LADA MALEŠ
SAŠA MLADENOVIĆ

OSNOVE PROGRAMIRANJA ZA WEB

SVEUČILIŠTE U SPLITU ◆ FILOZOFSKI FAKULTET

OSNOVE PROGRAMIRANJA ZA WEB

(HTML, JavaScript, XML, i XSL)

Nakladnik
Sveučilište u Splitu
Filozofski fakultet

Za nakladnika
Josip Milat, dekan

Za izdavačko povjerenstvo
Anči Leburić, predsjednica

Biblioteka
Suvremena nastava
Knjiga br. 4.

Recenzenti
Slavomir Stankov
Marko Rosić
Maja Štula

Lektorica
Sonja Krvavica

Grafička obrada i dizajn naslovnice
Lada Maleš
Saša Mladenović

Tisak
Redak, Split

CIP - Katalogizacija u publikaciji
SVEUČILIŠNA KNJIŽNICA U SPLITU

UDK 004.738.52(075.8)
004.438(075.8)

MALEŠ, Lada
Osnove programiranja za web
(HTML, JavScript, HML i XSL) / Lada Maleš, Saša Mladenović. - Split : Filozofski fakultet Sveučilišta, 2007. - (Biblioteka Suvremena nastava ; knj. 4)

ISBN 978-953-7395-21-6

I. Mladenović, Saša
I. Web dizajn -- Priručnik

470605015

ISBN 978-953-7395-21-6

Studija se objavljuje u tiskanom i elektroničkom obliku na službenoj web stranici Filozofskog fakulteta u Splitu, te na CD-ovima, prema odluci br. 849/07 donesenoj na sjednici Znanstveno-nastavnog vijeća FF Splita dana 19.ožujka, 2007. godine.

Osnove programiranja za web

(HTML, JavaScript, XML i XSL)

Lada Maleš, Saša Mladenović

Split, 2007.

PREDGOVOR

Razdoblje interneta počinje 1969. godine, ali tek devedesetih godina dvadeseta stoljeća doživljava svoju ekspanziju. Internet je do danas izmijenio načine komunikacije među ljudima. Poslati nekome elektroničku poštu umjesto pisma, kupiti knjigu webom umjesto u knjižari, platiti račune preko internetom umjesto na šalteru u banci i sl., postale su uobičajene pojave, još nedavno nezamislive. Sve usluge koje nam se nude preko mreže imaju svoju tehnološku podršku (infrastrukturnu, sklopovsku i programsku).

U ovim skriptama opisat ćemo osnove HTML-a, JavaScripta, XML-a i XSL-a. U prvom poglavlju ukratko su opisane neke tehnologije i alati za izradu aplikacija na webu. Drugo poglavlje opisuje kako izraditi web stranicu pomoću HTML-a. Programiranjem pomoću JavaScripta bavi se treće poglavlje, a u četvrtom poglavlju uči se XML. Peto poglavlje pokazuje kako povezati HTML i XML, dok se šesto poglavlje bavi XSL-om.

Skripta su namijenjena studentima Filozofskog fakulteta u Splitu, Centra za stručne studije Sveučilišta u Splitu i Fakulteta prirodoslovno-matematičkih znanosti i kineziologije u Splitu, koji u sklopu nastave slušaju opisane sadržaje. Naravno, skripta su namijenjena i svima onima koji se žele upoznati sa obrađenim temama.
Na tržištu postoji velik broj knjiga koje obrađuju iste teme. Kod pisanja ove knjige, vodili smo se idejom da čitateljima na jednom mjestu objedinimo te sadržaje. Nadamo se da ćemo na taj način olakšali usvajanje znanja iz područja web tehnologija, što nam je i bio cilj.

Autori

SADRŽAJ

1. Uvod

Velik broj ljudi kreće se webom potaknuti jednostavnim korištenjem preko grafičkog sučelja i relativno jeftinim pristupom nebrojenoj količini podataka svih vrsta. Mnoge tvrtke u korisnicima weba vide potencijalne kupce te u tom dijelu tržišta žele zauzeti što bolje pozicije. Web može biti izvrsna infrastruktura za novu aplikaciju. Internet pri tome osigurava zajednički prijenosni mehanizam, a sučelje web preglednika ujedno je i sučelje aplikacije.

Niz je primjena aplikacija na webu, npr. unos narudžaba od krajnjeg kupca preko on-line kataloga, podrška korisnicima, pregled najnovijih cjenika udaljenim komercijalistima, prodaja proizvoda na međunarodnom tržištu, rad iz kuće (pristup katalogu i izdavanje narudžbe) i sl.

Za realizaciju bilo kojeg od ovih zahtjeva nužne su zadovoljavajuće telekomunikacijske veze (pristup internetu mora biti dovoljno brz). Za mnoge aplikacije potrebno je programirati grafičko sučelje (GUI Graphical User Interface), a za neke je potrebno prikazati sliku proizvoda.

Prije nego se krene u razvoj web aplikacije, potrebno je razmotriti tradicionalne načine rješavanja problema, a ne razvijati web aplikaciju samo zato što je to moderno i zanimljivo.

Dizajn internet aplikacije je zahtijevan, jer se prije svega radi o klijent-poslužitelj arhitekturi aplikacije. Potrebno je dobro razlučiti koji se dijelovi aplikacije mogu izvršiti nezavisno od poslužitelja. Nadalje treba voditi računa kako većina resursa nije pod vašom kontrolom. Kakva korist od aplikacije koja je dobro dizajnirana, izrađena u najmodernijim alatima, ali nije dostupna korisniku kada mu to treba jer njegova internetska veza nije stabilna ili ako korisnik nije ni s kime u mogućnosti izmijeniti podatke jer se koristi protokolom koji je toliko napredan da ga ne koristi nitko drugi. Najgore je kada je web aplikacija u svom izvođenju toliko spora da ometa korisnika u radu. Treba istaknuti kako je aplikacija tu da pomogne korisnicima u izvršavanju njihovih svakodnevnih zadaća, a ne da bi lijepo izgledala. Također, ne smije se zaboraviti kako postoje i korisnici koji nemaju brzu internetsku vezu te da se na lokalnoj mreži uz internet/intranet aplikacije vjerojatno izvode i druge aplikacije.

1.1. Projektiranje aplikacije

Ključ uspješne izrade aplikacije uvijek je isti. Postupak projektiranja započinje definicijom zahtjeva. Na temelju zahtjeva izrađuju se detaljne tehničke specifikacije sustava. Nakon izrade aplikacije preostaje još testirati aplikaciju.

Korisnički zahtjevi trebaju sadržavati opis funkcija sustava i očekivano opterećenje. Nije svejedno hoće li aplikaciju istovremeno koristiti pet ili pet

tisuća korisnika. Isto tako, ne može se reći kako je svejedno hoće li se u bazi nalaziti pet ili pet milijuna slogova. O ovim zahtjevima ovisit će izbor alata i nekih tehničkih rješenja. Nije isplativo utrošiti vrijeme na optimizaciju SQL upita ako se zna kako će u bazi postojati samo stotinjak slogova. Treba voditi računa o pravilnom dimenzioniranju rješenja koje se predlaže. Nije razumno koristiti kamion nosivosti 20t za dostavu pizze na kućnu adresu. Potrebno je dobro razmisliti o tome prije prijedloga ili izbora alata za izradu aplikacije. Kao primjer pretpostavit ćemo kako je korisnik postavio sljedeći zahtjev pismenim putem:

Grupa Enterprise
Videoteka Star Trek
Poljudsko šetalište b.b.

ZAHTJEV ZA IZRADOM PROGRAMSKE PODRŠKE

Aplikacija koju tražimo služit će za odabir DVD naslova iz posebne ponude koju ima naša videoteka preko interneta. Cilj je omogućiti pojedincima uvid u izbor filmova koje nudimo po povoljnim cijenama kako bi se poboljšala kvaliteta usluge. Najčešće jednom dnevno (na kraju radnog vremena) katalog iz videoteke treba uskladiti s katalogom na webu. Potrebno je omogućiti brzo pojedinačno prelistavanje ponuđenih naslova. U posebnoj ponudi pojavit će se najviše 100 naslova. Katalog je namijenjen rasprodaji rabljenih DVD naslova. Aplikacija mora omogućiti slaganje naslova po naslovu, redatelju i cijeni. Katalog na webu mora biti dostupan i u tabličnom obliku. Zadovoljavajuće je pretpostaviti kako većina korisnika pristupa našim stranicama IE6.0 ili višim preglednikom. Ovi podaci dobiveni su praćenjem pristupa tvrtkinim stranicama.

Ovo je tipičan zahtjev korisnika koji nije detaljan, ali korisnik očekuje aplikaciju koja će biti zaokružena cjelina koja će mu pomoći u ostvarivanju cilja. U ovom slučaju aplikacija mora pomoći poboljšanju prodaje DVD naslova iz posebne ponude korisnika. Iako aplikacija može biti izrađena prema svim pravilima struke, to ne znači da će korisnik ispuniti svoje ciljeve samom uporabom aplikacije, i to je potrebno naglasiti. Funkcije aplikacije nisu dovoljno detaljne za izradu koda, a specificirat će se tijekom izrade aplikacije. Iz zahtjeva je jasno kako je potrebno izraditi internetsku aplikaciju. Iduća faza je izbor tehnologija i alata potrebnih za izradu aplikacije.

1.1.1. Izbor tehnologija i alata

Jedna od stvari na koje se ne smije zaboraviti je stalna promjena funkcionalnosti web sjedišta (eng. site). Mogu se već uočiti promjene nastale odmakom od statičkih prema dinamičkim stranicama. Novu ulogu u izboru tehnologija i alata imaju i brojni preglednici koji se pojavljuju na platformama kao što su mobilni telefoni i ručna računala. Uzimajući u obzir trendove tržišta, možemo zaključiti da bi se svaki web programer trebao usredotočiti na: HTML

4.01, CSS, XHTML, XML i XSL, skriptni jezik na strani poslužitelja i klijenta i upravljanje podacima uporabom SQL-a.

1.1.1.1. HTML (HyperText Markup Language)

HTML 4.01. zadnji je standard ovog tipa koji je izdao W3C (World Wide Web Consortium) i neće se više razvijati. Umjesto njega nastavlja se dalje sa standardom XHTML. Uporabom HTML 4.01 standarda buduće unapređivanje iz HTML u XHTML standard postaje vrlo jednostavan postupak. HTML se koristiti za izradu korisničkog sučelja i više o njemu bit će riječi u nastavku.

1.1.1.2. CSS (Cascading Style Sheets)

Stil (eng. style) određuje način prikaza elemenata HTML-a, što se u prethodnom standardu HTML 3.2, postizalo atibutima i oznakama unutar samog HTML koda. CSS omogućava da se jedanput definirani stilovi pohranjuju u datotekama izdvojenim od HTML dokumenata. Ovakav način organizacije omogućava promjenu izgleda stranica na webu jednostavnom ispravkom CSS datoteke. Prednost uporabe CSS datoteka naročito dolazi do izražaja kod izmjena boja ili vrste slova više stotina HTML dokumenata. Umjesto izmjene u svakom od HTML dokumenata, CSS omogućava izmjenu samo u jednoj datoteci.

1.1.1.3. XHTML (eXtensible Hyper Text Markup Language)

XHTML 1.1. najnoviji je HTML standard koji je objavila institucija W3C i predstavlja budućnost HTMLa. XHTML 1.0. postao je službena preporuka 26. siječnja 2000. Dobiveni status preporuke znači kako je specifikacija stabilna i kako je to sada novi web standard. U svibnju 2001. objavljen je XHTML 1.1 (nešto restriktivniji od prethodnog standarda). XHTML je reformulacija HTML 4.01 standarda u XML i može se odmah početi koristiti postojećim preglednicima uz poštivanje nekoliko jednostavnih pravila.

1.1.1.4. XML (eXtensible Markup Language)

XML je metajezik[1] koji je uveo W3C kako bi korisnici mogli razvijati svoje markup jezike, zavisne od njihovih potreba. XML standardizira na koji se način to radi. Važno je razumjeti kako XML nije zamjena za HTML. U razvoju web aplikacije XML se koristi za opis i prijenos podataka, dok se HTML koristi za prikaz podataka. Najbolji opis XML-a je: alat za prijenos informacija neovisan o programskoj podršci i sklopovlju (hardware and software independent tool).

[1] Metajezik definira pravila i simbole za kreiranje drugog jezika.

Važnost XML-a moguće je usporediti s važnošću HTML-a u samim počecima weba. Pokazatelji upućuju kako će XML postati najčešće korišten alat za prijenos i rad s podacima na webu.

1.1.1.5. XSL (eXtensible Stylesheet Language)

XSL je alat za transformaciju podataka. Podatke je potrebno isporučiti u različitim formatima, različitim preglednicima i različitim poslužiteljima. Kako bi se omogućila transformacija XML podataka, W3C je preporučio XSL standard. XSL može transformirati XML datoteku u format prepoznatljiv za web preglednik. Jedan od tih formata je HTML. Još jedan od danas popularnih formata je WML (jezik za označavanje korišten u mnogim mobilnim uređajima). XSL omogućava dodavanje, brisanje, slaganje, testiranje, odlučivanje o prikazu elemenata i još mnogo toga.

1.1.1.6. Skriptni jezik na strani poslužitelja i klijenta

Kako bi se omogućila isporuka dinamičkog sadržaja na webu, potrebno je izraditi skripte na strani poslužitelja, dok za provjeru unosa podataka i neke lokalne obrade treba izraditi skripte na strani klijenta. Postoje mnogi skriptni jezici, a jedan od onih koji se mogu koristiti na obje strane je JavaScript.

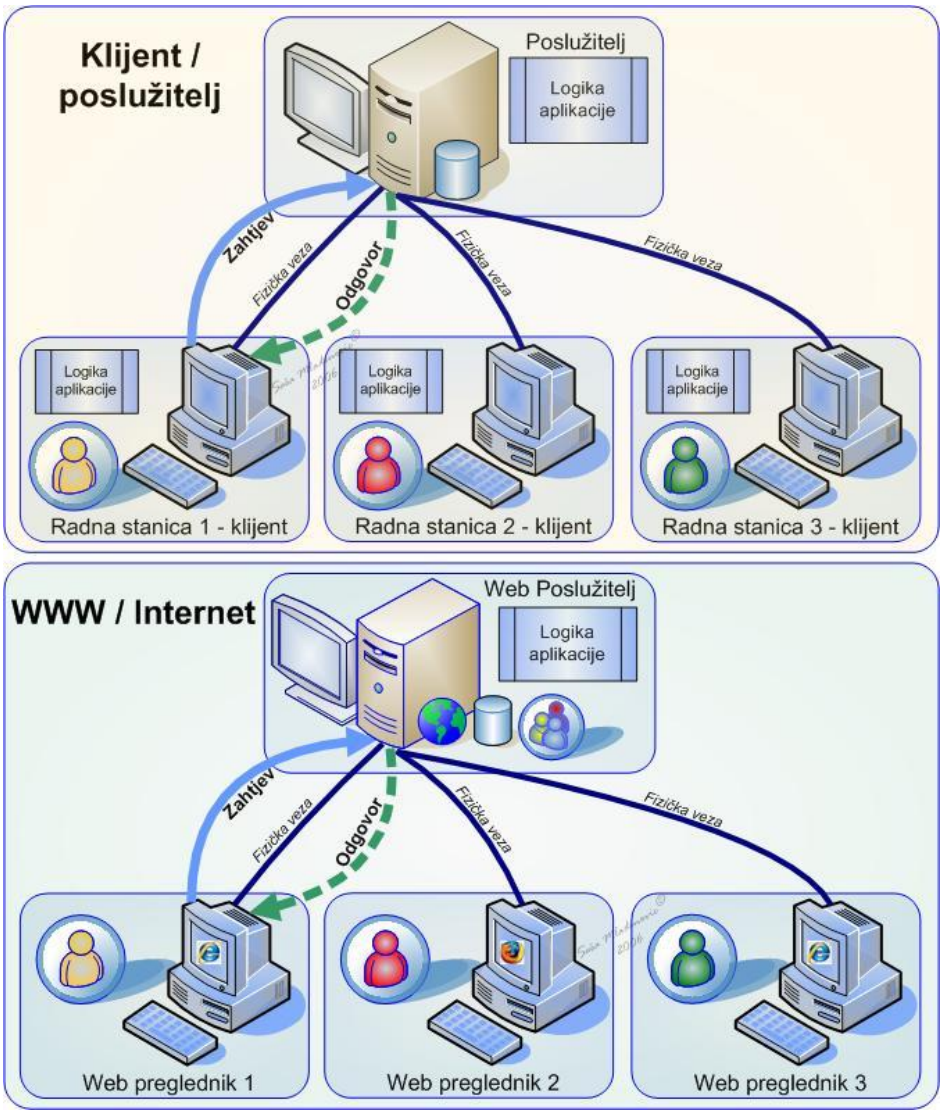
1.1.1.7. Upravljanje podacima uporabom SQL-a

SQL (Structured Query Language) standard je za pristup bazama podataka kao što su SQL Server, Oracle, Sybase, Access itd. Znanje SQL-a potrebno je svima koji žele pohraniti, izmijeniti ili preuzeti podatke iz baze podataka.

Kada se zna opća ideja za sustav koji treba realizirati, potrebno je donijeti odluku koji će se alati koristiti. Stoga na osnovi do sada opisanog za izradu sučelja koristit će se HTML, za prijenos podataka iz kataloga i njihov prikaz koristit će se XML i XSL i konačno za kontrolu unosa podataka potrebnih za pretragu upotrijebit će se JavaScript[2].
Kako bismo mogli izraditi traženu aplikaciju, potrebno je svladati barem osnove navedenih alata. Započet ćemo HTML-om i nastaviti JavaScriptom, XML-om i XSL-om.

Razlika između klasične klijent poslužitelj aplikacije i internetske aplikacije može se prikazati shemom:

[2] Nećemo koristiti bazu podataka koja će kao rezultat upita davati XML datoteku.



Shema 1. Usporedba klijent-poslužitelj i internetske aplikacije

2. Izrada web stranica pomoću HTML-a

HTML je kratica za **H**yper**T**ext **M**arkup **L**anguage. Datoteka napisana pomoću HTMLa je obična tekstualna datoteka koja u sebi sadrži oznake (engl. *markup tags*), koje kažu web pregledniku (engl. *browser*) kako treba prikazati stranicu. HTML dokument mora imati .htm ili .html ekstenziju.

2.1. Kratka povijest HTML-a

Početkom 90-ih godina prošlog stoljeća Tim Berners-Lee, fizičar zaposlen u CERN-u (Geneve) razvio je HTML, koji je populariziran upotrebom preglednika Mosaic. Tijekom 90-ih zbog naglog razvoja weba došlo je i do brzog razvoja HTML-a. Internet Engineering Task Force (IETF) 1995. razvija standard HTML 2.0 kako bi se uobličila već usvojena praksa izrade stranica[3]. HTML+ (1993) i HTML 3.0 (1995) predlagali su širu verziju HTML-a. Unatoč teškoćama pri dobivanju konsenzusa za standard, 1997. godine World Wide Web Consortium (W3C) donosi nov standard koji je usvojio uobičajenu praksu kod izrade web stanica i tako je nastao HTML 3.2. standard. Budući da HTML u početku nije imao zamišljene oznake za formatiranje (koje su ubačene u HTML 3.2.), nastaje HTML verzija 4.0.[4], koja dodaje upotrebu stilova (engl. *style sheets*), skripta, okvira (engl. *frame*) itd.. Također, novi standard omogućava da dodaci mogu biti smješteni izvan HTML dokumenta u posebnoj datoteci koja se poziva iz HTML dokumenta. Standard HTML 4.01. unosi neke manje dopune i ispravke te pravila koja omogućavaju lakši prelazak na XHTML (e**X**tensible **H**yper **T**ext **M**arkup Language) 1.0. standard. Uporabom HTML 4.01[5] standarda buduće unaprjeđivanje iz HTML u XHTML (HTML preformuliran kao XML aplikacija) standard postaje vrlo jednostavan postupak.[6] Stoga je W3C organizacija za standarde na području web tehnologija odlučila da HTML 4.01. bude zadnja verzija standarda. Dalje će se nastaviti s XHTML standardom.

Organizacije za standarde na Internetu

Poput mnogih popularnih tehnologija HTML je počeo kao neformalna specifikacija koju je koristila nekolicina ljudi. Kako se HTML sve više koristio, tako je postalo nužno formalizirati ga. Dvije su organizacije koje se brinu za standardizaciju web i općenito internet tehnologija, a to su W3C i IETF.

World Wide Web Consortium (W3C) osnovan je kako bi definirao standarde za HTML, a poslije za XHTML. Osim navedenih standarda W3C se brine za standardizaciju svih tehnologija koje su povezane s webom kao HTTP (Hyper Text Transfer Protocol), CSS

[3] Ne postoji službena verzija standarda HTML 1.0. jer je u to vrijeme postojalo više neslužbenih HTML standarda

[4] http://www.w3.org/TR/REC-html40/intro/intro.html#idx-HTML-2

[5] Popis svih oznaka HTML 4.01 standarda dan je u prilogu

[6] http://www.w3.org/MarkUp/

(Cascading Style Sheets), XML (eXtensible Markup Language) itd. Više o W3C može se pogledati na http://www.w3.org

Internet Engineering Task Force (IETF) organizacija je zadužena za definiranje i upravljanje svim aspektima internetske tehnologije. IETF izdaje RFC (Requests for Comments) dokumente u kojima definira određenu internetsku tehnologiju. Više o djelatnosti IETF može se pogledati na http://www.ietf.org

2.2. Alati za izradu HTML-a

Danas postoji veliki broj alata za izradu web stranica (nabrojit ćemo samo neke: Microsoft FrontPage, Macromedia Dreamweaver, Netscape Composer, Nvu ...[7]). Međutim, sve što je potrebno za izradu stranice je običan uređivač teksta kao na primjer Notepad i web preglednik za provjeru izgleda izrađenih web stranica. Osim toga poželjno je imati neki alat za obradu slike ili multimedijalnih datoteka. Svaki od ova dva pristupa: automatskog generiranja stranica nekim od alata ili „ručne“ izrade pomoću jednostavnog programa za obradu teksta ima svoje prednosti i mane.

„Ručna“ izrada web stranica	**Automatsko generiranje stranica**
+	+
• nije skupa (potreban je samo običan uređivač teksta) • omogućuje autoru suštinsko razumijevanje HTML-a u svim njegovim detaljima	• brzo i efikasno kreiranje i uređivanje stranica • u svakom trenutku vidljiv pravi izgled stranice • eliminiran problem sintaktičke ispravnosti koda
-	-
• relativno je spora • nema garancije da je ručno napisani HTML kod korektan	• skupo (alati za automatsko generiranje stranica uglavnom su komercijalni proizvodi)

Kod izrade web stranica možete se odlučiti na jedan od ta dva pristupa. Većina alata za automatsko generiranje web stranica su WYSIWYG (what-you-see-is-what-you-get), što znači da se već pri izradi web stranice vidi njezin izgled. Drugi pristup se zasniva na pisanje HTML oznaka pomoću jednostavnog uređivača teksta. U ovim skriptama bit će opisan drugi način jer je cilj svladati HTML, nakon čega je prelazak na bilo koji alat za automatsko generiranje web stranica brz i jednostavan. Također, razumijevanje HTML-a olakšava rad u bilo kojem alatu za izradu web stranica, jer većina ima opciju koja omogućava pisanje koda.

[7] http://www.microsoft.com/frontpage/; http://www.macromedia.com/; http://www.netscape.com/; http://www.nvu.com

2.3. Struktura HTML dokumenta

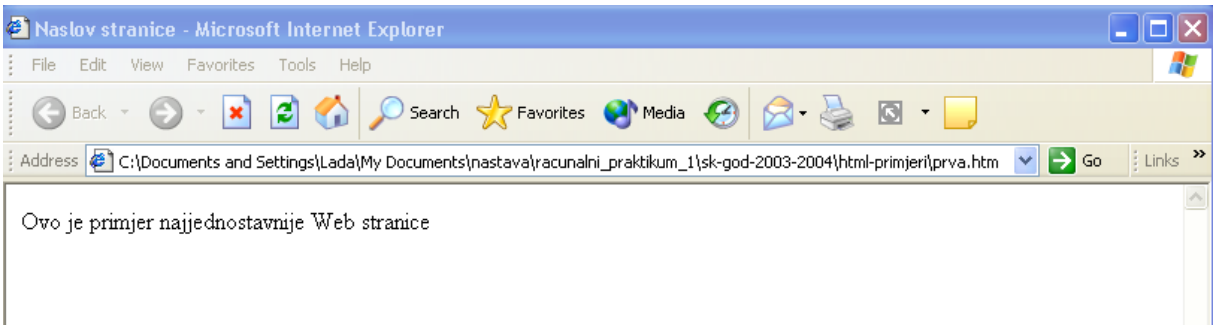
HTML dokument je tekstualna datoteka koja se sastoji od HTML elemenata koji su definirani pomoću HTML oznaka (engl. *tag*). Svaki HTML dokument ima svoju strukturu.

Primjer 1: Najjednostavniji HTML dokument. Napisan je u Notepadu i snimljen pod nazivom *prva.htm* Izgled datoteke u pregledniku prikazan je na Slici 1.

```
<html>
<head>
<title>Naslov stranice</title>
</head>
<body>
Ovo je primjer najjednostavnije Web stranice.
</body>
</html>
```

Primjer 1. HTML datoteka prva.htm

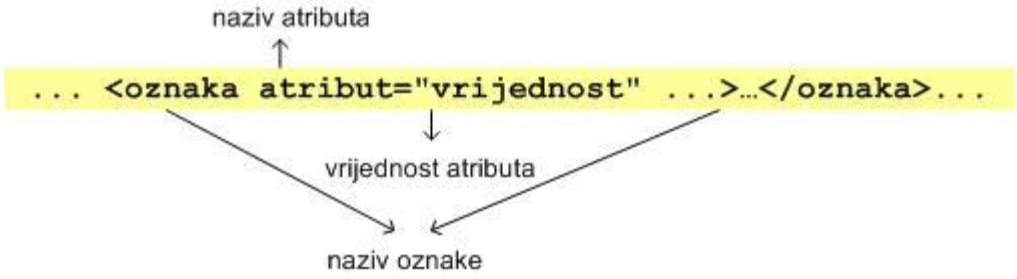
Datoteka *prva.htm* ovako izgleda u pregledniku:



Slika 1. Izgled datoteke prva.htm u web pregledniku

2.4. HTML elementi i oznake

HTML oznake koriste se za označavanje HTML elemenata i uokvirene su dvama znakovima **<** i **>**. Obično dolaze u paru gdje se prva naziva početna oznaka (engl. *start tag*), a druga zadnja oznaka (engl. *end tag*). Npr. <body> i </body>. Tekst koji se nalazi između početne i zadnje oznake naziva se sadržaj elementa. HTML oznake se mogu pisati velikim i malim slovima, ali u skladu s preporukom u HTML 4.0. standardu preporuča se pisati ih malim slovima.



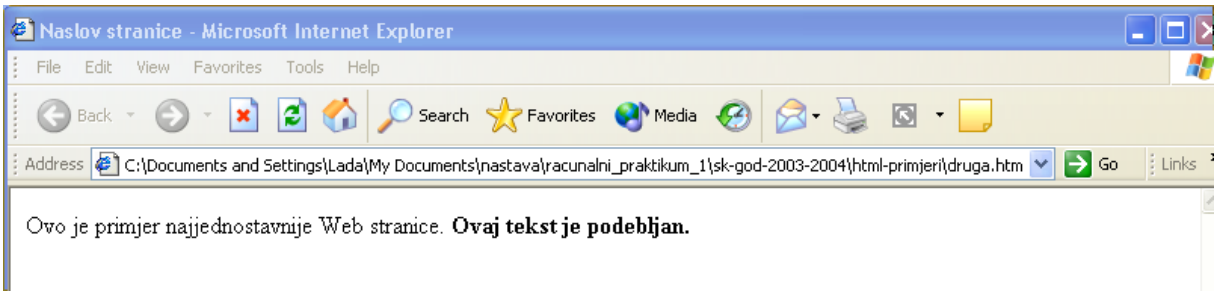
Nadopunit ćemo datoteku iz Primjera 1 još jednim HTML elementom.

Primjer 2: U datoteku iz Primjera 1 dodana je oznaka za podebljana slova.

```
<html>
<head>
<title>Naslov stranice</title>
</head>
<body>
Ovo je primjer najjednostavnije Web stranice.
<b>Ovaj tekst je podebljan.</b>
</body>
</html>
```

Primjer 2. HTML datoteka druga.htm

Datoteka *druga.htm* u web pregledniku izgleda ovako:



Slika 2. Izgled datoteke druga.htm u web pregledniku

`<b>Ovaj tekst je podebljan.</b>` HTML je element koji se sastoji od:
početne oznake:
sadržaja elementa: Ovaj tekst je podebljan.
zadnje oznake:

HTML element je također:

```
<body>
Ovo je primjer najjednostavnije Web stranice.
<b>Ovaj tekst je podebljan.</b>
</body>
```

2.4.1. Osnovne HTML oznake

Oznaka	Opis
<html>	Definira HTML dokument
<body>	Definira tijelo dokumenta
<h1> to <h6>	Definira veličinu naslova
<p>	Paragraf
** **	Prelazak u novi red
<hr>	Horizontalna crta
<!-->	Komentar

2.4.1.1. Oznaka <html>

Sav sadržaj HTML dokumenta, osim deklaracije tipa dokumenta, mora se naći unutar oznaka **<html>**.... **</html>**.

2.4.1.2. Oznaka <head>

Nakon oznake <html> slijedi zaglavlje koje počinje oznakom <head> i završava oznakom </head>. Unutar oznaka za zaglavlje obično se stavlja naslov stranice i metapodaci[8].

2.4.1.3. Oznaka <title>

Tekst koji se nalazi između oznaka <title> i </title> pojavit će se u naslovu web preglednika. Ovaj tekst nije moguće formatirati i piše se uvijek unutar zaglavlja.

2.4.1.4. Oznaka <body>

Unutar oznaka <body> i </body> nalazi se tijelo html dokumenta. Otvara se nakon oznake </head> i zatvara prije oznake </html>.

2.4.1.5. Oznake <h1> do <h6>

Gotovo svaki dokument je podijeljen na blokove teksta. Za dobru organizaciju teksta potrebno je imati i različite veličine naslova. U HTML-u su naslovi (engl. *heading*) definirani u 6 razina od <h1> do <h6>. HTML prije oznake naslova i nakon toga dodaje jedan prazan red.

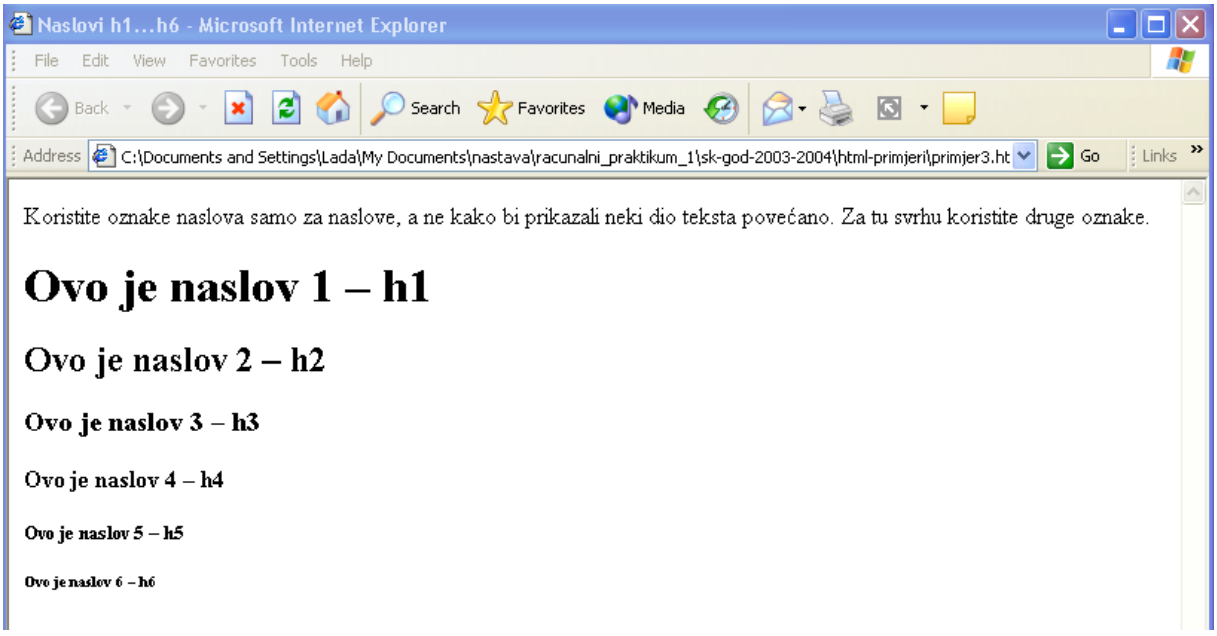
[8] Metapodaci su podaci koji opisuju podatke i sadrže informacije poput ključnih riječi, imena autora dokumenta ili bilo koji drugi podatak koji nije sadržan u samom dokumentu. Ove informacije ne prikazuju se u pregledniku.

Primjer 3: Oznake za naslove u HTML-u (od veličine h1 do h6)

```
<html>
<head>
<title>Naslovi h1...h6</title>
</head>
<body>
Koristite oznake naslova samo za naslove, a ne kako bi prikazali neki
dio  teksta povećano. Za tu svrhu koristite druge oznake.
<h1>Ovo je naslov 1 – h1</h1>
<h2> Ovo je naslov 2 – h2</h2>
<h3> Ovo je naslov 3 – h3</h3>
<h4> Ovo je naslov 4 – h4</h4>
<h5> Ovo je naslov 5 – h5</h5>
<h6> Ovo je naslov 6 – h6</h6>
</body>
</html>
```

Primjer 3.

Primjer 3. bi u pregledniku izgledao ovako:



Slika 3. Izgled web stranice iz Primjera 3.

Naslov može biti postavljen na sredinu korištenjem atributa *align* ili oznake [9]

[9] Uporaba oznake i atributa *center* je po HTML 4.01. zastarjela je, premda se još koristi. W3C preporuča korištenje CSS stilova

Primjer 3a i 3b: Poravnanje naslova na sredinu reda

```
<html>
<head>
<title>Poravnanje
teksta</title>
</head>
<body>
<h1 align="center">
Ovo je naslov 1
</h1>
<p> Naslov 1 je poravnat na
sredinu.</p>
</body>
</html>
```

Primjer 3a.

ILI

```
<html>
<head>
<title>Poravnanje teksta
</title>
</head>
<body>
<center>
<h1>Ovo je naslov 1</h1>
</center>
<p> Naslov 1 je poravnat na
sredinu.</p>
</body>
</html>
```

Primjer 3b.

2.4.1.6. Oznaka

Početak paragrafa, tj. odlomka označava se oznakom . Zatvaranje znakom nije obavezno[10].

Primjer 4: Uporaba odjeljaka (paragrafa) u web stranici

```
<html>
<head>
<title>Odlomci</title>
</head>
<body>
<p>Odlomak 1. Ovo je tekst koji se nalazi u odlomku 1</p>
<p>Odlomak 2. Opet malo teksta.</p>
<p>Odlomak 3. Još jednom novi odlomak.</p>
</body>
</html>
```

Primjer 4.

2.4.1.7. Oznaka

Oznaka <br> služi za prelazak u novi red. Bilo gdje u tekstu postavljena oznaka <br> prouzročit će prelazak u novi red.

Primjer 5: Prelazak u novi red (oznaka <br>)

```
<html>
<head>
<title>Prelazak u novi red</title>
```

[10] Premda zatvaranje oznakom nije obavezno u HTML 4.0., pravila za XHTML to nalažu

```
</head>
<body>
<p>Ako želite prijeći u novi red <br> morate upotrijebiti oznaku br
<br>.</p>
</body>
</html>
```

Primjer 5.

2.4.1.8. Oznaka <hr>

Oznaka <hr> definira horizontalnu crtu.

Pimjer 6: Ubacivanje horizontalne crte

```
<html>
<head><title>Horizontalna crta</title></head>
<body>
<p>Oznaka hr definira horizontalnu crtu:</p>
<hr><p>Odlomak 1</p>
<hr><p>Odlomak 2</p>
<hr><p>Odlomak 3</p>
</body>
</html>
```

Primjer 6.

Izgled i poravnanje horizontalne crte može se mijenjati koristeći atribute *width*, *size*, *align*.

