

Forme

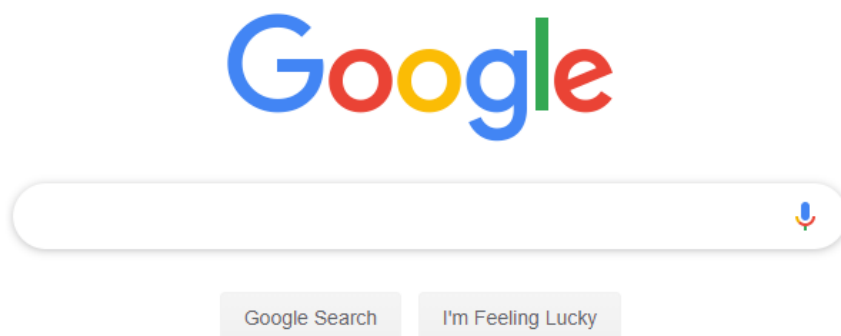
HTML dokument, sam po sebi, ne može mnogo toga da obezbedi korisniku, sem pukog pregleda informacija. Često postoji potreba da se određene informacije preuzmu od korisnika i proslede na dalju obradu bilo da je reč o obradi na serveru od strane neke serverske tehnologije ili obradu u lokalu korišćenjem klijentskih skripti. U takvim situacijama na scenu stupaju HTML forme.

Šta su HTML forme?

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.

Termin forma, odnosno formular, potiče od pojma koji je korišćen da označi štampani dokument koji sadrži polja (prazne prostore) za upis podataka. HTML je preuzeo taj koncept i prilagodio forme digitalnom funkcionisanju.

Verovatno se najpoznatija forma na webu nalazi na početnoj stranici sajta Google (slika 7.1).



Slika 7.1. HTML forma na sajtu Google

Forma na slici 7.1. dobro je poznata svima, a funkcioniše vrlo jednostavno. Pomoću nje se reči za pretragu unose u jedno tekstualno polje, a zatim se, pomoću dugmeta koje je na slici označeno kao *Google Search*, prosleđuju na dalju obradu.

Kako funkcionišu HTML forme?

Sama po sebi, HTML forma ne pruža nikakvu funkcionalnost, već samo neku vrstu šablona za prikupljanje informacija. Kada korisnik popuni formu unutar nekog HTML dokumenta, prikupljeni podaci se prosleđuju do skripte koja obavlja obradu prosleđenih podataka. Takav proces ilustrovan je slikom 7.2.



Slika 7.2. Prosleđivanje podataka forme na obradu

Na slici 7.2. prikazana su dva dokumenta. Dokument *contact.html* je HTML dokument koji sadrži HTML formu. Korisnik u takvu formu unosi podatke, nakon čega se oni prosleđuju dokumentu *contact_send.php*, koji sadrži kod za obradu prosleđenih podataka.

Forme i frontend razvoj

Forme su svojevrsan primer elementa koji se može koristiti za komunikaciju frontend i backend delova jedne web aplikacije. Iako se podaci koje je korisnik uneo unutar forme mogu obraditi i na klijentskom (frontend) delu aplikacije, najčešće se takav posao obavlja na serverskom delu od strane nekog od najpopularnijih backend jezika (PHP, Java, C#, Python...). Takva situacija je i na slici 7.2: podatke forme obrađuje logika napisana PHP jezikom.

Posao frontend programera uglavnom ne podrazumeva obradu podataka koje je korisnik uneo u formu. Frontend programer se brine o tome da forma izgleda onako kako je dizajnom predviđeno, da sadrži sve neophodne elemente i da korisniku na jednostavan i razumljiv način omogući unos traženih podataka. Vrhunac obrade formi za frontend programera ogleda se u validaciji podataka koje je korisnik uneo unutar forme. Bazična validacija može se obaviti korišćenjem HTML jezika, dok je za nešto napredniju validaciju podataka unetih u formu neophodno koristiti JavaScript jezik. Tako nešto, naravno, biće predmet kurseva koji slede.

Kako se kreiraju HTML forme?

HTML forma kreira se korišćenjem `form` elementa. `Form` element gradi se upotrebom otvarajućeg `<form>` i zatvarajućeg `</form>` taga, kao u sledećem primeru:

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

Element `form` može da sadrži attribute koji određuju način njegovog funkcionisanja. Attribute `form` elementa prikazani su tabelom 7.1.

