UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI FAKULTET
DEPARTMAN ZA
MATEMATIKU I INFORMATIKU

Milica Perišić

JavaScript u Web design-u

- diplomski rad -

Novi Sad, 2007

Predgovor

Autori *Web* prezentacija neprestano tragaju za alatima kojima će, sa što manje napora, njihove prezentacije biti još privlačnije. Ovo se naročito odnosi na stručnjake koji su mnogo vičniji pisanju, grafičkom dizajnu i oblikovanju izgleda stranica nego "suvom" programiranju. Pokretačka ideja iz koje je proizašao *JavaScript* je baš potreba za uvođenjem interaktivnosti u *HTML*[*].

JavaScript je osmislio *Brendan Eich* iz *Netscape*-a pod imenom *Mocha*, kasnije *LiveScript*, i konačno preimenovao u *JavaScript.* Preimenovanje *LiveScript*-a u *JavaScript* desilo se skoro istovremeno sa *Netscape*-ovom novinom u *Netscape Navigator browser*-u. Naime, u isto vremenskom periodu *Netscape* je osposobio *Navigator* za Java tehnologiju. *JavaScript* je prvi put uveden i , tako reći, "pušten u promet", u *Netscape*-ovom *browser*-u verzije 2.0B3 u decembru 1995.

I pored sličnih imena, *JavaScript* i Java su programski jezici koji samo spolja liče jedan na drugog. Programski jezik Java, kreiran od strane Sun Microsystems-a, je moćan i mnogo komplikovaniji objektno orijentisan programski jezik. U odnosu na Javu, *JavaScript* je dosta "slobodniji".

Da bi se izbegli nesporazumi oko zaštitnog imena, *Microsoft* je svoju implementaciju *JavaScript*-a nazvao *JScript*. *JScript* je prvi put zaživeo u *Internet Explorer*-u 3.0, koji je izdat avgusta 1996.

Ovaj diplomski rad obuhvata osnove programskog jezika *JavaScript*, od toga gde kod pisan u njemu smestiti pa sve do pravljenja vlastitih *JavaScript* funkcija, i predstavlja ga kroz primer *Web* prezentacije. Takođe, prožet je mnoštvom primera koji će upoznavanje sa ovim dosta popularnim i korisnim jezikom učiniti mnogo lakšim, a same pojmove dosta jasnijim.

Sintaksa *JavaScript*-a je pogodna za učenje i ne toliko komplikovana, mada se onima bez programerskog iskustva ipak preporučuje upoznavanje osnova programiranja jer dosta olakšavaju savladavanje *JavaScript*-a. Preporučuje se prethodno sticanje osnovnog znanja *HTML*-a.

Prvo poglavlje je uvod u kome se kratko opisuje *JavaScript* i nekoliko jezika slične primene. Takođe, dat je i opis primera *Web* stranice realizovane *JavaScript*-om.

Drugo poglavlje govori gde se *JavaScript* kod smešta i koji to načini postoje za njegovo pokretanje.

Treće poglavlje je sintaksa jezika *JavaScript* i neki osnovni principi programiranja.

[*] HyperText Markup Language

Četvrto poglavlje predstavlja upoznavanje sa objektima *JavaScript*-a kroz primere, zajedno sa prikazom *Web* stranica.

Peto poglavlje bavi se pojmom dinamičkog *HTML*-a i rešenjima potencijalnih problema vezanih za njegov prikaz u *browser*-u.

U šestom poglavlju skreće se pažnja na česte razloge problema, kao i na samu prevenciju.

Sedmo poglavlje prikazuje *Web* prezentaciju realizovanu JavaScriptom. Dati su prikazi svih stranica i njihovi opisi.

Zaključak ovog rada i rezime sadržani su u osmom poglavlju, zajedno sa kratkim odgovorom na pitanje šta je sledeći korak nakon savladavanja *JavaScript*-a.

Deveto poglavlje predviđeno je za dodatke. Dodaci su osmišljeni kao tabele sa ključnim rečima *JavaScript*-a, svojstvima, metodima i procedurama njegovih objekata, a takođe priložen je i spisak karakterističnih skriptova. Priložena su još dva skripta koji su deo propratne *Web* prezentacije ali nisu posebno obrađeni u tekstu.

Sadržaj

Poglavlje 1

Uvod

1.1 O *JavaScript*-u i njemu sličnim programskim jezicima

JavaScript pripada grupi tzv. *scripting* jezika. To su jezici koji se sastoje od redova izvršnog kompjuterskog koda koji se direktno umeću u *HTML* stranice.

Relativno novi programski jezik, takođe iz grupe scripting jezika, je *AJAX- Asynchronous JavaScript and XM*L[*]. Osnovna karakteristika *AJAX*-a je to što koristi asinhroni prenos podataka (*HTTP* zahteve) između *browser*-a i *Web* servera. Sa *AJAX*-om *JavaScript* skriptovi mogu komunicirati direktno sa serverom čineći aplikacije bržim i boljim.

WML tj. *Wireless Markup Language*, je jezik nastao iz *HTML*-a ali baziran je na *XML*-u i time dosta "strožiji". Ovaj skript jezik služi za pravljenje stranica koje se prikazuju u WAP *browser*-u i smatra se "lakšom" verzijom *JavaScript*-a. *WML* stranice sadrže samo reference na spoljašnje skriptove.

Skript jezik izveden iz Microsoft-ovog programskog jezika *Visual Basic*, naziva se *VBScript*. Skriptovi napisani u ovom jeziku umeću se u *HTML*, a *browser* čitajući *HTML* čita i izvršava skriptove *VBScript*-a.

JavaScript je jezik opšte namene, koristan i bez *HTML*-a. On se ugrađuje u servere, razvojne alate i druge *browser*-e. Od 2006. godine, najnovija verzija JavaScripta je *JavaScript* 1.7. Prethodna verzija, dakle 1.6, bila je u saglasnosti sa trećim izdanjem *ECMA-262*[**], kao i *JavaScript* 1.5.

Najvažnije osobine, pa samim tim mogućnosti koje *JavaScript* pruža svojim korisnicima/programerima su:

- *HTML* dizajnerima pruža programerski alat
- reagovanje na određene događaje (klik mišem, učitavanje stranice, …)
- čitanje i pisanje *HTML* elemenata
- provera podataka pre predaje serveru
- prepoznavanje *browser*-a
- pravljenje *cookies*-a

[*] E**X**tensible **M**arkup **L**anguage
[**] ECMA-262 Edition 3 može se pogledati i *download*-ovati u pdf formatu sa *www.ecma-international.org*

1.2 Primer *Web* prezentacija

Web prezentacija, kojom je u većini daljeg teksta predstavljen *JavaScript*, sastoji se iz nekoliko *Web* stranica i osmišljena je kao prezentacija modne ponude grada Novog sada.

Za programiranje ove *Web* prezentacije korišćen je *HTML* i *JavaScript*. Svi skriptovi su predstavljeni u radu, ako ne u izvornom onda u malo pojednostavljenom okruženju (gde se misli na dizajn stranice).

Prikaz stranica *Web* prezentacije sa kratkim opisom nalazi se u Poglavlju 7.

Poglavlje 2

Gde smestiti *JavaScript* kôd i kako se izvršava

Kako *JavaScript* nije *HTML*, da bi *browser* znao kako da prepozna koji delovi koda pripadaju skriptu, redovi skript koda ograđuju se parom oznaka <script> tj. </script>. U zavisnosti od *browser*-a oznaka <script> ima više atributa. Jedan od atributa zajedničkih za *Internet Explorer* i *Netscape* je *language*. Ovo je osnovni atribut zato što svaki *browser* prihvata različite jezike za skriptovanje. Dakle, redovi koda *JavaScript*-a pišu se između sledeće dve oznake:

```
<script language="JavaScript">
    ovde dolazi jedan ili više redova JavaScript-a
</script>
```

Ako oznaka za kraj skripta nije navedena, ne samo da skript neće pravilno raditi, već i *HTML* može da izgleda čudno na nekom mestu na stranici.