Primjer 6a: Atributi oznake <hr> (width, size, align, noshade)

```
<html>
<head>
<title>Horizontalna crta</title>
</head>
<body>
<p>Oznaka hr definira horizontalnu crtu:</p>
<hr width=75% size=3 align="left">
<p>Odlomak 1</p>
<hr width=200 size=2>
<p>Odlomak 2</p>
<hr size=4 noshade>
<p>Odlomak 3</p>
</body>
</html>
```

Primjer 6a.

2.4.1.9. Komentari u HTML-u

Oznaka za komenatar <!—služi za ubacivanje teksta u HTML kod, koji se neće prikazati u web pregledniku.

Primjer 7: Komentari u HTMLu

```
<html>
<head>
<title>Komentari</title>
</head>
<body>
<!—Komentar se ne prikazuje-->
<p>Ovo je paragraf</p>
</body>
</html>
```

Primjer 7.

2.4.2. Formatiranje teksta

2.4.2.1. Oznake za formatiranje

Oznake za formatiranje teksta dijele se na **logičke** i **fizičke** stilove. Oznake logičkog stila kažu pregledniku da taj tekst ima neko posebno značenje, kontekst ili namjenu. HTML 4.01. ne definira kako će takav tekst izgledati. To ovisi o samom pregledniku koji će ga formatirati tako da se razlikuje od običnog teksta. Za razliku od logičkog stila, pomoću oznaka fizičkog stila može se točno odrediti izgled teksta (npr. podebljani ili iskošeni tekst).

Oznake logički stil	Opis
<abbr>	Tekst koji predstavlja kraticu
<acronym>	Tekst koji predstavlja akronim
<cite>	Teksta koji predstavlja citat
<code>	Tekst koji predstavlja računalni kod
<dfn>	Tekst koji predstavlja frazu
****	Naglašeni tekst
<kbd>	Prikaz teksta koji bi trebao unijeti korisnik
<samp>	Naglašava se tekst koji je izvan uobičajenog konteksta
****	Pojačano naglašeni tekst
<var>	Prikaz varijabli

Oznake fizički stil	Opis
<b>	Podebljani tekst (ekvivalent oznaci)
<big>	Tekst koji će biti prikazan za jednu veličinu veću od prethodnog
<blink>	Tekst koji „blinka“ (pravila dobrog web dizajna kažu ne koristiti ga)
<i>	Iskošeni tekst
<small>	Tekst koji će biti prikazan za jednu veličinu manju od prethodnog
<s> <strike>	Precrtani tekst. Zastarjelo.
<sub>	Tekst u indeksu (subscript)
<sup>	Tekst u potenciji (superscript)
<tt>	Tekst sa razmakom između znakova
<u>	Podcrtani tekst. Zastarjelo.

Primjer 8: Formatiranje teksta u HTMLu (logički i fizički stilovi)

```
<html>
<head>
<title>Formatiranje</title>
</head>
<body>
<h2>Logički stilovi</h2>
<code>Računalni kod code</code>
```

```
<br><em>Naglašeni tekst -em</em>
<br><strong>Pojačano naglašeni tekst -strong</strong>
<br><var>Varijable –var</var>


<h2>Fizički stilovi</h2>
<i>Iskošeni tekst -i</i>
<br>Podebljani tekst –b</b>
<br><big>Tekst za 1 veličinu veći od prethodnog–big</big>
<br><small>Tekst za 1 veličinu manji od prethodnog -small</small>
<br>Ovaj tekst sadrži<sub>subscript</sub>
<br>Ovaj tekst sadrži<sup>superscript</sup>
<br><tt>Tekst s većim razmakom –tt</tt>
<br><b><i>Podebljani iskošeni tekst</i></b>
</body>
</html>
```

Primjer 8.

2.4.2.2. Predformatirani tekst

Kada se na web želi postaviti tekst koji je napisan u nekom uređivaču teksta i pri tom žele zadržati razmaci i novi redovi, koristi se oznaka <pre>. Tekst unutar oznaka <pre> ...</pre> bit će prikazan u istom obliku kao i u originalnom tekstu. Također, znakovi se formatiranja napisani unutar oznake <pre> u HTML kodu zadržavaju (za razliku od tako formatiranog teksta izvan <pre> oznake. Učinak ove oznake najbolje se vidi ako se isti tekst prikaže sa oznakom <pre> i bez nje.

Primjer 9a: U uređivaču teksta (npr. MSWordu) napisan je tekst koji je kopiran u HTML unutar oznake <pre>.
Primjer 9b: Isti tekst kao iz primjera 9a, ali bez oznake <pre>

```
<html>
<head>
<title>Predformatirani
tekst</title>
</head>
<body>
<pre>
      U devetom selu
      živi Antuntun.
      U njega je malo
      neobičan um.

      On posao svaki
      na svoj način radi,
      jaja za leženje
      on u vrtu sadi.
```

```
<html>
<head>
<title>Bez predformatiranog
teksta</title>
</head>
<body>
      U devetom selu
      živi Antuntun,
      U njega je malo
      neobičan um.

      On posao svaki
      na svoj način radi,
      jaja za leženje
      on u vrtu sadi.

      ...
```

```
      ...
</pre>
<p>Oznaka pre pogodna je za
pisanje koda:</p>
<pre>
for i = 1 to 10
     print i
next i
</pre>
</body>
</html>
```

Primjer 9a.

```
<p>Oznaka pre pogodna je za
pisanje koda:</p>
for i = 1 to 10
     print i
next i
</body>
</html>
```

Primjer 9b.

2.4.2.3. Specijalni znakovi u HTML-u

Neki znakovi u HTML-u kodu imaju posebno značenje, npr. znakovi < i > koji predstavljaju početak odnosno kraj HTML oznake. Ako želimo da preglednik prikaže takve znakove, potrebno je ubaciti odgovarajući entitet u HTML kod. Entitet se sastoji od znaka &, imena entiteta ili znaka # s brojem entiteta i znaka ;. Npr. Da bismo prikazali znak < u pregledniku, potrebno je napisati < ili <

Najčešće korišteni entiteti:

Oznaka	Opis	Ime entiteta	Broj entiteta
	razmak[11]		
<	manje	<	<
>	veće	>	>
&	and	&	&
"	navodnici	"	"
'	apostrof		'

još neki česti entiteti:

Znak	Opis	Ime entiteta	Broj entiteta
©	copyright	©	©
®	registered trademark	®	®
×	množenje	×	×
÷	dijeljenje	÷	÷

[11] Najčešći entitet u HTML-u je znak razmaka (engl. space). HTML poštuje upisane razmake, ali kad se u tekstu nalaze npr. 5 uzastopnik razmaka, u pregledniku će biti prikazan samo jedan. Zbog toga se u takvim slučajevima piše entitet

2.4.2.4. Prikaz hrvatskih slova

Postoji više standarda za prikaz hrvatskih znakova (iso-8859-2, windows-1250). U oznaci <meta> upisuje se koja se kodna stranica treba učitati (ona mora postojati i na klijentovu računalu). Promjena kodne stranice u IE nalazi se pod View->Encoding

Kodovi hrvatskih znakova:

Znak	Kod	Znak	Kod
Č	&#200;	**č**	&#232;
Ć	&#230;	**ć**	&#198;
Đ	&#208;	**đ**	&#240;
Š	&#138;	**š**	&#154;
Ž	&#142;	**ž**	&#158;

2.4.2.5. Oznaka <font>

HTML 3.2. standardizirao je oznaku <font> kako bi preglednici mogli prikazati različite stilove, veličinu i boje slova. Standard HTML 4.01. dopušta korištenje oznake <font>, ali se preporuča korištenje CSS stilova kako bi HTML kod bio pregledan. Oznaka <font> koristi atribute *face*, *size* i *color*.

Atribut *face* određuje stil slova

Npr. `<font face="Arial, Times New Roman">Tekst</font>`

U prethodnom primjeru vrijednost atributa znači da će tekst biti u stilu Arial. Ako ga nema, tada će se prikazati Times New Roman, a kada nijedan nije prisutan na računalu, tada će tekst biti prikazan u „default" stilu.

Atribut *size* određuje veličinu slova

Npr. `<font size=4>Tekst</font>` znači da će tekst biti prikazan u relativnoj veličini 4 (default je 3).

Relativna veličina slova ide od najmanje 1, do najveće 7. Nemoguće je odrediti kolika je stvarna veličina tako prikazanih slova jer to zavisi od preglednika. U pravilu svaka relativna veličina je za 20% veća od prethodne. Prije vrijednosti atributa *size* može se dodati znak + ili -. Na taj način se definira povećanje ili smanjenje veličine slova u odnosu na veličinu slova koja je prethodila. Ako veličina prijeđe zadane vrijednost (od 1 do 7), preglednik je zaokruži na najmanju 1 ili najveću 7.

Npr. `Tekst 1<font size=+2>Tekst koji je za 40% veći od prethodnog teksta 1</font>`

Korištenje size=+1 ili size=-1 ima isti efekt kao oznake <big> i <small>.

Boja slova mijenja se pomoću atributa *color*.

Npr. `<font color="#ffff00">Tekst</font>` obojit će tekst unutar oznake[12].

[12] Više o bojama u nastavku (Poglavlje 2.4.6.)

2.4.3. Liste

Liste se u HTML-u pojavljuju u tri oblika: neuređene (nenumerirane), uređene (numerirane) i definicijske.
Neuređene liste sastoje se od stavki ispred kojih se nalazi obično mali crni krug. Takva lista počinje oznakom <ul>, a svaka stavka ima ispred oznaku <li>. Uređena lista također se sastoji od stavki, samo što su te stavke nabrojene. Počinje sa oznakom <ol>, a svaka stavka ima ispred oznaku <li>. Unutar stavki liste (uređene i neuređene) mogu se nalaziti paragrafi, slike, linkovi, druge liste itd.

Oznake za liste:

Oznaka	Opis
** **	Definira uređenu listi
** **	Definira neuređenu listu
** **	Definira stavke
<dl>	Definira definicijsku listu
<dt>	Definira definicijski pojam
<dd>	Definira opis definicijskog pojma
<dir>	Zastarjelo. Bolje koristiti
<menu>	Zastarjelo. Bolje koristiti

Primjer 10a i 10b: Neuređena i i uređena lista

```
<html>
<head>
<title>Neuređena
lista</title>
</head>
<body>
Ovako izgleda neuređena
lista.
<ul>
<li>Jabuke</li>
<li>Kruške</li>
<li>Banane</li>
<li>Naranče</li>
</ul>
</body>
</html>
```

Primjer 10a

```
<html>
<head>
<title>Uređena lista</title>
</head>
<body>
Ovako izgleda neuređena
lista.
<ol>
<li>Jabuke</li>
<li>Kruške</li>
<li>Banane</li>
<li>Naranče</li>
</ol>
</body>
</html>
```

Primjer 10b

Definicijska lista nije lista koja se sastoji od stavaka, već nju čine lista pojmova i njihovih objašnjenja. Definicijska lista počinje sa oznakom <dl>, svaki pojam počinje oznakom <dt>, a objašnjenje pojma sa <dd>. Unutar objašnjenja

pojma, tj. oznake <dd>, mogu se nalaziti paragrafi, slike, linkovi, druge liste itd.

Primjer 11: Definicijska lista

```
<html>
<head>
<title>Definicijska lista</title>
</head>
<body>
Ovako izgleda definicijska lista.
<dl>
<dt>RAM</dt>
<dd>Random Access Memory</dd>
<dt>ROM</dt>
<dd>Read Only Memory</dd>
</dl>
</body>
</html>
```

Primjer 11.

Upotrebom atributa može se mijenjati izgled lista.

Primjer 12a: Atributi neuređene liste
Primjer 12b: Atributi uređene liste

```
<html>
<head>
<title>Neuređena lista</title>
</head>
<body>
Izgled liste može se mijenjati
upotrebom atributa type.
<ul type="disc">
<li>Jabuke</li>
<li>Kruške</li>
<li>Banane</li>
<li>Naranče</li>
</ul>

<ul type="circle">
<li>Jabuke</li>
<li>Kruške</li>
<li>Banane</li>
<li>Naranče</li>
</ul>

<ul type="square">
<li>Jabuke</li>
<li>Kruške</li>
<li>Banane</li>
```

```
<html>
<head>
<title>Uređena lista</title>
</head>
<body>
Izgled liste može se
mijenjati upotrebom atributa
type.
<ol type="A">
<li>Jabuke</li>
<li>Kruške</li>
<li>Banane</li>
<li>Naranče</li>
</ol>
<ol type="a">
<li>Jabuke</li>
<li>Kruške</li>
<li>Banane</li>
<li>Naranče</li>
</ol>
<ol type="I">
<li>Jabuke</li>
<li>Kruške</li>
<li>Banane</li>
<li>Naranče</li>
```

```
<li>Naranče</li>
</ul>
</body>
</html>
```

Primjer 12a.

```
</ol>
<ol type="i">
<li>Jabuke</li>
<li>Kruške</li>
<li>Banane</li>
<li>Naranče</li>
</ol>
</body>
</html>
```

Primjer 12b.

Liste se mogu i ugnježdavati, tj. definirati jednu listu unutar druge i pri tome nije važno koji tip se ugnježđuje u koji.

Primjer 13: Gniježđenje lista

```
<html>
<head>
<title>Gniježđenje lista</title>
</head>
<body>
Liste se mogu i gnijezditi.
<ol>
  <li>Voće</li>
  <ul>
    <li>Jabuke</li>
    <li>Banane</li>
  </ul>
<li>Povrće</li>
  <ul>
    <li>Mrkva</li>
    <li>Špinat</li>
 </ul>
<ol>
</body>
</html>
```

Primjer 13.

Napomena: Primjere od 10a do 13 mogu se napraviti radi boljeg pregleda u istoj HTML datoteci.

2.4.4. Ubacivanje multimedijalnih datoteka

Web preglednici mogu prikazati različite multimedijalne datoteke (slike, zvučne i videozapise). Najčešće se na web stranice ubacuju slike.

2.4.4.1. Slike

Standardi HTML i XHTML ne propisuju format slike. Međutim, u praksi preglednici obično preferiraju formate GIF (Graphics Interchange Format) i JPEG (Joint Photographic Experts Group), tako da ih se može smatrati *de facto* standardima.

GIF ili JPEG?
GIF format (.gif) podržava prekrivanje, prozirnu podlogu slike i animirane slike. Mana mu je ograničen broj boja koje može prikazati (256). Za razliku od GIF formata, JPEG (.jpeg) može prikazati desetke tisuća boja, stoga je pogodan za prikaz fotografija. JPEG format koristi algoritam koji može postići znatnu kompresiju slike. Na primjer, 200KB GIF se može reducirati na 30KB JPEG. Stoga format GIF je bolji za prikaz slika u kojima nema puno boja kao što su crteži i ikone, dok je JPEG bolji za slike s puno boja i za fotografije.

Sliku se u HTML postavlja upotrebom oznake <img> koja ima samo početnu oznaku. Sintaksa je sljedeća:
`<img src="url">`, gdje URL u atributu *src* označava lokaciju gdje se slika nalazi.
Npr. `<img src="ime_slike.ext">` HTML je element koji bi prikazao sliku *ime_slike.ext* u pregledniku.
Navođenjem samo imena slike, bez puta do direktorija, podrazumijeva da se slika nalazi u istom direktoriju kao i HTML datoteka. Inače, treba navesti put do direktorija (apsolutno ili relativno[13]).
Npr. relativni URL
`<img src="../ime_direktorija/ime_slike.ext">`.
Slika može biti i na webu:
npr.
`<img src="http://www.ffst.hr/ime_direktorija/ime_slike.ext">`
bi prikazalo sliku koja se nalazi na navedenom URL-u. Ovaj način se ne preporuča ako autor web stranice nije vlasnik slike koju prikazuje jer uklanjanjem slike s tog URL-a nestaje i slika s web stranice.
Atribut *alt* omogućava ispis alternativnog teksta koji će se pojaviti na mjestu slike ako se iz bilo kojeg razloga slika ne može učitati. Kad je slika učitana, taj tekst se pojavljuje iznad slike prilikom prelaska mišem preko nje.

[13] Bolje je koristiti relativni put do datoteke radi jednostavnijeg prebacivanja datoteka na različite operacijske sustave

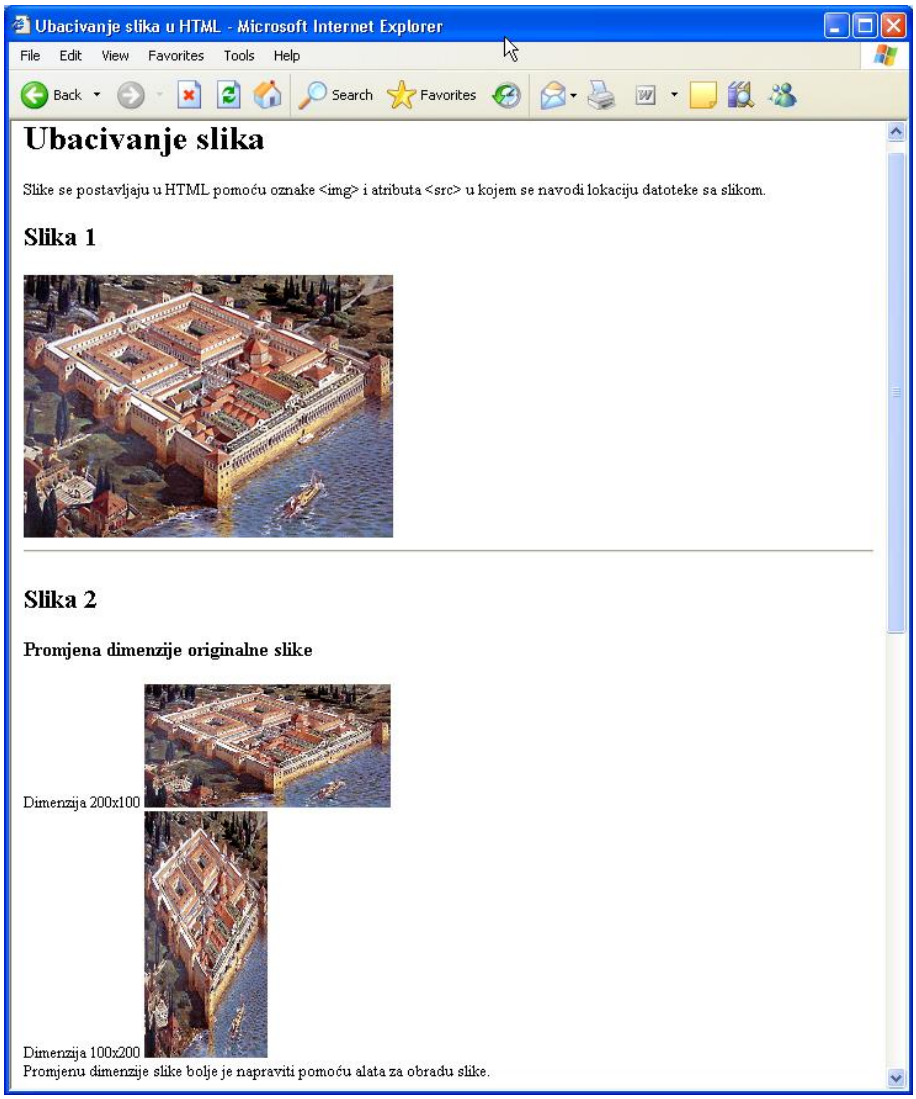
Svakoj slici se može definirati širina i duljina (treba paziti jer to mijenja proporcije slike)

Npr. `<img src="slika.gif" width="200" height="100">` prikazat će sliku s dimenzijom 200x100 pixela.

Slika se može poravnati unutar teksta u kojem se nalazi pomoću atributa *align,* i to po vrhu, sredini i dnu teksta (default je dno teksta).

Npr.

```
<img src="slika.gif" align="top">
<img src="slika.gif" align="middle">
<img src="slika.gif" align="bottom">
```



Slika 4. Izgled web stranice sa slikama

Slika se može poravnati u skladu s okolnim tekstom koristeći također atribut *align* i to lijevo i desno od teksta. HTML 4 i XHTML oznaku i atribut *center* smatraju zastarjelim. Kako bi se slika postavila na sredinu, može se

koristiti tablica u koju će se smjestiti slika. W3C za ovaj problem preporuča korištenje CSS stilova.

Npr.

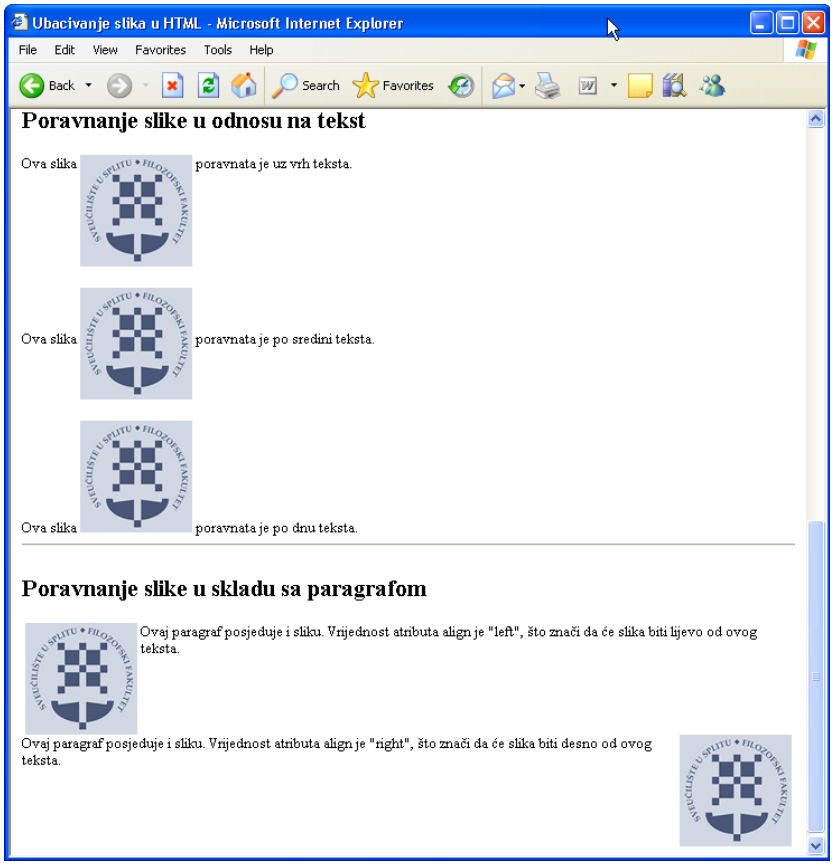
```
<img src="slika.gif" align="left">
<img src="slika.gif" align="right">
```

Primjer 14: U ovom primjeru je na internetu pronađena slika (u .gif formatu) i spremljena u isti direktorij gdje i HTML dokument (pod nazivom) *slika.gif*. Slika *znak.jpg*, snimljena je sa web stranice Filozofskog fakulteta u Splitu. Obje slike ubačene su u HTML dokument (Slika 4 i Slika 5)

```
<html>
<head>
<title>Ubacivanje slika u HTML</title>
</head>
<body>
<h1>Ubacivanje slika</h1>
<p>Slike se postavljaju u HTML pomoću oznake &lt;img&gt; i atributa
&lt;src&gt; u kojem se navodi lokaciju datoteke sa slikom.</p>
<h2>Slika 1</h2>
<img src="slika.gif" alt="ovdje se piše zamjenski tekst">
<hr>
<h2>Slika 2</h2>
<h3>Promjena dimenzije originalne slike</h3>
Dimenzija 200x100
<img src="slika.gif" width="200" height="100" alt="ovdje se piše
zamjenski tekst">
<br>Dimenzija 100x200
<img src="slika.gif" width="100" height="200" alt="ovdje se piše
zamjenski tekst">
<br>
Promjenu dimenzije slike bolje je napraviti pomoću alata za obradu
slike.
<hr>
<h2>Poravnanje slike u odnosu na tekst</h2>
<p>Ova slika <img src="znak.jpg" align="top"> poravnata je uz vrh
teksta.
<p>
Ova slika <img src="znak.jpg" align="middle"> poravnata je po sredini
teksta.
<p>
Ova slika <img src="znak.jpg" align="bottom"> poravnata je po dnu
teksta.
<hr>
<h2>Poravnanje slike u skladu sa paragrafom</h2>
<p><img src ="znak.jpg" align ="left" width="100" height="100">
Ovaj paragraf posjeduje i sliku. Vrijednost atributa align je "left",
što znači da će slika biti lijevo od ovog teksta.</p>
<br clear="all">
```

```
<p><img src ="slika.gif" align ="right" width="100" height="100">
Ovaj paragraf posjeduje i sliku. Vrijednost atributa align je
"right", što znači da će slika biti desno od ovog teksta.</p>
</body>
</html>
```

Primjer 14. Ubacivanje slika u HTML



Slika 5. Poravnanje slike u odnosu na tekst

2.4.4.2. Ostali multimedijalni formati

Osim slike u web stranicu se mogu uključiti i druge multimedijalne datoteke kao što su audio i videodatoteke. Način prikazivanja videodatoteke zavisi od preglednika. Internet Explorer upotrebljava oznaku <img> i svoje atribute controls, dynsrc, loop i start. Na taj način se može ubaciti video u sadržaj web stranice na isti način kao i sliku. Netscape koristi dodatne programe koji se zovu plug-in i zahtijevaju od korisnika da instalira odgovarajući program kako bi pogledao video. IE koristi AVI (Audio Video Interleave) format jer je on dostupan MS Windows operacijskim sustavom.

Npr. <img dynsrc="primjer.avi"> bit će prikazan u web pregledniku IE. Ostali preglednici ako ne mogu prikazati video, prikazat će na tom mjestu zamjensku sliku (npr. prazni prekriženi okvir). Zbog toga se preporuča dodati atribut src,

koji će na tom mjestu u slučaju nemogućnosti učitavanja videa prikazati izabranu sliku, tj. <img dynsrc="primjer.avi" src="slika.gif">
Korisno je dodati atribut controls koji dodaje uobičajenu dugmad kod prikazivanja videa (play, pause, stop, forward, …).

Većina web preglednika tretira audio kao zasebnu datoteku koja se pokreće nekim od pomoćnih programa, appletom ili plug-in programom. Internet Explorer posjeduje ugrađeni audiodekoder i podržava oznaku <bgsound> koja može pokrenuti audio u podlozi web stranice. IE podržava tri audioformata .wav, .au i .midi.
Npr. `<bgsound src="pozdrav.wav">`

Multimedijalni dokumenti se mogu referencirati pomoću oznake za link <a>. Kad preglednik naiđe na link koji nije HTML dokument, automatski poziva odgovarajući program za prikaz te datoteke ili popis programa ako ne zna sam povezati datoteku s odgovarajućim programom.
Npr. `<a href="pjesma.au">Poslušaj pjesmu</a>`

Osim audio i videozapisa na ovaj način mogu se uključivati i druge vrste datoteka. Na primjer, dokument u .pdf formatu koji se čita pomoću programa Adobe Acrobat Reader.

Uobičajeni formati multimedijalnih datoteka
Slika – GIF (.gif), JPEG (.jpg, .jpeg), TIFF (.tif, .tiff), PICT (.pic, .pict), PNG (.png)
Video – MPEG (.mpg, .mpeg), AVI (.avi), QuickTime (.qt, .mov)
Audio – AU (.au), WAV (.wav), AIFF (.aif, .aiff), MIDI (.midi, .mid)
Dokument – PostScript (.ps, .eps, .ai), Acrobat (.pdf)

2.4.5. Linkovi

Glavna osobina HTML je da omogućava međusobno povezivanje HTML dokumenta korištenjem hiperlinkova (u nastavku teksta samo link). Web stranice jedinstvene su upravo po tome što čitatelja kroz tekst ne vode linearno, već se može napraviti „skok“ na drugu web stranicu koja može biti na istom poslužitelju kao i na poslužitelju na najudaljenijem dijelu zemaljske kugle. Prijelaz je u oba slučaja isti i omogućava ga link. Link (veza) se postavlja u HTML dokument pomoću oznake <a> (engl. anchor) i može pokazivati na različita odredišta na internetu. Osim na drugi HTML dokument, može pokazivati na sliku, zvučni i videozapis itd.

Oznaka <a> koristi se na sljedeći način:

```
<a href="url odredišta">tekst linka</a>
```

URL odredišta može biti:

- Druga web stranica (najčešće)
  - npr. link na početnu stranicu Filozofskog fakulteta u Splitu:
    ```
    <a href="http://www.ffst.hr/">Filozofski
    fakultet</a>
    ```
- URL neke druge usluge
  - npr. link na FTP uslugu
    ```
    <a href="ftp://ftp.st.carnet.hr>Javni FTP
    poslužitelj - CARNet Split</a>
    ```
- Lokalna datoteka
  - u istom direktoriju kao i HTML dokument s linkom npr.
    ```
    <a href="datoteka.htm">
    ```
  - u nekom drugom direktoriju i u tom slučaju pišemo relativni put do tog direktorija npr: `<a href="../datoteka.htm">` ili
    ```
    <a href="dir1/datoteka.htm">
    ```
- Odjeljak HTML dokumenta (u tom slučaju prije odjeljka pišemo <a name="ime">)
  - u istoj dateoteci npr. `<a href="#ime">`
  - u drugoj datoteci `<a href="datoteka.htm#ime">`
- E-mail adresa: `<a href="mailto:ime.prezime@pmfst.hr">Ime Prezime</a>`

Relativni URLovi u linku
Relativno napisani URL predstavlja više od kraćeg zapisa. S obzirom na to da je relativan u odnosu na poslužitelj i direktorij, može se prebaciti cijeli skup dokumenata u drugi direktorij ili na drugi poslužitelj i pri tome nema potrebe promijeniti nijedan link.

Sam link, tj. njegovo polazište može biti tekst (kao u prethodnim primjerima) ili slika.

Npr. `<a href="datoteka.htm"><img src="slika.gif"></a>`

Primjer 15: Različite vrste linkova tj. odredišta linkova

```
<html>
<head>
<title>Linkovi</title>
</head>
<body>
<a name="vrh">
<center>
<h1>Linkovi</h1>
<p>
<b>Za postavljanje linkova koristi se oznaka &lt;a&gt; </b>
</center>
<hr width=75%>
<p>
Odredište linka može biti neki HTML dokument na internetu kao što je
početna stranica Filozofskog fakulteta u Splitu.
<br>
Ovo je link na <a href="http://www.ffst.hr/">Filozofski fakultet u
Splitu</a>
<p>
Linkovi mogu pokazivati i na usluge koje nisu WWW, kao npr. na javni
FTP poslužitelj.
<br>
Ovo je link na <a href="ftp://ftp.st.carnet.hr">javni FTP
poslužitelja CARNet Split</a>.
<hr>
<p>
Isto tako možemo postaviti link na neki lokalni dokument koji se
nalazi u istom direktoriju kao i HTML dokument u kojem se nalazi
link.
<p>
Sad ćemo postaviti link na neki od prethodno izrađenih primjera,
točnije na HTML datoteku sa slikama: <a href="primjer14.htm">primjer
web stranice sa slikama</a>.
<p>
Isto tako možemo napraviti link na neki lokalni HTML dokument koji
nije u istom direktoriju. U tom slučaju relativno se piše put do te
datoteke.
<hr>
<p>
Link može biti i slika. <br>
Ova slika je link na web stranicu Filozofskog fakuletata u Splitu<br>
<a href="http://www.ffst.hr/">
<img src="znak.jpg"></a>
<p>
Ako ne želimo rub oko slike, tada se u oznaku img dodaje atribut
border s vrijednosti 0.<br>
<a href="http://www.ffst.hr/">
<img src="znak.jpg" border=0></a>
<hr>
```

```
<p>
Često se na web stranicama pojavljuju i linkovi na e-mail adrese.<br>
Njih pišemo ovako <a href="mailto:ime.prezime@pmfst.hr">Ime
Prezime</a>.
<hr>
<p>
Odredište linka može biti i neki odjeljak HTML dokumenta. U tom
slučaju prije tog odjeljka mora se postaviti sidro, a link treba
pokazivati na to sidro.<br>
Ovo je link na <a href="#vrh">vrh</a> ovog dokumenta.
</body>
</html>
```

Primjer 15.

2.4.6. Boje

Tekst i podloga web stranice mogu se obojiti. Boja se može postaviti na tri načina:

- heksadecimalno #0000ff,
- RGB rgb(0,0,255) i
- imenom blue (što nije po standardu).

Obično se zadaju boje heksadecimalno i taj će način biti opisan.
Boje se postavljaju unutar oznake <body> i u tom slučaju se odnose na cijelo tijelo HTML dokumenta.
Npr. `<body bgcolor="#000000" text="#ffffff" link="#ff010a" vlink="#12ab89" alink="#404000">`

Atributi za bojenje HTML elemenata su:

- bgcolor – boja podloge web stranice
- text – boja teksta
- link – boju linka (još neposjećenog)
- vlink – boja posjećenog linka
- alink – boja aktivnog linka (u trenutku odabira)

Umjesto atributa bgcolor može se upotrijebiti atribut background i odabrati neku sliku kao podlogu web stranice. Npr. `<body background="podloga.gif">`

Heksadecimalni broj za bijelu je #ffffff, a #000000 za crnu. Sve boje se nalaze između ova dva heksadecimalna broja. Nakon znaka #, sljedeća dva znaka određuju količinu crvene, dva znaka iza toga količinu zelene i zadnja dva znaka količinu plave boje u toj boji. Znači da bi #ff0000 bila crvena, #00ff00 zelena, a #0000ff plava boja.

Svaki alat za izradu web stranica u pravilu ima paletu boja iz koje se odabire željena boja elementa web stranice.
Boje, stil i veličina slova mogu se mijenjati samo na nekim dijelovima teksta korištenjem oznake <font> na sljedeći način:
Npr. `<font color="#ff1101" face="Verdana, Arial" size="4">Tekst</font>`
Ako se navede više vrsta slova kao vrijednost atributa, te ako prva vrsta slova ne postoji, tada će se primijeniti druga itd.

Kada se radi o velikoj HTML datoteci ili o većem broju datoteka, ovaj način se ne preporuča. Umjesto toga bolje je koristiti CSS-a (Cascading Style Sheet), tj. stilove koji u posebnoj datoteci definiraju boje, vrste slova itd. Datoteka sa stilovima se pozove na početku svake HTML datoteke.

Primjer 16: HTML dokument u kojem su primjenjene oznake i atributi za boje

```
<html>
<head>
<title>Boje</title>
</head>
<body bgcolor="#000000" text="#ffffff" link="#00ff00" vlink="#0000ff"
alink="#12ab89">
<h2>Boje linkova</h2>
<hr>
<p>Na ovoj stranici ćemo pokazati kako se mogu postavljati boje u
HTML.
<p>
<a href="http://www.ffst.hr">Filozofski fakultet u Splitu</a>
<p>
<a href=“http://www.pmfst.hr“>Fakultet prirodoslovno-matematičkih
znanosti i kineziologije u Splitu</a>
<p>
<a href="http://www.w3schools.com/">HTML tutorial</a>
<hr>
<h2><font color="#ff3456"Boje teksta</font></h2>
<p>
<font color="#00ff00" size="4" face="Verdana">Ovaj tekst je zelene
boje, veličine 4, a vrsta slova je Verdana.</font>
<p>
<font size=5>M</font>ožete samo jednom slovu promijeniti veličinu.
<br>Također, možete samo <font color="#abcdef">jednoj</font> ili
<font color="#fedcba">više riječi</font> promijeniti boju ...
<br>Isto tako i <font face="Helvetica, Arial">vrstu slova.</font>
</body>
</html>
```

Primjer 16.

2.4.7. Tablice

Tablice u HTML ne koriste se samo za prikaz podataka, već i za upravljanje izgledom web stranica. Ubacivanjem teksta, slika i ostalih elemenata u ćelije tablice može se lakše i preciznije pozicionirati elemente na stranici. U odnosu na ostale oznake HTML-a, oznake za tablice su pretrpjele najviše izmjena u standardu 4.0.

Oznake za tablice:

Oznaka	Opis
<table>	Definira tablicu
<th>	Definira naslovni redak
<tr>	Definira red
<td>	Definira ćeliju tablice
<caption>	Definira opis tablice (caption)
<colgroup>	Definira grupu stupaca
<col>	Definira atribute za jedan ili više stupaca u tablici
<thead>	Definira zaglavlje tablice
<tbody>	Definira tijelo tablice
<tfoot>	Definira podnožje tablice

Za izradu tablica nužne su sljedeće HTML oznake:

- <table>...</table> za definiranje tablice
- <tr>..</tr> za definiranje retka
- <td>...</td> za definiranje ćelije u retku (<th>...</th> ako je ta ćelija naslov)

Tablica se kreira redak po redak oznakom <tr>. U svakom retku se definiraju stupci oznakom <td>.