Atribut	Vrednost(i)	Opis
action	URL	Definiše lokaciju na koju je potrebno proslediti podatke forme.
autocomplete	on/off	Definiše da li će forma imati autocomplete funkcionalnost, po kojoj će browser automatski popunjavati vrednosti polja na osnovu prethodnih korisničkih unosa; podrazumevana vrednost je on.
method	get post	Definiše HTTP metodu koja će se koristiti za prosleđivanje podataka forme.
enctype	application/x-www-form-urlencoded multipart/form-data text/plain	Definiše na koji način će podaci biti enkodovani prilikom prosleđivanja serveru; ova opcija ima efekta samo ukoliko se za slanje koristi metoda <u>POST</u> .
name		Definiše naziv forme
novalidate	novalidate	Definiše da podaci forme neće biti <u>validirani</u> prilikom prosleđivanja
target	_blank _self _parent _top	Definiše gde će se prikazati podaci koji se dobiju kao odgovor nakon prosleđivanja forme.

Tabela 7.1. Atributi form elementa

Od svih atributa prikazanih tabelom 7.1. najviše se koriste atributi `action` i `method`.

`Action` atribut definiše **kome** će podaci forme biti prosleđeni. To je uglavnom adresa do fajla sa serverskim skriptom, napisanom nekim od serverskih jezika. Takav podatak je moguće i izostaviti, a u toj situaciji će browser podrazumevati da se za obradu forme koristi ista ona strana na kojoj se forma nalazi.

Atribut `method` definiše način na koji će podaci biti prosleđeni i može imati dve vrednosti:

- **GET** - Prilikom korišćenja GET metode podaci forme prosleđuju se serveru kroz URL adresu, koja je sastavni deo zaglavlja HTTP zahteva (GET metoda ne poseduje telo, što je objašnjeno u uvodnom kursu ovog programa). S obzirom na to da je dužina URL-a ograničena na 8.192 karaktera, GET metoda nije podesna za prosleđivanje veće količine podataka. Takođe, prilikom prosleđivanja podataka forme GET metodom može doći do transliteracije ili transkripcije i neki karakteri mogu se promeniti ili izgubiti.
- **POST** - Kada se za prosleđivanje forme koristi POST metoda, podaci se smeštaju unutar tela HTTP zahteva. Kakav će njihov oblik biti prevashodno zavisi od vrednosti `enctype` atributa. Na primer, ukoliko se za vrednost atributa postavi `application/x-www-form-urlencoded`, podaci u telu POST zahteva biće formatirani na identičan način kao i prilikom slanja GET metodom. Razlikovaće se samo njihov položaj unutar strukture zahteva.

Kada koristiti GET, a kada POST metodu?

GET metodu najbolje je koristiti u situacijama koje ne zahtevaju kreiranje novih podataka. Na primer, ukoliko je na sajtu potrebno implementirati pretragu forma sa poljem za unos kriterijuma, podatke bi prosleđivala korišćenjem GET metode. Sa druge strane, forma za kreiranje korisničkog naloga na nekom sajtu podatke uglavnom prosleđuje korišćenjem POST metode.

Uzimajući u obzir opisane attribute koje form element može imati, može se napisati sledeći HTML kod:

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

U primeru je kreirana forma koja podatke prosleđuje skripti koja se nalazi unutar fajla *script.php*, kao metod slanja koristi se HTTP metoda POST, naziv id forme je *demo_form*, a rezultat koji se bude dobio od servera na zahtev upućen ovom formom biće prikazan na novoj stranici.

Elementi HTML forme

Prikazani form tagovi samo su kontejneri za elemente koji se mogu naći unutar HTML forme. Tako form element može da sadrži jedan ili više sledećih elemenata:

- <input>
- <textarea>
- <button>
- <select>
- <fieldset>
- <label>

Navedeni elementi zaduženi su za prikupljanje i prosleđivanje podataka. Svi oni će biti opisani u nastavku ove lekcije.

<input> element

Element input je najznačajniji element forme. Može se pojaviti u nekoliko različitih varijacija, i to u zavisnosti od vrednosti njegovog type atributa. Različite varijante input elementa prikazane su u tabeli 7.2.