Kod najnovijih *browser*-a postoji drugi atribut koji služi da ukomponuje sadržaj spoljašnje skript datoteke u tekući dokument. Atribut src pokazuje na datoteku koja sadrži skript kod. Takva datoteka mora se završavati nastavkom *.js*, a skup oznaka izgleda ovako:

```
<script language="javascript" src="mojskript.js"> </script>
```

Pošto su svi redovi koda u spoljašnjoj datoteci, ni jedan red nije uključen između početne i završne oznake.

Pozicija oznaka u dokumentu zavisi od potrebe. Ponekad ima smisla ugnezditi ove oznake između para oznaka <*head*> ... </*head*>, a ponekad je bitno da skript bude smešten na određenu lokaciju u okviru sekcije <*body*> ... </*body*>.
Najčešća pozicija skripta u dokumentu je u odeljku <*head*> :

```
<html>
<head>
<title> dokument </title>
<script language="javascript">
</script>
<body>
</body>
</html>
```

U jedan dokument moguće je smestiti neograničen broj parova oznaka <script>, u oba odeljka- <*head*> i <*body*>.

Svaki red koda koji se nalazi između oznaka <script> i </script> jeste naredba *JavaScript*-a. Da bi bio usklađen sa navikama iskusnih programera, *JavaScript* prihvata tačku sa zarezom na kraju svake naredbe ali ona je opciona. Znak za povratak na početak reda na kraju naredbe je dovoljan da *JavaScript* zna da je naredba završena.

Vrste poslova koje naredba obavlja su:

- ✓ definisanje ili inicijalizovanje promenljive
- ✓ dodela vrednosti svojstvu ili promenljivoj
- ✓ promena vrednosti svojstva ili promenljive
- ✓ pozivanje metoda objekta
- ✓ pozivanje funkcijske rutine
- ✓ donošenje odluke.

U zavisnosti od toga šta je potrebno da skript obavlja, postoje četiri mogućnosti kada da se on pokrene:

a) dok se dokument učitava
b) odmah posle učitavanja dokumenta
c) kao odgovor na akciju korisnika
d) kada ga pozove druga naredba skripta.

Odlučujući faktor je pozicija naredbe skripta u dokumentu.

Pre nego što se pozabavimo ovim mogućnostima, važno je, naročito za poslednje tri, da se prethodno upoznamo sa pojmom funkcije.

Funkcija definiše blok skript naredbi koje mogu biti pozvane da se izvrše u nekom trenutku posle njihovog učitavanja u *browser*. Funkcije su jasno vidljive unutar oznaka <script> zato što svaka definicija počinje rečju *function*, iza koje sledi ime funkcije (i zagrade).

a) Naredbe skripta koje se izvršavaju dok se dokument učitava nazivaju se momentane naredbe. Oznake <script> i naredbe se uključuju u telo dokumenta.

```
<html>
<head>
<title> onload script </title>
<body>
<script language="javascript">
        document.write('<embed width=0 height=0 hidden="true"
              autostart="true" volume="5" loop="false"
                    src="backgroundSong.mp3">');
</script>
</body>
</html>
```

Dati skript dokumentu daje muzičku pozadinu- pesma mp3 formata backgroundSong.mp3.

b) Funkcija se može pozvati odmah po učitavanju stranice. Objekat prozora ima proceduru za obradu događaja koja se naziva onload=. Za razliku od mnogih drugih, koje se aktiviraju kao odgovor na akciju korisnika, procedura onload= aktivira se odmah pošto se sve komponente stranice učitaju u *browser*. Procedura za obradu događaja onload= nalazi se unutar oznake <*body*>.

```
<html>
<head>
<title> onload script </title>
<script language="javascript">
      function done() {
            alert("stranica je učitana.");
      }
</script>
<body onload="done()">
      tekst tela dokumenta
</body>
</html>
```

c) Pokretanje skripta akcijom korisnika slična je pokretanju odgođenog skripta po učitavanju dokumenta. Skript se pokreće npr. klikom miša na dugme.

```
<html>
<head>
<title> onclick script </title>
<script language="javascript">
	function alertuser() {
		alert("pokrenuli ste skript!");
	}
	</script>
<body>
	<form>
	<input type="text" name="dugme"
				value="pritisni me!" onclick="alertuser()">
	</form>
</body>
</html>
```

Funkcija alertuser() je definisana nakon učitavanja dokumenta i čeka poziv. Ako nikad ne bude pozvana, ne postoje nikakvi sporedni efekti.

d) U pozivu funkcije pomoću druge naredbe skripta, glavnu ulogu imaju parametri. Znači, funkcije se definišu tako da primaju parametre od naredbe koja ih poziva.

```
<html>
<head>
<title> onclick script </title>
<script language="javascript">
	function kazi(poruka) {
		alert("dugme kaze:  " + poruka);
	}
	</script>
<body>
<form>
<input type="text" name="dugme" value="klikni me!"
				onclick="kazi('Kliknuli  ste me!')">
</form>
</body>
</html>
```

Definicija funkcije zahteva skup zagrada posle imena funkcije. Vrednosti unutar tih zagrada nazivaju se parametri. Parametri obezbeđuju mehanizam "prosleđivanja" vrednosti iz jedne naredbe u drugu pomoću poziva funkcije.

Kada funkcija primi prametre, ona ulazne vrednosti pridružuje imenima promenljivih koja su zadata u definiciji funkcije. *JavaScript* toleriše neslaganje broja parametara u definiciji funkcije, sa brojem parametara koje prosleđuje pozivajuća naredba. Isto tako, ako ih je poslato više nego što je zadato u definiciji funkcije, *JavaScript* ne pravi problem.

Poglavlje 3

Osnove jezika

3.1 Promenljive

Svaka napisana naredba *JavaScript*-a na neki sebi svojstven način obrađuje podatke koji joj se proslede, na primer, prikaz teksta na ekranu pomoću Javascript-a ili uključivanje i isključivanje radio dugmeta u obrascu. Svaki pojedinačni deo informacije u programiranju se naziva vrednost. U *JavaScript*-u postoji nekoliko tipova vrednosti. Formalni tipovi podataka su:

- String- niz znakova unutar navodnika
- Number- bilo koji broj koji nije pod navodnicima
- Boolean- logičko istinito(true) ili neistinito(false)
- Null- lišeno vrednosti
- Object- sva svojstva i metodi koji pripadaju objektu ili nizu
- Function- definicija funkcije

Najpogodniji način za rad sa podacima u skriptu je dodeljivanje podataka promenljivima.

Postoji nekoliko načina da se definiše promenljiva u *JavaScript*-u. Najčešći i najbolji je korišćenjem ključne reči *var* iza koje sledi ime promenljive. Dakle, da bi promenljiva pod nazivom *brojUcenika* bila deklarisana, naredba JavaScripta je:

var brojUcenika

Dodela vrednosti promenljivoj vrši se korišćenjem operatora dodele. Najčešće korišćen operator dodele je znak jednakosti.

brojUcenika=29

Ključna reč *var* se koristi samo za deklaraciju ili inicijalizaciju promenljive- tj. samo jednom za vreme postojanja neke promenljive u dokumentu.

JavaScript promenljiva može da sadrži bilo koji tip vrednosti. Što se tiče imena promenljivih, postoje određena pravila koja moraju biti zadovoljena i samim tim obezbeđuju pravilno imenovanje promenljivih. Za ime promenljive ne može se uzeti ni jedna ključna reč. Promenljiva ne može da sadrži razmak. Najsigurnije je da se ime sastoji od samo jedne reči. Ako ipak postoji potreba za imenom od više reči, preporučuje se korišćenje jedne od dve konvencije za združivanje dve reči u jednu. Prva je da se te dve reči razdvoje donjom crtom, dok u drugoj kombinacija reči počinje malim slovom a početno slovo druge reči je veliko.

broj_ucenika
brojUcenika

Kada je promenljiva deklarisana unutar funkcije, može joj se pristupiti samo unutar te funkcije. Kada se data funkcija napusti, promenljiva prestaje da važi. Ovakve promenljive se nazivaju lokalne i može postojati više lokalnih promenljivih sa istim imenom u različitim funkcijama. Ako je promenljiva deklarisana van funkcije, tada je slobodna za pristup svim funkcijama sa stranice. Oblast važenja ovih promenljivih tzv. globalnih, počinje od mesta gde su deklarisane pa sve do zatvaranja stranice.