Češći atributi oznake <table> su:

- border – određuje debljinu okvira tablice
- width – širina tablice koja se može zadati u pixelima i postotcima
- cellpadding – udaljenost sadržaja stranice od ruba tablice (u pixelima)
- cellspacing – razmak između dvije ćelije u tablici (u pixelima)

Default vrijednost atributa cellpadding i cellspacing je 1.

Primjer 17: HTML dokument s tablicama (upotrijebljeni atributi za širinu i rub tablice)

```
<html>
<head>
<title>Tablice</title>
</head>
<body>
<h1>Tablice</h1>
```

```
<hr>
<h2>Primjeri tablica</h2>
<p>Tablica s jednim retkom i dva stupca.<br>
<table>
<tr>
      <td>1</td><td>2</td>
</tr>
</table>
<hr>
<p>
Rub oko tablice može se dobiti pomoću atributa border u oznaci
&lt;table&gt;<br>
<p>Tablica s dva retka i tri stupca.<br>
<table border="1">
<tr>
      <td>1</td><td>2</td><td>3</td>
</tr>
<tr>
      <td>4</td><td>5</td><td>6</td>
</tr>
</table>
<hr>
<p>
Širina tablice određuje se pomoću atributa width u oznaci
&lt;table&gt; koji se može zadati u pixelima npr. width=200 i
postotcima npr. width="40%".<br>
<table border="1" width="200">
<tr>
      <td>1</td><td>2</td><td>3</td>
</tr>
<tr>
      <td>4</td><td>5</td><td>6</td>
</tr>
</table>
<p>
<table border="1" width="40%">
<tr>
      <td>1</td><td>2</td><td>3</td>
</tr>
<tr>
      <td>4</td><td>5</td><td>6</td>
</tr>
</table>
</body>
</html>
```

Primjer 17. Tablice

Primjer 18: HTML dokument s tablicama u kojima se koriste atributi za razmake unutar tablice

```
<html>
<head>
<title>Tablice</title>
</head>
<body>
<h1>Tablice </h1>
<hr>
<h2>Razmaci unutar tablice - atributi cellpadding i cellspacing</h2>
<p>
Oznaci &lt;table&gt; mogu se dodati i atributi cellpadding  i
cellspacing.<br>
Cellpading definira broj pixela za koji će sadržaj tablice biti
udaljen od ruba tablice. <br>
Cellspacing definira razmak (u pixelima) između dvije kućice u
tablici.<br>
Default je 1<br><p>
U ovoj tablici cellspacing=5, cellpadding=5
<table border="1" width="40%" cellspacing="5" cellpadding="5">
<tr>
      <td>1</td><td>2</td><td>3</td>
</tr>
<tr>
      <td>4</td><td>5</td><td>6</td>
</tr>
</table>
<p>
U ovoj tablici cellspacing=0, cellpadding=10
<table border="1" width="40%" cellspacing="0" cellpadding="10">
<tr>
<td>1</td><td>2</td><td>3</td>
</tr>
<tr>
      <td>4</td><td>5</td><td>6</td>
</tr>
</table>
<p>
U ovoj tablici cellspacing=10, cellpadding=0
<table border="1" width="40%" cellspacing="10" cellpadding="0">
<tr>
      <td>1</td><td>2</td><td>3</td>
</tr>
<tr>
      <td>4</td><td>5</td><td>6</td>
</tr>
</table>
</body>
</html>
```

Primjer 18. Razmaci unutar tablica

Sljedeći atributi mogu se koristiti kod svih oznaka za tablice, tj. <table>, <tr>, <td> i <th>:

- align (horizontalno poravnanje) s vrijednosti right, center, left
- valign (veritikalno poravnanje) s vrijednosti top, middle i bottom
- bgcolor (boja pozadine ćelije) – vrijednost je neki od heksadecimalnih broj boje

Primjer 19: HTML dokument s tablicama u kojima se koriste atributi za poravnanje tablice i unutar ćelija tablice

```
<html>
<head>
<title>Tablice</title>
</head>
<body>
<h1>Tablice</h1>
<hr><h2>Poravnanja - atributi align i valign</h2>
<p>
Atributi align i valign koriste se u svim oznakama.<br>
Vrijednosti atributa align su left, right i center.<br>
Vrijednosti atributa valign su top, bottom i middle.<br>
Unutar oznaka za tablice mogu se koristiti i atribut za boju
bgcolor.<br>
<p>
<table border="1" width="50%" cellspacing="2" cellpadding="2"
align="center">
<tr bgcolor="#00ff00" align="center">
      <td>1</td><td>2</td><td>3</td>
</tr>
<tr>
      <td>4</td><td bgcolor="#ff0000" align="right">5</td><td>6</td>
</tr>
</table>
<hr><p>
Kada se oznaka &lt;td&gt; zamijeni oznakom &lt;th&gt; tada se radi o
naslovu i tekst u toj ćeliji je napisan masnim slovima.<br>
<p>
<table border="1" width="50%" cellspacing="2" cellpadding="2"
align="center">
<tr bgcolor="#00ff00" align="center">
      <th>1</th><th>2</th><th>3</th>
</tr>
<tr>
      <td>4</td><td>5</td><td>6</td>
</tr>
</table>
</body>
</html>
```

Primjer 19. Tablice - poravnanja

Neki od atributa oznaka <td> i <th>:

- rowspan - proširuje ćeliju na onoliko redaka koliko se navede. Redci na koje se širi ćelija moraju biti definirani.
- colspan - proširuje ćeliju na onoliko stupaca koliko se navede. Stupci na koje se širi ćelija moraju biti definirani kao ćelije u sljedećem ili prethodnom retku.
- nowrap – omogućava da cijeli tekst bude u jednoj liniji (nema preloma).

Primjer 20: HTML dokument s tablicama u kojima se koriste atributi za spajanje ćelija u tablici

```
<html>
<head>
<title>Tablice</title>
</head>
<body>
<h1>Tablice</h1>
<hr>
<h2>Spajanje ćelija – atributi rowspan i colspan</h2>
<p>
Oznakama &lt;th&gt; i &lt;td&gt; mogu se dodijeliti argumenti rowspan
i colspan, koji se koristi u slučaju kad se ćelija visinom (rowspan)
ili širinom (colspan) proteže kroz područje drugih ćelija u
tablici.<br>
Treba voditi računa da broj ćelija mora odgovarati i nakon spajanja
redaka ili stupaca.<br>
<p>
U ovoj tablici su spojene dvije ćelije stupaca u jednu pomoću colspan
atributa.<br>
<table border="1" width="50%" cellspacing="2" cellpadding="2"
align="center">
<tr bgcolor="#00ff00" align="center">
      <td colspan=2>1</td><td>2</td>
</tr>
<tr>
      <td>4</td><td>5</td><td>6</td>
</tr>
<tr>
      <td>7</td><td>8</td><td>9</td>
</tr>
</table>
<hr>
<p>
U ovoj tablici su spojene dvije ćelije redaka u jednu pomoću rowspan
atributa.<br>
<table border="1" width="50%" cellspacing="2" cellpadding="2"
align="center">
<tr bgcolor="#00ff00" align="center">
      <th>1</th><th>2</td><th>3</td>
</tr>
```

```
<tr>
      <td>4</td><td rowspan="2" valign="bottom" align="center">5</td>
      <td>6</td>
</tr>
<tr>
      <td>7</td><td>8</td>
</tr>
</table>
</body>
</html>
```

Primjer 20. Spajanje ćelija u tablici

Kreiranje složenih tablica može se postići oznakama <thead>, <tfoot> i <tbody>. Navedene oznake koriste se za sljedeće:

- <thead> definira red ili redove koji predstavljaju zaglavlje tablice, može se pojaviti samo jedanput u tablici iza oznake <table>
- <tbody> se upotrebljava za podjelu tablice u dijelove, tj. grupiranje redova, može ih biti više unutar tablice
- <tfoot> definira podnožje tablice, može se upotrijebiti samo jedanput i postavlja se iza oznake <tbody>

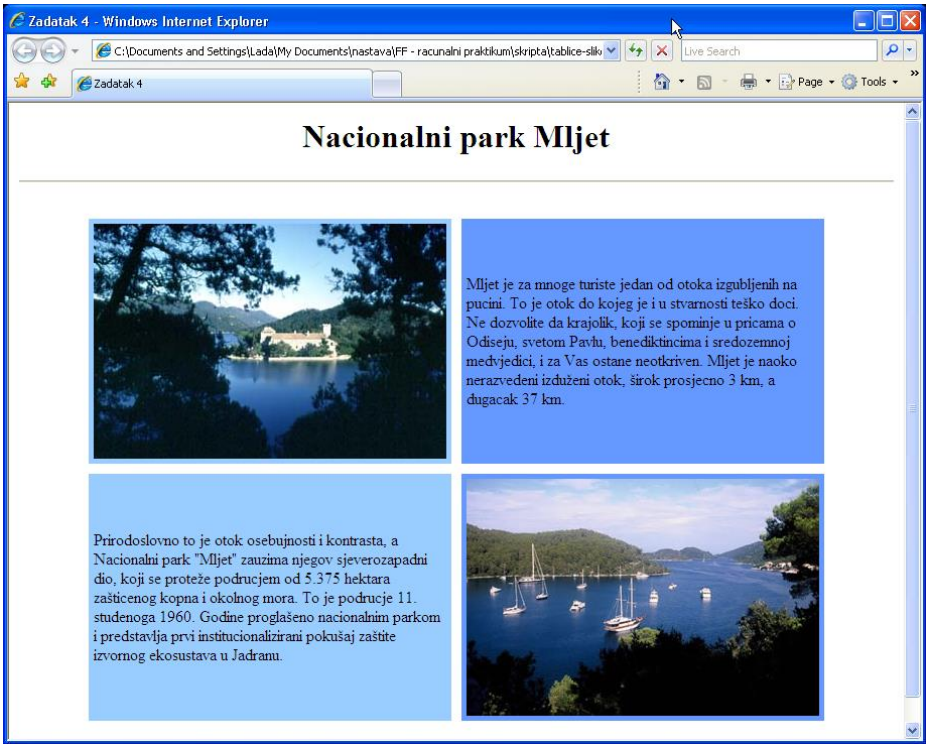
Unutar ćelija tablice može se osim teksta postaviti i bilo koji drugi HTML element (druga tablica, slika i sl).

Primjer 21: HTML dokument s tablicom koja sadrži u sebi drugu tablicu i slike.

```
<html>
<head>
<title>Tablice</title>
<body>
<h1 align="center">Nacionalni park Mljet</h1>
<hr>
<p>
<table align="center" width=62% cellspacing=10 cellpadding=5>
<tr>
<td width="50%" align="center" bgcolor="#99CCFF">
<img src="mljet.jpg"></td>
<td width="50%" bgcolor="#6699FF" align="left">
Mljet je za mnoge turiste jedan od otoka izgubljenih na pucini. To je
otok do kojeg je i u stvarnosti teško doci. Ne dopustiti da krajolik,
koji se spominje u pricama o Odiseju, svetom Pavlu, benediktincima i
sredozemnoj medvjedici, i za Vas ostane neotkriven.
Mljet je naoko nerazvedeni izduženi otok, širok prosjecno 3 km, a
dugacak 37 km. </td>
</tr>
<tr>
<td width="50%" bgcolor="#99CCFF" align="left">Prirodoslovno to je
otok osebujnosti i kontrasta, a Nacionalni park "Mljet" zauzima
```

```
njegov sjeverozapadni dio, koji se proteže podrucjem od 5.375 hektara
zašticenog kopna i okolnog mora. To je podrucje 11. studenoga 1960.
godine proglašeno nacionalnim parkom i predstavlja prvi
institucionalizirani pokušaj zaštite izvornog ekosustava u
Jadranu.</td>
<td width="50%" align="center" bgcolor="#6699FF">
<img src="mljet3a.jpg"></td>
</tr>
</table>
<p>
</body>
</html>
```

Primjer 21. Tablice - napredno



Slika 6. Izgled web stranice iz Primjera 21.

2.4.8. Okviri

Prostor za tijelo HTML dokumenta može se podijeliti na više dijelova pomoću okvira (engl. *frame*). U tom slučaju umjesto oznake <body> koristi se oznaka <frameset>. Unutar oznake <frameset> definira se koliko će biti okvira. Prozor se može podijeliti na redove i stupce (i oboje) pomoću atributa *rows* i *cols*. Atribut *rows* definira podjelu po redovima, a *cols* po stupcima. Za preglednike koji ne podržavaju okvire dodaje se oznaka <noframes> u kojoj se piše tijelo dokumenta.

Općenito može se pisati

```
<frameset rows="h1, h2, ..." cols="v1, v2, ..." >
```

Svakom okviru dodjeljuje se određeni prostor koji se može zadati kao:

- fiksna vrijednost u pixelima, npr. cols="500, 250", što znači da je prostor podijeljen na dva stupca, prvi od 500, a drugi od 250 pixela (ako te veličine ne odgovaraju širini ekrana, tada ih preglednik proporcionalno prilagodi)
- u postocima, npr. rows="10%, 90%", što znači da je prostor podijeljen u dva retka od kojih je prvi visok 10%, a drugi 90% raspoloživog prostora
- kao prostorni omjer, npr. cols="2*,3*", što znači da je prostor podijeljen na stupce u omjeru 2:3
- znakom *, npr. cols="100,200,*", gdje * koja označava preostali prostor. U ovom slučaju radi se o tri stupca: prvi širok 100 px, drugi 200px, a treći ostatak prostora

Pomoću oznake <frame> i njezinih atributa određuje se koji dokument će se učitati u pojedini okvir i na koji način.

```
<frame src="url" name="ime_prozora"
marginwidth="vrijednost" marginheight="vrijednost"
scrolling="yes|no|auto" noresize>
```

Nedostatak korištenja okvira je što se mora voditi računa o više HTML dokumenata koji kreiraju jednu stranicu. Isto tako otežan je ispis stranice s okvirima.

Primjer 22: HTML dokument u kojem je web stranica podijeljena na dva dijela (dva stupca). U prvom se učitava URL http://www.unist.hr/, a u drugom URL http://www.ffst.hr/

```
<html>
<head>
<title>Stranica sa okvirima</title>
</head>
<frameset cols="50%,50%">
<frame src="http://www.unist.hr/">
<frame src="http://www.ffst.hr/">
```

```
</frameset>
<noframes>
Ovo bi vidjeli oni koji imaju preglednik koji ne prepoznaje okvire.
</noframes>
</html>
```

Primjer 22. HTML dokument koji definira okvire

Primjer 23a: HTML dokument u kojem je web stranica podijeljena na dva dijela (dva stupca). U prvom se učitava HTML dokument *izbornik.htm*, a u drugom URL http://www.ffst.hr/. Desni okvir nazvan je "desno".

```
<html>
<head>
<title>Stranica sa okvirima</title>
</head>
<frameset cols="20%,80%" border=0>
<frame src="izbornik.htm" noresize marginwidth=20 marginheight=20>
<frame src="http://www.ffst.hr" name="desno">
</frameset>
</html>
```

Primjer 23a. HTML dokument koji definira okvire

Primjer 23b: HTML dokument *izbornik.htm* koji se učitava u lijevom okviru iz Primjera 23a. Datoteka *izbornik.htm* sastoji se od tri linka od kojih se prvi otvara u novom prozoru (target="_blank"), a ostala dva u desnom okviru (target="desno"). Atribut target ima vrijednost "desno" jer je u datoteci s okvirima pridijeljen taj naziv okviru, tj. postavljen je atribut name="desno".

```
<html>
<head>
</head>
<body>
<a href="http://www.pmfst.hr" target="_blank">Fakultet prirodoslovno-
matematičkih znanosti i kineziologije u Splitu</a>
<br><br>
<a href="http://www.fesb.hr" target="desno">FESB</a>
<br><br>
<a href="http://www.gradst.hr" target="desno">Gradjevinski
fakultet</a>
</body>
</html>
```

Primjer 23b. Datoteka *izbornik.htm*

Datoteke koje se učitavaju u okvirima imaju prazno zaglavlje, tj. naslov web stranice postavlja se u datoteci u kojoj se definiraju okviri.

Atribut target može imati sljedeće vrijednosti:

- target="_blank" – odredište linka se prikazuje u novom prozoru
- target="_self" – odrediše linka se prikazuje u istom okviru kao i njegovo polazište (nije potrebno pisati jer je to default)
- target="_parent" – odrediše linka se prikazuje u neposrednom roditeljskom okviru (ako ga nema, onda djeluje kao i _self)
- target="_top" – odredište linka se prikazuje preko cijelog prozora.

Okviri se mogu gnijezditi. Pretjerana upotreba okvira nije preporučljiva.

Primjer 24: HTML dokument u kojem je web stranica podijeljena na tri okvira. Prvo u dva retka (prvi 10%, a drugi 90%). Drugi redak podijeljen je u dva stupca (20% i 80%)

```
<html>
<head>
<title>Gnježdenje okvira</title>
</head>
<frameset rows="10%,90%" border=0>
      <frame src="gore.htm" name="gore" noresize scrolling=no>

      <frameset cols="20%,80%">
            <frame src="stupac1.htm" name="lijevo" noresize>
            <frame src="stupac2.htm" name="desno">
      </frameset>
</frameset>
</html>
```

Primjer 24. Gniježđenje okvira

2.4.9. Zaglavlje HTML dokumenta

Oznaka koja se najčešće pojavljuje u zaglavlju HTML dokumenta je <meta> i služi za ubacivanje dodatnih podataka o dokumentu (web stranici). Obično su ti podaci namijenjeni web pregledniku, pretraživačkom programu ili služe za opis sadržaja dokumenta. Sadržaj unutar oznake <meta> ne prikazuje se u pregledniku.
Npr.

```
<meta name="description" content="opis stranice">
<meta name="keywords" content="HTML tutorial">
<meta name="author" content="Ime Prezime">
<meta name="revised" content="06.01.2004.">
```

Pretraživački mehanizmi (tražilice) koriste meta oznake pri indeksiranju stranica. Stoga se u oznaku <meta> mogu dodavati ključne riječi koje opisuje web stranicu. Međutim, budući da su mnogi autori web stranica u <meta> oznaci ponavljali ključne riječi kako bi postigli bolji rezultat kod pretraživačkih

mehanizama, mnogi pretraživači su prestali koristiti sadržaj <meta> oznake kod indeksiranja.

Alati za izradu web stranica obično u <meta> oznaku dodaju naziv samog editora u kojem je web stranica izrađena.
Npr. `<meta name="generator" content="Microsoft FrontPage4.0">`
Oznaka <meta> koristi se za redirekciju u slučaju da se URL web stranice promijenio.

```
<meta http-equiv="Refresh" content="5;
url=http://www.novi_url.hr">
```

Primjer 25: HTML dokument u kojem je u oznaci <meta> uključena redirekcija, tj. 6 sekundi nakon učitavanja dogodit će se redirekcija na drugi HTML dokument, tj. URL http:/www.pmfst.hr

```
<html>
<head>
<meta http-equiv="Refresh" content="6;url=http://www.pmfst.hr">
</head>
<body>
<p>
Ova web stranica je premještena.
Novi URL je <a href="http://www.pmfst.hr">http://www.pmfst.hr</a>
koji će se učitati za 6 sekundi.
<br>Ako se to ne dogodi, kliknite na gornji link!
</p>
</body>
</html>
```

Primjer 25. Redirekcija

Odabir kodne stranice piše se također unutar oznake <meta>

```
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-2">
```

Osim oznake <meta> u zaglavlju se može referencirati vanjska datoteka u kojoj se nalaze stilovi ili se unutar oznake <style> (koja se smješta u zaglavlje) piše sam stil. Također se može postaviti oznaka <script> s kodom koji tada postaje dostupan i prije korištenja.

2.4.10. Forme

Komunikacija između vlasnika i posjetitelja web stranice moguća je uporabom formi. Korisnik popunjava obrazac na web stranici i odabirom dugmeta „Submit“ šalje formu poslužitelju ili na elektroničku adresu. Poslužitelj predaje informacije odgovarajućem programu koji obrađuje podatke i može poslati odgovor (obično u obliku HTML-a)[14]. Poslužitelj šalje odgovor pregledniku koji ga prikazuje korisniku. Kod formi koje se šalju elektroničkom poštom podaci se dostavljaju na definiranu adresu i nema obavijesti korisniku o slanju tih podataka. Forme se prikazuju u pregledniku oznakom <form>.
Atributi oznake <form>:

- Method - opisuje kako će podaci prikupljeni u formi biti poslani procesoru forme. Postoje dva načina POST i GET
- Action - definira URL adresu aplikacije koja će prihvatiti podatke iz forme i obraditi ih. Osim aplikaciji, podaci se mogu poslati i elektroničkom poštom.

POST ili GET

POST metodom preglednik šalje podatke u dva koraka: prvo kontaktira poslužitelj koji je naveden u atributu action i tek onda šalje podatke.

GET metoda kontaktira poslužitelj i u jednom koraku šalje podatke iz forme (odvaja ih znakom &).

POST metoda se češće koristi jer je prikladnija za duže forme, sigurniji je prijenos podataka, ali traži od autora poznavanje pisanja aplikacija za poslužitelj.

GET metoda koristi se za kraće forme s manje podataka.

Npr.

```
<form method=”post” action=”http://www.pmfst.hr/cgi-bin/forma.cgi”>
```

Znači da se na URLu http://www.pmfst.hr u direktoriju *cgi-bin* nalazi aplikacija *forma.cgi* koja će primiti i obraditi podatke iz forme. Podaci se šalju POST metodom.
Prosljeđivanje podataka iz forme direktno na elektroničku poštu (bez ikakve obrade ili uobličavanja) može se postići definiranjem linka mailto u URLu.
Npr.

```
<form method=”post” action=”mailto:rpraktikum@pmfst.hr”>
```

Ova metoda trebala bi se koristiti samo za najjednostavnije forme i to samo u slučaju kada preglednik ima integriran program za elektroničku poštu.

[14] Odgovor može biti zahvala za popunjavanje obrasca, upozorenje ako obrazac nije ispravno popunjen, upute kako popuniti obrazac i sl.

Forme se sastoje od kontrola za unos podataka (npr. okvir za tekst, padajući izbornici). Kontrole se prikazuju u pregledniku pomoću oznake <input> koja se postavlja unutar oznake <form>.
Svaka od kontrola ima istu sintaksu:
`<input type=”vrsta_kontrole” name=”naziv_kontrole”>`
Dva najvažnija atributa oznake <input> su name kojom se imenuje kontrola (kako bi se povezao uneseni podatak s kontrolom) i type koji određuje o kakvoj vrsti kontrole je riječ.
Postoji deset uobičajeno korištenih kontrola ulaza na <form>:

1. Text – kreira polje za unos teksta.
2. Password – kreira posebno polje za unos teksta. Upisani znaci ne pojavljuju se na zaslonu kako bi se sačuvala tajnost podataka.
3. Checkbox – kreira okvir u koji se upisuje znak za potvrdu.
4. Radio – slična je kontroli checkbox, važi za cijelu grupu, ali se samo jedna opcija može odabrati.
5. Hidden – omogućava autoru postavljanje imena i vrijednosti kontrola koje korisnici ne moraju mijenjati.
6. Submit – kreira standardnu tipku koja se koristi za naredbu pregledniku za izvođenje neke akcije definirane u elementu <form>.
7. Image – omogućava uporabu slike umjesto tipke Submit.
8. Reset – kreira tipku sličnu Submit tipki kojom se svim elementima forme vraćaju početne vrijednosti.
9. Select – daje listu opcija koje korisnik može izabrati. Lista se može prikazati kao padajući izbornik ili kao okvir.
10. Textarea – kreira područje za unos teksta koji može biti duži od jednog reda.

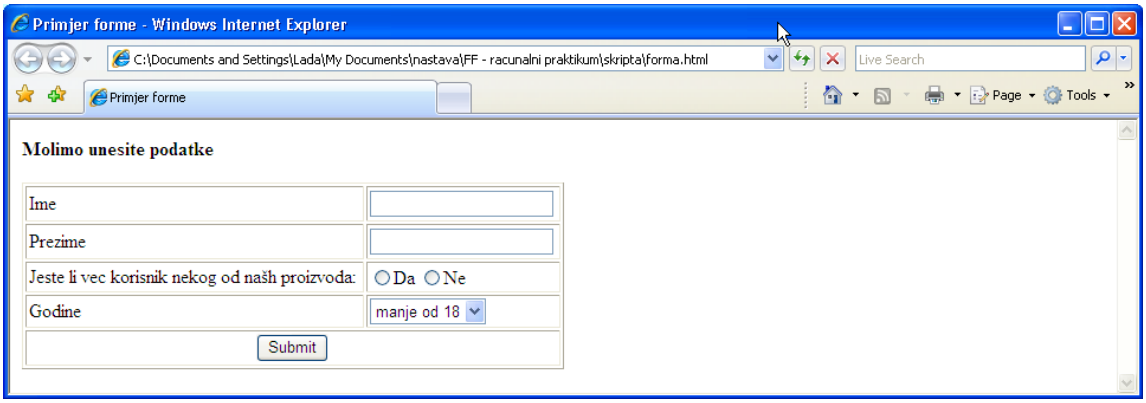
Primjer 26a: HTML dokument koji se sastoji od forme za unos podataka[15]. Upisuju se podaci o imenu, prezimenu, godinama i korištenju proizvoda. Dugme „Submit“ služi za potvrdu upisa.

```
<html>
<head>
<title>Primjer forme</title>
</head>
<body>
<form action="http://www.ffst.hr/forme" method="post">
  <p><strong>Molimo unesite podatke</strong></p>
  <table width="50%"  border="1" cellspacing="2" cellpadding="2">
    <tr>
      <td>Ime</td>
      <td><input name="ime" type="text"></td>
    </tr>
```

[15] Napomena: ovo je samo izgled forme u pregledniku, podaci se mogu poslati i obraditi samo ako se na poslužitelju postavi odgovarajuća aplikacija

```
    <tr>
      <td>Prezime</td>
      <td><input name="prezime" type="text"></td>
    </tr>
    <tr>
      <td>Jeste li već korisnik nekog od naših proizvoda: </td>
      <td><label><input type="radio" name="proizvod" value="Da">Da
</label>
        <label><input type="radio" name="proizvod" value="Ne">Ne
</label></td>
    </tr>
    <tr>
      <td><label>Godine </label></td>
      <td><label>
<select name="godine">
        <option>manje od 18</option>
        <option>18-30</option>
        <option>31-40</option>
        <option>41-50</option>
        <option>51 i više</option>
      </select>
      </label></td>
    </tr>
    <tr>
      <td colspan="2"><div align="center">
        <input type="submit" name="Submit" value="Submit">
      </div></td>
    </tr>
  </table>
  </form>
</body>
</html>
```

Primjer 26a. Forma



Slika 7. Izgled web stranice iz Primjera 26a.

Primjer 26b: HTML dokument iz primjera 26a izmijenjen je u atributu action u oznaci <form>, tako da se podaci šalju na elektroničku adresu

Potrebno je samo izmijeniti oznaku <form> iz prethodnog primjera u:

```
<form action="mailto:ime.prezime@ffst.hr" method="post"
onSubmit="window.alert('Ova forma biti će poslana elektroničkom
poštom')">
…
</form>
```

Primjer 26b. Forma (podaci se šalju elektroničkom poštom)

2.5. Pitanja

1. Zaokruži HTML oznake:
    a. <b>
    b. <bold>
    c. <body>
    d. <image>
2. Zaokruži HTML oznake:
    a. <pre>
    b. <bold>
    c. <body>
    d. <image>
3. HTML oznaka za naglašeni tekst je:
    a. <b>
    b.
    c.
    d. <u>
4. HTML oznaka za stupac tablice je:
    a. <table>
    b. <tr>
    c. <thead>
    d. <td>
5. HTML oznaka za redak tablice je:
    a. <table>
    b. <tr>
    c. <thead>
    d. <td>
6. Za ubacivanje slika koristi se HTML oznaka:
    a. <a>
    b. <image>
    c. <img>
    d. <src>
7. Zaokruži HTML oznake:
    a. <table>
    b. <bold>
    c. <body>
    d. <image>

8. HTML oznaka za link je:
    a. <a>
    b. <href>
    c. <url>
    d. <link>
9. HTML oznaka za neuređene liste je:
    a.
    b.
    c. <dl>
    d. <dd>
10. HTML oznaka za uređene liste je:
    a.
    b.
    c. <dl>
    d. <dd>
11. HTML atribut za boju podloge je:
    a. backbround
    b. color
    c. bgcolor
    d. bgcol
12. Napiši osnovnu strukturu HTML dokumenta

_____
_____
_____
_____
_____
_____
_____
_____

2.6. Postavljanje osobnih web stranica na poslužitelj

Kreirane HTML dokumente i sve ostale datoteke (npr. slike) potrebno je postaviti na poslužitelj kako bi bile dostupne internetom. U nastavku će biti opisano postavljanje web stranica na CARNetove poslužitelje.[16]
Kako bi se datoteke prebacile na poslužitelj potrebno je imati neki od FTP programa. Treba voditi računa da se naslovna stranica mora zvati index.html. Nakon spajanja na poslužitelj FTP programom, datoteke se prebacuju u direktorij public_html[17]. Direktorij public_html je poddirektorij Vašeg kućnog direktorija (http://naziv-posluzitelja/~vaslogin/) na poslužitelju. Dostupnost prebačenih datoteka na webu može se provjeriti na URLu http://www.nazivposlužitelja/~vaslogin
Npr. http://www.ffst.hr/~lmales

FTP (File Transfer Protocol)
FTP protokol koristi se za prijenos datoteka s jednog računala na drugo. (obično s poslužitelja na osobno računalo ili obrnuto). Definiciju FTP-a daje RFC 959. Njegova neobičnost je u tome što ostvaruje istovremeno dvije veze s računalom. Prva veza koristi telnet kako bi prijavila korisnika za rad i obradila naredbe, a druga se veza koristi za proces prijenosa datoteka. Prva se veza drži stalno aktivnom, a druga se zatvara prilikom svakog prijenosa datoteke. Danas postoji niz FTP klijenata (programa koji su namijenjeni klijentskim računalima) i imaju grafičko sučelje. U prošlosti su se koristili ftp programi kojima se pristupalo preko telnet protokola i programa.

Uobičajeni problemi
Na lokalnom računalu kreirane web stranice ne prikazuju se isto na internetu (ne rade linkovi, ne prikazuju se slike i sl)

- Provjeriti sve URLove, tj. ispraviti ih u relativne jer direktoriji koji postoje na Vašem računalu, npr. MyDocuments ne postoje na poslužitelju
- Provjeriti jesu li nazivi pisani malim ili velikom slovima, jer za razliku od Windows operacijskih sustava UNIX ili Linux operacijski sustav koji može biti na poslužitelju razlikuje velika i mala slova u imenu datoteka
- Izbrisati znak razmaka ili promijeniti naša slova iz imena datoteka, ako je greška u tim datotekama
- Provjeriti jesu li poddirektoriji (ako ste ih koristili) preneseni

[16] Ova skripta prvenstveno je namijenjena studentima koji imaju prostor na fakultetskom poslužitelju za postavljanje svojih web stranica u skladu s pravilima CARNeta
[17] Direktorij public_html već je kreiran u Vašem kućnom direktoriju na većini CARNetovih poslužitelja.

2.7. CSS (Cascading Style Sheet)

CSS stilovi definiraju kako će izgledati HTML elementi, a dodani su u standardu HTML4.0. Više puta do sad je spomenuto da W3C preporuča uporabu CSS stilova umjesto nekih oznaka (posebno oznake <font>). Njihova osnovna prednost je što na taj način HTML kod postaje „čistiji“, a jednom definiran stil može se primijeniti na više datoteka.
Tri su načina primjene stilova: u zasebnoj datoteci, uključen u HTML datoteku pomoću oznake <style> u zaglavlju HTML dokumenta i unutar HTML oznake.
Najčešće se koristi prvi način u kojem se stilovi spremaju u datoteku čija ekstenzija je .css. Spremanje stilova u jednu datoteku omogućava da se promjenom u jednoj datoteci izmjene sve HTML datoteke koje primjenjuju stilove iz te datoteke.
Kad se stilovi nalaze u zasebnoj datoteci, pozivaju se unutar oznake <head>:
Npr:

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

Ako je stil uključen unutar datoteke, tada se definira u zaglavlju:
Npr:

```
<head>
<style type="text/css">
hr {color: red}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
</style>
</head>
```

Stilovi se mogu uključiti u HTML oznaku:
Npr:

```
<h1 style="color: blue; font-style: italic">Ovaj naslov
je plave boje</h1>
```

U ovom poglavlju samo smo htjeli naglasiti važnost korištenja stilova pri izradi web stranica, ali ih nećemo obrađivati.

3. JavaScript

JavaScript je programski jezik poput mnogih drugih, ali se izdvaja uporabom za realizaciju programske podrške koja se koristi na Internetu, kako na strani klijenta tako i na strani poslužitelja.

3.1. Uvod u JavaScript

Osnove programiranja JavaScriptom ne razlikuju se značajno od osnova programiranja u programskom jeziku C ili nekom drugom jeziku. Ipak, često se pojavljuju nedoumice prilikom spominjanja Jave i JavaScripta.

3.1.1. JavaScript i Java

JAVA je programski jezik koji je razvila tvrtka Sun Microsystems. Java je jezik čiji se izvorni kod postupkom kompajliranja pretvara u binarni oblik. JavaScript je jezik koji je originalno razvila tvrtka Netscape pod nazivom LiveScript. JavaScript je interpreterski orijentiran jezik. Java i JavaScript su dva potpuno različita programska jezika unatoč sličnosti u nazivu jezika. Svi programski jezici, pa i ova dva, imaju neke sličnosti.
JavaScript trenutno je jedini jezik za pisanje skripta koji podržavaju svi popularni web preglednici. Netscape Navigator podržava samo JavaScript dok Microsoft Internet Explorer podržava i JavaScript i VBScript. JavaScript može se koristiti i za pisanje skripta na strani poslužitelja.

3.1.2. Interpreteri i kompajleri

Prije definiranja razlika između interpretiranih i kompajliranih programa, potrebno je definirati pojam izvorni kod. Izvorni kod je niz naredbi napisanih u tekstualnom obliku od kojih je sačinjen program. Svi programski jezici kreću od izvornog koda koji se tada u zavisnosti od jezika interpretira ili kompajlira.
Jezici koji izvorni kod interpretiraju u pravilu su jednostavniji za programiranje, a sporiji prilikom izvođenja. Svaki put prilikom izvođenja potrebno je intrepretirati kod, i to liniju po liniju u zavisnosti od toka izvršavanja programa. Tok programa određen je grananjem i petljama koje se izvršavaju u programu. Jezici koji kompajliraju kod, uobičajeno imaju složeniju sintaksu i zahtijevaju striktno poštivanje pravila prilikom programiranja. Kod ovako orijentiranih jezika prvo je potrebno ispisati izvorni kod, a nakon toga predati ga kompajleru koji kao rezultat daje izvršni kod u binarnom obliku. Na Windows platformi izlaz iz kompajlera najčešće ima nastavak .exe. Izvršni program koji generira kompajler obično je moguće izvoditi na točno određenoj platformi (operativnom sustavu). Značajna prednost za programera je činjenica da izvorni kod nije čitljiv nakon kompajliranja, a druga je prednost što prestaje biti bitno u kojem je programskom jeziku pisan neki program.

JAVA je jezik koji iz izvornog koda kompajliranjem dolazi do izvršnog koda koji ne ovisi o platformi (operacijskom sustavu) na kojoj će se program izvršavati. Nezavisnost o platformi osigurava preglednik preko JAVA Virtual Machine i interpretera za JavaScript. Iz ovoga je lako zaključiti kako je vrlo bitno za koji se preglednik pišu skripti.

JavaScript može proširiti korisnost web stranica u odnosu na one stranice koje se oslanjaju samo na HTML. Uporabom JavaScripta moguće je provjeravati točnost unosa podataka od strane korisnika, kreirati zanimljive efekte i otvarati prozore koji će se pojaviti kada se dogodi (okine, eng. *triggers*) neki od predefiniranih događaja. Ako se JavaScript kombinira sa CSS-om, tada se dobiva stranica koja je poznata pod nazivom Dinamička HTML stranica.

3.1.3. O JavaScriptu

JavaScript je interpreterski orijentiran programski jezik koji je moguće uključiti u HTML dokument (web stranicu). Pojam interpreterski orijentiran znači da će se u preglednik učitati cijela stranica, a JavaScript kod će se izvršiti po okidanju nekog događaja. Za vrijeme izvođenja programa kod se interpretira liniju po liniju. Postoje brojni događaji, poput klika na dugme ili završetka učitavanja stranice koji će prouzročiti okidanje, a time i izvršavanje nekog dijela koda.

