - *type="url"*
- Polje za email - *type="email"*

Nema potrebe ih posebno objašnjavati jer podešavaju polje tako da se pripreme za unos odgovarajućeg sadržaja. Na sam prikaz i funkcionalnosti utiče to koliko je browser sposoban da ih obradi. Pažljivo ih koristite i testirajte.

## HTML5 placeholder

Još jedna novina je placeholder opcija koja omogućuje prikaz teksta pre korisnikove interakcije. Na primer, možemo u polju za pretragu upisati: „Search...“. Ta reč bi se sklonila čim polje dobije fokus (kad korisnik klikne na njega i krene sa unosom).

Dovoljno je da dodamo atribut *placeholder* sa željenom reči ili rečenicom kao vrednošću. Pogledajmo na primeru:

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

### Pitanje u jedinici

Izbacite uljeza (savet: onaj koji nije *input* tag)

- **textarea**
- text
- password
- radio button

## **Najvažnije iz lekcije**

- Forme služe za sakupljanje informacija od posetilaca sajta i prosleđivanje serveru na dalju obradu.
- Forme nemaju funkcionalnost same po sebi već zavise od drugih skripti.
- Svi elementi jedne forme moraju biti obuhvaćeni form tagom.
- Kontrole forme se moraju pažljivo postaviti i odrediti pomoću atributa.