3.2 Operatori

3.2.1 Aritmetički operatori

Operator	Opis	Primer	Rezultat
+	Sabiranje	x=2 y=2 x+y	4
-	Oduzimanje	x=5 y=2 x-y	3
*	Množenje	x=5 y=4 x*y	20
/	Celobrojno deljenje	15/5 5/2	3 2.5
%	Ostatak deljenja	5%2 10%8 10%2	1 2 0
++	Povećanje za 1	x=5 x++	x=6
- -	Smanjenje za 1	x=5 x--	x=4

3.2.2 Operatori dodeljivanja

Operator	Primer	Objašnjenje
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
***=**	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

3.2.3 Operatori poređenja

operator	Opis
==	Jednako
!=	Nije jednako
>	Veće
<	Manje
>=	Veće ili jednako
<=	Manje ili jednako

3.2.4 Logički operatori

Operator	Opis
&&	AND
||	OR
!	NOT

3.2.5 String operator

String je obično tekst (pod navodnicima). Spajanje dve ili više String promenljive vrši se pomoću operatora +:

txt1="Dobar"
txt2="dan!"
txt3=txt1+txt2

Promenljiva txt3 sada je "Dobardan!"

Za razmak između dve string promenljive dovoljno je ugnezditi razmak između njih ili napraviti razmak posle jedne tj. pre druge promenljive.

txt1="Dobar"
txt2="dan!"
txt3=txt1+" "+txt2

ili

txt1="Dobar "
txt2="dan!"
txt3=txt1+txt2

3.2.6 Uslovni operator

JavaScript ima i jedan uslovni operator (ternarni) koji dodeljuje vrednost promenljivoj na osnovu nekog uslova. Sintaksa je sledeća:

imePromenljive= (uslov)? vrednost1:vrednost2

Dakle, ako je uslov zadovoljen, promenljivoj se dodeljuje vrednost vrednost1, inače vrednost2.

Prioritet operatora u *JavaScript*-u

Prioritet	Operator	Napomena
1	() [] function()	 Vrednost indeksa niza Bilo koji poziv funkcije
2	! ~ - + - - typeof void delete	Logičko NOT NOT nad bitovima Negacija Povećanje za 1 Smanjenje za 1 Brisanje elemenata niza/objekta
3	* / %	Množenje Deljenje Modulo
4	+ -	Sabiranje Oduzimanje
5	<< > >>	Pomeranje u operacijama nad bitovima
6	< <= > >=	Operatori poređenja
7	== !=	Jednakost
8	&	AND nad bitovima
9	^	XOR nad bitovima
10	|	OR nad bitovima
11	&&	Logičko AND
12	||	Logičko OR
13	?	Uslovni izraz
14	= += -= *= /= %= <<= >= >>= &= ^= |=	Operatori dodele
15	,	Zarez (za razdvajanje parametara)

3.3 Kontrolne strukture i petlje

Vrste naredbi koje donose oduke i ponavljaju se u petlji u programiranju se nazivaju kontrolne strukture. Važan deo komandne strukture je uslov. Svaki uslov je jedan logički izraz koji dobija vrednost *true* ili *false*. *JavaScript* obezbeđuje nekoliko vrsta kontrolnih struktura za različite programske situacije.

3.3.1 Konstrukcija *if* i *if.. else*. Switch

Najjednostavnija odluka u programu jeste praćenje neke grane ili putanje programa ako je ispunjen određen uslov. Formalna sintaksa za ovu konstrukciju je:

```
if (uslov) {
kod koji se izvršava ako je vrednost izraza true
}
```

Ako su umesto jedne grane potrebne dve ili više koje obrada treba da prati, koristi se *if.. else* tj. *if.. else if.. else* konstrukcija.

Formalna definicija *if.. else* je:

```
if (uslov) {
        kod koji se izvršava ako je vrednost izraza uslov true
}
else {
        kod koji se izvršava ako je vrednost izraza uslov false
}
```

Konstrukcija *if.. else if.. else* je pogodna kada je potrebno pratiti nekoliko izvršnih linija.

Sintaksa:

```
if (uslov1) {
        kod koji se izvršava ako je vrednost izraza uslov1 true
}
else if (uslov2) {
        kod koji se izvršava ako je vrednost izraza uslov2 true
}
else {
        kod koji se izvršava ako ni jedan od izraza uslov1 i uslov2 nema
vrednsot true
}
```

Pod nekim okolnostima, odluka tipa *true* ili *false* nije dovoljna za obradu podataka u skriptu. Svojstvo objekta ili vrednost promenljive mogu sadržavati bilo koju od nekoliko vrednosti i potreban je poseban put izračunavanja za svaku od njih. U *JavaScript*-u postoji kontrolna struktura koju koriste mnogi jezici. Na početku strukture se identifikuje o kom izrazu se radi i svakoj putanji izvršavanja dodeljuje se oznaka koja odgovara određenoj vrednosti. U pitanju je *switch* naredba.

```
switch(izraz) {
  case oznaka1:
        naredbe
        break
  case oznaka2:
        naredbe
        break
  default:
        naredbe
}
```

Izraz može biti bilo koji znakovni niz ili numerička vrednost. Naredba *default* obezbeđuje nastavak po putanji izvršavanja kada vrednost izraza ne odgovara ni jednoj oznaci naredbe case.

Naredba *break*, koja služi za izlazak iz petlje, ovde ima značajnu ulogu. Naime, ako nije navedeno *break* posle svake grupe naredbi u *case* granama, izvršiće se sve naredbe iz svake *case* grane bez obzira na to da li je nađena odgovarajuća oznaka.

3.3.2 Petlja *for*. Konstrukcije *while* i *do.. while*

Za neke *JavaScript* skriptove bitno je da mogu da kruže kroz svaki element niza ili svaku stavku obrasca. Jedna od struktura *JavaScript*-a koja dozvoljava ponavljanje prolazaka jeste petlja *for*.

```
for (pocetniIzraz; uslov; korak) {
    naredbe
}
```

Tri naredbe unutar zagrada (parametri petlje *for*) igraju ključnu ulogu. *pocetniIzraz* u petlji *for* izvršava se samo jednom, prvi put kada se pokreće petlja. Najčešća primena početnog izraza je dodela imena i početne vrednosti promenljivoj brojača petlje. Koristi se naredba *var* koja i deklariše ime promenljive i dodeljuje joj početnu vrednost. Kada je promenljiva brojača definisana u početnom izrazu, izraz *uslov* obično definiše do koje vrednosti brojač petlje treba da ide pre nego što se kruženje završi. Poslednja izraz, *korak,* izvršava se na kraju svakog izvršenja petlje nakon što su se pokrenule

sve naredbe unutar *for*-a. Obično je to povećavanje brojača petlje za jedan za svaki sledeći prolaz, tzv. inkrementacija vrednosti.