Netscape je tvrtka koja je kreirala JavaScript, ali je jezik nakon toga standardiziran od European Computer Manufacturers Association (ECMA). Danas postoji više inačica JavaScripta (1.0, 1.1, 1.2,...), a jezik se kontinuirano razvija s razvojem interneta i web tehnologija.

3.1.4. Pitanja

1. JavaScript je interpreterski orijentiran jezik (Da ili Ne)
2. JAVA je _____ orijentiran programski jezik i _____ o platformi na kojoj se izvodi.
3. JAVA i JavaScript proizvod su iste tvrtke (Da ili Ne)
4. Microsoft Internet Explorer podržava skript jezike:
    a. JavaScript
    b. JAVA
    c. Basic
    d. VBScript
    e. C++
    f. Perl
5. JavaScript podržana je u većini novih web preglednika (Da ili Ne)

3.2. JavaScript sintaksa

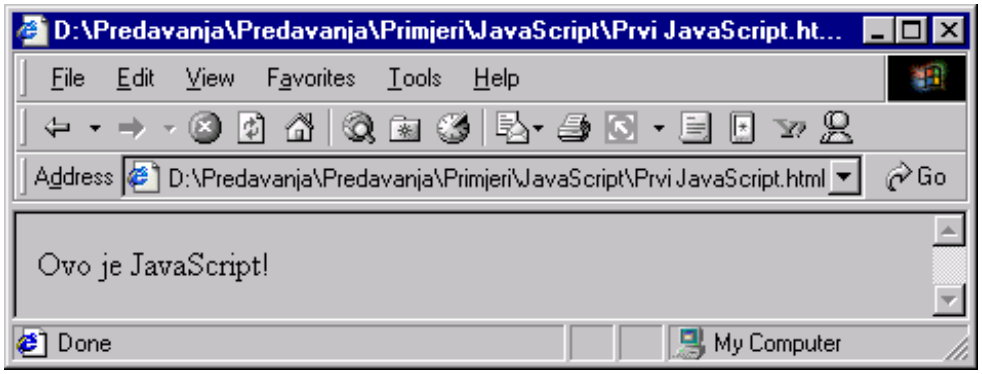
3.2.1. Uključivanje JavaScripta u HTML dokument

JavaScript kod dodaje se u HTML dokument uporabom oznake <script>

Atribut:	Vrijednost:	Opis:
Type	text/ecmascript text/javascript text/jscript text/vbscript text/vbs text/xml	OVAJ JE ATRIBUT OBVEZAN ! Tip skripta (MIME type of the script)
language	javascript livescript vbscript *other*	Zastarjeli atribut. Ne koristiti!
charset	*charset*	Oznaka kodne stranice koja se koristi
Defer	false true	Indikator koji kaže kako preglednik može pričekati učitavanje ostatka dokumenta prije početka izvršavanja skripta.
Src	*url*	URL prema datoteci koja sadržava skript

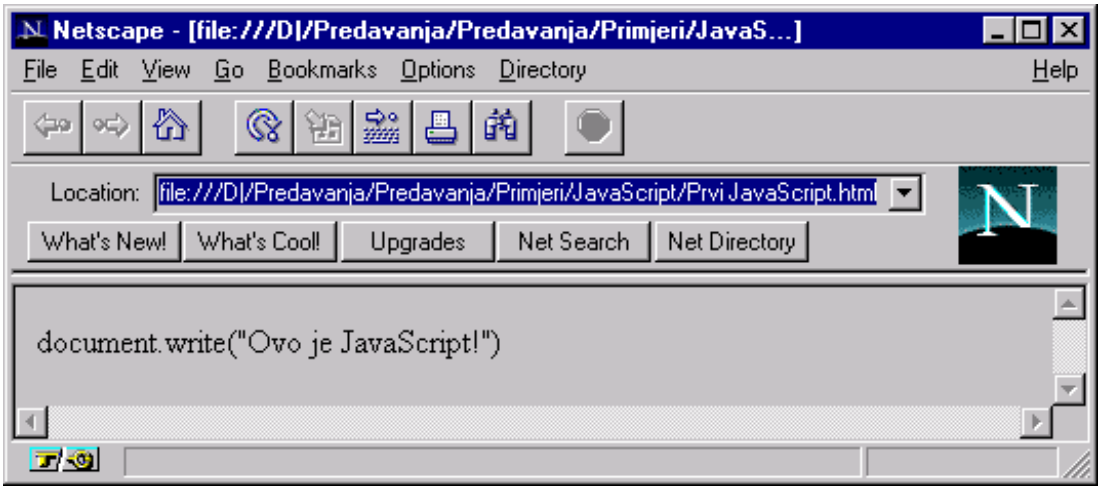
```
<html>
<head>
</head>
<body>
<script type="text/javascript">
  document.write("Ovo je JavaScript!")
</script>
</body>
</html>
```

Primjer 27. Prvi JavaScript.html



Slika 8

Identičan primjer učitan u stari preglednik koji ne podržava JavaScript izgleda ovako:



Slika 9

Ako se oznaka <script> postavi u zaglavlje dokumenta (u slučaju da stranicu učitava preglednik koji ne podržava oznaku <script>), taj će dio HTML dokumenta biti zanemaren od preglednika. Starija generacija preglednika zanemaruje oznake koje nije u mogućnosti interpretirati.

JavaScript kod koji se stavlja u tijelo dokumenta kod starijih preglednika bit će prikazan kao sadržaj dokumenta. Kako bi se izbjeglo prikazivanje koda skripta koji preglednik nije u mogućnosti izvršiti, moguće je koristiti HTML oznake komentara <!--...-->.

U primjeru bi to izgledalo ovako:

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
  <!-- Skrivanje JavaScript koda od starijih preglednika
  .
  .       (JavaScript kod)
  .
  // kraj skrivanja JavaScript koda -->
</script>
</body>
</html>
```

Primjer 28.

Oznake ***//*** u JavaScriptu imaju značenje komentara koji će spriječiti interpretiranje te linije koda.

> **Napomena:** stavljanje oznake za JavaScript komentar ispred HTML oznake za početak komentara (//<!--) kod starijih preglednika rezultira ispisom oznake JavaScript komentara kao sadržaja dokumenta. Po logici stvari to ne bi trebao biti slučaj, ali je to činjenično stanje.

3.2.2. Smještanje JavaScripta

Skripti se izvršavaju odmah nakon učitavanja u preglednik. Ovaj način obrade skripta nije uvijek poželjan. Ponekad je skript potrebno izvršiti za vrijeme učitavanja stranice, a nekad samo kada se aktivira neki od okidača.
Skripti koji će se izvršavati po pozivu ili na određeni događaj, stavljaju se u zaglavlje HTML dokumenta. Ovim se osigurava dostupnost skripta (skript je učitan) prije korištenja.
Skripti koji će se izvršiti za vrijeme učitavanja stranice stavljaju se u tijelo HTML dokumenta. Takvi skripti uobičajeno služe za generiranje sadržaja stranice.
Broj skripta koji se mogu uključiti u HTML dokument nije ograničen. Dopušteno je definirati skripta i u zaglavlju i u tijelu dokumenta.

3.2.3. Sintaksa i konvencije

Pisanje u bilo kojem jeziku prate pravila i konvencije. Hrvatski jezik zahtijeva uporabu velikog slova na početku rečenice, a kraj rečenice označava se točkom, uskličnikom ili upitnikom. To je sintaksa ili gramatika jezika. Svaki programski jezik ima sličan niz pravila koja se nazivaju sintaksa jezika.

JavaScript ima nekoliko pravila i konvencija:

3.2.3.1. Osjetljivost na velika i mala slova

JavaScript je jezik koji razlikuje velika od malih slova, što znači kako se riječi iznos, Iznos i IZNOS tretiraju kao različite riječi.

3.2.3.2. Točka-zarez

Svi izrazi trebaju završavati točka-zarezom (;). Točka-zarez razdvaja dva izraza.

```
var x = 0; var y = 10;
```

3.2.3.3. Razmaci

JavaScript kao i HTML ignorira razmak, tab i novi red koji se pojavljuju unutar izraza. JavaScript raspoznaje razmake, tabove i novi red kao dijelove stringa.

var x = 0; isto je kao i var x=0;

Rezultat obaju primjera isti je, ali je dobro koristiti razmake kako bi se poboljšala čitljivost koda. Nije dobro naredbu razvući preko više redova, iako je to sintaksom JavaScripta dopušteno.

Prazna mjesta obvezna su kako bi se razdvojile naredba i varijabla. Primjerice varx=0; nije ispravno napisan izraz jer je razmak između var i x neophodan za pravilno tumačenje izraza.

3.2.3.4. String i navodnici

String je niz znakova omeđenih jednostrukim ili dvostrukim navodnicima ('jednostruki', “dvostruki”).
Dopušteno je gnježđenje dvostrukih navodnika unutar jednostrukih i obratno.('Rekao je: “JavaScript je zanimljiv jezik.” ')
Češće će se naići na ovakav primjer:

```
<html>
<head>
</head>
<body>
<input type="Button" value="Kliknite na dugme"
onclick="window.alert('Kliknuli ste dobro');">
</body>
</html>
```

Primjer 29. String i navodnici.html

U ovom primjeru dvostruki navodnici koriste se kako bi se odredio atribut HTML oznake. Iz tog razloga poruku koja će se pojaviti u prozoru trebalo je omeđiti jednostrukim navodnicima.

Slika 10

3.2.3.5. Znak backslash (\) i stringovi

Backslash ima posebno značenje u JavaScript stringu. Ovaj znak slijedit će drugi koji nije moguće otipkati na tipkovnici.

Primjerice, ako želimo riječ “Kliknuli” prikazati u jednom redu, riječ “ste” u drugom redu, a riječ “dobro” u trećem redu, string bi izgledao ovako:

```
'Kliknuli\nste\ndobro'
```

\n je kontrolni znak koji zamjenjuje dva znaka CR i LF (oznake za novi red i početak reda). Rezultat je:

Slika 11

Kombinacija backslash znaka i slova obično se naziva escape sekvencom. Neke od uobičajenih sekvenci su:

- \b backspace – jedno mjesto prema natrag
- \f form feed – početak stranice
- \n new line – novi red
- \r carriage return – početak reda
- \t tab
- \' jednostruki navodnik
- \” dvostruki navodnik

Posljednje dvije sekvence bitne su jer omogućavaju ispis navodnika bez prethodne interpretacije, što je naročito važno za značenje jednostrukih navodnika u engleskom jeziku.

3.2.3.6. Otvaranje i zatvaranje zagrada

Sve otvorene zagrade moraju se zatvoriti! Ovo uključuje (), [], i {}.

```
winpop = window.open('primjer1.html','popup','scrollbars=yes')

      if ( x[0] == 10 )
                              {
            x[0] = 0;
            x[1] = 0;
            }
```

Vitičaste zagrade { } koriste se za omeđivanje više JavaScript izraza.
Uglate zagrade [ ] dio su posebne podatkovne strukture koja se naziva polje.
Okrugle zagrade ( ) omeđuju funkcije ili argumente metode.

3.2.3.7. Komentari

Komentar se naznačuje dvostrukom kosom crtom //. Ovakav način označavanja komentara prihvatljiv je za komentar koji je ograničen na jednu crtu koda, ali ako se želi komentirati veći dio teksta u kodu, poželjno je koristiti /* kao početnu oznaku komentara i */ kao završnu oznaku komentara.

U primjeru bi to izgledalo ovako:

```
/* Komentari često služe programerima
Kako bi objasnili logiku koja stoji iza nekog
dijela koda ili razlog iz kojeg je kod izmijenjen.
Ovo se čini kako bi se lakše razumjele akcije
programera nakon nekog duljeg vremena.
 */
```

3.2.3.8. Varijable i imena funkcija

Kao programer potrebno je izabrati i dodijeliti nazive varijabli i funkcija.
Nazivi varijabli i funkcija moraju pratiti nekoliko jednostavnih pravila:

Prvi znak mora biti slovo abecede (malo ili veliko slovo), donja crta (_) ili oznaka za dolar ($). Oznaka $ ne preporučuje se jer je podržana tek nakon inačice 1.1 JavaScripta.
NIJE DOPUŠTENO koristiti brojeve kao prvi znak u nazivu
NIJE DOPUŠTENA upotreba razmaka u nazivu
NIJE DOPUŠTENA upotreba rezerviranih riječi u nazivu

Evo nekoliko primjera ispravnih naziva

1. X
2. zbroj_dva_broja
3. x13
4. _sve
5. $iznos_u_dolarima

Preporučuje se pri izboru imena birati nazive koji će asocirati na varijablu ili funkciju koja se piše. Kako JavaScript razlikuje mala od velikih slova, uobičajeno je umjesto naziva zbroj_dva_broja pisati ZbrojDvaBroja. Nezavisno od toga koja se konvencija koristi, bitno je znati kako je bilo koja konvencija bolja od nikakve konvencije.

3.2.3.9. Rezervirane riječi

Postoje brojne riječi koje su sastavni dio JavaScript jezika. Te riječi nije dopušteno koristiti za nazive varijabli i funkcija jer interpreter ne bi bio u mogućnosti razlučiti što je izvorna naredba jezika, a što varijabla ili funkcija.

Lista rezerviranih riječi u JavaScriptu:

abstract	delete	innerWidth	Packages	status
alert	do	instanceof	pageXOffset	statusbar
arguments	document	int	pageYOffset	stop
Array	double	interface	parent	String
blur	else	isFinite	parseFloat	super
boolean	enum	isNaN	parseInt	switch
Boolean	escape	java	personalbar	synchroniz
break	eval	length	print	this
byte	export	location	private	throw
callee	extends	locationbar	prompt	throws
caller	final	long	protected	toolbar
captureEvents	finally	Math	prototype	top
case	find	menubar	public	toString
catch	float	moveBy	RegExp	transient
char	focus	moveTo	releaseEvents	try
class	for	name	resizeBy	typeof
clearInterval	frames	NaN	resizeTo	unescape
clearTimeout	Function	native	return	unwatch
close	function	netscape	routeEvent	valueOf
closed	goto	new	scroll	var
confirm	history	null	scrollbars	void
const	home	Number	scrollBy	watch
constructor	if	Object	scrollTo	while
continue	implements	open	self	window
Date	import	opener	setInterval	with
debugger	in	outerHeight	setTimeout	FALSE
default	Infinity	outerWidth	short	TRUE
defaultStatus	innerHeight	package	static	

Današnji alati automatski označe ključnu riječ drugom bojom u samom uređivaču koda, pa su sve rjeđe pogreške tipa korištenja rezerviranih riječi za nazive varijabli. Važno je odabrati dobar razvojni alat kako bi se greške prilikom pisanja koda minimizirale. Pored komercijalnih razvojnih alata sve češće mogu se naći kvalitetna rješenja temeljena na otvorenom kodu.

3.2.4. Pitanja

1. Koji su od nabrojenih naziva ispravni nazivi funkcija ili varijabli?
    a. Y
    b. 100posto
    c. vrlo veliki broj
    d. break
    e. substractTwoNumbers
    f. Ulica_Stanovanja

2. Nadopunite kod zagradama na mjesta koja su obilježena __

    If __ z__0__ == 0__ ___
        x = 2;
        __

3. Nadopunite rečenicu: Svaki izraz treba završiti _____.

4. JavaScript razlikuje velika i mala slova (Da ili Ne)

5. Dobro je dodavati komentare u kodu (Da ili Ne)

6. Dodijelite zagrade iz stupca A načinu uporabe u stupcu B

Stupac A	**Odgovor (slovo iz stupca A)**	**Stupac B**
a. { }		dio su posebne podatkovne strukture koja se naziva polje
b. [ ]		omeđuju funkcije ili argumente metode
c. ( )		koriste se za omeđivanje više JavaScript izraza

3.3. Osnove programiranje JavaScriptom

3.3.1. Deklariranje varijabli

Varijabla je ime koje se daje memorijskoj lokaciji računala u kojoj se spremaju podaci. Varijabla je kontejner za informaciju koju se želi pohraniti. Vrijednost varijable može se mijenjati za vrijeme izvođenja skripta. Varijabli se može pristupiti preko imena varijable kako bi se pročitala ili izmijenjila njezina vrijednost.
Deklaracija varijable (kreiranje varijable) vrši se ključnom riječi var:

```
var x;
var y;
var zbroj;
```

Više varijabli može se deklarirati jednom var ključnom riječi

```
var x,y,zbroj;
```

Varijablu je u JavaScriptu moguće kreirati i direktnom inicijalizacijom vrijednosti varijable:

```
x=0;
y=0;
zbroj=0;
```

Iako je ovakav način dopušten, zbog preglednosti je bolje prije inicijalizacije izvršiti deklaraciju ili barem kombinirati deklaraciju i inicijalizaciju kao što se može vidjeti u ovom primjeru:

```
var x=1, y=3, zbroj=0;
```

Ako se posebno ne inicijalizira, deklarirana varijabla poprima posebnu vrijednost JavaScripta undefined (nedefiniranu) vrijednost.

3.3.2. Tipovi varijabli

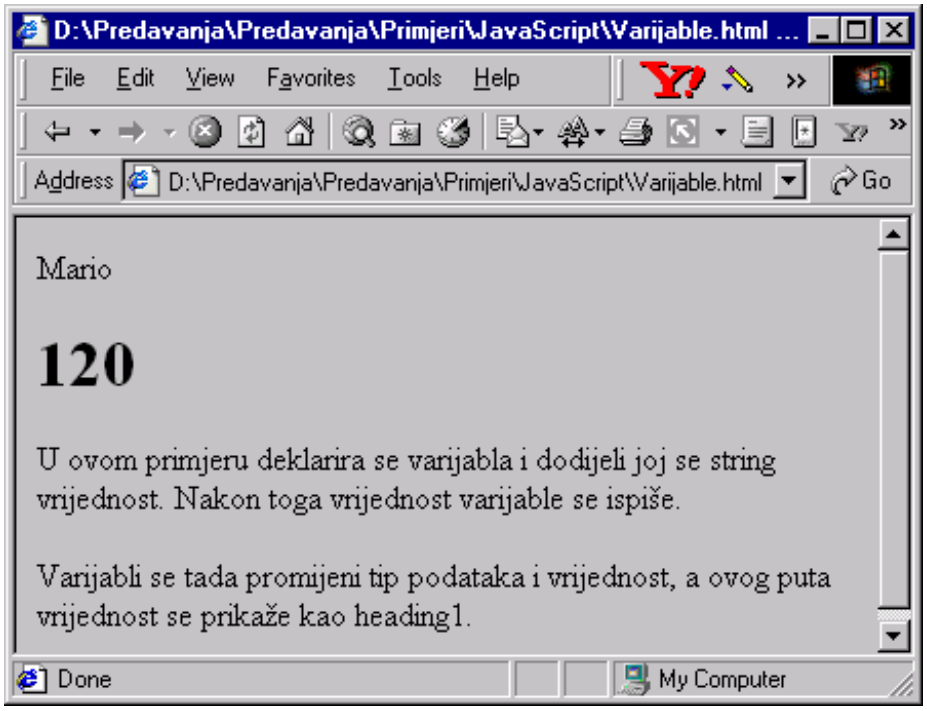
Velika razlika između JavaScripta i drugih jezika poput JAVE i C-a je u tome što JavaScript nije orijentiran prema tipu varijable. To znači da varijabla u JavaScriptu može pohraniti bilo koji tip podataka iako ga se nije definiralo prilikom deklaracije varijable. Ovo omogućava izmjenu tipa podatka varijable za vrijeme izvođenja skripta.

```
var x = 10;
x = "deset";
```

U ovom primjeru varijabli x prvo je dodijeljena cjelobrojna vrijednost 10, a nakon toga string vrijednost riječi deset.

```
<html>
<body>
<script type="text/javascript">
   var ime = "Mario"
   document.write(ime)
   ime=120
   document.write("<h1>"+ime+"</h1>")
</script>
<p>U ovom primjeru deklarira se varijabla i dodijeli joj se string
vrijednost. Nakon toga vrijednost varijable se ispiše.</p>
<p>Varijabli se tada promijeni tip podataka i vrijednost, a ovog puta
vrijednost se prikaže kao heading1.</p>
</body>
</html>
```

Primjer 30. Varijable



Slika 10.

3.3.3. Uporaba operatora

Operatori djeluju na varijable. U primjeru je korišten operator dodjele vrijednosti varijabli =. Još jedan od operatora je +.

```
var x = 1, y = 3, zbroj = 0;
zbroj = x + y
```

Ovaj djelić koda deklarira varijable x,y i zbroj. Dodijeli varijablama x, y i zbroj vrijednosti 1, 3 i 0 respektivno. Druga crta skripta zbraja vrijednosti x i y i dodijeli tu vrijednost varijabli zbroj. Vrijednost varijable zbroj bit će 4 nakon izvršenja koda.

Drugi operatori koriste se za uspoređivanje:

```
var x = 1, y = 3, zbroj = 0;
if ( zbroj == 0 ) {
            zbroj = x + y;
}
```

Dodjela vrijednosti		
Operator	**Primjer**	**Rezultat**
=	I = 5	i jednako 5
+=	I += 5	i jednako i + 5
-=	I -= 5	i jednako i – 5
***=**	I *= 5	i jednako i * 5
/=	I /= 5	i jednako i / 5
%=	I %= 5	i jednako i %5
++	i++	i jednako i+1
--	i--	i jednako i-1
-	X – Y	Oduzmi vrijednost Y od vrijednosti X
+	X + Y	Pribroji vrijednost Y vrijednosti X
/	X / Y	Podijeli X sa Y
*****	X * Y	Pomnoži vrijednost X sa vrijednosti Y
%		Modulo

Logički operatori		Operatori nad bitovima	
Operator	**Značenje**	**Operator**	**Značenje**
==	jednako	**~**	Ne
!=	nije jednako	**<<**	Posmak (shift) u lijevo
>	veće od	**>>**	Posmak (shift) u desno
>=	veće ili jednako	**>>>**	Unsigned posmak (shift) u desno
<	manje od	**&**	i AND
<=	manje ili jednako	**^**	XOR
||	logičko ili	**|**	OR
&&	logičko i		

3.3.4. Vidljivost varijable

Kad se varijabla deklarira unutar funkcije, ona je vidljiva samo unutar te funkcije. Kada se funkcija napusti, varijabla se uništi. Ove varijable nazivaju se lokalne varijable. Moguće je koristiti varijable s istim imenom u više funkcija jer je svaka od njih vidljiva samo unutar funkcije u kojoj je deklarirana.
Ako se varijabla deklarira izvan funkcije, tada je ona vidljiva od svih funkcija. Život ovako definirane varijable počinje njezinom deklaracijom, a završava zatvaranjem stranice.

3.3.5. Upravljačke strukture

3.3.5.1. Uvjetne naredbe

Uvjetne naredbe omogućavaju selektivno izvođenje dijelova programa.
If je osnovna upravljačka naredba koja omogućuje programu provjeravanje nekog uvjeta te nastavak izvršavanja u zavisnosti od rezultata tog testa.

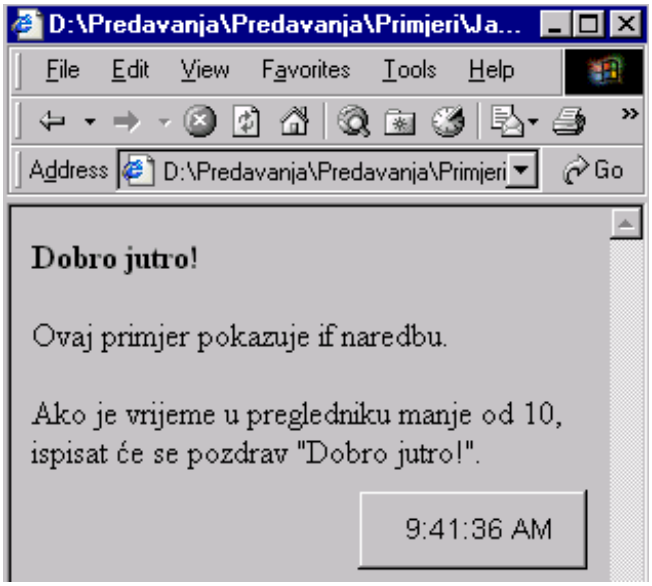
Primjer jednostrukog odabira:

```
if ( (x==1)&&(y==3))
{      sum = y-x;
}
```

U ovom izrazu provjerava se je li vrijednost varijable x jednaka 1 i je li vrijednost varijable y jednaka 1. U ovom slučaju oba uvjeta moraju biti zadovoljena jer se između njih nalazi operator (&&) logičko i. Ako ovaj uvjet nije zadovoljen, izraz se neće izvršiti.
Kod dvojnog izbora postoji mogućnost izvršavanja jedne od dviju naredbi u zavisnosti od uvjeta koji je ispunjen.

```
<html>
<body>
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time < 10)
{
document.write("<b>Dobro jutro!</b>")
}
</script>
<p>
Ovaj primjer pokazuje if naredbu.<p>Ako je vrijeme u pregledniku
manje od 10,
ispisat će se pozdrav "Dobro jutro!". </p>
</body>
</html>
```

Primjer 31. Jednostruki odabir



Slika 11.

Primjer dvojnog odabira:

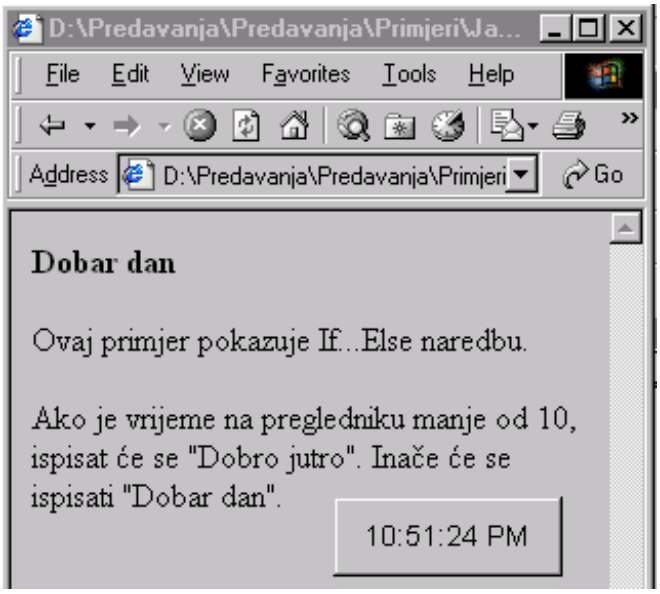
```
if (sum==0)
   {
      sum=x+y;
   }
else
   {
            subtotal=sum;
   }
```

Ovaj izraz čita se kao: ako je sum jednak 0, onda je sum jednak zbroju x i y; inače subtotal je jednak sum.

```
<html>
<body>

<script type="text/javascript">
var d = new Date()
var time = d.getHours()

if (time < 10)
   {
    document.write("<b>Dobro jutro.</b>")
   }
else
   {
    document.write("<b>Dobar dan</b>")
   }
</script>
```

```
<p>
Ovaj primjer pokazuje If...Else naredbu.<p>Ako je vrijeme na
pregledniku manje od 10,
 ispisat će se "Dobro jutro". Inače će se ispisati "Dobar dan".
</body>
</html>
```

Primjer 32. Dvojni odabir, if ... else



Slika 12.

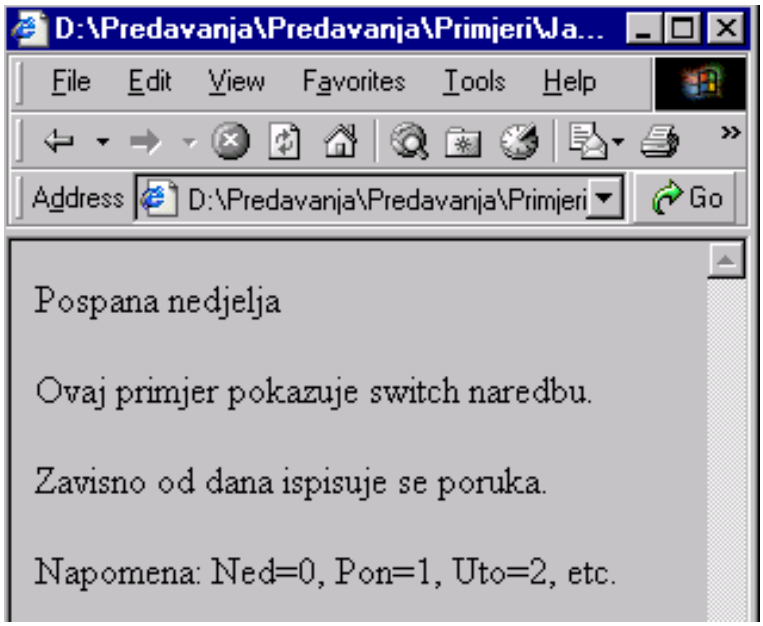
Switch – case naredba je pogodna kod višestrukog grananja programa. Ona omogućava izbor jednog između više mogućih putova daljnjeg izvođenja programa, što ovisi o vrijednosti koju poprima varijabla odnosno izraz.

Primjer za switch-case:

```
switch(n)
{
      case 1: // počni od ove točke ako je n jednak 1
            // kod
      break;

      case 2: // počni od ove točke ako je n jednak 2
            // kod
      break;
.
.
.
default; // ako nije zadovoljen ni jedan od prethodnih uvjeta izvrši
// kod
break;
}
```

```
<html>
<body>
<script type="text/javascript">
var d = new Date()
Dan=d.getDay()
switch (Dan)
{
      case 5:
            document.write("Konačno petak")
      break
      case 6:
            document.write("Savršena subota")
      break
      case 0:
            document.write("Pospana nedjelja")
      break
      default:
            document.write("Jedva čekam kraj tjedna!")
}
</script>
<p>Ovaj primjer pokazuje switch naredbu.</p>
<p>Zavisno od dana ispisuje se poruka.</p>
<p>Napomena: Ned=0, Pon=1, Uto=2, itd.</p>
</body>
</html>
```

Primjer 33. Poruke ovisne o danu u tjednu



Slika 13.

3.3.5.2. Petlje

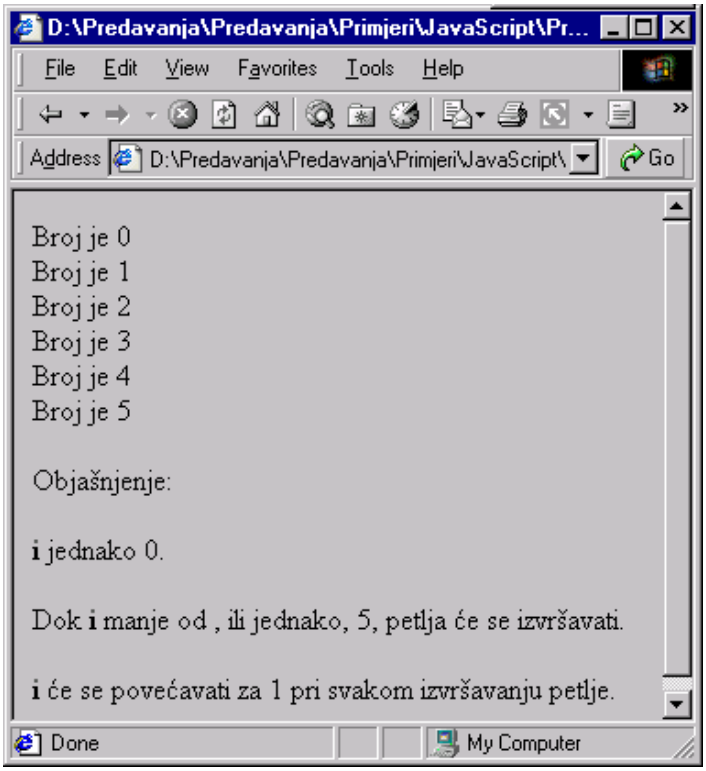
Petlja je programska struktura koja izvršava naredbe koje se nalaze unutar strukture sve dok je zadovoljen uvjet. U trenutku kada uvjet prestane vrijediti, petlja se završava.

While

While petlja izvršava jednu ili više naredbi, sve dok je uvjet zadovoljen. While petlje posebno su korisne kada se ne zna koliko puta treba ponoviti petlju, ali je poznato u kojem slučaju petlju treba napustiti.

```
<html>
<body>
<script type="text/javascript">
i = 0
while (i <= 5)
 {
  document.write("Broj je " + i);
  document.write("<br />");
  i++;
 }
</script>
<p>Objašnjenje:</p>
<p><b>i</b>  jednako 0.</p> <p>Dok <b>i</b> manje od , ili jednako,
5, petlja će se izvršavati.</p> <p><b>i</b> će se povećavati za 1 pri
svakom izvršavanju petlje.</p>
</body>
</html>
```

Primjer 34. While petlja



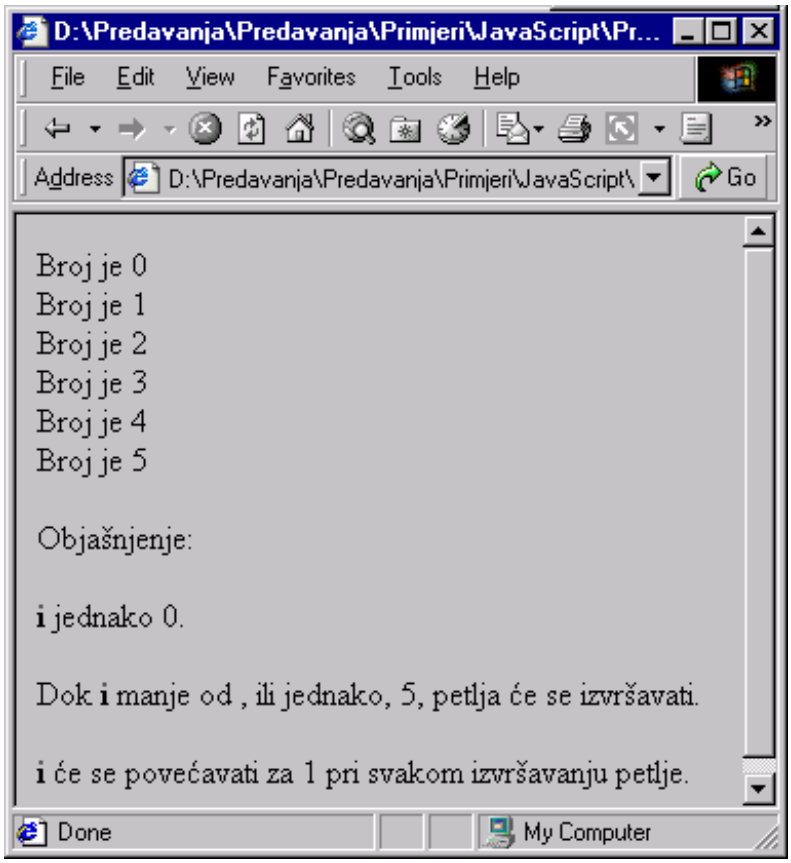
Slika 14.

Do ... while

Ova petlja identična je prethodnoj, ali s razlikom što će se svakako izvršiti barem jedanput.