Atribut	Opis
text	Element za unos teksta
password	Element za unos lozinki
radio	Element za selektovanje jedne od više ponuđenih opcija
submit	Element za prosleđivanje podataka forme
checkbox	Checkbox, kontrola koja može imati dva stanja: on i off
color	Kontrola za odabir boje (color picker)
datetime-local	Kontrola za odabir datuma i vremena
email	Element za unos email adrese
file	Dugme koje aktivira prozor za odabir fajla
hidden	Skriveni element
month	Element za odabir meseca i godine
number	Element za unos isključivo brojeva
range	Slider element za odabir bročane vrednosti
search	Element za unos teksta za pretragu
tel	Element za unos telefonskog broja
time	Kontrola za unos vremena
url	Element za unos URL adrese
week	Element za unos nedelje i godine

Tabela 7.2. Atributi form elementa

Napomena

Narandžastom bojom obeleženi su atributi koji su se pojavili u HTML5 jeziku.

Element `input` kreira se korišćenjem samozatvarajućeg taga, jer ne može posedovati nikakav sadržaj, već isključivo attribute. U nastavku će biti prikazane neke od najznačajnijih varijanti `input` HTML elemenata.

input (text)

Input element za unos jednostavnog teksta kreira se postavljanjem vrednosti `type` atributa na `text`, kao u sledećem primeru:

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

Pored atributa `type`, u primeru je navedena vrednost i za atribut `name`. To je veoma bitan podatak za skriptu koja će obraditi podatke ovakve forme, jer će na osnovu vrednosti `name` atributa podaci biti identifikovani.

U kombinaciji sa elementima forme za unos teksta, često se koristi i element `label`, čija uloga je da identifikuje polje za unos i obezbedi neku bližu informaciju o takvom polju. Kombinacija `label` i `input` kontrola prikazana je sledećim primerom:

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

Prikazani kod proizvodi efekat kao na slici 7.3.

Color:

Slika 7.3. Label i input elementi na stranici

Bitno je primetiti da `label` element poseduje atribut `for`, kojim se vezuje za određeni `input` element. Vrednost atributa `for`, `label` elementa mora se poklopiti sa vrednošću `id` atributa `input` elementa na koji se `label` element odnosi.

Element `input` poznaje atribut sa nazivom `placeholder`, kojim se može definisati kratak tekst koji će opisati vrednost koja se očekuje da bude uneta u tekstualnu kontrolu. Sledeći primer ilustruje upotrebu tog atributa:

```
<label for="color">Color: </label>
<input type="text" name="color" id="color" placeholder="enter color...">
```

Ovoga puta, prikazani kod će imati efekat kao na slici 7.4.

Color:

Slika 7.4. Input element sa placeholder vrednošću

input (password)

Kada je potrebno obezbediti korisniku unos lozinke, koristi se `input` element čija je vrednost `type` atributa `password`, baš kao u sledećem primeru:

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

Prikazani kod proizvodi efekat kao na slici 7.5.



Slika 7.5. Input kontrola za unos lozinke

Sa slike 7.5. može se videti da nema teksta pored kontrole, jer u kodu ne postoji `label` element. Sa slike se još može videti i to da ovaj tip `input` elementa sakriva karaktere koji se unose. Ipak, `input` element `password` tipa ni na koji način ne garantuje sigurnost prilikom slanja podataka lozinke, već samo sprečava da se lozinka fizički vidi unutar polja za unos.

input (radio)

HTML omogućava i kreiranje radio button kontrole, i to kao `input` elementa sa `radio` vrednošću `type` atributa. Korišćenjem Radio button kontrole korisnik ne unosi nikakve vrednosti, već bira jednu od unapred ponuđenih. I ovo je kontrola koja se najčešće koristi u kombinaciji sa `label` elementom, a HTML kod koji ilustruje kreiranje radio buttona je sledeći:

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

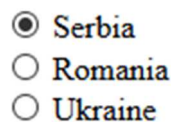
<br>

<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>
```

Prikazani kod proizvešće efekat kao na slici 7.6.



Slika 7.6. Grupa input kontrola tipa radio

Za kreiranje svake stavke, odnosno mogućeg izbora, koristi se zaseban `input` element sa tipom `radio`. Da bi browser znao da više različitih `input` tagova kreiraju jednu grupu, neophodno je postaviti identične vrednosti `name` atributa. U primeru je vrednost `name` atributa `county` i na taj način je moguće odabrati samo jednu državu od ponuđene tri. U protivnom, odnosno, da vrednosti `name` atributa nisu iste, svaki `radio button` ponašao bi se kao zasebna celina i ne bi bio postignut željeni efekat.

U primeru je za svaki `input` element definisana i vrednost `value` atributa. To je atribut pomoću koga se definiše vrednost koja će biti prosleđena na obradu.

Za identifikaciju `radio button` kontrola na strani koriste se `label` elementi na isti način na koji su korišćeni i sa `text input` elementima. Vrednost atributa `for` `label` elementa poklapa se sa vrednošću `id` atributa pripadajuće `radio button` kontrole.

Ukoliko je potrebno da neki `radio button` bude čekiran prilikom učitavanja stranice, koristi se boolean atribut `checked`. Na primer, ukoliko je potrebno da nakon učitavanja stranice čekirani `radio button` bude onaj koji ukazuje na državu sa nazivom *Romania*, dovoljno je uraditi sledeće:

```
<input name="country" type="radio" id="Romania" value="Romania" checked>
```

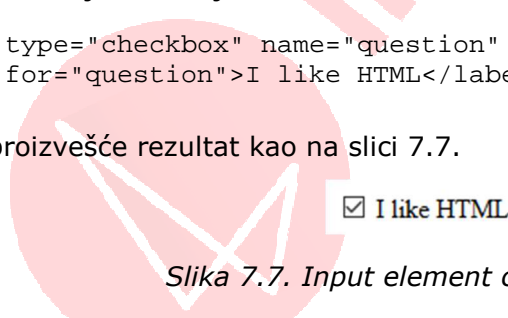
input (checkbox)

U prethodnom pasusu prikazani su HTML element za kreiranje `radio` kontrole i tehnike kojima se postiže kreiranje grupe takvih kontrola, iz kojih se može odabrati samo jedna. HTML poseduje i element koji dozvoljava kreiranje `checkbox` kontrole. To je kontrola koja može imati dva stanja: *on/off*, odnosno *true/false*, odnosno *čekirano/nečekirano*. Za razliku od `radio` kontrola, `checkbox` se uglavnom upotrebljava samostalno, kada je od korisnika potrebno dobiti jednostavan da/ne odgovor na pitanje.

Sledeći primer ilustruje kreiranje `checkbox` kontrole korišćenjem `input` elementa:

```
<input type="checkbox" name="question" value="Like" id="question">  
<label for="question">I like HTML</label>
```

Prikazani kod proizvešće rezultat kao na slici 7.7.



☒ I like HTML

Slika 7.7. Input element checkbox tipa

U prethodnom primeru, pored `input checkbox` elementa, definisan je i jedan `label` element koji prikazuje tekst korisniku. `Label` i `input checkbox` elementi međusobno su povezani identičnim vrednostima za `id` i `for` attribute. Na taj način dovoljno je kliknuti na površinu `label` elementa (na tekst labele) i klik će biti prosleđen `checkbox` elementu.

`Input checkbox` element poseduje atribut `value`, koji definiše vrednost koja će biti prosleđena zajedno sa formom u slučaju da je `checkbox` čekiran.

Baš kao i kod `input radio` elementa, i `checkbox` element može se na stranici pojaviti u čekiranom početnom stanju korišćenjem atributa `checked`. Sledeći kod to ilustruje:

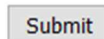
```
<input type="checkbox" name="question" value="Like" id="question"  
checked>
```

input (submit)

Input element može se pojaviti u još jednom obliku. To je oblik koji dozvoljava kreiranje kontrole za prosleđivanje podataka forme. Tako nešto postiže se postavljanjem vrednosti atributa `type` na `submit`, kao u sledećem primeru:

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

Atribut `value` definiše tekst koji će biti prikazan unutar dugmeta na strani. Tako je efekat koji proizvodi navedeni kod prikazan na slici 7.8.



Slika 7.8. Input element submit tipa

input (hidden)

Jedan od input elemenata jeste i onaj koji nema svoju vizuelnu reprezentaciju na stranici, a dobija se postavljanjem tipa na `hidden`. Često se naziva i skriveno polje. Iako možda pomisao na skriveno polje unutar forme deluje kontradiktorno, ovakva kontrola je moguća i često je u upotrebi. Skriveno polje, kao što mu naziv kaže, *nije vidljivo* za korisnika, ali se može koristiti da privremeno sačuva neke podatke. Kada se kaže da ovakvo polje nije vidljivo, prevashodno se misli na prikaz na stranici. Polje tipa `hidden` svakako je vidljivo u kodu strane, a što je još značajnije, vrednost polja `hidden` prosleđuje se zajedno sa svim ostalim vrednostima jedne forme.

Sledeći primer ilustruje kreiranje jednog skrivenog polja.

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

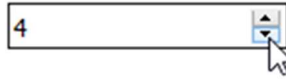
Skriveno polje se najčešće koristi da se u njega, prilikom učitavanja stranice, upišu neki podaci koji će biti prosleđeni skripti na obradu. Na primer, zamislite formular za sugestije i komentare na nekom sajtu koji prodaje proizvode. Ukoliko je korisnik ulogovan prilikom pristupa formi, sajt može u skriveno polje upisati njegov identifikacioni broj. Takav podatak za korisnika nema nikakvu važnost, pa je korišćenje `hidden` polja za čuvanje ovakve vrednosti idealan izbor. Na taj način, kada korisnik unese podatke i prosledi formu, prosleđuje se i njegov identifikacioni broj.

input (number)

HTML5 donosi dosta novih input elemenata, među kojima je i input element tipa `number`. Ovaj element omogućava unos brojčane vrednosti korišćenjem kontrole koja poseduje Up i Down tastere za odabir brojčane vrednosti (slika 7.9). Sledeći kod ilustruje način na koji se kreira input `number` element:

```
<input type="number" name="quantity" min="1" max="5">
```

Pored standardnih atributa, koji su već razmatrani u dosadašnjem toku kursa, input kontrola tipa `number` poznaje i dva karakteristična atributa: `min` i `max`. Njima se definiše opseg vrednosti koje se mogu odabrati korišćenjem ovakve kontrole. To se može videti i na slici 7.9, koja ilustruje efekat koji prikazani kod ima na strani.



Slika 7.9. Input element number tipa

Kao što je već rečeno, `input` element tipa `number` na stranici se renderuje kao kontrola kod koje se vrednost bira korišćenjem Up i Down tastera sa desne strane elementa.

input (color)

Još jedan `input` element koji predstavlja novinu u HTML5 jeziku jeste `input` element tipa `color`. On omogućava odabir boje korišćenjem ugrađenog color pickera operativnog sistema. Sledeći kod ilustruje kreiranje ovog elementa:

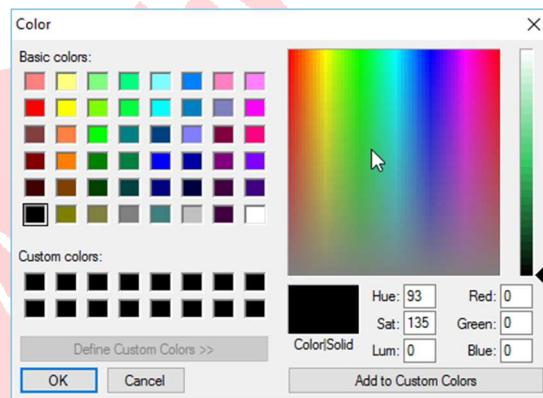
```
<input type="color" name="favcolor">
```

Na stranici se ovaj element renderuje kao na slici 7.10.



Slika 7.10. Input element color tipa

Klikom na dugme dobija se prozor za odabir boje, kao na slici 7.11.



Slika 7.11. Color picker koji se prikazuje aktiviranjem input color elementa

input (range)

HTML5 donosi i kontrolu koja omogućava odabir brojske vrednosti u definisanom opsegu koja se najčešće na stanici renderuje kao slider kontrola. Takva kontrola se kreira korišćenjem `input` elementa tipa `range`, kao u sledećem primeru:

```
<input type="range" name="points" min="0" max="10">
```

Efekat koji prikazani kod proizvodi je kao na slici 7.12.



Slika 7.12. Input element range tipa

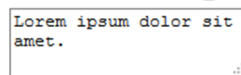
<textarea> element

Da bi se korisniku omogućio unos višelinijuskog teksta, koristi se element `textarea`. Za razliku od `input text` elementa, `textarea` se ne kreira korišćenjem samozatvarajućeg taga, te zahteva otvarajući i zatvarajući tag, između kojih je moguće uneti tekst koji će se prikazati u kontroli prilikom učitavanja strane. Ukoliko takav tekst korisnik ne obriše, biće prosleđen zajedno sa podacima.

Primer kreiranja `textarea` elementa je sledeći:

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

Efekat koji prikazani kod proizvodi je kao na slici 7.13.



Slika 7.13. Textarea element

<button> element

Nešto ranije u ovoj lekciji prikazan je element za kreiranje kontrole za prosleđivanje podataka forme. Naravno, reč je o `input` elementu tipa `submit`. HTML poznaje još jedan element slične namene. Reč je o `button` elementu.

Element `button` se koristi kako bi se na strani kreiralo dugme. Ipak, za razliku od `input submit` elementa, `button` element se ne kreira korišćenjem samozatvarajućeg taga, odnosno poseduje svoj početak i kraj. To omogućava ovom elementu da prihvati sadržaj proizvoljnog oblika i da se tako `button` kontrola kreira korišćenjem različitih elemenata. Upravo tako nešto ilustruje sledeći primer:

```
<button type="button">  
    
  Finish  
</button>
```

U primeru je prikazan jedan `button` element koji u sebi sadrži sliku i tekst. Na `button` elementu definisana je i vrednost `type` atributa. Kao vrednost je postavljen tekst *button*, što ovaj element proglašava običnim dugmetom, bez specijalne namene. Pored ove vrednosti, mogu se koristiti i vrednosti `submit` i `reset`, čime se postiže kreiranje dugmića specijalnih namena. Tip `submit` definiše da je reč o dugmetu koje će aktivirati prosleđivanje podataka forme, a tip `reset` da je reč o dugmetu koje će resetovati podatke forme na početne vrednosti.

<select> element

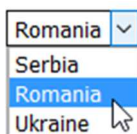
Nešto ranije prikazan je element koji omogućava korisniku odabir jedne od više ponuđenih opcija. To je bio `input` element tipa `radio`. HTML poznaje još jedan element sa istom namenom, ali u nešto drugačijem obliku. Reč je o elementu `select`.

`Select` element omogućava kreiranje drop down kontrole u HTML dokumentu. `Select` element se kreira korišćenjem `<select>` i `</select>` tagova, dok se stavke koje će biti ponuđene korisniku kreiraju korišćenjem `<option>` i `</option>` tagova.

Sledeći primer ilustruje kreiranje drop down kontrole.

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

Prikazani kod proizvodi efekat kao na slici 7.14.



Slika 7.14. Element `select`

Kao što je već rečeno, svaki `option` element unutar `select` elementa predstavlja jednu vrednost koju korisnik može da odabere. Svaki `option` element mora sadržati i `value` atribut, koji definiše vrednost stavke koja će biti prosledjena na obradu.

Slično kao i kod `input radio` elementa i kod `option` elementa moguće je odabrati stavku koja će biti selektovana kada se stranica učita. To se postiže korišćenjem atributa `selected`.

Napomena

Prilikom navođenja boolean atributa `selected` i nešto ranije atributa `checked`, može se primetiti izvesna razlika. Kada je u pasusu o `input radio` kontroli definisan `checked` atribut, to je učinjeno direktnim navođenjem njegove vrednosti. Sa druge strane, prilikom navođenja `selected` atributa, definisan je i njegov naziv i vrednost. O čemu je reč?

Zapravo, oba pristupa su ispravna i u prvom slučaju primenjena je tehnika koja se naziva minimizacija atributa. Tako su sledeće dve linije potpuno analogne:

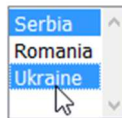
```
<option value="Romania" selected="selected">Romania</option>
i
<option value="Romania" selected>Romania</option>
```

Identično važi i za mnoge druge HTML attribute: `hidden`, `multiple`, `readonly`, `required`, `selected`...

Element `select` omogućava selekciju više vrednosti korišćenjem atributa `multiple`. Sledeći primer to ilustruje:

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

Efekat koji proizvodi prikazani kod ilustrovan je slikom 7.15.



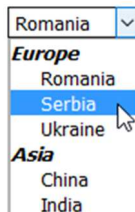
Slika 7.15. Select element sa multiple atributom

Na slici 7.15. vidi se da `select` element sa `multiple` atributom ima nešto drugačiji izgled na stranici, koji više odgovara list kontroli. Naravno, razlog tome jeste omogućavanje korisniku da selektuje više stavki. Selekcija više stavki se postiže pomoću Control tastera na PC-u i Command tastera na Mac računarima.

Još jedan HTML element koji je moguće koristiti unutar `select` elementa jeste element `optgroup`. Ovaj element omogućava grupisanje srodnih stavki unutar drop down kontrole. Primer upotrebe ovakvog elementa je sledeći:

```
<select>
  <optgroup label="Europe">
    <option value="romania">Romania</option>
    <option value="serbia">Serbia</option>
    <option value="ukraine">Ukraine</option>
  </optgroup>
  <optgroup label="Asia">
    <option value="chine">China</option>
    <option value="india">India</option>
  </optgroup>
</select>
```

Prikazani kod proizvodi efekat kao na slici 7.16.



Slika 7.16. Select element sa dve grupe kreirane korišćenjem `optgroup` elementa

Tag `<optgroup>` dozvoljava definisanje dva atributa. Naziv grupe definiše se atributom `label`, koji će biti prikazan u drop down listi, dok atribut `disabled` definiše da li će selektovanje stavki grupe biti onemogućeno.

Pitanje

Izbacite element koji nije input tag:

- a) **textarea**
- b) text
- c) password
- d) radio button

Objašnjenje

Unutar formi, polja za unos teksta i lozinki, ali i radio-dugmići za omogućavanje odabira, kreiraju se korišćenjem input samozatvarajućih tagova i odgovarajućih vrednosti type atributa. Textarea tag je drugačiji, poseban – nije input sa atributom type i zato ne spada u ovu grupu.

Grupisanje podataka forme

Srodni elementi HTML forme mogu se grupisati korišćenjem elementa `fieldset`. Tako grupisani elementi biće jasno izdvojeni i na samoj stranici. Sledeći primer to ilustruje:

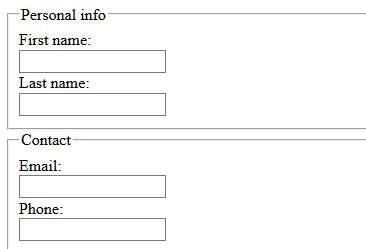
```
<form action="script.php" method="post" name="demo_form">
<fieldset>
  <legend>Personal info</legend>
  <label for="name">First name:</label><br>
  <input type="text" name="name" id="name"><br>

  <label for="surname">Last name:</label><br>
  <input type="text" name="surname" id="surname"><br>
</fieldset>

<fieldset>
  <legend>Contact</legend>
  <label for="email">Email:</label><br>
  <input type="text" name="email" id="email"><br>

  <label for="phone">Phone:</label><br>
  <input type="text" name="phone" id="phone">
</fieldset>
</form>
```

U primeru je kreirana forma sa dva `fieldset` regiona. Svaki `fieldset` element kao podelement poseduje element `legend`, unutar koga se definiše naslov grupe. Prikazani primer na stranici će proizvesti rezultat kao na slici 7.17.



Slika 7.17. Izgled forme sa definisanim regionima korišćenjem fieldset elementa

Element `fieldset` kreira jasno izdvojene grupe kontrola koje se nalaze unutar forme. Grupe kontrola se na stranici unutar browsera jasno izdvajaju dodavanjem okvira. Pritom se naziv grupe definiše elementom `legend`.

Napomena

Na današnjem webu, forme su jedan od elemenata web sajtova i web aplikacija koji se može susresti u najrazličitijim oblicima. kada se govori o njihovom izgledu. HTML jezik ne obezbeđuje gotovo nikakve mogućnosti za uticanje na izgled formi i njihovih elemenata. Tako nešto obavlja se korišćenjem jezika CSS i biće posebno objašnjeno u jednoj od narednih lekcija ovog kursa.

Rezime

- Forme predstavljaju alat za sakupljanje informacija od posetilaca web sajta.
- Prikupljeni podaci forme prosleđuju se do skripte koja obavlja obradu takvih podataka.
- HTML forma kreira se korišćenjem `form` elementa.
- Element `form` može da sadrži atribute koji određuju način njegovog funkcionisanja.
- Atribut `action` definiše kome će podaci forme biti prosleđeni.
- Atribut `method` definiše način na koji će podaci biti prosleđeni.
- Element `input` je najznačajniji element forme i može se pojaviti u nekoliko različitih varijacija, i to u zavisnosti od vrednosti njegovog `type` atributa.
- Unos jednostavnog teksta unutar forme postiže se korišćenjem `input` elementa sa vrednošću `type` atributa `text`.
- Unos višelinijskog teksta unutar forme postiže se korišćenjem elementa `textarea`.
- Prosleđivanje podataka forme postiže se korišćenjem `input` elementa tipa `submit`.