Sledeći primer je skript koji ispisuje brojeve od 0 do 10, svaki u posebnom redu. Znači, početna vrednost je 0, a krajnja, maksimalna, je 10. Inkrementacija iznosi 1. Dakle, sve dok *i* ne dobije vrednost 10, skript ispisuje brojeve svaki put povećacajući brojač *i* za 1.

```
<html>
<body>
<script language="javascript">
for (var i=0; i<=10; i++) {
        document.write("Broj " + i)
        document.write("<br>")
}
</script>
</body>
</html>
```

Svaki od parametara *for* petlje je opcioni. Ponekad se vrednost brojača petlje kontroliše na osnovu vrednosti koje se izvršavaju unutar tela petlje. Pogledajmo primer. Poslednji parametar- inkrementacija vrednosti- je izostavljen.

```
<html>
<body>
<script language="javascript">
document.write("Parni brojevi u otvorenom intervalu od 1 do 10 su:");
for (var i=1; i<=10) {
        if (i%2 == 0) {
                document.write("Broj " + i+ "je paran!");
                document.write("<br>");
                i += 2;
        }
        else {
                i++;
        }

}
</script>
</body>
</html>
```

Petlja *for* nije jedina vrsta ponavljačke petlje u *JavaScript*-u. Druga naredba, *while*, postavlja malo drugačiju petlju. *While* petlja pretpostavlja da će naredbe skripta doći u stanje u kome se automatski izlazi iz petlje.

```
while (uslov) {
        naredbe
}
```

Ova petlja izvodi akciju sve dok izraz *uslov* ne dobije vrednost false.

JavaScript nudi još jednu konstrukciju petlje zvanu *do.. while*. Formalna sintaksa za ovu konstrukciju je sledeća:

```
do {
        naredbe
}
while (uslov)
```

Razlika između *while* i *do.. while* petlje je ta što se u *do.. while* petlji naredbe izvršavaju bar jednom pre nego što se uslov ispita, dok u petlji *while* to nije slučaj.

3.3.3 *Break* i *continue*

U neke konstrukcije petlje skript ulazi samo kada se ispuni određeni uslov, pa više nemaju potrebu da nastave kruženje kroz ostatak vrednosti u opsegu brojača petlje. Za izlazak iz petlje koristi se naredba *break*. Ona govori *JavaScript*-u da izađe iz petlje *for* a izvršavanje skripta se nastavlja odmah iza te petlje, (dakle iza vitičaste zagrade koja zatvara telo petlje).

```
for (var i=0; i< array.length; i++) {
        if (array[i]. property == wantedValue) {
                naredbe;
                break
        }
}
```

Petlja *for* omogućava i da se preskoči izvršavanje naredbi unutar nje zbog samo jednog uslova. Može da postoji neka vrednost brojača za koju nije poželjno da se naredbe izvrše. Naredba *continue* primorava skript da, ako je uslov ispunjen, pređe na sledeći korak tj. poveća vrednost brojača i nastavi petlju od te vrednosti.

```
for(var i=1; i <=20; i++){
        if (i==13) {
        continue;
        naredbe
                ...
```

U slučaju ugnežđenih *for* petlji, naredba *continue* utiče na petlju *for* u čiji trenutni opseg spada i konstrukcija *if*.

3.4 Funkcije

Ono što se u drugim programskim jezicima naziva procedura ili potprogram, *JavaScript* naziva funkcijom. Funkcije možemo definisati kao blokove koda koji se mogu izvršavati više puta a pokrenuti nekom akcijom ili pozivom funkcije koji može biti bilo gde na stranici. Definicije funkcija pišu se u *head* odeljku.

```
<html>
<head>
<script type="text/javascript">
        function imeFunkcije (parametri) {
                naredbe
        }
</script>
</head>
…
```

Parametri su vrednosti koje funkcija prima od naredbe koja je poziva. Ako funkcija nema parametre, pišu se samo zagrade. Funkcija je u stanju da vrati vrednost naredbi koja je pozvala, ali ne obavezno.

Naredba return se koristi upravo u tom slučaju, dakle kada je potrebno odrediti vrednost koju funkcija vraća; i sve funkcije koje vraćaju neku vrednost moraju imati return naredbu.

```
function prod(a,b) {
        x=a*b
        return x
}
```

Kada funkcija primi vrednost parametara od naredbe koja ju je pozvala, te vrednosti se dodeljuju svojstvu arguments objekta function. Ovo svojstvo je niz vrednosti, gde je svaka vrednost parametar dodeljena članu niza čiji indeksi počinju nulom.

Ugnežđivanjem je moguće staviti jednu funkciju unutar druge. Ugnežđenoj funkciji, međutim, može pristupiti samo funkcija koja je obuhvata. Takođe, sve promenljive definisane u spoljnoj funkciji, dostupne su unutrašnjoj, ali obrnuto ne važi.

3.5 Nizovi

Nizovi u *JavaScript*-u su jedna od najkorisnijih struktura podataka. Elementi niza su numerisani, počevši od nule, i redni broj predstavlja indeks datog elementa. Da bi se pristupilo nekom elementu niza, potrebno je znati njegovo ime i indeks. Pošto vrednosti indeksa počinju nulom, broj elementata niza (što je određeno svojstvom niza length), uvek je za jedan veći od najvećeg indeksa niza. Elementi u nizovima *JavaScript*-a mogu da pripadaju bilo kom tipu podataka, uključujući i objekte.

Niz je smešten u promenljivoj tako da kada se pravi niz, promenljivoj se dodaje novi objekat niza. Posebna ključna reč- *new*- prethodi pozivu funkcije za generisanje niza u *JavaScript*-u, koja u memoriji obezbeđuje prostor za niz. Međutim, u *JavaScript*-u veličinu niza moguće je promeniti bilo kad.

new mojNiz = new Array(duzina)

Popunjavanje niza vrši se operatorom dodele, prethodno se u uglastim zagradama navodi indeks elementa.

mojNiz [0] = "Novi Sad"
mojNiz [1] = "Beograd"
mojNiz [2] = "Sombor"

Postoji i kraći način formiranja niza. Umesto serije naredbi dodele, podaci se predaju kao parametri konstruktoru *Array()*:

mojNiz = new Array ("Novi Sad", "Beograd", "Sombor")

Za pristup podacima niza ključ je indeks niza. Ime niza i indeks u uglastim zagradama dobijaju vrednost sadržaja te lokacije u nizu.

Da bi se obrisao sadržaj koji je zauzimao prostor na određenoj lokaciji niza, dovoljno je postaviti ga na null ili prazan znakovni niz. Međutim, stvarno brisanje elementa niza moguće je operatorom *delete*.

```
mojNiz.length;          // rezultat je 3
delete mojNiz[1];
mojNiz.length = 3;      // rezultat je ponovo 3
mojNiz[1];              // rezultat je undefined
```

Primetimo da operator *delete* briše dati element ali ne narušava strukturu niza! Dakle, sadržaj koji se nalazio na datoj poziciji više ne postoji, ali uređenost niza je sačuvana i nema potrebe za premeštanjem elemenata i menjanjem njihovih indeksa da bi se očuvala uređenost niza. Znači, mesto sa datim indeksom postoji u nizu, ali je prazno.

Poglavlje 4

Objekti *JavaScript*-a

Skriptabilni *browser* ima dosta posla oko pravljenja programskih objekata, koji generalno predstavljaju vidljive elemente *HTML* stranice u prozoru *browser*-a. U vidljive objekte spadaju slike i elementi obrazaca. Međutim, postoje objekti koji nisu tako očigledni, ali imaju smisla kada su uzmeu u obzir *HTML* oznake koje su korišćene da bi se napravio sadržaj stranice. Da bi se pomoglo skriptovima da kontrolišu ove objekte, ali i autorima da pronađu neki sistem u mnoštvu objekata na stranici, definisan je model objektata dokumenta (*Document Object Model, DOM*). Model je nešto kao prototip ili plan organizacije objekata na stranici.

Svaki objekat je na neki način jedinstven. Tri najvažnije odlike objekta definišu šta je on: kako izgleda, kako se ponaša i kako ga skript kontroliše. To su svojstva (*properties*), metodi i procedure za obradu događaja (*event handlers*).

Svaki fizički objekat ima skup karakteristika koji ga definiše: oblik, boju, težinu, i još mnogo drugih koje ga razlikuju od ostalih objekata. Svaka od ovih osobina naziva se svojstvo, i svako od svojstava ima neku vrstu vrednosti koja mu je dodeljena. Svojstva *HTML* objekta najčešće se podešavaju atributima *HTML* oznaka. Prisustvo *JavaScript*-a često dodaje opcione atribute čija se inicijalna vrednost može modifikovati kada se dokument učita.