```
<html>
<body>
<script type="text/javascript">
i = 0
do
{
  document.write("Broj je " + i);
  document.write("<br />");
  i++;
} while (i <= 5)
</script>
<p>Objašnjenje:</p>
<p><b>i</b>  jednako 0.</p>
<p>Dok <b>i</b> manje od , ili jednako, 5, petlja će se
izvršavati.</p>
<p><b>i</b> će se povećavati za 1 pri svakom izvršavanju petlje.</p>
</body>
</html>
```

Primjer 35. Do ... while petlja



Slika 15.

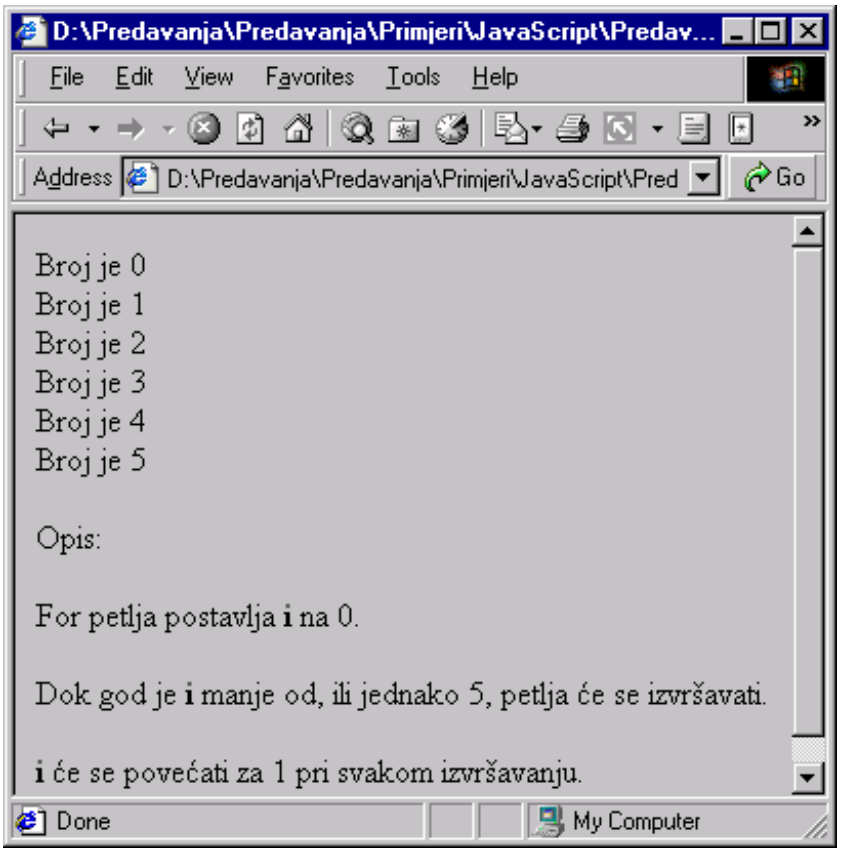
Nema razlike u izvršavanju jer je u oba slučaja početni uvjet zadovoljen.

For

For petlje su korisne kada se točno zna broj ponavljanja petlje.

```
<html>
<body>
<script language="JavaScript">
for (i = 0; i <= 5; i++)
 {
  document.write("Broj je " + i);
  document.write("<br>");
 }
</script>
<p>Opis:</p>
<p>For petlja postavlja <b>i</b> na 0.</p>
<p>Dok god je <b>i</b> manje od, ili jednako 5, petlja će se
izvršavati.</p>
<p><b>i</b> će se povećati za 1 pri svakom izvršavanju.</p>
</body>
</html>
```

Primjer 36. For petlja



Slika 16.

3.3.6. Funkcije

Rješavanje složenih programskih problema moguće je zahvaljujući dekompoziciji problema. Funkcije su strukturni blokovi koji dopuštaju dekompoziciju.
Uporaba takvih strukturnih blokova ima tri prednosti:

1. Za vrijeme rada na jednom strukturnom bloku, sva pozornost usmjerava se na jedan, manji dio problema.
2. Moguća je podjela zadaća na više programera.
3. Moguće je iskoristiti gotov blok na nekom drugom mjestu u programu.

Kako bi se učinkovito mogao pokazati poziv funkcije, koristit će se tip prozora *Alert*. Način na koji se poziva ovaj prozor iz JavaScripta je:

```
alert("Poruka za prikaz korisniku")
```

3.3.6.1. Definicija funkcije

Da bi se kreirala funkcija, potrebno joj je definirati ime, argumente i neke naredbe:

```
function MojaFunkcija(argument1,argument2,itd)
 {
   neke naredbe
 }
```

Funkcija bez argumenata mora uključivati okrugle zagrade :

```
function MojaFunkcija()
 {
   neke naredbe
 }
```

Argumenti su varijable koje će se koristiti u funkciji. Vrijednosti varijabli bit će vrijednosti koje su se prenijele prilikom poziva funkcije.
Stavljanjem funkcije u zaglavlje dokumenta osiguravate dostupnost funkcije prije njezina poziva.

Neke funkcije vraćaju vrijednost izrazu koji ih je pozvao:

```
function rezultat(a,b)
 {
   c=a+b
   return c
 }
```

3.3.6.2. Poziv funkcije

Funkcija se ne izvršava prije poziva funkcije. Može se pozvati funkcija koja sadržava argumente:

```
MojaFunkcija(argument1,argument2,itd)
```

ili ona koja ne sadržava argumente:

```
MojaFunkcija ()
```

3.3.6.3. Return naredba

Funkcija koja vraća rezultat mora koristiti return naredbu. Ova naredba specificira vrijednost koja će biti vraćena pozivatelju funkcije

```
function total(a,b)
 {
   rezultat=a+b
   return rezultat
 }
```

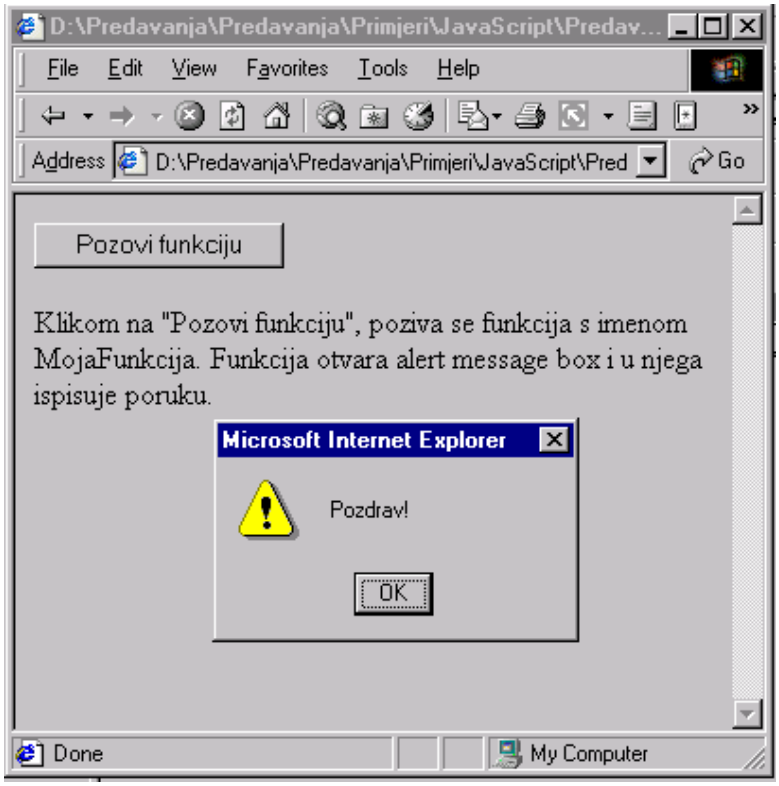
Prilikom poziva ove funkcije potrebno je navesti oba argumenta:

```
suma=total(2,3)
```

Varijabla suma poprima vrijednost 5.

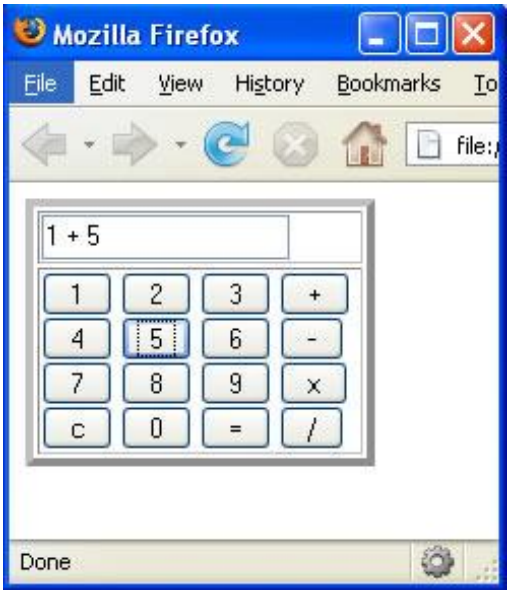
```
<html>
 <head>
<script type="text/javascript">
function MojaFunkcija()
  {
    alert("Pozdrav!")
  }
</script>
 </head>
 <body>
<form>
<input type="button"
   onclick="MojaFunkcija()"
   value="Pozovi funkciju">
</form>
<p>Klikom na "Pozovi funkciju", poziva se funkcija s imenom
MojaFunkcija. Funkcija otvara alert message box i u njega ispisuje
poruku.</p>
 </body>
</html>
```

Primjer 37. Poziv funkcije



Slika 17.

Pozivom funkcije moguće je izraditi i jednostavan JavaScript kalkulator. Za vježbu pokušajte izraditi kod za kalkulator koji će izgledati kao na slici 17b



Slika 17b.

3.3.7. JavaScript objekti

Najjednostavnije je zamisliti objekt kao varijablu s nizom podvarijabli. U JavaScriptu postoji niz objekata koji pojednostavljuju rješavanje različitih tipova zadaća. Važno je upoznati se s objektima koji se najčešće koriste.

3.3.7.1. Booleov objekt

Brojčani podaci i stringovi imaju neograničen broj mogućih vrijednosti, dok Booleovi podaci imaju samo dvije moguće vrijednosti, true ili false.

Metode	Opis	NN	IE	ECMA
toString()	Vraća string Booleovu vrijednost. (true or false)	3.0	3.0	1.0
valueOf()	Vraća vrijednost specificiranog objekta	4.0	4.0	1.0

3.3.7.2. String objekt

String je niz znakova koji kod JavaScripta predstavlja tekst. Jedna od mogućnosti JavaScripta je spajanje stringova sa '+' operatorom i to tako da se drugi string doda prvome, na primjer:

```
ime = "Pero " + "Perić";              // rezultat je "Pero Perić"
recenica = "Moje ime je" + " " + ime;// rezultat je "Moje ime je Pero
Perić"
```

Za duljinu stringa odnosno, broj znakova u stringu koristi se lenght svojstvo (property) stringa. Ako varijabla sadrži string, duljini stringa se može pristupiti na sljedeći način:

```
ime.lenght
```

Postoji više načina pomoću kojih se može raditi sa stringovima. Na primjer za otkrivanje zadnjeg znaka stringa može se koristiti sljedeći izraz:

```
zadnje_slovo=ime.charAt(ime.lenght-1)
```

Za izdvajanje drugog, trećeg i četvrtog znaka stringa, može se koristiti sljedeći izraz:

```
podstring=ime.substring(1,4);
```

Za pronalazak pozicije prvog slova 'r' u stringu ime, može se koristiti sljedeći izraz:

```
r=ime.indexOf('r');
```

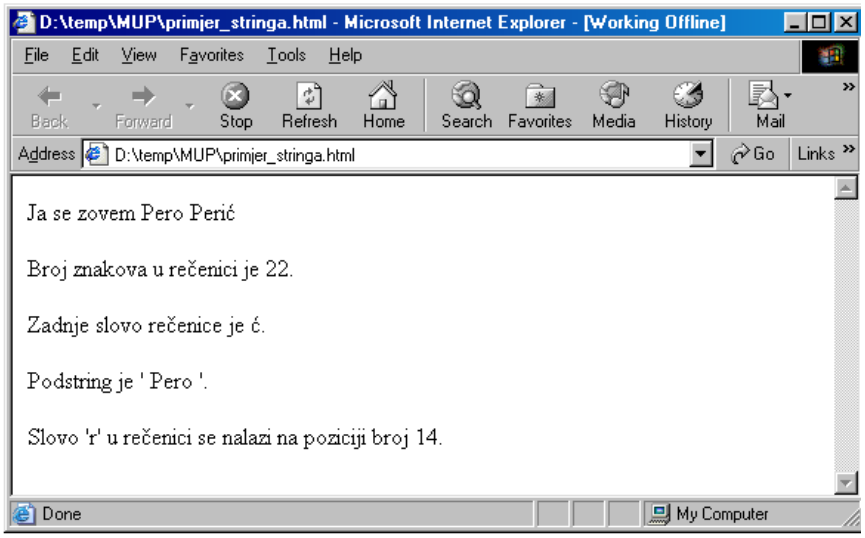
Postoji još niz drugih metoda kojima se može upravljati stringovima. Metode i opisi metoda upravljanja stringovima se nalaze u tablici.

Metode	Opis	NN	IE	EC MA
length	Vraća duljinu stringa	2.0	3.0	1.0
anchor()	Vraća string kao anchor: <a>string	2.0	3.0	
big()	Vraća string, formatiran sa big: <big>string</big>	2.0	3.0	
blink()	Vraća string formatiran s treptanjem: <blink>string</blink>	2.0		
bold()	Vraća podebljani string: string	2.0	3.0	
charAt()	Vraća znak na određenom mjestu	2.0	3.0	1.0
charCodeAt()	Vraća unicode od znaka na određenom mjestu	4.0	4.0	1.0
concat()	Vraća spojena dva stringa	4.0	4.0	
fixed()	Vraća string formatiran kao teletype text: <tt>string</tt>	2.0	3.0	
fontColor()	Vraća string u određenoj boji: string	2.0	3.0	
fontSize()	Vraća string u određenoj veličini: string	2.0	3.0	
fromCharCode()	Inverzna funkcija funkcije charCodeAt.	4.0	4.0	
indexOf()	Vraća mjesto prve pojave znaka ili –1 ako tog znaka nema.	2.0	3.0	
italics()	Vraća string u kurzivu, italic: string	2.0	3.0	
lastIndexOf()	Isto kao indexOf, ali s desna na lijevo.	2.0	3.0	
link()	Vraća string kao hyperlink: string	2.0	3.0	
match()	Ponaša se slično kao indexOf i lastIndexOf, ali vraća znakove koji se podudaraju ili "null", umjesto brojčane vrijednosti.	4.0	4.0	
replace()	Zamijeni znak s odabranim znakom.	4.0	4.0	
search()	Vraća cjelobrojnu vrijednost, tako da string sadrži određeni znak, inače vraća -1.	4.0	4.0	
slice()	Vraća string koji sadrži određeni znak definiran indexom.	4.0	4.0	
small()	Vraća string formatiran na small: string	2.0	3.0	
split()	Mijenja određeni znak zarezom.	4.0	4.0	1.0
strike()	Vraća string formatiran prekrižen: <strike>string</strike>	2.0	3.0	
sub()	Vraća string formatiran subscript: _{string}	2.0	3.0	
substr()	Vraća specificirane znakove. (14,7) vraća 7 znakova, počevši od 14og znaka.	4.0	4.0	
substring()	Vraća znakove koji su specificirani. (14,7) vraća znakove između 7og i 14og.	2.0	3.0	1.0
sup()	Vraća string formatiran superscript: ^{string}	2.0	3.0	
toLowerCase()	Vraća string konvertiran u mala slova	2.0	3.0	1.0
toUpperCase()	Vraća string konvertiran u velika slova	2.0	3.0	1.0

Sva svojstva objekata i metode se koriste na isti način kao u navedenim primjerima.

```
<html>
<body>

<script type="text/javascript">
ime = "Pero " + "Perić";
recenica = "Ja se zovem" + " " + ime;
document.write("<p>" + recenica + "</p>")
document.write("<p> Broj znakova u rečenici je " + recenica.length
+".</p>")

zadnje_slovo=recenica.charAt(recenica.length-1)
document.write("<p>Zadnje slovo rečenice je " + zadnje_slovo +
".</p>")

podstring=recenica.substring(12,16);
document.write("<p>Podstring je ' " + podstring + " '.</p>")

r=recenica.indexOf('r');
document.write("<p>Slovo 'r' u rečenici se nalazi na poziciji broj "
+ r + ".</p>")

</script>

</body>
</html>
```

Primjer 38. Primjeri za string objekt



Slika 18

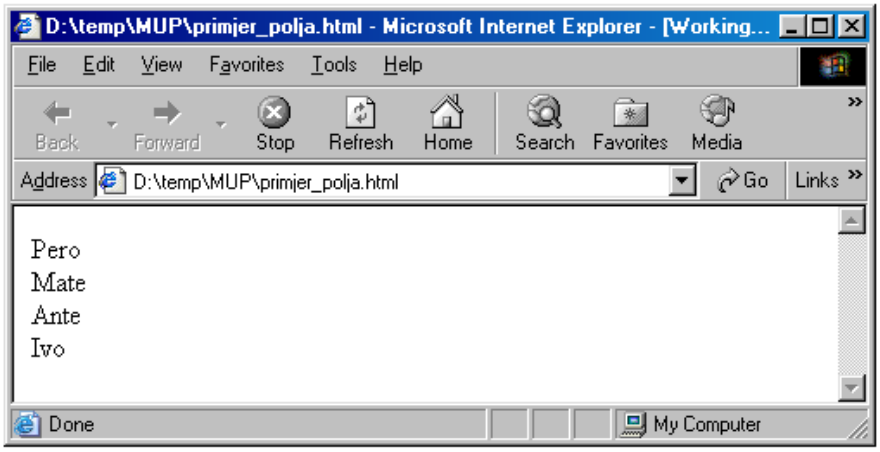
3.3.7.3. Objekt polja

Polje je niz podataka, kao i objekt. Kao što podatak unutar objekta ima ime, svaki podatak u polju ima broj ili index. Kod JavaScripta polja su indeksirana upisivanjem indeksa unutar uglatih zagrada nakon imena polja. Polje može

sadržavati bilo koji tip JavaScript podataka kao i reference na druga polja, objekte ili funkcije. JavaScript ne podržava izravno višedimenzionalne nizove. Polje se definira ključnom riječi new Array.

```
<html>
<body>

<script type="text/javascript">
var imena = new Array(4)
imena[0] = "Pero"
imena[1] = "Mate"
imena[2] = "Ante"
imena[3] = "Ivo"

for (i=0; i<4; i++)
 {
   document.write(imena[i] + "<br>")
 }
</script>
</body>
</html>
```

Primjer 39. Primjeri za objekt polja



Slika 19

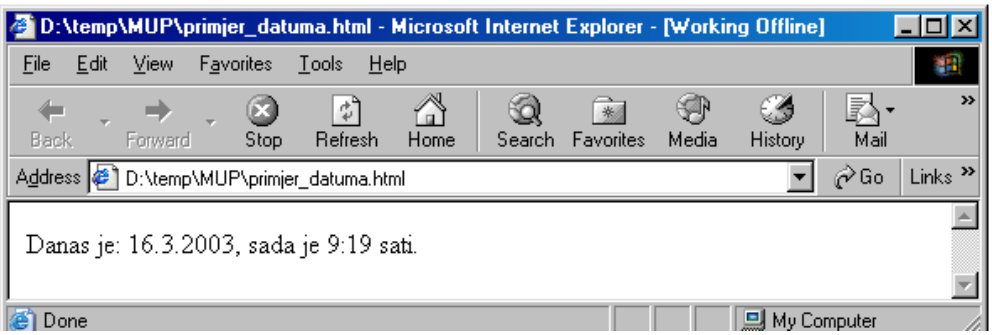
Metode	Opis	NN	IE	ECMA
length	Vraća broj elemenata polja	3.0	4.0	1.0
concat()	Vraća polje nastalo lijepljenjem dvaju polja	4.0	4.0	1.0
join()	Vraća string slijepljenih vrijednosti svih elemenata polja	3.0	4.0	1.0
reverse()	Vraća polje obrnutim redoslijedom	3.0	4.0	1.0
slice()	Vraća specificirani dio polja	4.0	4.0	
sort()	Vraća sortirano polje	3.0	4.0	1.0

3.3.7.4. Objekt datuma

JavaScript omogućuje tip ili klasu objekta koji predstavlja datum i vrijeme, i može se iskoristiti za manipuliranje tim tipom objekta. Date objekt kod JavaScripta se radi operatorom new i Date() konstruktorom.

```
<html>
<body>
<script type="text/javascript">
var d = new Date()
document.write("Danas je: ")
document.write(d.getDate())
document.write(".")
document.write(d.getMonth() + 1)
document.write(".")
document.write(d.getFullYear())
var s = new Date()
document.write(", sada je ")
document.write(s.getHours())
document.write(":")
document.write(s.getMinutes())
document.write("  sati.")
</script>
</body>
</html>
```

Primjer 40. Primjeri za objekt polja



Slika 20

Metode	Opis	NN	IE	ECMA
Date()	Vraća novi objekt datuma	2.0	3.0	1.0
getDate()	Vraća datum od Date objekta. (1-31)	2.0	3.0	1.0
getDay()	Vraća dan u tjednu. (0-6)	2.0	3.0	1.0
getMonth()	Vraća mjesec u godini. (0-11)	2.0	3.0	1.0
getFullYear()	Vraća godinu. (2000)	4.0	4.0	1.0
getYear()	Vraća godinu kao vrijednost od 4 znaka (ili godinu sa 2 znaka ako je datum manji od 1.1. 2000). Koristiti getFullYear za zamjenu !!	2.0	3.0	1.0

Metode	Opis	NN	IE	ECMA
getHours()	Vraća sat između 0 i 23	2.0	3.0	1.0
getMinutes()	Vraća minute. (0-59)	2.0	3.0	1.0
getSeconds()	Vraća sekunde. (0-59)	2.0	3.0	1.0
getMilliseconds()	Vraća milisekunde. (0-999)	4.0	4.0	1.0
getTime()	Vraća milisekunde protekle od 1/1-1970	2.0	3.0	1.0
getTimezoneOffset()	Vraća vremensku razliku između vremena na računalu i GMT	2.0	3.0	1.0
getUTCDate()	Vraća datum podešen prema World Time Standard, UTC = Universal Coordinated Time. Za povrat na lokalno vrijeme koristiti getDate metodu	4.0	4.0	1.0
getUTCDay()	Vraća UTC dan	4.0	4.0	1.0
getUTCMonth()	Vraća UTC mjesec	4.0	4.0	1.0
getUTCFullYear()	Vraća UTC 4 znamenke za godinu	4.0	4.0	1.0
getUTCHourc()	Vraća UTC sat	4.0	4.0	1.0
getUTCMinutes()	Vraća UTC minute	4.0	4.0	1.0
getUTCSeconds()	Vraća UTC sekunde	4.0	4.0	1.0
getUTCMilliseconds()	Vraća UTC millisekunde	4.0	4.0	1.0
parse()	Vraća string vrijednost datuma koliko je millisekundi proteklo od 1/1-1970.	2.0	3.0	1.0
setDate()	Postavlja novi datum u Date objekt	2.0	3.0	1.0
setFullYear()	Postavlja novu godinu u Date objekt	4.0	4.0	1.0
setHours()	Postavlja novi sat u Date objekt. (0-23)	2.0	3.0	1.0
setMilliseconds()	Postavlja nove millisekunde u Date objekt. (0-999)	4.0	4.0	1.0
setMinutes()	Postavlja nove minute u Date objekt. (0-59)	2.0	3.0	1.0
setMonth()	Postavlja novi mjesec u Date objekt. (0-11)	2.0	3.0	1.0
setSeconds()	Postavlja nove sekunde u Date objekt. (0-59)	2.0	3.0	1.0
setTime()	Postavlja milisekunde nakon 1/1-1970	2.0	3.0	1.0
setYear()	Postavlja novu godinu u Date objekt	2.0	3.0	1.0
setUTCDate()	Postavlja UTC Date objekt	4.0	4.0	1.0
setUTCDay()	Postavlja UTC Date objekt	4.0	4.0	1.0
setUTCMonth()	Postavlja UTC Date objekt	4.0	4.0	1.0
setUTCFullYear()	Postavlja UTC Date objekt	4.0	4.0	1.0
setUTCHourc()	Postavlja UTC Date objekt	4.0	4.0	1.0
setUTCMinutes()	Postavlja UTC Date objekt	4.0	4.0	1.0
setUTCSeconds()	Postavlja UTC Date objekt	4.0	4.0	1.0
setUTCMilliseconds()	Postavlja UTC Date objekt	4.0	4.0	1.0
toGMTString()	Vraća string vrijednost datuma.	2.0	3.0	1.0
toLocaleString()	Vraća string vrijednost datuma.	2.0	3.0	1.0
toString()	Vraća string vrijednost datuma.	2.0	4.0	1.0

3.3.7.5. Matematički objekt

Brojevi su osnovni tip podataka koje nije potrebno dodatno objašnjavati. Kod JavaScripta, za razliku od drugih programskih jezika poput C-a, svi brojevi su decimalne vrijednosti (*floating-point*). JavaScript programi rade s brojevima koristeći osnovne matematičke operatore kao što su: + za zbrajanje, - za oduzimanje, * za množenje, i / za dijeljenje. Kao dodatak ovim osnovnim aritmetičkim operacijama, JavaScript podržava kompleksne matematičke operacije pomoću velikog broja matematičkih funkcija koje su dio JavaScript jezika. Sve matematičke funkcije su spremljene kao svojstva (properties) Math objekta. Na primjer, za izračun sinusa numeričke vrijednosti x može se koristiti sljedeći izraz:

```
sinus_x=Math.sin(x);
```

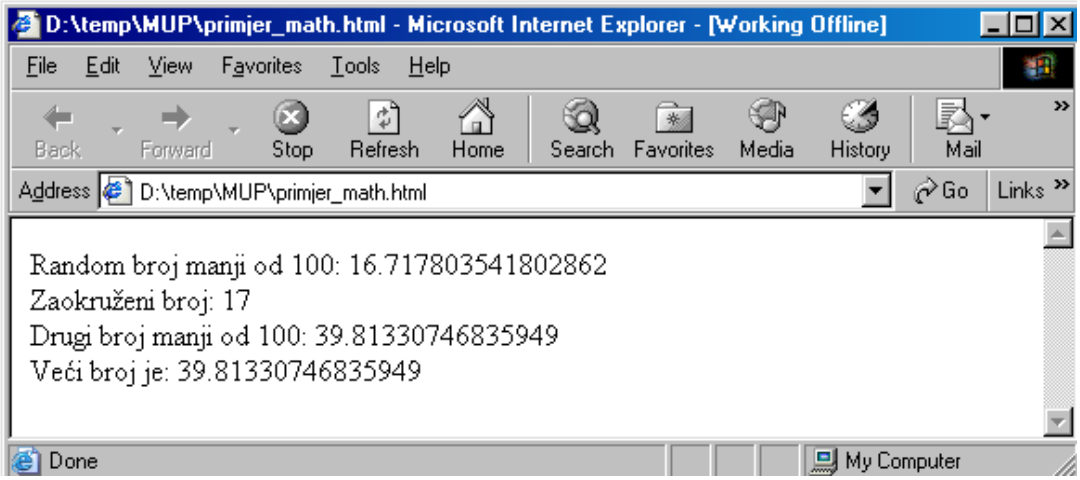
Za izračun korijena brojčane vrijednosti može se koristiti sljedeći izraz:

```
korijen=Math.sqrt(x*x+y*y)
```

Postoji nekoliko specijalnih matematičkih vrijednosti korištenih kod JavaScripta. Kada neka numerička vrijednost postane veća od najvećeg mogućeg broja, rezultat je beskonačna vrijednost koju JavaScript prikazuje kao Infinity. Isto tako kada vrijednost nekog broja postane manja od najmanje moguće, rezultat je -Infinity. Drugi specijani slučaj je kada je vraćena vrijednost matematičke operacije (kao kod dijeljenja s nulom) nedefinirani rezultat ili greška. U ovom slučaju rezultat JavaScripta je vrijednost Not-a-Number koja se prikazuje kao Nan. Not-a-Number vrijednost se ne može usporediti s drugim brojevima pa čak ni sa samim sobom! Iz tog razloga specijana funkcija isNan()potrebna je za testiranje ovih vrijednosti.

```
<html>
<body>
<script type="text/javascript">

random_broj=Math.random()*100;
document.write("Random broj manji od 100:  " + random_broj)
document.write("<br>Zaokruženi broj: " + Math.round(random_broj))

drugi_broj=Math.random()*100;
document.write("<br>Drugi broj manji od 100:  " + drugi_broj)

document.write("<br>Veći broj je:  " +
Math.max(random_broj,drugi_broj))
</script>
</body>
</html>
```

Primjer 41. Primjeri za matematički objekt



Slika 21

Svojstvo	Opis	NN	IE	ECMA
E	Vraća bazu prirodnog logaritma	2.0	3.0	1.0
LN2	Vraća prirodni logaritam od 2	2.0	3.0	1.0
LN10	Vraća prirodni logaritam od 10	2.0	3.0	1.0
LOG2E	Vraća logaritam po bazi 2 od E	2.0	3.0	1.0
LOG10E	Vraća logaritam po bazi 10 od E	2.0	3.0	1.0
PI	Vraća PI	2.0	3.0	1.0
SQRT1_2	Vraća kvadratni korijen od 0.5	2.0	3.0	1.0
SQRT2	Vraća kvadratni korijen od 2	2.0	3.0	1.0

Metode	Opis	NN	IE	ECMA
abs()	Vraća apsolutnu vrijednost broja	2.0	3.0	1.0
acos()	Vraća arccos broja	2.0	3.0	1.0
asin()	Vraća arcsin broja	2.0	3.0	1.0
atan()	Vraća arctan broja	2.0	3.0	1.0
atan2()	Vraća kut od x osi do točke	2.0	3.0	1.0
ceil()	Vraća najbliži cijeli broj veći ili jednak broju	2.0	3.0	1.0
cos()	Vraća kosinus broja	2.0	3.0	1.0
exp()	Vraća bazu logaritma na neki eksponent	2.0	3.0	1.0
floor()	Vraća najbliži cijeli broj manji ili jednak broju.	2.0	3.0	1.0
log()	Vraća logaritam broja	2.0	3.0	1.0
max()	Vraća veći od dva broja	2.0	3.0	1.0
min()	Vraća manji od dva broja	2.0	3.0	1.0
pow()	Vrijednost na neku potenciju	2.0	3.0	1.0
random()	Vraća slučajni broj	2.0	3.0	1.0
round()	Zaokruži broj na najbliži cijeli broj	2.0	3.0	1.0
sin()	Sinus broja	2.0	3.0	1.0
sqrt()	Kvadratni korijen broja	2.0	3.0	1.0
tan()	Tangens broja	2.0	3.0	1.0

3.3.8. Elementi forme i događaji (events)

JavaScript Form objekt predstavlja HTML formu. Forme su uvijek elementi forms[] polja, koje je svojstvo Document objekta. Forma se pojavljuje u polju redoslijedom kojim se pojavljuje u dokumentu. Tako se document.forms[0], odnosi na prvu formu u dokumentu, a zadnjoj formi se može pristupiti izrazom document.forms[document.forms.lenght]. Najbitnije svojstvo Form objekta je elements[] polje, koje sadrži JavaScript objekte (različitih tipova) koji predstavljaju različite elemente forme. Elementi se kao i forme pojavljuju u polju redoslijedom kojim se pojavljuju u dokumentu, pa se izraz document.forms[1].elements[2] odnosi na treći element druge forme u dokumentu.
Postoji nekoliko mogućih HTML form elemenata i odgovarajućih JavaScript objekata, koji su ispisani u tablici:

Objekt	HTML Tag	type Property	Opis i događaji
Button	<INPUT TYPE=button>	"button"	Dugme; onClick().
Checkbox	<INPUT TYPE=checkbox>	"checkbox"	Kućica za označavanje; onClick().
FileUpload	<INPUT TYPE=file>	"file"	Polje za unos imena dokumenta za upload na web server; onChange().
Hidden	<INPUT TYPE=hidden>	"hidden"	Podatak se prima na poslužitelj zajedno sa ostalim podacima koji se nalaze između <form> oznaka. Ovaj podatak nije vidljiv korisniku jer je atribut type postavljen na vrijednost hidden (skriven). Na ovaj element ne mogu se vezivati događaji (events) koji služe za upravljanje dinamikom HTML stranice.
Option	<OPTION>	*none*	Pojedini element Select objekta za koje vrijede događaji na Select objektu a ne na Option objektu.
Password	<INPUT TYPE=password>	"password"	Polje za unos lozinki-uneseni znakovi nisu vidljivi korisniku; onChange().
Radio	<INPUT TYPE=radio>	"radio"	Kućice za označavanje od kojih samo jedna može biti označena; onClick().
Reset	<INPUT TYPE=reset>	"reset"	Dugme koje poništava podatke iz forme; onClick().
Select	<SELECT>	"select-one"	Lista ili padajući izbornik s kojeg može biti izabrana samo jedna opcija. onChange().Pogledati Option object.
Select	<SELECT MULTIPLE>	"select-multiple"	Lista s koje može biti izabrano više opcija; onChange(). Pogledati Option object.

Objekt	HTML Tag	type Property	Opis i događaji
Submit	<INPUT TYPE=submit>	"submit"	Dugme koje šalje formu; onClick().
Text	<INPUT TYPE=text>	"text"	Jednolinijsko polje za unos podataka; onChange().
Textarea	<TEXTAREA>	"textarea"	Višelinijsko tekst polje za unos podataka; onChange().

Dva druga svojstva koja imaju svi elementi forme su *name* i *value*. Kada je forma primljena, korisnikovi podaci su poslani web poslužitelju u formi *name/value* parova. Ova svojstva određuju ime pod kojim se svaki element primi i vrijednost koja je primljena za taj element.
Name svojstvo je read-only string i njegova vrijednost je određena NAME atributom u HTML oznaci definiranom u formi. NAME atribut je opcionalan, ali podatak s elementa ne može biti primljen ako taj atribut nije određen.
Value svojstvo je slično *name* svojstvu. To svojstvo je read/write string za sve elemente forme i sadrži podatke koji su poslani s formom. *Value* svojstvo je određeno VALUE atributom HTML taga koji je definiran u formi.
Ako se želi ispisati trenutni datum i vrijeme u tekst polju za unos unutar forme, može se koristiti sljedeći izraz:

```
danas=nedjelja;
document.moja_forma.dan.value=danas;
```

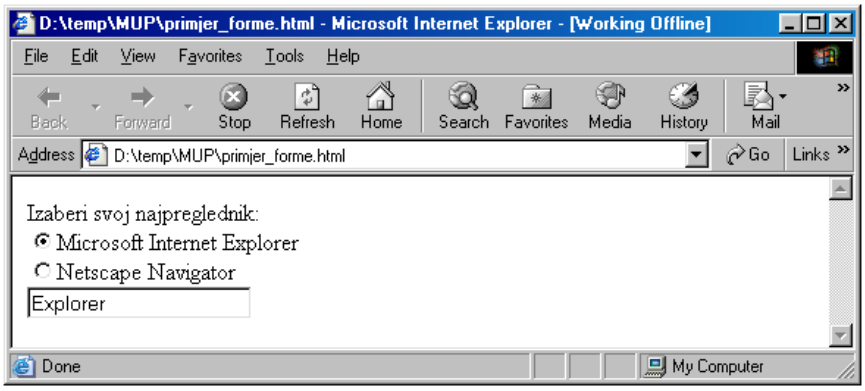
Ako se želi dodati datum na već postojeći string, može se koristiti:

```
document.moja_forma.dan.value+=danas;
```

Sve upotrebe *value* svojstva nisu ista. Za *text* i *textarea* objekte value svojstvo je string koji se nalazi u polju za unos. Namještanje *value* svojstva tih objekata mijenja tekst koji se nalazi u polju za unos.
Za *Button, Reset,* i *Submit* objekte *value* svojstvo predstavlja tekst ispisan na dugmetu.
Za *Checkbox* i *Radio* objekte *value* svojstvo predstavlja string vrijednost koja je poslana s formom ako su *Checkbox* ili *Radio* označeni. Posebno svojstvo *checked* se koristi kod tih objekata za ispitavanje stanja kućica, odnosno da li je neka kućica označena ili ne. To je booleov objekt, znači može biti *true* ili *false*.
Select objekt je još jedan poseban slučaj. Prikazuje padajući meni opcija koji omogućuje korisniku da izabere jedan od ponuđenih. Te opcije nisu označene <SELECT> tagom već <OPTION> tagom, pa tako *select* objekt nema *value* svojstva, i iznimka je u pravilu da svi form elementi imaju *value*.