Ako svojstvo liči na opisni pridev za objekat (*name, value*, ...), onda je metod glagol. Metod je sve u vezi sa akcijom objekta. On ili nešto radi objektu, ili nešto sa objektom što ima uticaja na ostale delove skripta ili dokumenta. Oni su neka vrsta komandi, ali takvih čija su ponašanja povezana sa određenim objektima. Jedan objekat može imati neodređen broj metoda koji su s njim povezani uključujući i ni jedan. Imena metoda se završavaju parom zagrada a da bi se metod pokrenuo, naredba *JavaScript*-a mora sadržati njegovu referncu, preko njegovog objekta.

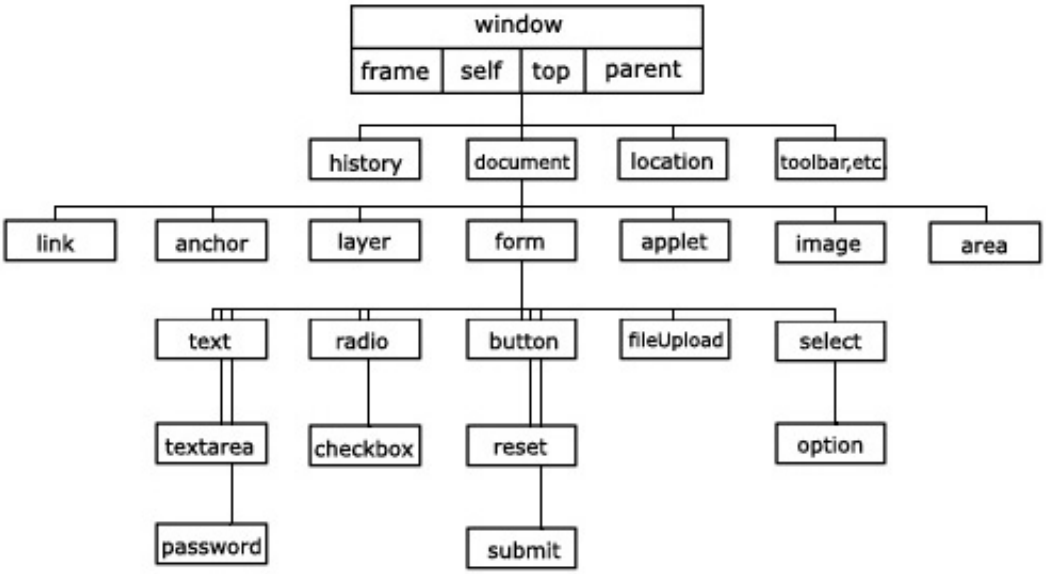
document.orderForm.submit()

Ponekad metod zahteva da se s njim pošalju i dodatne informacije tj. parametri ili argumenti, koji se tada navode unutar zagrada.

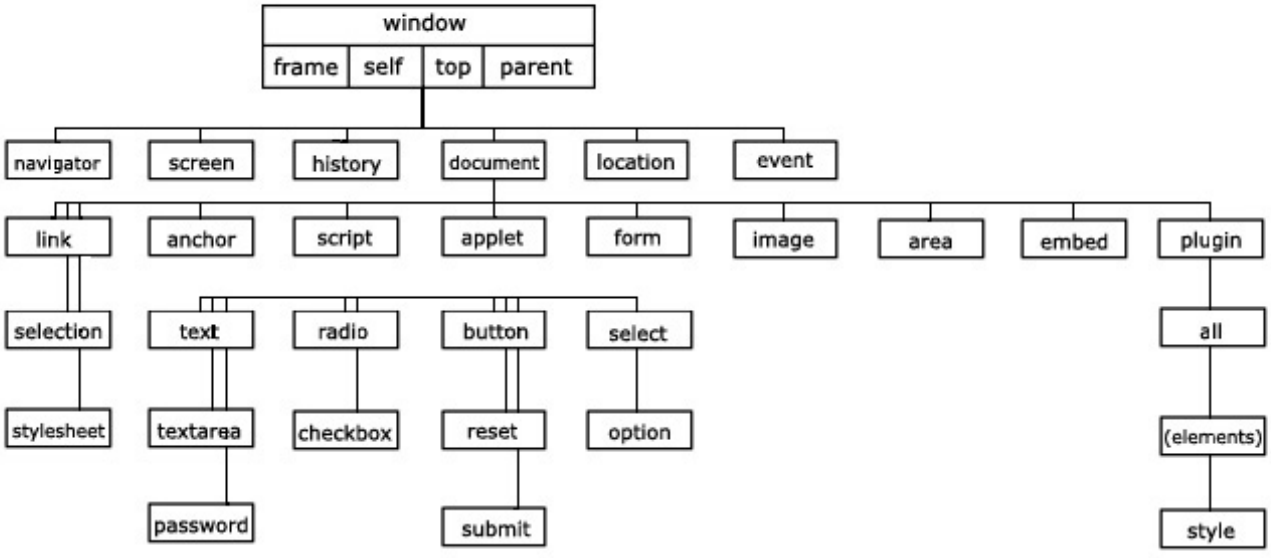
Poslednja karakteristika objekta *JavaScript*-a je procedura za obradu događaja. Događaji (*events*) su akcije koje se odvijaju u dokumentu, obično kao posledica akcije korisnika kao što su pritisak mišem na dugme ili upisivanje teksta u polje za unos. Gotovo svaki objekat *JavaScript*-a u

dokumentu prima događaj neke vrste. Da li će objekat uraditi bilo šta drugo kao odgovor na događaj, zavisi od atributa koji se unosi u definiciju *HTML* objekta. Atribut se sastoji od imena objekta i znaka jednakosti, kao i uputstva šta da se radi kada određeni događaj otpočne.

Slika 1: Hijerarhija objekata *Navigator*-a



Slika 2: Hijerarhija objekata *Internet Explorer*-a



4.1 Objekat *window*

Objekat prozor tj. *window*, nalazi se na vrhu hijerarhije objekata *JavaScript*-a i predstavlja mesto za sadržaj *HTML* dokumenta u prozoru *browser*-a. Kako se sve akcije dokumenta odigravaju unutar prozora, prozor je na vrhu hijerarhije objekata *JavaScript*-a. Njegove fizičke granice sadrže *JavaScript*. Pored onog dela prozora gde se smešta sadržaj dokumenta, sfera uticaja prozora uključuje dimenzije prozora i sve ostale potrebne sastavne delove koji okružuju oblast sa sadržajem, kao što su *scrollbars, toolbars, menu, statusbar*, i dr.

Svojstva i metodi objekta prozora mogu se u skriptu referencirati na nekoliko načina. Najčešći način je da se u referencu uključi objekat prozora:

window.imeSvojstva
window.imeMetoda(parametri)

Kada pri referenciranju skript pokazuje na prozor u kome je smešten dokument, za objekat prozora postoji i sinonim- *self*. Nije poželjno da se *self* kombinuje u referencama u skriptu za prozor s jednim okvirom; kada postoji više okvira *self* svakako olakšava čitanje i ispravku koda.

self.imeSvojstva
self.imeMetoda(parametri)

Nije uvek potrebno praviti novi objekat window koristeći kod *JavaScript*-a.

Skript ne pravi glavni prozor *browser*-a, to čini korisnik aktiviranjem *browser*-a ili otvaranjem URL-a ili datoteke iz menija *browser*-a. Taj prozor je automatski formiran valjan objekat window čak iako je prazan. Ali skript može generisati bilo koji broj potprozora ako je otvoren glavni prozor (i ako sadrži dokument čiji će skript otvoriti potprozor).

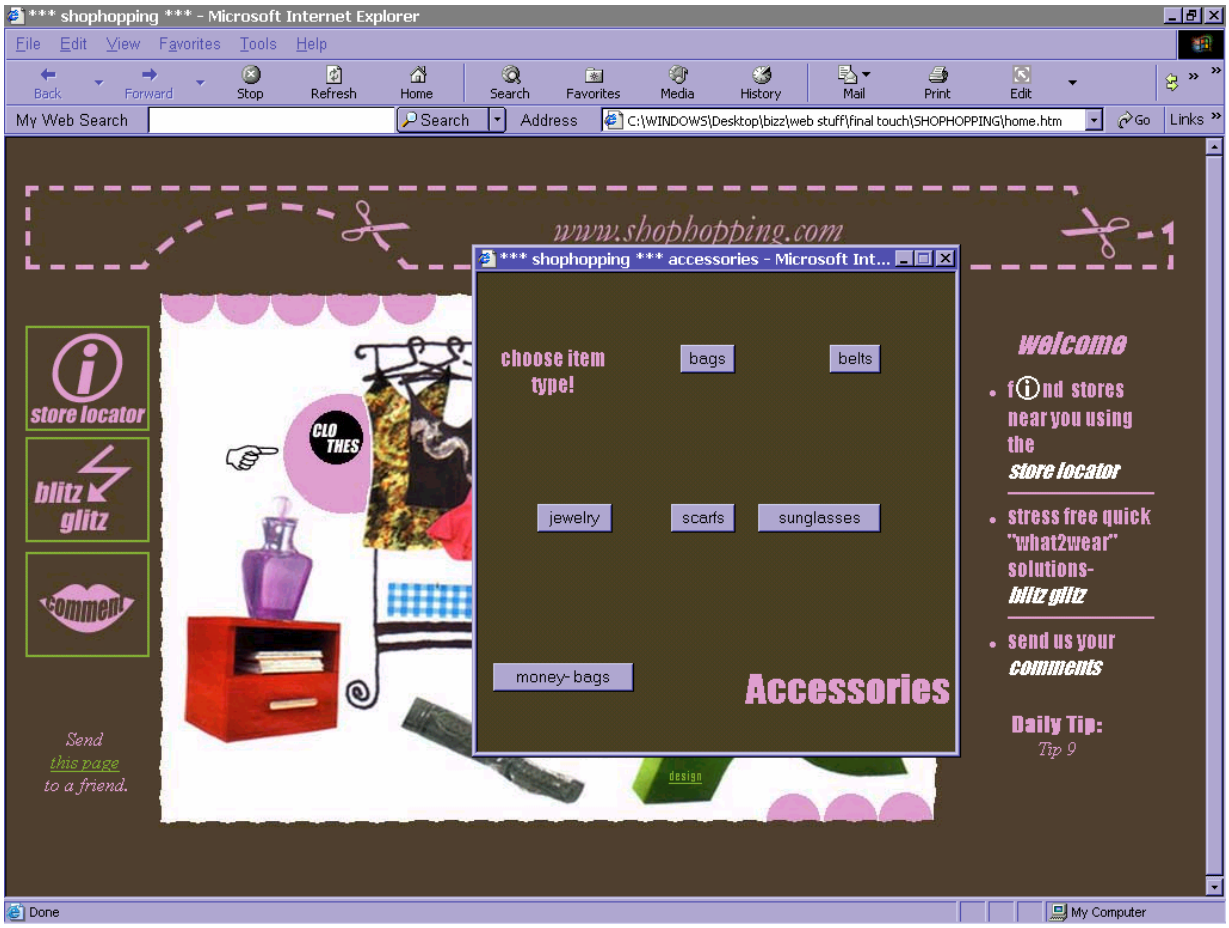
Metod koji pravi novi prozor je *window.open()* i sadrži više parametara koji definišu karakteristike prozora.

window. open("URL", "imeProzora", "osobineProzora")

osobineProzora je znakovni niz koji se sastoji od liste atributa, razdvojenih zarezima. Atributi koji se mogu kontrolisati u novonapravljenim prozorima su: *toolbar, location, directories, status, menubar, scrollbar, resizable, copzhistory, width, height.*

```
<html><head><title>otvaranje prozora</title>
<script>
function openAccessories{
        window.open ("accessories.html", "accessories", width= 100,
height= 100, scrollbars= no,    resizable= no, menubar= yes,
location= yes ")
}
</script>
</head>
<body>
<form>
<input type="button" value="accessories" onclick= "openAccessories()">
</form>
</body>
</html>
```

Na slici dole, vidi se novootvoreni manji prozor kog je "definisao" i otvorio gornji skript.



Čim se napravi drugi prozor u okruženju *Web* stranice, vazno je obratiti pažnju na upravljenje prozorskim slojevima. Korisnik vrlo lako može izgubiti manji prozor iza većeg. Metod *window. focus()* stavlja navedeni prozor ispred

svih prozora. Nasuprot *focus()* je *blur()* koji odabrani prozor gura u pozadinu svih otvorenih prozora.

Statusbar na dnu prozora *browser*-a redovno prikazuje URL veze kada na nju postavite kursor miša. I druge poruke se pojavljuju na *statusbar*-u u vreme kad se dokument učitava. *JavaScript* se može zgodno iskoristiti za prikazivanje vlastite poruke na *statusbar*-u u željenom trenutku. Svojstvu *window.status* se može pridružiti neki drugi tekst. Da bi se promenio tekst u vezi, akcija se pokreće procedurom za obradu događaja *onMouseOver*= povezanog objekta. Osobitost korišćenja ove naredbe za podešavanje *statusbar*-a je da dodatna naredba *return true* mora da bude deo procedure za obradu događaja. Radi jednostavnijeg podešavanja svojstva *window.status*, naredbe skripta se najčešće pišu kao skriptovi u redu u sklopu definicije procedure za obradu događaja. *HTML* oznaci a jednostavno se dodaju naredbe skripta:

```
<a href="http://home.mySite.com"
 onMouseOver=
"window.status='posetite homepage mog website- a' ; true">
mySite
</a>
```

Još jedan zanimljiv i koristan metod objekta window je *setInterval. setInterval()* se koristi kada je potrebno da skript pozove funkciju ili više puta izvrši neki izraz sa fiksnim kašnjenjem između poziva te funkcije i izraza. Tipične aplikacije su animacije, pomeranjem objekta po stranici kontrolisanom brzinom (pogledati primer slajd u odeljku Objekat *image*)

Objekat prozora reaguje na nekoliko događaja koje prouzrokuje sistem i korisnik, ali verovatno najčešće korišćen je događaj koji se aktivira odmah čim se cela stranica učita- *onload*=. Prednost korišćenja ovog događaja je što pruža sigurnost o tome da su svi objekti u modelu jer ako se dozvoli pristup elementu objekta dok objekta još nije učitan, može se desiti greška u skriptu. Procedure za obradu događaja objekta prozora smeštaju se unutar oznake *body*.

4.1.1 *Popup* prozori

JavaScript omogućava kreiranje tri "iskakajuća" prozora tj. *popup box*-a:

1. *alert box* (upozorenje)
2. *confirm box*(dijalog za potvrdu)
3. *prompt box* (odzivni okvir za dijalog).