```
<html>
<head>
<script type="text/javascript">
function provjeri(browser)
{
document.forms[0].odgovor.value=browser
}
</script>

</head>
<body>

<form>
Izaberi svoj najpreglednik:<br>
<input type="radio" name="preglednik" onclick="provjeri(this.value)"
value="Explorer">Microsoft Internet Explorer<br>

<input type="radio" name="preglednik" onclick="provjeri(this.value)"
value="Netscape">Netscape Navigator<br>

<input type="text" name="odgovor">
</form>

</body>
</html>
```

Primjer 42. Primjeri za elemente forme i događaje



Slika 21

3.3.9. Pitanja

1. Varijablama u JavaScriptu mora se strogo odrediti tip (Da – Ne)

2. Više varijabli može se deklarirati odjednom (Da – Ne)

3. Ključna riječ _____ koristi se za deklaraciju varijabli

4. Spojite operatore s njihovim funkcijama

A	Odgovor (slovo iz stupca A)	B
a) =		Dodjela vrijednosti
b) = =		Zbrajanje
c) +		Jednakost

5. Napišite if strukturu koja će provjeriti je li vrijednost varijable p veća ili jednaka 99 i ako je to tako tu vrijednost dodijelit će varijabli total

6. Napišite funkciju koja će pomnožiti dva broja i vratiti rezultat pozivatelju funkcije.

3.4. DOM (Document Object Model)

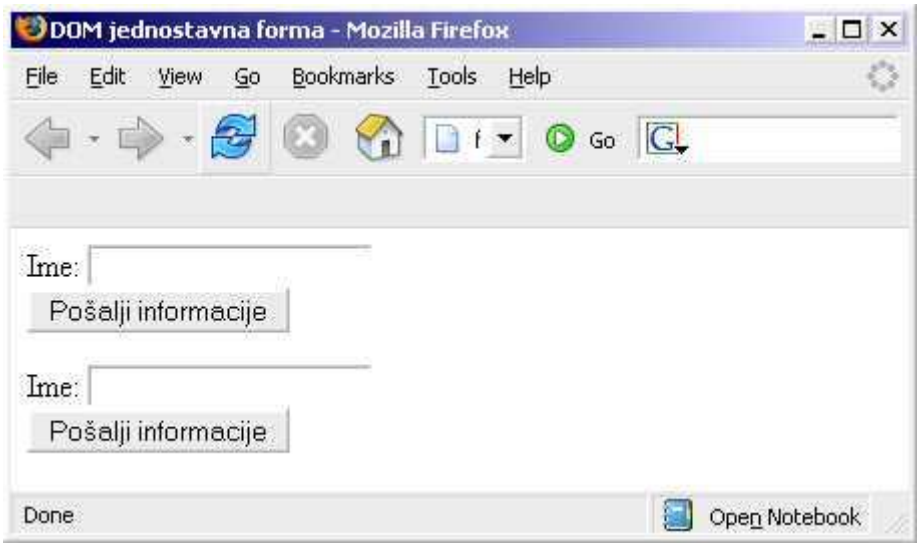
Objekt je skup varijabli (parametara) i funkcija (metoda). Web preglednik pruža mogućnost pristupa nizu predefiniranih objekata. Prozor preglednika u kojem se prikazuje stranica naziva se window object (objekt prozora). HTML stranica koju prikazuje preglednik naziva se document object (objekt dokumenta). Objekt dokumenta vjerojatno je najčešće korišteni objekt JavaScripta na klijentovoj strani.
HTML elementi koji se dodaju stranici proširuju hijerarhiju objekata. Primjer je form element i elementi koji se nalaze unutar form elementa. Ovo znači da je moguće referencirati svaki od dodanih objekata.
DOM je skup objekata koje JavaScript jeziku dodaje preglednik.
window.document.forms[0] - odnosi se na prvu formu unutar html dokumenta. Forme su u DOM-u implementirane kao polja. Ako postoji više od jedne forme u dokumentu, brojevi forme počinju od 0 i rastu za 1.

window.document.Forma1 - odnosi se na formu unutar html dokumenta naziva Forma1.

window.document.Forma1.Ime.value - odnosi se na vrijednost upisanu u textbox naziva Ime u formu unutar HTML dokumenta naziva Forma1 od strane klijenta.

```
<html>
 <head>
            <title>
                  DOM jednostavna forma
            </title>
  </head>
      <body>
            <form name="Forma1">
                  Ime:
                  <input type="text" name="Ime" >
                  <br />
                  <input type="button" value="Pošalji informacije">
            </form>
            <form name="Forma2">
                  Ime:
                  <input type="text" name="Prezime" >
                  <br />
                  <input type="button" value="Pošalji informacije">
            </form>
      </body>
</html>
```

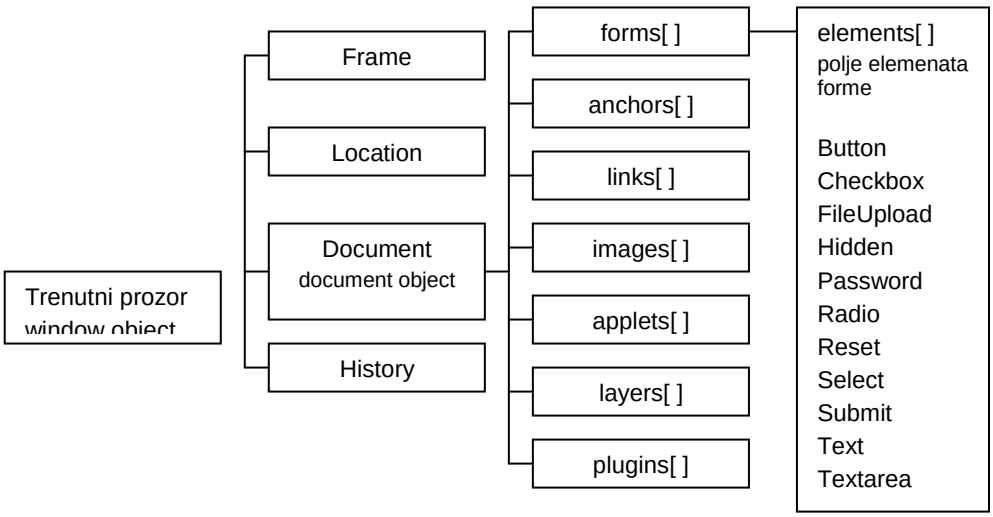
Primjer 43. Primjeri za elemente forme i događaje



Slika 22.

Iako je moguće referencirati objekt na formi bilo preko imena forme bilo preko njezina položaja u polju formi, preporučuje se koristiti referenciranje preko imena radi jednostavnijeg praćenja naziva objekata.

Dijagram koji ilustrira Document Object Model (DOM)



Slika 23.

3.4.1. Događaji (events)

Događaji su okidači koji pozivaju (pokreću) jednu od definiranih funkcija. Klijent strana aplikacije (JavaScript program) neće se izvršiti (interpretirati) dok je ne pokrene neki događaj. Događaj može biti akcija poput klika na neki objekt ili prijelaz miša preko nekog objekta. Provjera unosa na formi u pravilu se pokreće kada korisnik klikne na «Pošalji», odnosno kada se na objektu okine onClick događaj. U JavaScriptu možemo registrirati okidanje događaja za koje slijedi opis.

3.4.1.1. onClick()

Događaj koji okida kada se klikne na elemente:

- button
- checkbox
- radio button
- reset button
- submit button

Primjer uporabe ovog događaja

```
<input type=”button” value=”Klikni me”
onClick=”window.alert(‘Kliknuli ste’);”>
```

3.4.1.2. onSubmit()

Ovaj događaj okida kada korisnik šalje rezultate popunjene na formi. Uobičajeno je taj događaj vezati na provjeru podataka forme.

Primjer uporabe ovog događaja

```
<form action=”http://www.primjeri.hr/formtest.asp” onSubmit=”return
provjera();”>
```

U ovom primjeru kada korisnik klikne na dugme *Submit* izvršit će se funkcija provjera(). Ako svi podaci na formi uspješno prođu test, funkcija će vratiti true, a podaci će se tada proslijediti poslužitelju. U protivnom korisnik će morati ispraviti ili dopuniti podatke na formi.

3.4.1.3. onMouseOver()

Ovaj događaj okida kada se korisnik pozicionira mišem na hyperlink.

```
<a href="http://www.primjeri.hr/"
      onMouseOver="window.status=' Java Script i programiranje na
webu!'; return true;">
```

3.4.1.4. onMouseOut()

Ovaj događaj okida kada se korisnik pozicionira mišem izvan hyperlinka.

```
<a href="http://www. primjeri.hr/"
      onMouseOut="window.status=' Java Script i programiranje na
webu!'; return true;">
```

3.4.1.5. onFocus()

Objekt forme postaje aktivan. Ovaj događaj okida kada se korisnik tabulatorom ili mišem postavi na:

- password
- text
- textarea
- FileUpload

U HTML formi.

```
<INPUT TYPE="TEXT" NAME="Mjesec"
 onFocus="window.status=('Upišite mjesec od 01 do 12');
return true;">
```

3.4.1.6. onChange()

Ovaj događaj okida kada korisnik napušta objekt, a vrijednost objekta se promijenila.

```
<INPUT TYPE="TEXT" NAME="Mjesec"
onChange="window.status=('Vrijednost se promijenila!!!!'); return
true;" >
```

3.4.1.7. onBlur()

Ovaj događaj okida kada korisnik napusti objekt u HTML formi bez obzira na to je li se vrijednost objekta promijenila ili ne.

3.4.1.8. onLoad()

Ovaj događaj okida nakon što preglednik učita dokument.

3.4.1.9. onUnload()

Ovaj događaj okida kada se dokument počne učitavati.

Tablični prikaz događaja i elemenata HTML forme

JavaScript Events and HTML Form Elements

Events / HTML Elements	Blur	Click	Change	Focus	Load	Mouseover	Select	Submit	Unload
Button		X							
Checkbox		X			X				X
Document								X	
Form						X			
Link		X							
Radio		X							
Reset		X							
Selection	X		X	X					
Submit		X							
Text	X		X	X			X		
Textarea	X		X	X			X		

Primjer 44. Primjeri za elemente forme i događaje

3.5. Pitanja

1. Objekt je skup _____ i _____.

2. Objekti koji su ugrađeni u JavaScript jezik su:

   _____
   _____
   _____
   _____
   _____

3. DOM je skup objekata koje JavaScript jeziku dodaje preglednik. (Da –Ne)

4. Događaji su ključni za izvođenje skripta (Da – Ne)

5. Koji od događaja bi koristili za pokretanje skripta koji se treba izvršiti nakon što se HTML dokument učitao u preglednik
   a. onClick()
   b. onSubmit()
   c. onLoad()
   d. onMouseOver()
   e. onUnload()

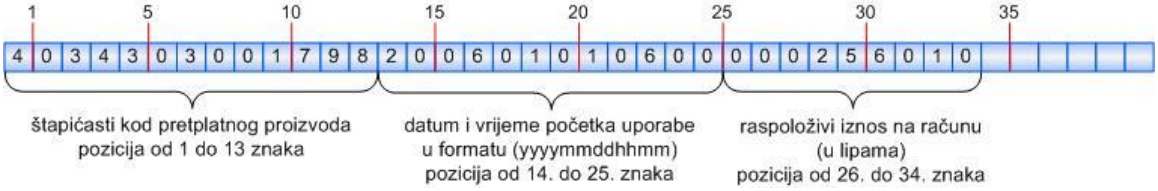
6. Događaji su povezani na određene HTML elemente (Da – Ne)

7. Napišite skript za provjeru ispravnosti upisanog e-maila koji će se izvršiti čim se korisnik makne s polja za unos.

4. XML – eXtensible Markup Language

XML je proširiv jezik za označavanje temeljen na SGML-u (Standard Generalized Markup Language), koji se već dugo koristi u sofisticiranim i veoma složenim aplikacijama za izdavaštvo. SGML je jezik koji je značajno složeniji od XML-a (osnovna specifikacija SGML-a iznosi 150 strana, dok osnovna specifikacija XML iznosi 35 strana). XML je u svojem nastajanju optimiziran za upotrebu na internetu na način da se iz SGML-a izbacilo sve što je nepotrebno za WWW. XML je (pojednostavnjena verzija SGML-a) vrlo brzo prihvaćen kao standard za zapis i razmjenu podataka između računalnih sustava i na webu.

U praksi računalni sustavi i baze podataka sadrže podatke u nekompatibilnim formatima. Jedna od redovitih zadaća programera je napisati dio koda koji čita pristigle podatke i/ili koji zapisuje podatke u određenom obliku. Različiti sustavi koriste različite formate. Ako neki sustav razmjenjuje podatke s više drugih računalnih sustava, tada se javlja potreba za višestrukim pisanjem koda za konverziju podataka (za svaki od sustava s kojim se razmjenjuju podaci). Za razmjenu se najčešće koristi tekstualni format podataka. Strukturu zapisa podataka moguće je prikazati kao:



Slika 24.

Na prethodnoj slici možemo vidjeti ustaljeni način definiranja formata zapisa podataka koji se oslanja na sljedeći postupak:

1. Između dviju strana dogovorno se odredi struktura podataka za razmjenu (kao na prethodnoj slici). Zapravo se definira struktura retka.
2. Datoteka za razmjenu se popunjava podacima na način da se na određeno mjesto (definirano specifikacijom) upiše željeni podatak.
3. Datoteka se čita na način da se prema specifikaciji odredi na kojoj se poziciji nalaze željeni podaci.

Jedan od primjera razmjene podataka je dogovor između dviju tvrtki koje razmjenjuju podatke o pretplatnicima, koji onda mogu koristiti usluge bilo kojeg od poslovnih partnera.

Ovaj način razmjene podataka je težak, neučinkovit i nestandardan. Izrada programske podrške je mukotrpna i podložna pogreškama jer se za svaki podatak treba konzultirati dokumentacija, a zatim pisati programski kod koji na određeno mjesto stavlja podatak određene duljine. Teško je i zamisliti kako to

izgleda u velikim sustavima koji imaju velike količine podataka u bazama podataka, koje imaju veliki broj polja u retcima tablice. Ovaj način rada je već dugo u praksi i mane takvog načina rada su više nego očite. Rješenje je pronađeno u korištenju XML jezika koji pruža mogućnost opisa podataka.

4.1. Osnove XML-a

XML se koristi za strukturiranje i opis podataka u obliku koji je lako čitljiv različitim aplikacijama. HTML je jezik namijenjen definiranju prikaza podataka, dok je XML namijenjen opisu podataka.

```
<?xml version="1.0"?>
<poruka>
  <salje>Jana</salje>
  <prima>Marin</prima>
  <tekst>Ovo je poruka!</tekst>
</poruka>
```

Primjer 45. Primjeri za jednostavan XML dokument

Primjetno je kako ovaj dokument ne sadrži nikakvu informaciju o tome kako bi podaci trebali izgledati (npr. vrsta i veličina fonta …) za razliku od HTML jezika.

```
<html>
<body>
<h2>PORUKA</h2>
<p>
<b>Ovo je poruka!</b>
</p>
</body>
</html>
```

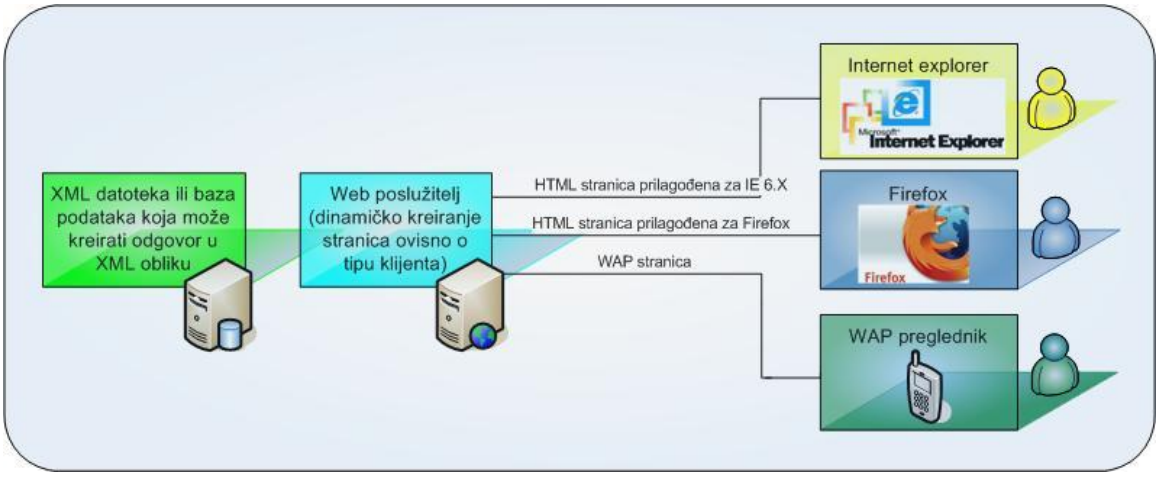
Primjer 46. Primjeri za jednostavan HTML dokument

Uočljivo je kako je velik dio sadržaja XML dokumenta zapravo opis samih podataka. Lako je pročitati XML dokument i razumjeti što je sadržaj dokumenta (Jana šalje poruku Marinu sadržaja “Ovo je poruka!”). Zbog toga se XML dokumenti često nazivaju i samoopisujući dokumenti. Ne treba više voditi računa na kojoj se poziciji (i koje je duljine) nalazi podatak.

Mnogi ljudi smatraju XML zamjenom za HTML. Iako je to dijelom točno, istina je da se ova dva jezika prije nadopunjuju nego što su konkurencija jedan drugome. Najčešća veza XML-a i HTML-a u upotrebi na WWW-u je kada se XML koristi za zapis podataka, a HTML za određivanje prikaza tih istih podataka (koristeći HTML oznake kako je opisano u prethodnim poglavljima i kako će tek biti opisano u idućim poglavljima).

Prednost ove podjele je u tome što se sadržaj (u našem primjeru XML dokument sadrži poruku) može lako promijeniti bez zadiranja u HTML kod (koji može biti jako složen zbog formatiranja prikaza podataka), a i obrnuto: izgled prikazanih podataka se može promijeniti bez potrebe za izmjenom XML dokumenta. Ta podjela ima za posljedicu nekoliko važnih prednosti u odnosu na integraciju podataka i definicije izgleda u jednu cjelinu, tj. HTML stranicu:

- za različite klijente može se iz istih podataka kreirati različit prikaz (npr. ovisno o pretraživaču koji klijent koristi, a to može biti Internet Explorer s najnaprednijim mogućnostima ili mobitel koji koristi WAP protokol, generira se ili HTML ili WAP stranica prilagođena klijentu)
- dinamičko ažuriranje stranica koje prikazuju podatke iz neke baze podataka jednostavno se postiže (izgled same stranice je definiran u HTML obliku, a podaci iz baze podataka dobivaju se u XML obliku).



Slika 25: Jedan od načina korištenja XML-a na webu za dinamičke stranice

U prethodnom primjeru XML podaci su zapisani ili u nekoj datoteci ili mogu egzistirati u nekoj bazi podataka iz koje se mogu dobiti postavljanjem upita (podatke može ažurirati neka druga aplikacija). Na strani poslužitelja se, ovisno o vrsti klijenta, vrši generiranje (popunjavanje) stranice s dobivenim podacima.

Generiranje stranica se može vršiti i direktnom transformacijom XML-a u željeni oblik (HTML ili neki drugi) pomoću XSL-a (jezika namijenjenog za transformacije XML dokumenata iz jednog oblika u drugi), ali o tome u idućim poglavljima.

Još jednom vrijedi napomenuti da je generiranje različitih prezentacija istih podataka lako ostvarivo zahvaljujući razdvajanju podataka od definicije prikaza.

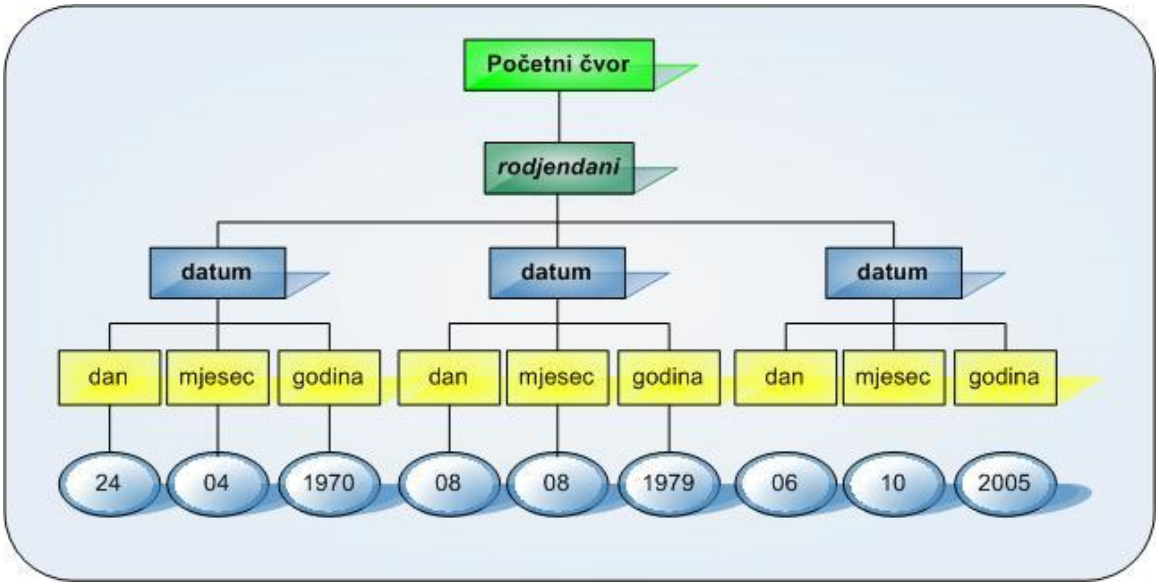
Pojavom XML standarda razmjena podataka se bitno pojednostavljuje i olakšava. Sami podaci koji se razmjenjuju sadržavat će opis podataka. Programerski dio posla je značajno olakšan jer postoje komercijalni proizvodi koji sadrže metode za čitanje, kreiranje, brisanje i pretraživanje XML dokumenata. Takvi proizvodi zovu se **XML parseri**. Programerski dio posla se svodi na pozivanje određenih metoda koje XML parser podržava. Između ostalog XML parseri sadrže metode za validaciju XML dokumenta (XML dokument mora poštivati stroga XML pravila).

4.2. Struktura XML dokumenta

Svaki dobro napisani XML dokument ima strukturu stabla. XML stablo je struktura sastavljena od povezanih čvorova (eng. *node*) na čijem vrhu je početni čvor (root node). Na početni čvor su povezani drugi čvorovi tzv. potomci (eng. *child node*), a na svaki od tih potomaka mogu biti povezani njegovi potomci itd. Grafički prikaz XML stabla je vrlo sličan obiteljskom stablu potomaka jednog pretka. Najkorisnija karakteristika strukture stabla je činjenica da svaki čvor i njegovi potomci također predstavljaju strukturu stabla. Dakle, stablo je hijerarhijska struktura sastavljena od stabala od kojih je svako stablo sastavljeno od više manjih stabala.

```
<?xml version="1.0" encoding="ISO-8859-2" ?>
<rodjendani>
      <datum>
            <dan>24</dan>
            <mjesec>04</mjesec>
            <godina>1970</godina>
      </datum>
      <datum>
            <dan>08</dan>
            <mjesec>08</mjesec>
            <godina>1979</godina>
      </datum>
      <datum>
            <dan>06</dan>
            <mjesec>10</mjesec>
            <godina>2005</godina>
      </datum>
</rodjendani>
```

Primjer 47: Podaci o rođendanima u XML obliku



Slika 26: Hijerarhijska struktura XML dokumenta je slična strukturi stabla

Na prethodnom primjeru prikazano je stablo XML dokumenta. Uočimo da postoji samo jedan početni čvor (eng. *root node*) i samo jedan početni element (eng. *document element*). Početni element rođendani sadrži svoje direktne potomke, a to su elementi datum. Svaki od elemenata datum sadrži svoje pripadajuće elemente dan, mjesec, godina. Svaki od tih elemenata ima pripadajući čvor koji sadrži vrijednost samog elementa (čvorovi na dnu prikazanog XML stabla).

4.3. Osnovni dijelovi XML dokumenta

XML dokument se sastoji od deklaracija, elemenata, atributa, procesnih instrukcija i komentara.

4.3.1. Prolog

Prolog je opcionalni dio XML dokumenta i on može sadržavati dvije također opcionalne komponente (XML deklaraciju i deklaraciju tipa dokumenta):

XML deklaracija označava verziju XML specifikacije po kojoj je rađen dokument. XML deklaracija koja sadrži informaciju o verziji XMLa izgleda ovako:

```
<?xml version="1.0" ?>
```

Napomena:
XML deklaracija mora biti napisana malim slovima, mora biti napisana na samom početku XML dokumenta (u prvoj liniji i bez vodećih praznih mjesta).

XML deklaracija može sadržavati i informaciju o kodnoj stranici koja se koristi prilikom kreiranja samog dokumenta. XML podržava Unicode i preporuča se koristiti ga gdje god je to moguće.

Primjer XML deklaracije koja sadrži informaciju o kodnoj stranici:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Deklaracija tipa dokumenta sastoji se od pravila (kao i svaki drugi jezik i XML može imati gramatiku) koja XML dokument mora poštivati. Ta deklaracija može također upućivati na neku vanjsku datoteku i mora se nalaziti odmah iza XML deklaracije. Primjer deklaracije izgleda ovako:

```
<!DOCTYPE listaproizvoda SYSTEM "proizvodi.dtd">
```

Deklaracija kaže da je XML dokument tipa listaproizvoda i da poštuje pravila koja su navedena u datoteci proizvodi.dtd.
Opis definicije tipa dokumenta je izvan opsega ovog skripta pa se neće razmatrati, ali ukratko se može reći kako ova definicija određuje koje elemente i u kojem međusobnom odnosu XML dokument može sadržavati. Provjeru valjanosti XML dokumenta moguće je vršiti na temelju te definicije.

4.3.2. Elementi

Kao što je već napomenuto XML je jezik koji koristi oznake za opis podataka. XML element je sastavljen od početne oznake, završne oznake i podatka između tih dviju oznaka. Početna i završna oznaka opisuju podatak koji je ujedno i vrijednost elementa. Naziv XML elementa definiran je tim dvjema oznakama.

```
<grad>Split</grad>
```

U navedenom primjeru prikazan je XML element *grad* čija je vrijednost Split. Bitno je napomenuti kako je XML jezik koji razlikuje velika i mala slova u imenima elemenata.
Nazivi elemenata nisu unaprijed definirana već ih se proizvoljno može kreirati. Naravno, poželjno je da nazivi budu smisleni pa je tako naziv elementa *grad* puno bolji od naziva *a11a*.
Elementi mogu imati različite vrste sadržaja. Kao što je već napomenuto jedan XML element čini sve između početne i završne oznake elementa uključujući i te dvije oznake.
Elementi mogu sadržavati druge elemente i/ili tekstualni sadržaj (odnosno vrijednost elementa). Neki elementi mogu imati prazan sadržaj.

U našem primjeru elementi rođendani i datum imaju za svoj sadržaj druge elemente, dok elementi dan, mjesec i godina imaju tekstualni sadržaj.

4.3.3. Atributi

Elementi mogu imati jedan ili više atributa. Atribut sadrži dodatnu informaciju za neki element i sastoji se od imena atributa i njegove pripadajuće vrijednosti. Vrijednost atributa mora biti navedena u navodnicima.
Primjer XML elementa s pripadajućim atributom:

```
<grad postbroj=”21000”>Split</grad>
```

U ovom primjeru XML elementu grad pridružen je atribut postbroj čija je vrijednost 21000.

4.3.4. Procesne instrukcije

Procesne instrukcije su također opcija unutar XML dokumenta. Namjena im je prosljeđivanje informacija aplikaciji koja obrađuje XML dokument. Procesna instrukcija počinje sa <?, nakon toga ide ime procesne instrukcije i parametri (ako ih ima) i završava sa ?>.

```
<?xml version='1.0'?>
<?word document="test.doc" ?>
<root>
...
</root>
```

Primjer 48: Procesna instrukcija

Imena procesnih instrukcija ne smiju počinjati sa xml ili XML jer su te procesne instrukcije rezervirane XML standardom.

```
<?xml version='1.0'?>
<?xml-stylesheet type="text/xsl" href="knjiga.xsl"?>
<root>
...
</root>
```

Primjer 49: Procesna instrukcija kojom se referencira XSL dokument kojim će se transformirati ovaj XML dokument. (XSL transformacije će se obraditi u nastavku).

4.3.5. Komentari

Komentari sadrže informaciju za autora i/ili korisnika XML dokumenta. Mogu se nalaziti bilo gdje u dokumentu. Komentar počinje sa <!-- i završava sa -->.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- ovaj dokument kreiran je u petak -->
<datum>
<dan>30</dan>
<mjesec>10</mjesec>
<godina>1969</godina>
</datum>
```

Primjer 50: Komentari unutar XML dokumenta

Komentar se ne može staviti unutar oznaka elementa ili na mjesto vrijednosti atributa.

4.4. XML sintaksa

Sintaksna pravila XML-a su jednostavna i moraju se poštivati za razliku od HTML-a, koji također ima pravila, ali se neka od njih ne moraju poštivati.
Svi XML elementi moraju imati početnu i završnu oznaku.
Ako se napiše sljedeće u HTML-u, internet pretraživač nam neće javiti grešku iako ovi elementi nemaju završnu oznaku.

```
<p>Ovo je tekst
<p>Ovo je drugi tekst
```

Ali ako se isto želi napisati u XML-u, moraju se napisati i završne oznake, inače će aplikacija koja obrađuje XML javiti grešku. XML elementi moraju imati završne oznake kao u sljedećem primjeru:

```
<p>Ovo je tekst</p>
<p>Ovo je drugi tekst</p>
```

Napomena:
Ako XML element ne sadrži ništa, tada se može koristiti skraćeni način zapisa elementa.

Umjesto

```
<praznielement></praznielement>
```

Može se pisati

```
<praznielement />
```

4.4.1. XML oznake

XML oznake su osjetljive na velika i mala slova. Stoga XML oznake (tj. imena elemenata) moraju biti identično napisane s obzirom na velika i mala slova jer XML razlikuje npr. <oznaka> od <OZNAKA>.
Primjer ispravno napisane oznake:

```
<oznaka>Ovo je dobro!</oznaka>
```

Primjer neispravno napisane oznake:

```
<oznaka>Ovo nije dobro!</OZnaka>
```

4.4.2. XML elementi

U XML dokumentu elementi mogu sadržavati druge elemente (sub-elemente) koji moraju biti pravilno ugniježđeni odnosno oznake se moraju zatvarati poštujući redoslijed kojim su i otvarane.
Primjer za ispravno gniježdenje elemenata:

```
<prvi>
<drugi>
<treci>Ovo je dobro!</treci>
</drugi>
</prvi>
```

Primjer za neispravno gniježdenje elemenata:

```
<prvi>
<drugi>
<treci>Ovo nije dobro!</drugi>
</treci>
</prvi>
```

4.4.3. XML dokument

Prvi element u nekom XML dokumentu je početni element (root element) koji sadrži sve ostale elemente u XML dokumentu. XML mora sadržavati samo jedan početni element.

```
<?xml version=”1.0”?>
<pocetni>
  <prvi>
    <drugi>
      <treci>.....</treci>
    </drugi>
  </prvi>
</pocetni>
```

Primjer 51: Ispravan XML dokument (sadrži samo jedan početni element)

```
<?xml version=”1.0”?>
<pocetni>
  <drugi>
    <treci>.....</treci>
  </drugi>
</pocetni>
<prvi>
  ...
</prvi>
```

Primjer 52: Neispravan XML dokument (sadrži dva početna elementa)

4.4.4. Vrijednosti atributa

XML specifikacija zahtijeva da vrijednosti atributa budu upisane u navodnicima. Ovdje vidimo još jednu razliku u odnosu na HTML koji je tolerantan o ovom pitanju.

```
<?xml version="1.0"?>
<poruka datum=”01/10/06”>
      <prima>Tomo</prima>
      <salje>Janica</salje>
      <tekst>Ovo je dobro!</tekst>
</poruka>
```

Primjer 53: Ispravan XML dokument (vrijednost atributa je u navodnicima)

```
<?xml version="1.0"?>
<poruka datum=01/10/06>
      <prima>Tomo</prima>
      <salje>Janica</salje>
      <tekst>Ovo je dobro!</tekst>
</poruka>
```

Primjer 54: Neispravan XML dokument (vrijednost atributa nije u navodnicima)

4.5. XML prostor imenovanja (*namespace)*

Do sada su se razmatrala pravila koja omogućavaju kreiranje ispravnih XML dokumenata. Treba napomenuti kako jedna karakteristika XML dokumenta može stvoriti problem koji nije očit na prvi pogled. Rečeno je kako XML standard ne definira nazive elemenata, već ih zadaje sam autor. Ova činjenica može dovesti do toga da različiti autori koriste iste nazive elemenata za različite podatke odnosno dolazi do pojave nejedinstvenih naziva. Ovo je izraženo kada se XML podaci iz različitih izvora prikupljaju na jedno mjesto.

Uzmimo slučaj neke tvrtke koja se bavi prodajom različitih proizvoda koje nabavlja od različitih dobavljača.

Ta tvrtka prima cjenike od svojih dobavljača i na temelju njih formira vlastiti cjenik.

Dakle, XML dokument za dobavljača autoguma može izgledati ovako:

```
<?xml version="1.0"?>
<autogume>
	<guma>
		<proizvodjac>Goodyear</proizvodjac>
		<model>Club</model>
		<id>1234</id>
		<cijena>300</cijena>
	</guma>
	<guma>
		<proizvodjac>Sava</proizvodjac>
		<model>Efecta</model>
		<id>1235</id>
		<cijena>250</cijena>
	</guma>
</autogume>
```

XML dokument za dobavljača amortizera može izgledati ovako:

```
<?xml version="1.0"?>
<amortizeri>
	<amortizer>
		<proizvodjac>Monroe</proizvodjac>
		<model>Sensatrack</model>
		<id>2331</id>
		<cijena>400</cijena>
	</amortizer>
	<amortizer>
		<proizvodjac>Q H</proizvodjac>
		<model>nepoznat</model>
		<id>2332</id>
		<cijena>200</cijena>
	</amortizer>
</amortizeri>
```

Kada tvrtka prikupi sve cjenike može generirati jedan jedinstveni cjenik koji izgleda ovako:

```
<?xml version="1.0"?>
<artikli>
	<autogume>
		<guma>
			<proizvodjac>Goodyear</proizvodjac>
			<model>Club</model>
			<id>1234</id>
			<cijena>300</cijena>
		</guma>
		<guma>
			<proizvodjac>Sava</proizvodjac>
			<model>Efecta</model>
			<id>1235</id>
			<cijena>250</cijena>
		</guma>
	</autogume>
	<amortizeri>
		<amortizer>
			<proizvodjac>Monroe</proizvodjac>
			<model>Sensatrack</model>
			<id>2331</id>
			<cijena>400</cijena>
		</amortizer>
		<amortizer>
			<proizvodjac>Q H</proizvodjac>
			<model>nepoznat</model>
			<id>2332</id>
			<cijena>200</cijena>
		</amortizer>
	</amortizeri>
</artikli>
```

Uočljivo je pojavljivanje istih imena elemenata za semantički različite podatke što može dovesti do krive interpretacije podataka. Kako bi se izbjegli ovi problemi, XML uvodi mogućnost stavljanja prefiksa ispred imena elementa. Svaki taj prefiks je vezan za jedan jedinstveni string (najbolje je koristiti URL tvrtke jer je on već sam po sebi jedinstven).

Sada bi cjenik dobavljača autoguma mogao izgledati ovako (Za sve elemente ovog dokumenta dodan je prefiks d1 koji je vezan za URL prvog dobavljača. Svako je ime elementa sastavljeno od imena elementa i prefiksa koje predstavlja jedinstveni URL. Na ovaj način je ime elementa jedinstveno definirano).

XML cjenik prvog dobavljača sada bi trebao izgledati ovako:

```
<?xml version="1.0"?>
<d1:autogume xmlns:d1=”http://www.dobavljac1.hr”>
      <d1:guma>
            <d1:proizvodjac>Goodyear</d1:proizvodjac>
            <d1:model>Club</d1:model>
            <d1:id>1234</d1:id>
            <d1:cijena>300</d1:cijena>
      </d1:guma>
      <d1:guma>
            <d1:proizvodjac>Sava</d1:proizvodjac>
            <d1:model>Efecta</d1:model>
            <d1:id>1235</d1:id>
            <d1:cijena>250</d1:cijena>
      </d1:guma>
</d1:autogume>
```

XML cjenik drugog dobavljača sada bi trebao izgledati ovako:

```
<?xml version="1.0"?>
<d2:amortizeri xmlns:d2=”http://www.dobavljac2.hr”>
      <d2:amortizer>
            <d2:proizvodjac>Monroe</d2:proizvodjac>
            <d2:model>Sensatrack</d2:model>
            <d2:id>2331</d2:id>
            <d2:cijena>400</d2:cijena>
      </d2:amortizer>
      <d2:amortizer>
            <d2:proizvodjac>Q H</d2:proizvodjac>
            <d2:model>nepoznat</d2:model>
            <d2:id>2332</d2:id>
            <d2:cijena>200</d2:cijena>
      </d2:amortizer>
</d2:amortizeri>
```

Kada se ova dva cjenika spoje, dobija se sljedeći XML dokument:

```
<?xml version="1.0"?>
<artikli>
      <d1:autogume xmlns:d1=”http://www.dobavljac1.hr”>
            <d1:guma>
                  <d1:proizvodjac>Goodyear</d1:proizvodjac>
                  <d1:model>Club</d1:model>
                  <d1:id>1234</d1:id>
                  <d1:cijena>300</d1:cijena>
            </d1:guma>
            <d1:guma>
                  <d1:proizvodjac>Sava</d1:proizvodjac>
                  <d1:model>Efeckta</d1:model>
                  <d1:id>1235</d1:id>
                  <d1:cijena>250</d1:cijena>
```

```
            </d1:guma>
      </d1:autogume>
      <d2:amortizeri xmlns:d2="http://www.dobavljac2.hr">
            <d2:amortizer>
                  <d2:proizvodjac>Monroe</d2:proizvodjac>
                  <d2:model>Sensatrack</d2:model>
                  <d2:id>2331</d2:id>
                  <d2:cijena>400</d2:cijena>
            </d2:amortizer>
            <d2:amortizer>
                  <d2:proizvodjac>Q H</d2:proizvodjac>
                  <d2:model>nepoznat</d2:model>
                  <d2:id>2332</d2:id>
                  <d2:cijena>200</d2:cijena>
            </d2:amortizer>
      </d2:amortizeri>
</artikli>
```

Ovaj novi XML dokument sadrži jedinstvena imena elemenata. Aplikacija koja to zna iskoristiti neće imati problema s nazivima. XML prostor imenovanja (namespace, xmlns) neophodno je koristiti u primjeni XSL.

4.6. Sažetak

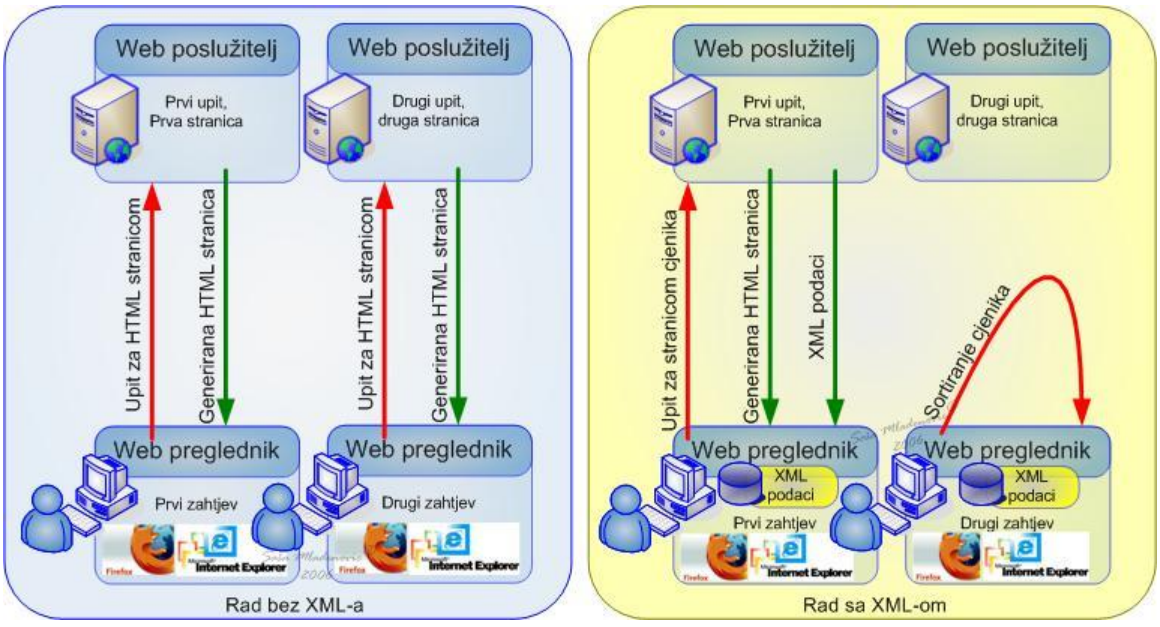
XML je jednostavan i standardan način za razmjenu strukturiranih tekstualnih podataka između računalnih aplikacija. Dio zasluge za svoj uspjeh XML duguje lakom pisanju i lakoj čitljivosti XML podataka koristeći pritom samo obični tekstualni editor. XML vrlo dobro ispunjava dva zahtjeva:

1. Razdvajanje podataka od prezentacije. Potreba za ovim razdvajanjem podataka (npr.: podaci o sportskim rezultatima) od načina njihove prezentacije na različitim uređajima sve je prisutnija jer sve više različitih vrsta uređaja ima pristup internetu. Organizacije koje ulažu mnogo novca u sustave za obradu informacija imaju realne zahtjeve za isporukom podataka ne samo internetskim pretraživačima instaliranim na PC računala (koji su opet prisutni u različitim verzijama), već i WAP mobilnim telefonima pa i novijim TV uređajima.
2. Razmjena podataka između aplikacija. Različiti sustavi imaju često potrebu za razmjenom informacija. Npr.: sustav za zaprimanje narudžbi koji omogućava plaćanje karticama mora imati mogućnost slanja podataka o narudžbi (broj kartice i iznos) nekom drugom sustavu specijaliziranom upravo za to (uobičajeno se radi o autorizacijskom centru kartičara). Umjesto razmjene podataka u raznoraznim formatima (što uvijek donosi dodatne troškove za razvoj aplikacija za prebacivanje podataka u prikladni format), korištenjem XML kao standarda lako se prepoznaje struktura samih podataka pa se i podaci lako mogu čitati i koristiti.

5. Povezivanje HTML-a i XML-a

Za sada smo se upoznali sa osnovnim pojmovima XML-a i logičan korak dalje je primjena XML na webu (što nam je i bio cilj). U prethodnim poglavljima vidjeli smo kako se rade statičke i dinamičke stranice bez upotrebe XML-a. Uvidjeli smo i koje su potencijalne prednosti korištenja XML-a.
Razmotrimo najprije kako se odvija proces prikaza podataka na nekim web stranicama. Dinamičke HTML stranice se generiraju na strani poslužitelja i rezultat toga postupka je stranica koja ima sjedinjene podatke sa definicijom samog izgleda podataka. Generirana stranica se tada prebacuje na stranu klijenta gdje se i prikazuje. Ako korisnik poželi vidjeti iste podatke u nekom drugom redoslijedu (sortirane po nekom drugom kriteriju), ponavlja se cijeli postupak (zahtjev ide prema poslužitelju, generira se nova stranica i šalje se klijentu). Karakteristika ovog procesa je sporost reagiranja na korisnikovu inicijativu.

Uzrok te sporosti je što su podaci integrirani s definicijom izgleda. Zbog toga je klijentu teško izvući podatke iz HTML stranice. Ako klijent ne može izvaditi podatke, ne može ni obaviti sortiranje ili filtriranje podataka.



Slika 27. Komunikacija između web preglednika i poslužitelja bez XML-a i s njim

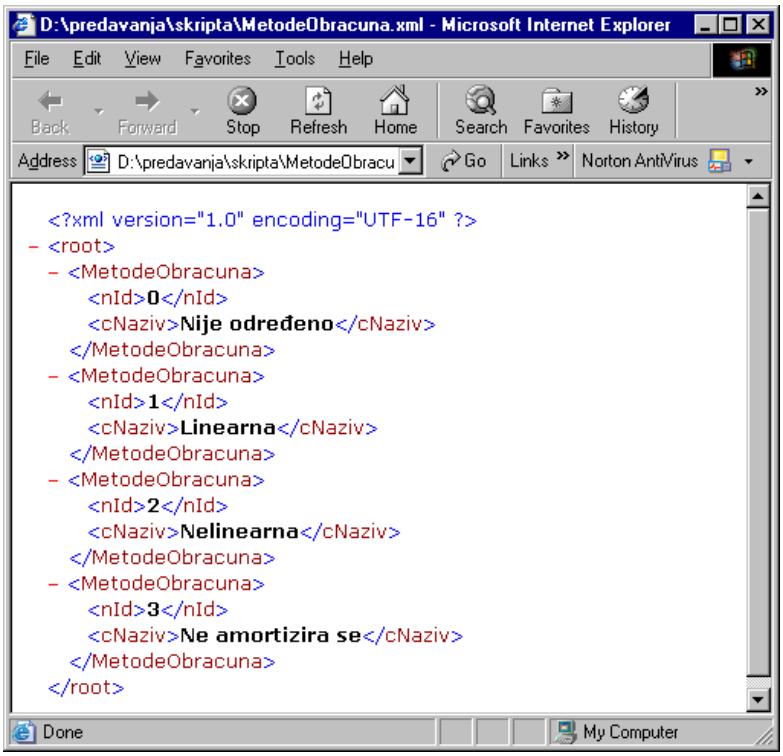
Rješenje se nalazi u korištenju XML-a. Podaci umjesto da budu združeni s HTML kodom bit će odvojeni i kao takvi poslani klijentu. Popunjavanje HTML stranice podacima s odvija se na strani klijenta (korištenjem JavaScripta). Napomenimo da je sada lako ostvariti različite vrste operacija nad podacima (izdvajanje skupa podataka, sortiranje i slično) na strani klijenta. Osim što je

lako, osvježavanje stranice se odvija vrlo brzo jer su podaci već prisutni na strani klijenta.
Ostaje samo upitati se kakvu podršku pruža internetski pretraživač koji stoji na raspolaganju (Internet Explorer najvjerojatnije).
Internet Explorer pruža podršku za:

- pregledavanje XML dokumenata
- umetanje XML dokumenata u HTML kao Data Islands
- vezivanje XML podataka i HTML elemenata
- transformiranje XML-a pomoću XSL-a.

5.1. XML *data islands*

Najjednostavija provjera podržava li web preglednik XML ili ne je otvoriti XML datoteku u web pregledniku. Rezultat bi trebao izgledati kao na slici.



Primjer 55: Pregled XML dokumenta u Internet Exploreru

XML podaci mogu biti integrirani unutar HTML stranice upotrebom HTML oznake <xml>. Ta oznaka služi za kreiranje data islanda (otok s podacima). Zgodno je promatrati XML podatke kao otok koji je vidljiv u moru HTML dijela koda web stranice. Data islandi su zapravo XML podaci koji su referencirani ili uključeni u HTML stranicu. Ako su XML podaci referencirani, tada se XML podaci nalaze u nekoj eksternoj datoteci.

Ako se XML podaci uključuju u samu HTML stranicu, tada se cijeli XML dokument upisuje između oznaka <xml> i </xml> kojima se označava početak i kraj data islanda.

```
<xml id="xmlData">
<?xml version="1.0"?>
      <poruka>
            <prima>Tomo</prima>
            <salje>Janica</salje>
            <tekst>Ovo je XML data island!</tekst>
      </poruka>
</xml>
```

Primjer 56: Izgled XML data islanda kada su podaci uključeni u HTML stranicu

Napomena:
HTML oznaka <xml> ima atribut id čija vrijednost je zapravo ime po kojem će se referencirati data island negdje u HTML ili JavaScript kodu.

Kada se XML podaci nalaze u nekoj vanjskoj datoteci, tada se oni mogu uključiti u data island podešavanjem vrijednosti src atributa u oznaci XML, na vrijednost koja ukazuje na XML datoteku. Atribut src može upućivati na lokalnu datoteku ili na neku datoteku koja se nalazi bilo gdje na internetu.
Primjer XML data islanda kada se koristi referenca na XML dokument koji se nalazi u istom direktoriju kao i HTML datoteka:

```
<xml id="xmlData" src=”xmlData.xml”></xml>
```

```
<?xml version="1.0"?>
      <poruka>
            <prima>Tomo</prima>
            <salje>Janica</salje>
            <tekst>Ovo je eksterna XML datoteka!</tekst>
      </poruka>
```

Primjer 57: XML datoteka xmlData koju referencira prethodni primjer

Primjer XML data islanda koji koristi referencu na XML dokument u URL obliku koji se može nalaziti bilo gdje na webu:

```
<xml id="xmlData" src=”http://www.test.com/xmlData.xml”></xml>
```

Kako HTML stranica odnosno web aplikacija može koristiti više data islanda vidimo da je vrlo jednostavno koristiti XML podatke s više različitih izvora s interneta.

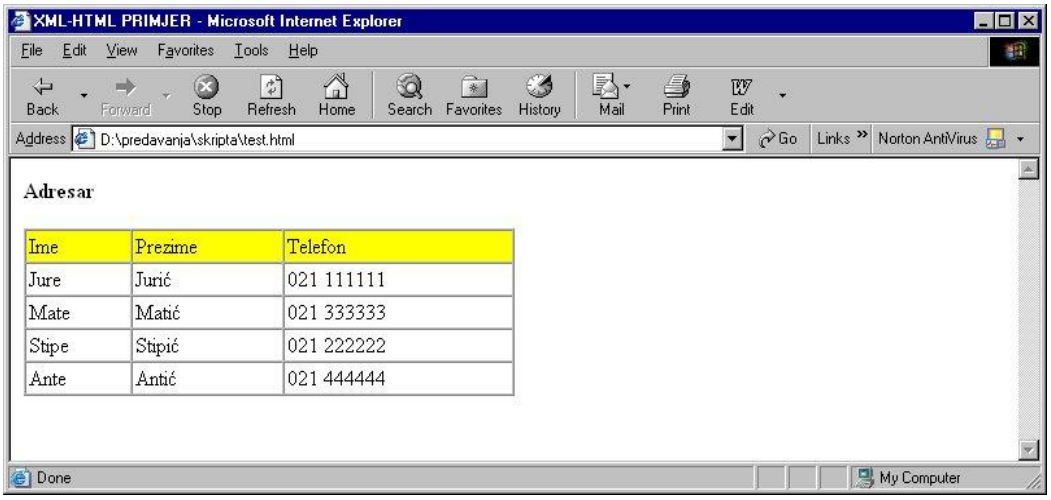
5.2. Povezivanje HTML elemenata s XML podacima

Atribut id koji je dodijeljen XML oznaci koristi se za referenciranje podataka iz data islanda. Korištenjem HTML oznaka (koje omogućavaju referenciranje data islanda) možemo prikazati (i formatirati) XML podatke. Neke od HTML oznaka koje mogu prihvatiti reference za data islande: div, span, textarea, img, table, select, button, a, frame, applet, param. Pogledajmo jednu HTML stranicu koja sadrži XML podatke i prikazuje ih u obliku jedne tablice:

```
<html>
      <head>
            <title>XML-HTML PRIMJER</title>
      </head>
<body>
<xml id="xmlAdresar">
<?xml version="1.0" ?>
<kontakti>
      <osoba>
            <ime>Jure</ime>
            <prezime>Jurić</prezime>
            <telefon>021 111111</telefon>
            <mobitel>098 111111</mobitel>
            <adresa>Z 90</adresa>
      </osoba>
      <osoba>
            <ime>Mate</ime>
            <prezime>Matić</prezime>
            <telefon>021 333333</telefon>
            <mobitel>098 333333</mobitel>
            <adresa>H 10</adresa>
      </osoba>
      <osoba>
            <ime>Stipe</ime>
            <prezime>Stipić</prezime>
            <telefon>021 222222</telefon>
            <mobitel>098 222222</mobitel>
            <adresa>K 20</adresa>
      </osoba>
      <osoba>
            <ime>Ante</ime>
            <prezime>Antić</prezime>
            <telefon>021 444444</telefon>
            <mobitel>098 444444</mobitel>
            <adresa>E 40</adresa>
      </osoba>
</kontakti>
</xml>
<p><b>Adresar</b></p>
      <table   datasrc="#xmlAdresar"   border="1"   width="50%"
```

```
      cellspacing="0" cellpadding="2">
      <thead>
            <tr bgcolor="yellow">
            <td>Ime</td><td>Prezime</td><td>Telefon</td>
            </tr>
      </thead>
      <tr>
            <td><div datafld="ime"></div></td>
            <td><div datafld="prezime"></div></td>
            <td><div datafld="telefon"></div></td>
      </tr>
      </table>
</body>
</html>
```

Napomena:
Uočimo tabličnu strukturu podataka koji se prikazuju. Naime podaci imaju jedan zajednički element kontakti, a unutar se nalaze elementi osoba. Svaki od elemenata osoba ima svoje pripadajuće elemente. XML podaci se dakle mogu interpretirati kao jedna tablica koja ima retke (elementi osoba) sa pripadajućim kolonama (ime, prezime, telefon, mobitel, adresa). Ovo je važno uočiti radi lakšeg razumijevanja referenciranja XML podataka iz HTML elemenata.



Slika 28: Prikaz prethodne HTML stranice u Internet Exploreru

U primjeru možemo primijetiti da su HTML elementi povezani s podacima u XML data islandu korištenjem atributa *datasrc, datafld i datapagesize*:

- *datasrc atribut* se koristi za referenciranje XML data islanda na temelju ID
- *datafld* se koristi za referenciranje stupca XML podataka (sjetimo se tablične strukture XML podataka)

- *datapagesize* atribut koji koristi HTML element <table> za određivanje koliko redaka XML-a će se odjednom prikazati u tablici

Proučimo malo detaljnije prethodni primjer. XML podaci su navedeni unutar elementa <xml> i atribut id ima vrijednost xmlAdresar. HTML element <table> se koristi za prikaz podataka i ovom elementu je postavljen atribut datasrc na vrijednost koja odgovara vrijednosti atribut id od data islanda (u ovom slučaju xmlAdresar).

Primijetimo da na listi HTML elemenata koji podržavaju povezivanje XML-a i HTML-a ne postoji HTML element <td>. Za kopiranje vrijednosti elemenata iz XML u određena polja tablice koristit ćemo se naravno sa <td>, ali ćemo se poslužiti malim trikom.

Unutar elementa <td> stavit ćemo element koji ima podršku za povezivanje XML-a i HTML-a. Unutar elementa koristit ćemo atribut datafld (referencira stupac) i u njega ćemo upisati vrijednost stupca koji želimo prikazati.

Prethodni primjer bez dodatnih definiranja prikaza tablice bi izgledao ovako:

```
<table datasrc="#xmlAdresar">
      <tr>
            <td><div datafld="ime"></div></td>
            <td><div datafld="prezime"></div></td>
            <td><div datafld="telefon"></div></td>
      </tr>
</table>
```

Kada se koristi HTML element <table> za prikaz XML podataka, svi podaci (tj. redci, odnosno slogovi ako XML datoteku zamislimo kao bazu podatka) iz XML-a se prikazuju u tablici. Ako podataka ima malo (kao u našem primjeru), to ne predstavlja problem, ali ako je podataka puno, tada će podaci biti prikazani u jednoj velikoj tablici, što će biti nepregledno. Atribut datapagesize može se tada iskoristiti kako bi se ograničio broj slogova koji se istovremeno prikazaju u tablici. Vrijednost toga atributa određuje broj redaka tablice koji će biti vidljivi na zaslonu. Naravno, u tom slučaju potrebno je omogućiti korisniku pristup i ostalim podacima.

JavaScript vidi elemente HTML stranice kao objekte oslanjajući se na DHTML Document Object Modela. Svaki objekt ima svoje metode i vrijednosti (property).

Objekt table ima metode koje se koriste u tu svrhu:

- *firstPage* (prikaži prvu stranicu podataka)
- *lastPage* (prikaži zadnju stranicu podataka)
- *nextPage* (prikaži sljedeću stranicu podataka)
- *previousPage* (prikaži prethodnu stranicu podataka).

Uporabom ovih metoda i JavaScripta moguće je jednostavno napisati korisničko sučelje za pregled svih stranica tablice.

U ovom poglavlju razmatrano je korištenje HTML elementa <table> za prikaz podataka iz XML dokumenta. Kao što je već napomenuto, osim <table> elementa postoje i drugi elementi za povezivanje XML-a i HTML-a.

Ovdje je potrebno naglasiti kako za svaki data island Internet Explorer kreira i ADO (Active Data Object) recordset objekt. Taj objekt sadrži podatke iz XML data islanda u obliku skupa slogova. Korištenjem toga objekta dizajneru web stranice pružaju se dodatne mogućnosti za navigaciju kroz XML podatke, dodavanje i brisanje podataka iz tog objekta.

HTML elementi (osim elementa table) povezuju se samo s trenutnim slogom (prvi slog je trenutan po defaultu) iz ADO recordset objekta. Navigacija kroz slogove recordset objekta može se programirati u JavaScript jeziku korištenjem metoda i vrijednosti koje posjeduje ADO recordset objekt.

```
<html>
      <head>
            <title>XML-HTML PRIMJER</title>
      </head>
<body>
<xml id="xmlAdresar">
<?xml version="1.0" ?>
      <kontakti>
       ... (kao u prethodnom primjeru)
      </kontakti>
</xml>
<p><b>Adresar</b></p>
<p>
<span datasrc="#xmlAdresar" datafld="ime"></span><br>
<span datasrc="#xmlAdresar" datafld="ime"></span><br>
<span datasrc="#xmlAdresar" datafld="ime"></span><br>
</p>
</body>
</html>
```

Primjer 58: HTML stranica koja prikazuje samo sadržaj prvog sloga iz recordset objekta

Slika 28: izgled prethodne HTML stranice

Naravno, za poziv metode recordset objekta potrebno je taj objekt nekako referencirati. Data island objekt ima ime identične vrijednosti atributa id XML elementa. Taj objekt između ostalih ima i vrijednost (property) tipa recordset, pa referenca na taj njega izgleda ovako (u našem slučaju): *xmlAdresar.recordset .*

Recordset objekt ima sljedeće metode za navigaciju po slogovima:

- *MoveFirst()* - idi na prvi slog
- *MovePrevious()* - idi na prehodni slog
- *MoveNext()* - idi na sljedeći slog
- *MoveLast()* - idi na zadnji slog

Recordset objekt ima sljedeće vrijednosti (property):

- *AbsolutePosition* – trenutno aktivni slog (redak)
- *RecordCount* – ukupan broj redaka

Pozivanjem ovih metoda ADO recordset objekta mijenja se trenutno aktivni slog, a samim time i sadržaj HTML elemenata povezanih sa recordset objektom.

5.3. Sažetak

Primjena povezivanja HTML i XML je korisna i možemo je jednostavno i brzo iskoristiti. Upravo onoliko koliko je jednostavno koristiti to povezivanje, toliko su jednostavne i funkcije koje je moguće podržati. Ako nešto nije predviđeno standardom, tada to ili uopće nije moguće ili je vrlo teško izvedivo (pomoću nekih trikova).

U sljedećem poglavlju ćemo se susresti s jednim od jezika temeljenim na XML standardu. Radi se o jeziku koji pruža gotovo neograničene mogućnosti manipulacije i transformacije XML podataka u željeni oblik.

6. XSL - eXtensible Stylesheet Language

XSL je jezik kojem je prema svojoj prvoj specifikaciji primarna namjena transformiranje XML dokumenata iz jednog oblika u drugi. Treba napomenuti da je taj jezik mnogo više od jezika sposobnog za transformiranje XML dokumenta u HTML (ili neki drugi tekstualni) format.

XSL se sastoji od tri dijela:

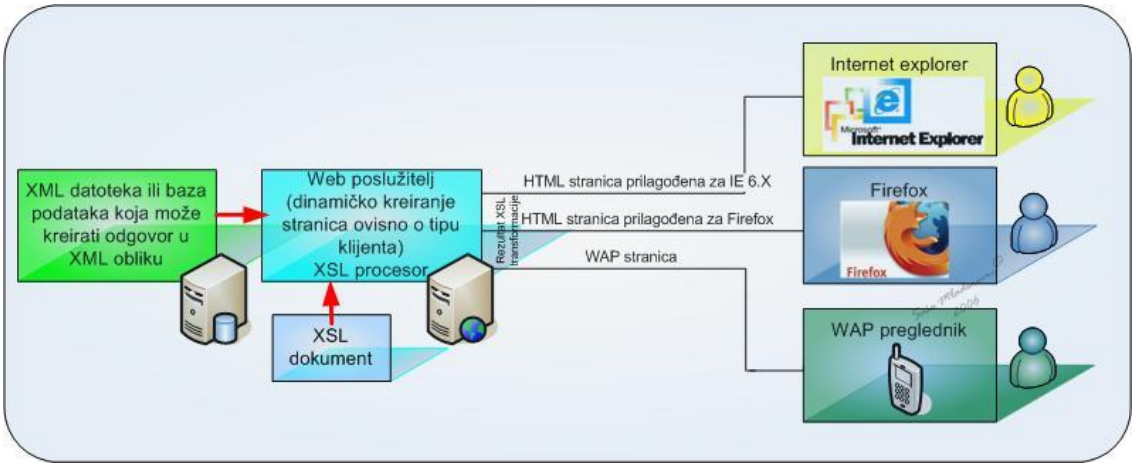
XSLT (XSL Transformations) jezik je kojim se definiraju transformacije na XML dokumentu

XPATH je jezik koji omogućava selektiranje dijelova XML dokumenta (na kojima će se izvršiti neka transformacija ili odrediti ukupan broj nekih elemenata)

XSL-FO (XSL Formatting objects) jezik je kojim se može potpuno definirati izgled podataka iz XML dokumenta. Njime se nećemo baviti.

U ovom dijelu razmotrit će se XSL transformacije (XSLT) i XPATH. Upotreba XSLT-a bez XPATH-a je zapravo beskorisna jer je praktički nemoguće napraviti nešto ozbiljnije ako se ne može obraditi samo neki skup podataka nego uvijek obrađivati cijeli XML dokument.

XSLT je jezik koji je različit od konvencionalnih programskih jezika jer je baziran na predlošcima (template rules) koji definiraju kako će XML dokument biti obrađen. Za razliku od konvencionalnih programskih jezika koji su sekvencijalni, predlošci se mogu navoditi bilo kojim redom, jer je u svojoj prirodi XSLT deklarativan jezik.



Slika 29: Shematski prikaz XSL transformacije

XSLT predlošci određuju kakav će biti rezultat kada XSL procesor pronađe određeni uzorak u ulaznom XML dokumentu. Drugim riječima u XSL dokumentu se deklarira što rezultat treba sadržavati kada XSL procesor naiđe na element IME u ulaznom XML dokumentu.

6.1. Primjer XSL transformacije

Prije nego se upustimo u proučavanje XSL jezika, proučimo jedan jednostavan primjer transformacije XML dokumenta u HTML tablicu.

XML dokument sadrži podatke koje treba prikazati na HTML stranici (datoteka people.xml). Iz primjera je vidljivo kako je datoteka pristigla od stranog partnera. Ponovno dolazi do izražaja problem razmjene podataka i razumijevanja podataka koje tumače različiti korisnici koji mogu imati jezične, kulturološke i druge razlike:

```
<?xml version="1.0" encoding=”UTF-8” ?>
<?xml-stylesheet type="text/xsl" href="people.xsl"?>
<PEOPLE>
     <PERSON PERSONID="p1">
          <NAME>Mark Wilson</NAME>
          <ADDRESS>Somewhere        Circle,        Canberra,
          Australia</ADDRESS>
          <TEL>(++612) 12345</TEL>
          <FAX>(++612) 12345</FAX>
          <EMAIL>markwilson@somewhere.com</EMAIL>
     </PERSON>
     <PERSON PERSONID="p2">
          <NAME>Tracey Wilson</NAME>
          <ADDRESS>Zootle    Road,    Cape    Town,    South
          Africa</ADDRESS>
          <TEL>(++2721) 531 9090</TEL>
          <FAX>(++2721) 531 9090</FAX>
          <EMAIL>TraceyWilson@somewhere.com</EMAIL>
     </PERSON>
     <PERSON PERSONID="p3">
          <NAME>Jodie Foster</NAME>
          <ADDRESS>30 Animal Road, New York, USA</ADDRESS>
          <TEL>(++1) 3000 12345</TEL>
          <FAX>(++1) 3000 12345</FAX>
          <EMAIL>JodieFoster@somewhere.com</EMAIL>
     </PERSON>
```

```
      <PERSON PERSONID="p4">
            <NAME>Lorrin Maughan</NAME>
            <ADDRESS>1143 Winners Lane, London, UK</ADDRESS>
            <TEL>(++94) 17 12345</TEL>
            <FAX>(++94) 17 12345</FAX>
            <EMAIL>LorrinMaughan@somewhere.com</EMAIL>
      </PERSON>
      <PERSON PERSONID="p5">
            <NAME>Steve Rachel</NAME>
            <ADDRESS>90210    Beverly    Hills,    California,
            USA</ADDRESS>
            <TEL>(++1) 2000 12345</TEL>
            <FAX>(++1) 2000 12345</FAX>
            <EMAIL>SteveRachel@somewhere.com</EMAIL>
      </PERSON>
</PEOPLE>
```

Ovaj XML dokument sadrži osim podataka i referencu na XSL dokument (datoteka people.xsl) kojim se vrši transformacija. U ovom slučaju datoteka people.xsl se mora nalaziti u istom direktoriju u kojem je i people.xml datoteka. Inače, ova referenca može biti i u URL obliku.
Primjer XSL reference unutar samog XML dokumenta

```
<?xml-stylesheet type="text/xsl" href="people.xsl"?>
```

Ako u Internet Exploreru otvorimo XML dokument koji ima referencu na XSL dokument, tada će se automatski obaviti XSL transformacija i bit će prikazan rezultat transformacije, a ne XMLdokument.

Dakle, imamo podatke u XML obliku i želimo kreirati HTML stranicu koja prikazuje podatke u tabličnom obliku otprilike ovako:

Name	Address	Tel	Fax	Email
Mark Wilson	Somewhere Circle, Canberra, Australia	(++612) 12345	(++612) 12345	Mark.Wilson@somewhere.com
Tracey Wilson	Zootle Road, Cape Town, South Africa	(++2721) 531 9090	(++2721) 531 9090	Tracey.Wilson@somewhere.com
Jodie Foster	30 Animal Road, New York, USA	(++1) 3000 12345	(++1) 3000 12345	Jodie.Foster@somewhere.com
Lorrin Maughan	1143 Winners Lane, London, UK	(++94) 17 12345	(++94) 17 12345	Lorrin.Maughan@somewhere.com
Steve Rachel	90210 Beverly Hills, California, USA	(++1) 2000 12345	(++1) 2000 12345	Steve.Rachel@somewhere.com

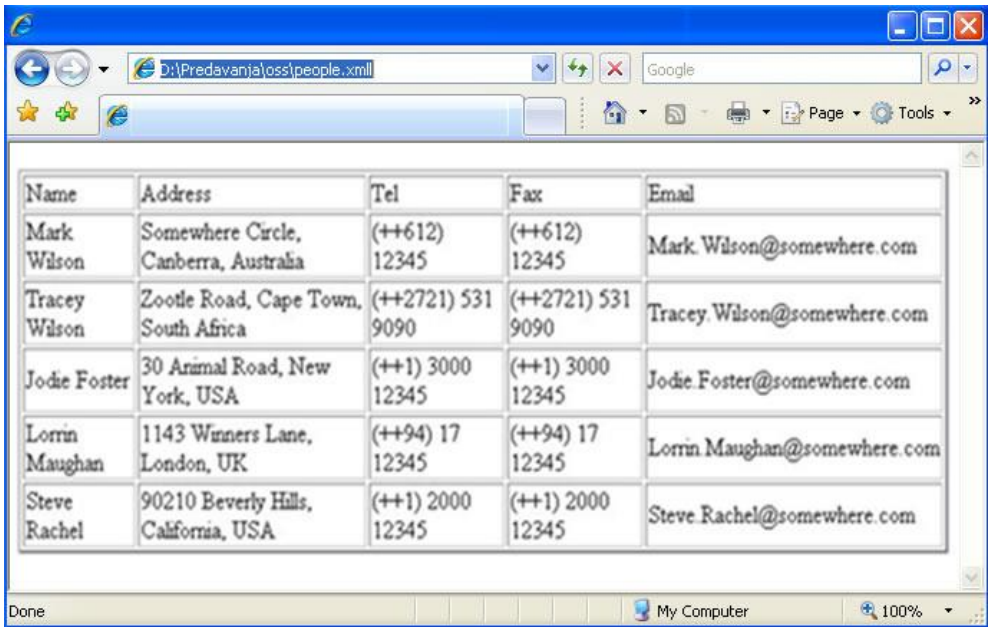
HTML kod stranice koju treba generirati izgleda ovako:

```
<table border="2">
<tr>
<td>Name</td>
<td>Address</td>
<td>Tel</td>
<td>Fax</td>
<td>Email</td>
</tr>
<tr>
<td>Mark Wilson</td>
<td>Somewhere Circle, Canberra, Australia</td>
<td>(++612) 12345</td>
<td>(++612) 12345</td>
<td>Mark.Wilson@somewhere.com</td>
</tr>
...
<table>
```

XSL dokument koji obavlja ovu transformaciju (datoteka people.xsl):

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:template match="/">
<html>
<body>
<table border="2">
<tr>
            <td>Name</td>
            <td>Address</td>
            <td>Tel</td>
            <td>Fax</td>
            <td>Email</td>
      </tr>
      <xsl:for-each select="PEOPLE/PERSON">
      <tr>
         <td><xsl:value-of select="NAME"/></td>
         <td><xsl:value-of select="ADDRESS"/></td>
         <td><xsl:value-of select="TEL"/></td>
         <td><xsl:value-of select="FAX"/></td>
         <td><xsl:value-of select="EMAIL"/></td>
      </tr>
      </xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Primjer 59: Datoteka people.xsl kojom se obavlja ova transformacija



Name	Address	Tel	Fax	Email
Mark Wilson	Somewhere Circle, Canberra, Australia	(++612) 12345	(++612) 12345	Mark.Wilson@somewhere.com
Tracey Wilson	Zootle Road, Cape Town, South Africa	(++2721) 531 9090	(++2721) 531 9090	Tracey.Wilson@somewhere.com
Jodie Foster	30 Animal Road, New York, USA	(++1) 3000 12345	(++1) 3000 12345	Jodie.Foster@somewhere.com
Lorrin Maughan	1143 Winners Lane, London, UK	(++94) 17 12345	(++94) 17 12345	Lorrin.Maughan@somewhere.com
Steve Rachel	90210 Beverly Hills, California, USA	(++1) 2000 12345	(++1) 2000 12345	Steve.Rachel@somewhere.com

Slika 30: Rezultat prethodne transformacije u Internet Exploreru

Princip XSL transformacije je sljedeći:
XSL dokument sadrži predloške (jedan ili više njih). Svaki predložak definira za koje elemente XML dokumenta će se sadržaj predloška (sve ono što se nalazi između oznaka predloška, tj. <xsl:template> i </xsl:template>) proslijediti na izlaz (rezultat).

XSL procesor čita XML dokument od početka (odnosno od početnog čvora prema kraju XML dokumenta) i za svaki čvor odnosno element pokušava pronaći odgovarajući XSL predložak. Ako ga nađe, tada će sadržaj tog predloška preusmjeriti na izlaz. Nakon toga XSL procesor nastavlja dalje s čitanjem XML dokumenta i pretraživanjem predložaka. Konačni rezultat transformacije će biti zbir sadržaja predložaka za koje je pronađen odgovarajući element u XML dokumentu. (Napomena: ako neki predložak definira da se za element <PERSON> na izlaz pošalje “nešto”, tada će se to “nešto” u krajnjem rezultatu pojaviti onoliko puta koliko XML dokument ima elemenata <PERSON>).

Bitno je napomenuti da je krajnji rezultat ovisan o redoslijedu elemenata u XML dokumentu.
U našem slučaju postoji samo jedan predložak koji je definiran za početni čvor (match=”/”). Sve ono što se nalazi unutar toga predloška kopirat će se na izlaz. Izuzetak su neki drugi XSLT elementi koje će onda XSL procesor zasebno obraditi i umjesto njih na izlaz kopirati nešto drugo.

6.2. XSLT elementi

XSL je jezik baziran na XML-u i ima svoj rječnik. Prisjetimo se da XML ne definira unaprijed imena elemenata, nego ih autor XML dokumenta sam određuje. XSLT koristi unaprijed zadana imena elemenata koja imaju unaprijed definirano značenje. Drugim riječima, XSL uvodi svoj rječnik koji je standard i kao takav ima neka svoja pravila korištenja. Odnosno svaki XSL dokument je ujedno i XML dokument i kao takav mora poštivati pravila i sintaksu XML-a. (Uočimo da XSL dokument people.xsl počinje XML deklaracijom <?xml version="1.0"?>.)

6.2.1. XSL *namespace*

XSL elementi moraju nekako biti prepoznatljivi XSL procesoru u odnosu na ostale elemente XSL dokumenta. U tu svrhu na početku svakog XSL dokumenta navodi se element xsl:stylesheet koji definira XSL namespace i verziju XSL standarda prema kojoj je napisan XSL dokument.