Metod *alert()* generiše okvir za dijalog koji prikazuje svaki tekst koji se preda kao parametar, a dugme OK omogućava korisniku da ukloni upozorenje. Browser ubacuje reči koje ukazuju na to da je *alert box*

upozorenje *JavaScript*-a- *JavaScript Alert*, i ovaj tekst se ne može menjati skriptom; samo se sadržaj druge poruke može menjati.

sintaksa: *alert("sometext")*

Evo krajnje efikasnog primera primene *alert box*-a. U pitanju je skup radio dugmadi čiji atributi value sadrže imena boja. For petljom skript nalazi odabrano dugme i upozorava korisnika.

```
<html>
<head>
<title>alert box boje</title>
<script language= "javascript">
function kojaBoja() {
	var form= document.forms[0]
	for (var i=0; i< form.boje.length; i++) {
		if (form.boje[i].checked) {
			break
		}
	}
	alert('odabrali ste' + form.boje[i].value + '!')
}
</script>
</head>
<body>
<form>
<input type="radio" name="boje" value= "crvena" onClick="kojaBoja() ">
<input type="radio" name="boje" value= "plava" onClick="kojaBoja() ">
<input type="radio" name="boje" value= "zuta" onClick="kojaBoja() ">
</form>
</body>
</html>
```

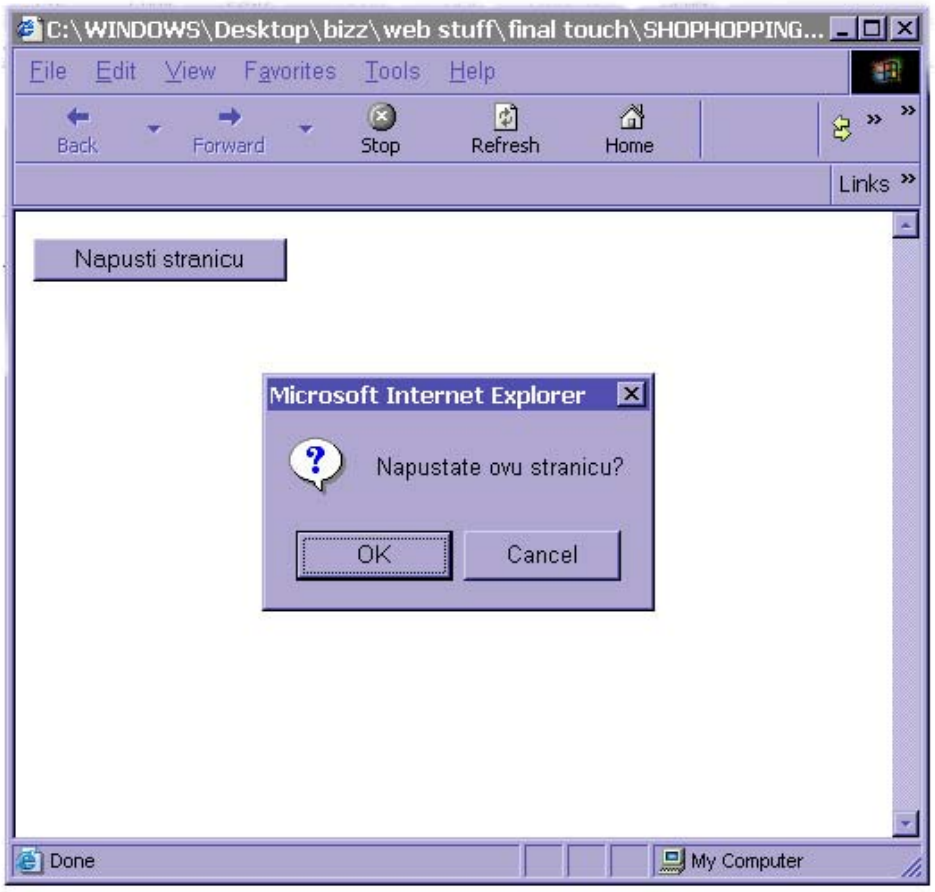
Vidimo da, ako je svojstvo *checked true* tj. data opcija je odabrana, *break* naredba uslovljava izlazak iz petlje, i indeks odabrane opcije pronađen pomoću *for* petlje koristi se za ispis poruke *alert box*-a.

Drugi stil *popup box*-a prikazuje dva dugmeta: *Cancel* i *OK*, i naziva se *confirm box. window.confirm()* je jedan od metoda koji vraća vrednost: *true* ako korisnik pritisne OK i false ako pritisne Cancel. Ovaj okvir za dijalog može se koristiti kao podsticaj korisniku da donese odluku o tome kako da se nastavi skript. Izlazna vrednost može se koristiti i kao uslov u nekoj if konstrukciji.

sintaksa: *confirm("sometext")*

Sledeći primer prikazuje kako to korisnik preko confirm box-a može uticati na dalji "tok" skripta:

```
<html>
<head>
<script language = " javascript">
function potvrdi() {
        var odgovor = confirm("Napustate ovu stranicu?")
        if (odgovor){
                alert("Dovidjenja!")
                window.location = "http://www.im.ns.ac.yu/";
        }
        else{
                alert("Hvala sto ostajete!")
        }
}
</script>
</head>
<body>
<form>
<input type="button" onclick="potvrdi()" value="Napusti Stribog">
</form>
</body>
</html>
```

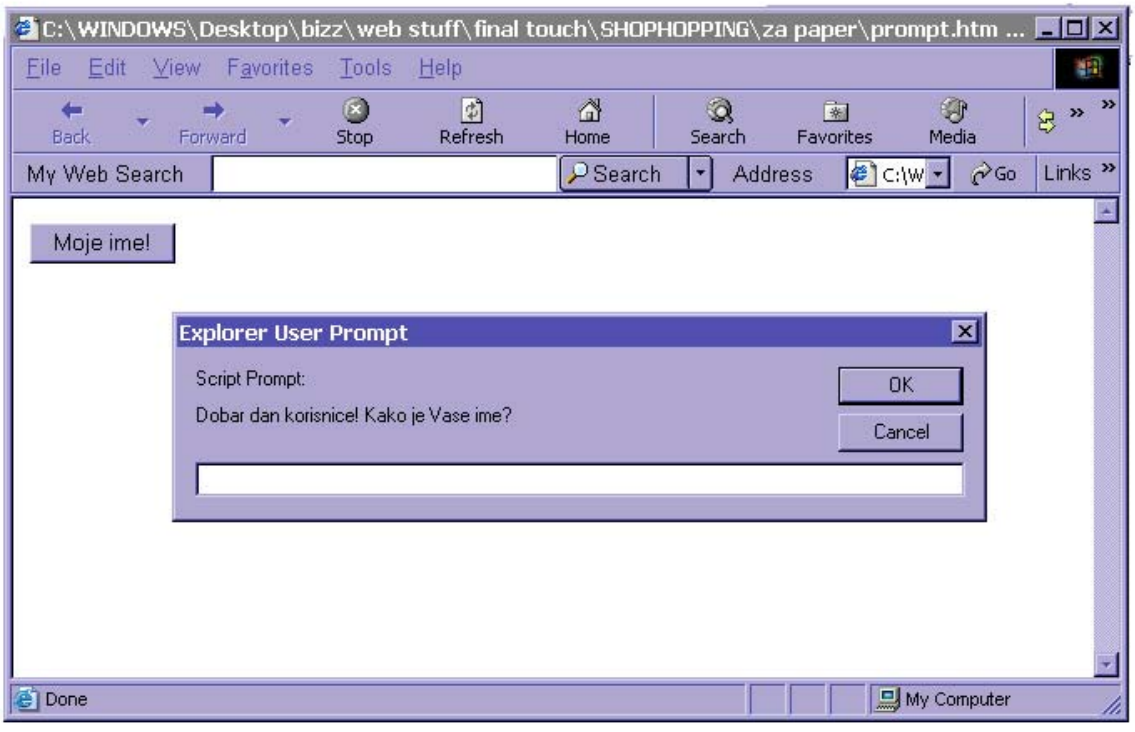


Dakle, ako korisnik potvrdi odlazak pritiskajući OK (promenljiva odgovor je true), skript ga šalje na homepage Instituta za matematiku (www.im.ns.ac.yu). Inače, iskače alert box koji zahvaljuje korisniku što ostaje.