Ta deklaracija izgleda ovako:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 version="1.0">
…
</xsl:stylesheet>
```

Primjer 60: XSL deklaracija

XSL namespace definira prefiks xsl (npr.: xsl:template) koji će biti zajednički za sve elemente koji pripadaju XSL jeziku. Navedenom deklaracijom XSL procesor će moći lako razlučiti xsl elemente od ostalih elemenata XSL dokumenta.

6.2.2. xsl:template

Već je napomenuto kako XSLT koristi predloške za definiranje transformacije i moglo bi se reći da su predlošci osnova XSLT-a. XSLT se može koristiti i bez predložaka, ali se onda ne mogu iskoristiti sve njegove mogućnosti.

Predlošci se definiraju pomoću oznaka xsl:template. Svaki predložak sadrži i atribut match. Atribut match koristi uzorke (patterns) za određivanje elemenata XML dokumenta za koje se izvodi predložak.

Selektiranje početnog čvora (root node) XML dokumenta

match="/"	Selektiranje početnog čvora XML dokumenta

```
<?xml version="1.0"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 version="1.0">
<xsl:template match="/">
<html>
<xsl:apply-templates/>
</html>
</xsl:template>

<xsl:template …>
…
</xsl:template>
…
</xsl:stylesheet>
```

Primjer 61: Predložak koji se izvodi za početni čvor

Ovo je uobičajen primjer korištenja predložaka. Čest je slučaj potreba za transformiranjem XML dokumenta u neki drugi oblik (npr. HTML). Tada je za početni čvor ulaznog dokumenta logično kreirati predložak, a koji će kreirati početne oznake HTML-a unutar kojih će se nalaziti sve ostalo.

Ovdje se unutar predloška koristi xsl element xsl:apply-templates. Želi li se razumjeti čemu ovaj element služi, potrebno je pogledati što bi bio rezultat sljedećeg XSL dokumenta.

```
<?xml version="1.0"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 version="1.0">
<xsl:template match="/">
<html>
</html>
</xsl:template>
<xsl:template …>
…
</xsl:template>
…
</xsl:stylesheet>
```

Primjer 62: Predložak koji često ne odgovara našim potrebama

Razlika između ova dva predloška je u tome što će u prvom slučaju XSL procesor za početni čvor dati na izlaz <html> i izvoditi ostale predloške i tek nakon što dođe do kraja dokumenta na izlaz dodati </html>. XSL element <xsl:apply-templates> predstavlja na neki način poziv procedure čiji će rezultat (izvođenje drugih predložaka) biti stavljen na mjesto ovog XSL elementa.
Drugi XSL dokument sadrži predložak koji za početni čvor daje rezultat <html></html>. Nakon toga XSL procesor nastavlja čitanjem XML dokumenta i pozivanjem odgovarajućih predložaka tako da rezultat neće biti korektan (rezultat bi mogao izgledati ovako <html></html> <body></body>…)

Selektiranje elementa XML dokumenta

Ako se želi kreirati predložak koji će biti pozvan kada XSL procesor pročita određeni element XML dokumenta, tada se za atribut match postavlja kao vrijednost ime tog elementa.

match=”ime elementa”	Selektiranje nekog elementa XML dokumenta

```
<?xml version="1.0"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 version="1.0">
<xsl:template match="/">
<html>
<body>
<xsl:apply-templates/>
</body>
</html>
</xsl:template>
<xsl:template match="PERSON">
<p>ELEMENT PERSON FOUND</p>
</xsl:template>
</xsl:stylesheet>
```

Primjer 63: Predložak koji se izvodi za određeni element XML dokumenta

Rezultat ove transformacije biti će (primijenjen na dokument people.xml)

```
<html>
<body>
<p>ELEMENT PERSON FOUND</p>
<p>ELEMENT PERSON FOUND</p>
<p>ELEMENT PERSON FOUND</p>
<p>ELEMENT PERSON FOUND</p>
<p>ELEMENT PERSON FOUND</p>
```

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

Napomena: Za svaki element PERSON u XML dokumentu će se izvršiti predložak (za taj element).

6.2.3. xsl:value-of

Za sada smo se susreli sa XSL dokumentima koji vrše transformacije ulaznog XML dokumenta bez čitanja podataka (iz tog dokumenta) i njihova upisivanja u rezultat. Međutim, najčešći oblik transformacije je onaj gdje se podaci iz ulaznog dokumenta kopiraju u nekom drugom obliku (ili npr. sortirani po abecedi ili filtrirani po nekom kriteriju). Bez te mogućnosti XSL transformacije bi bile neupotrebljive tako da se očekuje da i XSL posjeduje takve mogućnosti. XSL element xsl:value-of je jedan takav element. On ima atribut select kojim se definira ime elementa čija se vrijednost kopira u rezultat.

```
<?xml version="1.0"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 version="1.0">
<xsl:template match="/">
<html>
<body>
<xsl:apply-templates/>
</body>
</html>
</xsl:template>
<xsl:template match="PERSON">
<p><xsl:value-of select=”NAME” /></p>
</xsl:template>
</xsl:stylesheet>
```

Primjer 64: XSLT element xsl:value-of

Napomena: U ovom primjeru korišten je XSL element xsl:value-of koji kopira sadržaj navedenog elementa. Atribut select je relativan u odnosu na trenutni element XML dokumenta na kojem je XSL procesor. U ovom slučaju to konkretno znači da XSL procesor poziva predložak (<xsl:template match="PERSON">) kada naiđe na element PERSON unutar XML dokumenta. Unutar toga predloška se nalazi XSLT element <xsl:value-of select=”NAME”> koji kopira vrijednost elementa NAME koji je potomak trenutnog elementa (a to je PERSON).

Rezultat ove transformacije biti će (primijenjen na dokument people.xml)

```
<html>
<body>
<p>Mark Wilson</p>
<p>Tracey Wilson</p>
<p>Jodie Foster</p>
<p>Lorrin Maughan</p>
<p>Steve Rachel</p>
</body>
</html>
```

6.2.4. xsl:for-each

U našem slučaju svaki element PERSON ima samo jedan element NAME kao potomak. Kada bi postojalo više potomaka NAME za jedan element PERSON, <xsl:value-of select=”NAME” /> bi kopirao samo vrijednost prvog elementa NAME. Nekad nam je cilj ispisati sadržaj svih potomaka nekog elementa i tada koristimo XSLT element <xsl:for-each>. Ovaj element ima također atribut select kojim se definira za koje sve elemente treba izvršiti određeni dio transformacije.

```
<?xml version="1.0"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 version="1.0">
<xsl:template match="/">
<html>
<body>
<xsl:apply-templates/>
</body>
</html>
</xsl:template>
<xsl:template match="PEOPLE">
<xsl:for-each select=”PERSON”>
<p><xsl:value-of select=”NAME” /></p>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

Primjer 65: XSL dokument koji koristi <xsl:for-each>

Dva prethodna XSL dokumenta imaju isti rezultat, ali rade na malo drukčiji način. Oba primjera imaju predložak za početni čvor i u tome se ne razlikuju. Razlika je u drugom predlošku: prvi je za element PERSON, a drugi je za element PEOPLE. Dakle, XSL procesor će pozvati predložak iz prvog

dokumenta više puta (jer se element PERSON pojavljuje u XML dokument više puta) i svaki put će predložak ispisati samo jedno ime. Predložak iz drugog XSL dokumenta će biti pozvan samo jedanput (jer se element PEOPLE pojavljuje samo jedanput), ali se unutar njega nalazi petlja (<xsl:for-each>) koja se ponavlja za sve elemente PERSON koji su direktni potomci elementa PEOPLE. Petlja za svaki element PERSON ispisuje sadržaj njegova direktnog potomka NAME.

Rezultat ove transformacije će biti isti kao i rezultat prethodne transformacije, ali ostvaren na drugi način:

```
<html>
<body>
<p>Mark Wilson</p>
<p>Tracey Wilson</p>
<p>Jodie Foster</p>
<p>Lorrin Maughan</p>
<p>Steve Rachel</p>
</body>
</html>
```

6.2.5. xsl:sort

U prethodna dva primjera dobili smo ispis sadržaja elemenata NAME, ali što ako želimo prikazati podatke iz XML dokumenta sortirane, ili samo neke koje zadovoljavaju određeni uvjet. Za ovaj prvi zahtjev koristit ćemo XSLT element <xsl:sort>. Ovaj element ima između ostalih dva atributa koja se najčešće koriste:
atribut select kojim se definira po čemu će podaci biti poredani (vrijednost ovoga atributa je ime elementa po kojem se sortira)
atribut order kojim se definira kreira li se rastući ili padajući niz (dvije su moguće vrijednosti: ascending (default) ili descending)

```
<?xml version="1.0"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 version="1.0">
<xsl:template match="/">
<html>
<body>
<xsl:apply-templates/>
</body>
</html>
```

```
</xsl:template>
<xsl:template match="PEOPLE">
<xsl:for-each select=”PERSON”>
  <xsl:sort select=”NAME” order=”descending” />
<P><xsl:value-of select=”NAME” /></P>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

Primjer 66: XSLT element xsl:sort

Rezultat ove transformacije su podaci sortirani u padajućem poretku:

```
<html>
<body>
<p>Tracey Wilson</p>
<p>Steve Rachel</p>
<p>Mark Wilson</p>
<p>Lorrin Maughan</p>
<p>Jodie Foster</p>
</body>
</html>
```

6.2.6. xsl:if

Do sada smo upoznali mogućnosti ispisa i sortiranja podataka iz XML dokumenta. Nedostaje nam mogućnost filtriranja podataka i ovdje ćemo se upoznati s jednim od načina: korištenjem XSLT elementa <xsl:if>. Ovaj element pruža nam mogućnost izvođenja određenog dijela XSL koda ovisno o postavljenom uvjetu. Sadrži atribut test čija vrijednost je logički izraz. Ako je logički uvjet ispunjen, dio koda unutar toga elementa se izvršava, inače ne.

```
<?xml version="1.0"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 version="1.0">
<xsl:template match="/">
<html>
<body>
<xsl:apply-templates/>
</body>
</html>
</xsl:template>
<xsl:template match="PEOPLE">
<xsl:for-each select=”PERSON”>
  <xsl:if test=”NAME=’Jodie Foster’” >
<p><xsl:value-of select=”NAME” /></p>
```

```
<p><xsl:value-of select=”ADDRESS” /></p>
<p><xsl:value-of select=”TEL” /></p>
<p><xsl:value-of select=”EMAIL” /></p>
</xsl:if>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

Primjer 67: XSLT element xsl:if

U ovome primjeru naveden je logički uvjet koji je ispunjen samo kad programska petlja dođe do elementa PERSON koji sadrži element NAME s vrijednosti ‘Jodie Foster’. Taj uvjet će biti ispunjen samo za jedan element PERSON i tada će biti ispisan sadržaj njegovih direktnih potomaka.

Napomena:
Trenutno aktivni element unutar petlje nije element NAME nego element PERSON. Da nije tako, tada nam ova XSL transformacija ne bi dala željene vrijednosti jer element NAME nema direktne potomke NAME, ADDRESS, TEL i EMAIL.

Rezultat ove transformacije su podaci koji zadovoljavaju navedeni uvjet::

```
<html>
<body>
<p>Jodie Foster</p>
<p>30 Animal Road, New York, USA</p>
<p>(++1) 3000 12345</p>
<p>JodieFoster@somewhere.com</p>
</body>
</html>
```

6.3. XSL transformacije iz JavaScripta

Do sada smo naučili kako se u XSL dokumentu definira transformacija XML dokumenta. Naučili smo kako se kreiraju XML i XSL dokumenti i kako se unutar samog XML dokumenta može upisati referenca na XSL dokument. Međutim, ovakav način referenciranja XSL dokumenata je jednostavan za korištenje, ali često i nezadovoljavajući.

Naime, često se tek tijekom izvođenja aplikacije određuje kojim XSL dokumentom se obavlja transformacija, npr. kada aplikacija ima sučelje kojim korisnik može sortirati podatke po želji.

Za postizanje prethodno navedenog morat ćemo se koristiti Microsoft XML parserom koji dolazi uz Internet Explorer. XML parser nam služi za čitanje, kreiranje, mijenjanje i manipulaciju XML dokumentima.

Iz JavaScripta možemo eksplicitno kreirati Microsoft.XMLDOM ActiveX objekt i u njega učitati željenu XML datoteku (napomena: i XSL dokument je XML dokument).
Microsoft XMLDOM objekt sadrži brojne vrijednosti (property) i metode od kojih ćemo mi sada koristiti samo neke.

Metode XMLDOM objekta:
load - učitavanje datoteke u XML dokument
loadXML - učitava string u XML dokument
transformNode - transformira XML dokument pomoću navedenog XSL dokumenta (metoda vraća rezultat transformacije)

Vrijednosti XMLDOM objekta:
async - "false" sprječava daljnje izvođenje programa dok se cijeli XML dokument ne učita (true dozvoljava)

```
<html>
<body>
<script language="Javascript">

  var oXmlDoc = new ActiveXObject("Microsoft.XMLDOM");
  oXmlDoc.async = false;
  oXmlDoc.load("people.xml");

  var oXslDoc = new ActiveXObject("Microsoft.XMLDOM");
  oXslDoc.async = false;
  oXslDoc.load("people.xsl");

  // XSL transformacija
  document.write(oXmlDoc.transformNode(oXslDoc)
</script>
</body>
</html>
```

Primjer 68: Učitavanje XML i XSL dokumenata u XMLDOM

Prethodni primjer učitava XML i XSL dokument i rezultat XSL transformacije upisuje u HTML stranicu.

Međutim, primjena XSL nam dopušta razne kombinacije. Moguće je iskoristiti XSL za generiranje dijela HTML koda koji se upisuje u neki HTML element pomoću innerHTML metoda.
Primjer popunjavanje sadržaja HTML elementa rezultatom transformacije

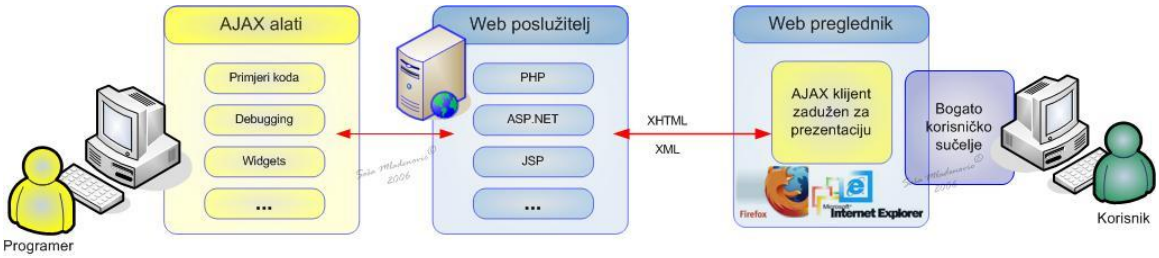
```
tblpodaci.innerHTML(oXmlDOC.transformNode(oXslDOC)
```

Naravno, u ovom slučaju XSL transformacija ne treba generirati zaglavlje HTML stranice, već samo onaj dio koda koji treba upisati u HTML element.

Moguće je kreirati XSL dokument koji prepisuje XML dokument u samog sebe, ali sa sortiranim podacima. Ovaj XSL dokument se može iskoristiti za popunjavanje XML data islanda sa sortiranim podacima. Povezivanje HTML elementa s XML data islandom ostaje nedirnuto, ali nakon što se promijeni sadržaj XML data islanda, mijenja se i prikaz (npr. tablice).
Primjer učitavanje u jedan XML data island rezultata transformacije drugog XML data islanda

```
xmlSort.loadXML(xmlOrig.transformNode(xslSort))
```

Uporabom JavaScript jezika, XML-a i XSL-a moguće je poboljšati osjećaj interaktivnosti s korisnikom. U posljednje vrijeme zajednica programera prepoznala je mogućnosti kombinacije ovih temeljnih tehnologija pa se one jednim imenom nazivaju AJAX. Najjednostavnije je ideju AJAX-a prikazati slikom:



Slika 31: Ideja AJAX-a

AJAX postavlja JavaScript tehnologiju i "*XMLHttpRequest*" objekt između web forme i poslužitelja. Nakon što korisnik završi s unosom podataka u formu, oni se šalju JavaScript skripti a ne direktno na poslužitelj. Tada skripta šalje podatke poslužitelju na obradu, te korisnik uopće nije svjestan ikakvog toka obrade podataka jer se stranica u pregledniku ne osvježava. Nakon obrade, poslužitelj vraća podatke JavaScript funkciji koja odlučuje što s njima dalje napraviti. Popularnost AJAX-a revitalizira znanje osnovnih tehnologija (JavaScripta, XML-a i XSL-a pored XHTML-a).

7. Literatura

DREYFUS, M.,: *Code_en_Stock JavaScript*, CampusPress, Paris, 2001

FLANAGAN, D.,: *JavaScript, The Definitive Guide*, 4th ed., O'Reilly, 2002.

FLANAGAN, D.,: *JavaScript Pocket Reference*, O'Reilly, 1998

GRAHAM, I.S.,: *HTML Stylesheet Source Book*, Wailey Computer Publishing, 1997

HAYES, D.,: *Vodič kroz HTML i XHTML*, 3. izdanje, Miš, Zagreb 2002

MEYER, E.A.,: *Cascading Style Sheets, The Definitive Guide*, 2nd ed., O'Reilly, 2004.

MORISS, B.,: *Formation a ... HTML*, Microsoft Press, Quebeck, 2000

MUSCIANO, C., KENNEDY, B.: *HTML & XHTML, The Definitive Guide*, 5th ed., O'Reilly, 2002.

YOUNG, M.J.,: *Formation a ... XML*, Microsoft Press, Quebeck, 2000

W3 SCHOOLS, http://www.w3schools.com/

W3C WORLD WIDE WEB CONSORTIUM, http://www.w3c.org/

8. Dodaci

8.1. Dodatak 1

8.1.1. Popis svih oznaka HTML 4.01./XTML 1.0. standarda[18]

Oznake su poredane po abecedi.

- NN: označava prvu verziju Netscapea koji podržava tu oznaku
- IE: označava prvu verziju Interner Explorera koja podržava tu oznaku
- DTD: označava u kojem od XHTML 1.0 DTD je oznaka dopuštena (S=Strict, T=Transitional, and F=Frameset)

Tag	Description	NN	IE	DTD
<!--...-->	Defines a comment	3.0	3.0	STF
<!DOCTYPE>	Defines the document type			STF
<a>	Defines an anchor	3.0	3.0	STF
<abbr>	Defines an abbreviation	6.2		STF
<acronym>	Defines an acronym	6.2	4.0	STF
<address>	Defines an address element	4.0	4.0	STF
<applet>	Deprecated. Defines an applet	2.0	3.0	TF
<area>	Defines an area inside an image map	3.0	3.0	STF
<b>	Defines bold text	3.0	3.0	STF
<base>	Defines a base URL for all the links in a page	3.0	3.0	STF
<basefont>	Deprecated. Defines a base font	3.0	3.0	TF
<bdo>	Defines the direction of text display	6.2	5.0	STF
<big>	Defines big text	3.0	3.0	STF
<blockquote>	Defines a long quotation	3.0	3.0	STF
<body>	Defines the body element	3.0	3.0	STF
** **	Inserts a single line break	3.0	3.0	STF
<button>	Defines a push button	6.2	4.0	STF
<caption>	Defines a table caption	3.0	3.0	STF
** **	Deprecated. Defines centered text	3.0	3.0	TF
<cite>	Defines a citation	3.0	3.0	STF
<code>	Defines computer code text	3.0	3.0	STF
<col>	Defines attributes for table columns		3.0	STF
<colgroup>	Defines groups of table columns		3.0	STF
<dd>	Defines a definition description	3.0	3.0	STF
<del>	Defines deleted text	6.2	4.0	STF
<dir>	Deprecated. Defines a directory list	3.0	3.0	TF
** **	Defines a section in a document	3.0	3.0	STF

[18] Preuzeto sa http://www.w3cschools.com

<dfn>	Defines a definition term		3.0	STF
<dl>	Defines a definition list	3.0	3.0	STF
<dt>	Defines a definition term	3.0	3.0	STF
****	Defines emphasized text	3.0	3.0	STF
<fieldset>	Defines a fieldset	6.2	4.0	STF
<font>	Deprecated. Defines text font, size, and color	3.0	3.0	TF
<form>	Defines a form	3.0	3.0	STF
<frame>	Defines a sub window (a frame)	3.0	3.0	F
<frameset>	Defines a set of frames	3.0	3.0	F
<h1> to <h6>	Defines header 1 to header 6	3.0	3.0	STF
<head>	Defines information about the document	3.0	3.0	STF
<hr>	Defines a horizontal rule	3.0	3.0	STF
<html>	Defines an html document	3.0	3.0	STF
<i>	Defines italic text	3.0	3.0	STF
<iframe>	Defines an inline sub window (frame)	6.0	4.0	TF
<img>	Defines an image	3.0	3.0	STF
<input>	Defines an input field	3.0	3.0	STF
<ins>	Defines inserted text	6.2	4.0	STF
<isindex>	Deprecated. Defines a single-line input field	3.0	3.0	TF
<kbd>	Defines keyboard text	3.0	3.0	STF
<label>	Defines a label for a form control	6.2	4.0	STF
<legend>	Defines a title in a fieldset	6.2	4.0	STF
** **	Defines a list item	3.0	3.0	STF
<link>	Defines a resource reference	4.0	3.0	STF
<map>	Defines an image map	3.0	3.0	STF
<menu>	Deprecated. Defines a menu list	3.0	3.0	TF
<meta>	Defines meta information	3.0	3.0	STF
<noframes>	Defines a noframe section	3.0	3.0	TF
<noscript>	Defines a noscript section	3.0	3.0	STF
<object>	Defines an embedded object		3.0	STF
** **	Defines an ordered list	3.0	3.0	STF
<optgroup>	Defines an option group	6.0	6.0	STF
<option>	Defines an option in a drop-down list	3.0	3.0	STF
<p>	Defines a paragraph	3.0	3.0	STF
<param>	Defines a parameter for an object	3.0	3.0	STF
<pre>	Defines preformatted text	3.0	3.0	STF
<q>	Defines a short quotation	6.2		STF
<s>	Deprecated. Defines strikethrough text	3.0	3.0	TF
<samp>	Defines sample computer code	3.0	3.0	STF
<script>	Defines a script	3.0	3.0	STF
<select>	Defines a selectable list	3.0	3.0	STF
<small>	Defines small text	3.0	3.0	STF
****	Defines a section in a document	4.0	3.0	STF
<strike>	Deprecated. Defines strikethrough text	3.0	3.0	TF
****	Defines strong text	3.0	3.0	STF

<style>	Defines a style definition	4.0	3.0	STF
<sub>	Defines subscripted text	3.0	3.0	STF
<sup>	Defines superscripted text	3.0	3.0	STF
<table>	Defines a table	3.0	3.0	STF
<tbody>	Defines a table body		4.0	STF
<td>	Defines a table cell	3.0	3.0	STF
<textarea>	Defines a text area	3.0	3.0	STF
<tfoot>	Defines a table footer		4.0	STF
<th>	Defines a table header	3.0	3.0	STF
<thead>	Defines a table header		4.0	STF
<title>	Defines the document title	3.0	3.0	STF
<tr>	Defines a table row	3.0	3.0	STF
<tt>	Defines teletype text	3.0	3.0	STF
<u>	Deprecated. Defines underlined text	3.0	3.0	TF
** **	Defines an unordered list	3.0	3.0	STF
<var>	Defines a variable	3.0	3.0	STF
<xmp>	Deprecated. D			

Oznake su poredane porema namjeni.

Start tag	Purpose	NN	IE
Basic Tags			
<!DOCTYPE>	Defines the document type		
<html>	Defines a html document	3.0	3.0
<body>	Defines the body element	3.0	3.0
<h1> to <h6>	Defines header 1 to header 6	3.0	3.0
<p>	Defines a paragraph	3.0	3.0
** **	Inserts a single line break	3.0	3.0
<hr>	Defines a horizontal rule	3.0	3.0
<!--...-->	Defines a comment	3.0	3.0
Char Format			
<b>	Defines bold text	3.0	3.0
<font>	Deprecated. Defines the font face, size, and color of text	3.0	3.0
<i>	Defines italic text	3.0	3.0
****	Defines emphasized text	3.0	3.0
<big>	Defines big text	3.0	3.0
****	Defines strong text	3.0	3.0
<small>	Defines small text	3.0	3.0
<sup>	Defines superscripted text	3.0	3.0
<sub>	Defines subscripted text	3.0	3.0
<bdo>	Defines the direction of text display	6.2	5.0
<u>	Deprecated. Defines underlined text	3.0	3.0

Output			
<pre>	Defines preformatted text	3.0	3.0
<code>	Defines computer code text	3.0	3.0
<tt>	Defines teletype text	3.0	3.0
<kbd>	Defines keyboard text	3.0	3.0
<var>	Defines a variable	3.0	3.0
<dfn>	Defines a definition term		3.0
<samp>	Defines sample computer code	3.0	3.0
<xmp>	Deprecated. Defines preformatted text. Use <pre> instead	3.0	3.0
Blocks			
<acronym>	Defines an acronym	6.2	4.0
<abbr>	Defines an abbreviation	6.2	
<address>	Defines an address element	4.0	4.0
<blockquote>	Defines an long quotation	3.0	3.0
** **	Deprecated. Defines centered text	3.0	3.0
<q>	Defines a short quotation	6.2	4.0
<cite>	Defines a citation	3.0	3.0
<ins>	Defines inserted text	6.2	4.0
<del>	Defines deleted text	6.2	4.0
<s>	Deprecated. Defines strikethrough text	3.0	3.0
<strike>	Deprecated. Defines strikethrough text	3.0	3.0
Links			
<a>	Defines an anchor	3.0	3.0
<link>	Defines a resource reference	4.0	3.0
Frames			
<frame>	Defines a sub window (a frame)	3.0	3.0
<frameset>	Defines a set of frames	3.0	3.0
<noframes>	Defines a noframe section	3.0	3.0
<iframe>	Defines an inline sub window (frame)	6.0	4.0
Input			
<form>	Defines a form	3.0	3.0
<input>	Defines an input field	3.0	3.0
<textarea>	Defines a text area	3.0	3.0
<button>	Defines a push button	6.2	4.0
<select>	Defines a selectable list	3.0	3.0
<optgroup>	Defines an option group	6.0	
<option>	Defines an item in a list box	3.0	3.0
<label>	Defines a label	6.2	4.0
<fieldset>	Defines a fieldset	6.2	4.0
<legend>	Defines a title in a fieldset	6.2	4.0

Lists			
** **	Defines an unordered list	3.0	3.0
** **	Defines an ordered list	3.0	3.0
** **	Defines a list item	3.0	3.0
<dir>	Deprecated. Defines a directory list	3.0	3.0
<dl>	Defines a definition list	3.0	3.0
<dt>	Defines a definition term	3.0	3.0
<dd>	Defines a definition description	3.0	3.0
<menu>	Deprecated. Defines a menu list	3.0	3.0
Images			
<img>	Defines an image	3.0	3.0
<map>	Defines an image map	3.0	3.0
<area>	Defines an area inside an image map	3.0	3.0
Tables			
<table>	Defines a table	3.0	3.0
<caption>	Defines a table caption	3.0	3.0
<th>	Defines a table header	3.0	3.0
<tr>	Defines a table row	3.0	3.0
<td>	Defines a table cell	3.0	3.0
<thead>	Defines a table header		4.0
<tbody>	Defines a table body		4.0
<tfoot>	Defines a table footer		4.0
<col>	Defines attributes for table columns		3.0
<colgroup>	Defines groups of table columns		3.0
Styles			
<style>	Defines a style definition	4.0	3.0
** **	Defines a section in a document	3.0	3.0
****	Defines a section in a document	4.0	3.0
Meta Info			
<head>	Defines information about the document	3.0	3.0
<title>	Defines the document title	3.0	3.0
<meta>	Defines meta information	3.0	3.0
<base>	Defines a base URL for all the links in a page	3.0	3.0
<basefont>	Deprecated. Defines a base font	3.0	3.0
Programming			
<script>	Defines a script	3.0	3.0
<noscript>	Defines a noscript section	3.0	3.0
<applet>	Deprecated. Defines an applet	3.0	3.0
<object>	Defines an embedded object		3.0
<param>	Defines a parameter for an object	3.0	3.0

8.1.2. ISO-8859 entiteti – simboli

Result	Description	Entity Name	Entity Number
	non-breaking space		
¡	inverted exclamation mark	&iexcl;	&#161;
¤	currency	&curren;	&#164;
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
¦	broken vertical bar	&brvbar;	&#166;
§	section	&sect;	&#167;
¨	spacing diaeresis	&uml;	&#168;
©	copyright	&copy;	&#169;
ª	feminine ordinal indicator	&ordf;	&#170;
«	angle quotation mark (left)	&laquo;	&#171;
¬	negation	&not;	&#172;
	soft hyphen	&shy;	&#173;
®	registered trademark	&reg;	&#174;
™	trademark	&trade;	&#8482;
¯	spacing macron	&macr;	&#175;
°	degree	&deg;	&#176;
±	plus-or-minus	&plusmn;	&#177;
²	superscript 2	&sup2;	&#178;
³	superscript 3	&sup3;	&#179;
´	spacing acute	&acute;	&#180;
µ	micro	&micro;	&#181;
¶	paragraph	&para;	&#182;
·	middle dot	&middot;	&#183;
¸	spacing cedilla	&cedil;	&#184;
¹	superscript 1	&sup1;	&#185;
º	masculine ordinal indicator	&ordm;	&#186;
»	angle quotation mark (right)	&raquo;	&#187;
¼	fraction 1/4	&frac14;	&#188;
½	fraction 1/2	&frac12;	&#189;
¾	fraction 3/4	&frac34;	&#190;
¿	inverted question mark	&iquest;	&#191;
×	multiplication	&times;	&#215;
÷	division	&divide;	&#247;

8.1.3. ISO-8859 entiteti – znakovi

Result	Description	Entity Name	Entity Number
À	capital a, grave accent	À	À
Á	capital a, acute accent	Á	Á
Â	capital a, circumflex accent	Â	Â
Ã	capital a, tilde	Ã	Ã
Ä	capital a, umlaut mark	Ä	Ä
Å	capital a, ring	Å	Å
Æ	capital ae	Æ	Æ
Ç	capital c, cedilla	Ç	Ç
È	capital e, grave accent	È	È
É	capital e, acute accent	É	É
Ê	capital e, circumflex accent	Ê	Ê
Ë	capital e, umlaut mark	Ë	Ë
Ì	capital i, grave accent	Ì	Ì
Í	capital i, acute accent	Í	Í
Î	capital i, circumflex accent	Î	Î
Ï	capital i, umlaut mark	Ï	Ï
Ð	capital eth, Icelandic	Ð	Ð
Ñ	capital n, tilde	Ñ	Ñ
Ò	capital o, grave accent	Ò	Ò
Ó	capital o, acute accent	Ó	Ó
Ô	capital o, circumflex accent	Ô	Ô
Õ	capital o, tilde	Õ	Õ
Ö	capital o, umlaut mark	Ö	Ö
Ø	capital o, slash	Ø	Ø
Ù	capital u, grave accent	Ù	Ù
Ú	capital u, acute accent	Ú	Ú
Û	capital u, circumflex accent	Û	Û
Ü	capital u, umlaut mark	Ü	Ü
Ý	capital y, acute accent	Ý	Ý
Þ	capital THORN, Icelandic	Þ	Þ
ß	small sharp s, German	ß	ß
à	small a, grave accent	à	à
á	small a, acute accent	á	á
â	small a, circumflex accent	â	â
ã	small a, tilde	ã	ã
ä	small a, umlaut mark	ä	ä
å	small a, ring	å	å
æ	small ae	æ	æ
ç	small c, cedilla	ç	ç
è	small e, grave accent	è	è
é	small e, acute accent	é	é

ê	small e, circumflex accent	ê	ê
ë	small e, umlaut mark	ë	ë
ì	small i, grave accent	ì	ì
í	small i, acute accent	í	í
î	small i, circumflex accent	î	î
ï	small i, umlaut mark	ï	ï
ð	small eth, Icelandic	ð	ð
ñ	small n, tilde	ñ	ñ
ò	small o, grave accent	ò	ò
ó	small o, acute accent	ó	ó
ô	small o, circumflex accent	ô	ô
õ	small o, tilde	õ	õ
ö	small o, umlaut mark	ö	ö
ø	small o, slash	ø	ø
ù	small u, grave accent	ù	ù
ú	small u, acute accent	ú	ú
û	small u, circumflex accent	û	û
ü	small u, umlaut mark	ü	ü
ý	small y, acute accent	ý	ý
þ	small thorn, Icelandic	þ	þ
ÿ	small y, umlaut mark	ÿ	ÿ

8.1.4. Ostali entiteti koje podržava HTML

Result	Description	Entity Name	Entity Number
Œ	capital ligature OE	&OElig;	&#338;
œ	small ligature oe	&oelig;	&#339;
Š	capital S with caron	&Scaron;	&#352;
š	small S with caron	&scaron;	&#353;
Ÿ	capital Y with diaeres	&Yuml;	&#376;
ˆ	modifier letter circumflex accent	&circ;	&#710;
˜	small tilde	&tilde;	&#732;
	en space		
	em space		
	thin space		
	zero width non-joiner	&zwnj;	&#8204;
	zero width joiner	&zwj;	&#8205;
	left-to-right mark	&lrm;	&#8206;
	right-to-left mark	&rlm;	&#8207;
–	en dash	&ndash;	&#8211;
—	em dash	&mdash;	&#8212;
‘	left single quotation mark	&lsquo;	&#8216;
’	right single quotation mark	&rsquo;	&#8217;
,	single low-9 quotation mark	&sbquo;	&#8218;
“	left double quotation mark	&ldquo;	&#8220;
”	right double quotation mark	&rdquo;	&#8221;
„	double low-9 quotation mark	&bdquo;	&#8222;
†	dagger	&dagger;	&#8224;
‡	double dagger	&Dagger;	&#8225;
…	horizontal ellipsis	&hellip;	&#8230;
‰	per mille	&permil;	&#8240;
‹	single left-pointing angle quotation	&lsaquo;	&#8249;
›	single right-pointing angle quotation	&rsaquo;	&#8250;
€	euro	&euro;	&#8364;

8.2. Dodatak 2

8.2.1. XSLT elementi

Oznake su poredane po abecedi.

- N: označava prvu verziju Netscapea koji podržava tu oznaku
- IE: označava prvu verziju Interner Explorera koja podržava tu oznaku

Element	Description	IE	N
apply-imports	Applies a template rule from an imported style sheet	6.0	
apply-templates	Applies a template rule to the current element or to the current element's child nodes	5.0	6.0
attribute	Adds an attribute	5.0	6.0
attribute-set	Defines a named set of attributes	6.0	6.0
call-template	Calls a named template	6.0	6.0
choose	Used in conjunction with <when> and <otherwise> to express multiple conditional tests	5.0	6.0
comment	Creates a comment node in the result tree	5.0	6.0
copy	Creates a copy of the current node (without child nodes and attributes)	5.0	6.0
copy-of	Creates a copy of the current node (with child nodes and attributes)	6.0	6.0
decimal-format	Defines the characters and symbols to be used when converting numbers into strings, with the format-number() function	6.0	
element	Creates an element node in the output document	5.0	6.0
fallback	Specifies an alternate code to run if the processor does not support an XSLT element	6.0	
for-each	Loops through each node in a specified node set	5.0	6.0
if	Contains a template that will be applied only if a specified condition is true	5.0	6.0
import	Imports the contents of one style sheet into another. **Note:** An imported style sheet has lower precedence than the importing style sheet	6.0	6.0
include	Includes the contents of one style sheet into another. **Note:** An included style sheet has the same precedence as the including style sheet	6.0	6.0
key	Declares a named key that can be used in the style sheet with the key() function	6.0	6.0
message	Writes a message to the output (used to report errors)	6.0	6.0
namespace-alias	Replaces a namespace in the style sheet to a different namespace in the output	6.0	

number	Determines the integer position of the current node and formats a number	6.0	6.0
otherwise	Specifies a default action for the <choose> element	5.0	6.0
output	Defines the format of the output document	6.0	6.0
param	Declares a local or global parameter	6.0	6.0
preserve-space	Defines the elements for which white space should be preserved	6.0	6.0
processing-instruction	Writes a processing instruction to the output	5.0	6.0
sort	Sorts the output	6.0	6.0
strip-space	Defines the elements for which white space should be removed	6.0	6.0
stylesheet	Defines the root element of a style sheet	5.0	6.0
template	Rules to apply when a specified node is matched	5.0	6.0
text	Writes literal text to the output	5.0	6.0
transform	Defines the root element of a style sheet	6.0	6.0
value-of	Extracts the value of a selected node	5.0	6.0
variable	Declares a local or global variable	6.0	6.0
when	Specifies an action for the <choose> element	5.0	6.0
with-param	Defines the value of a parameter to be passed into a template	6.0	6.0