Poslednji okvir za dijalog objekta prozora- *prompt box*, prikazuje zadatu poruku i obezbeđuje polje za unos teksta u koje korisnik unosi odgovor. Dva dugmeta, Cancel i OK, omogućavaju korisniku da ukloni okvir za dijalog sa dva efekta: otkazivanje cele operacije ili prihvatanje unetog. Metod window.prompt() ima dva parametra. Prvi je poruka koja se prikazuje korisniku. U polju za unos može se priložiti podrazumevani odgovor kao drugi parametar metoda. Ako to nije potrebno, navodi se prazan niz- "". Pritiskom na dugme Cancel, metod vraća *null* (u uslovu neke if konstrukcije smatra se za false); pritiskom na OK vraća se vrednost unetog znakovnog niza.

sintaksa: *prompt("sometext", "defaultvalue")*

```
<head>
<script type="text/javascript">
function prompter() {
var reply =
        prompt("Dobar dan korisnice! Kako je Vase ime?", "")
alert (reply)
}
</script>
</head>
<body>
<input type="button" onclick="prompter()" value="Moje ime!">
</body>
```

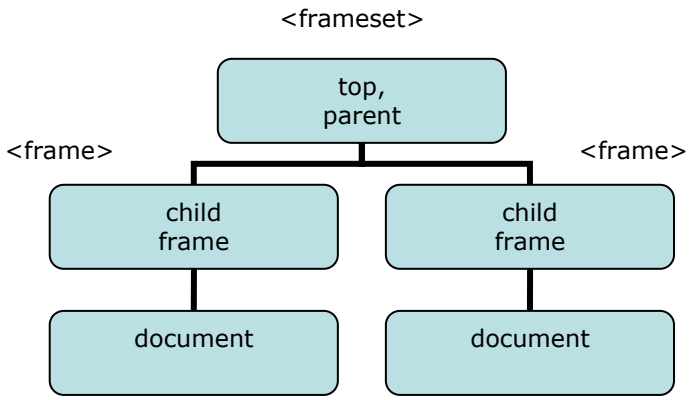


Skript u gornjem primeru koristi prompt box da bi od korisnika dobio informaciju o njegovom imenu, a zatim ga prikazuje u alert box-u.

4.2 Rad sa okvirima- objekat *frame*

Dobra strana *JavaScript*-a sa stanovišta korisnika je ta što omogućava da neke akcije u jednom prozoru ili okviru utiču na ono što se dešava u drugim okvirima i prozorima.

Učitavanjem jednog običnog *HTML* dokumenta u *browser* pravi se model koji kreće s jednim objektom prozora i dokumentom koji on sadrži. Najviši rang u hijerarhiji modela je window i reference počinju sa window ili self (ili *document*, pošto se tekući prozor podrazumeva). Čim se dokument koji koristi okvire učita u *browser*, tačna struktura tog modela zavisi od okvira koji su definisani u tom dokumentu. Model brine samo o strukturi. Dokument koji postavlja okvire učitava se u prozor parent. Svaki od okvira definisan u parent-u je child.



Roditeljski prozor za postavljanje okvira nema ni jedan objekat *document*. Kako *HTML* datoteka za postavljanje okvira nema oznaku *body* ili neki drugi centralni element dokumenta, nijedan objekat dokumenta u ovom delu modela se ne učitava u *browser*. Svaki od child okvira sadrži objekat dokumenta čiji sadržaj se vidi u prozoru *browser*-a; i struktura je takva da je svaki dokument potpuno nezavisan od drugog.

Ako skript u svom prozoru ima pristup nekom objektu, funkciji ili globalnoj promenljivoj, njemu se može pristupiti pomoću drugog skripta iz drugog okvira u hijerarhiji, pod uslovom da su oba dokumenta na istom *Web* serveru. Referenca u skriptu će morati da koristi jedan od tri moguća izbora u hijerarhiji parent-child, a to su: parent- child, child- parent i child- child.

Najređi smer korišćenja referenci je parent- child. Sa stanovišta parent-a, on sadrži dva ili više okvira, što znači da su okviri takođe uskladišteni u modelu kao niz objekata okvira. Okvir se može obeležiti indeksom u nizu ili

imenom unutar oznake frame. Dakle, reference od roditelja- parent, do potomka- child, slede jedan od sledeća dva modela:

frames[n].objFuncVarName
imeFrame.objFuncVarName

gde *objFuncVarName* predstavlja objekat, funkciju ili globalnu promenljivu kojoj se pristupa. Vrednosti indeksa za okvire zasnivaju se na redosledu po kome se pojavljuju oznake frame u dokumentu za postavljanje okvira.

Ne retko se skript smešta u parent, u zaglavlje, koji ostali child okviri koriste kao biblioteku skriptova. Učitavanjem u osnovu okvira, ti skriptovi se učitavaju samo jednom. Sa stanovišta child, sledeći nivo u hijerarhiji je parent, pa je referenca do objekta na tom nivou:

parent. objFuncVarName

Kada se prozor roditelja nalazi na samom vrhu hijerarhije objekata koji su trenutno učitani, moguće je obratiti mu se kao vrhovnom prozoru:

top.objFuncVarName

Kada dođe do toga da komuniciraju dva child okvira, referenca mora da nađe izlaz iz tekućeg okruženja ka tački koja je zajednička za oba child okvira, uglavnom parent. Kada se referenca nađe na nivou parent, ostatak reference se izvodi kao da se kreće od roditelja.

parent.frames[n].objFuncVarName
parent.imeFrame.objFuncVarName

Svojstva, metodi i procedure za obradu događaja objekta frame isti su kao i oni objekta window. Značajna razlika između prozora i okvira je u procedurama *onLoad*= i *onUnload=.* Kako svaki dokument koji se učitava u okvir može imati vlastite procedure *onLoad*= u svojoj oznaci *body*, okvir koji sadrži taj dokument prima učitani sadržaj kada i sam dokument završi učitavanje. Ali postavljač okvira koji pravi okvir prima zaseban događaj učitavanja kada su svi okviri završili učitavanje svojih dokumenata. Taj događaj je uhvaćen u proceduri *onLoad*= u oznaci frameset.

Sledeći *HTML* kod deli stranicu na dva okvira, vertikalno:

```
<html>
<head>
<title>okviri</title>
</head>
	<frameset cols="20%, 80%">
		<frame name="menu" src="menu.html" noresize=true
frameborder=0>
		<frame name="main" src="main.html" noresize=true
frameborder=0>
	</frameset>
</html>
```

Okvir "*menu*" otvara dokument menu.html u kome se nalazi *HTML* kod koji sadrži i skript za padajući meni, a okvir "*main*" otvara početnu stranicu *main.html*. Komunikacija između ova dva okvira- da se odabrana opcija sa padajućeg menija otvara u okviru "*main*"- uspostavlja se pomoću skripta za padajući meni. Zadnji red *JavaScript* koda postavlja svojstvo location okvira *main* na vrednost odabrane opcije. (zajednička tačka dvaju okvira je parent). Pogledajmo *JavaScript* kod.

```
<html>
<head>
<script>
	// funkcija za padajuci dropdown meni
	function DropDownMenu(entered){
	    with (entered){
	       ref=options[selectedIndex].value;
	       splitcharacter=ref.lastIndexOf("&");
	       if (splitcharacter!=-1) {
	          loc=ref.substring(0,splitcharacter);
	          target=ref.substring(splitcharacter+1,1000).toLowerCase();
	       }
	       else {
	                loc=ref;
	                target="_self";
	       }
	       lowloc=loc.toLowerCase();
	       if (lowloc=="false") {return;}
	       parent.main.location=loc;
	    }
	}
</script>
</head>
```

```
<body bgcolor=#4d4029>
<p align=center>
      <font size="2" face="Impact" color="#89ae3b"> menu </font><br>

      // ponudjene opcije dropdown menija
      <select onChange="DropDownMenu(this)">
            <option value="false"> occasion </option>
            <option value="casual.htm"> casual </option>
            <option value="formal.htm"> formal </option>
            <option value="night.htm"> night </option>
      </select>
</p>
</body>
</html>
```

4.3 Objekti location i history

Svaka *Web* stranica koja je prikazana u *browser*-u, naziva se lokacija i ima svoj URL- Uniform Resource Locator[*]. Browser ovu informaciju smešta u objekat location, a i dok se krećemo *Web*-om, čuva URL-ove posećenih stranica u objektu history čiji sadržaj je moguće pregledati iz menija *browser*-a. Odatle je moguće i vratiti se na neku prethodno posećenu stranicu.

Ponekad objekat u hijerarhiji predstavlja nešto što nema fizičku reprezentaciju, kao što je prozor ili dugme. To je slučaj sa objektom lokacije. Ovaj objekat predstavlja URL koji je učitan u dokument. Za kretanje skriptova kroz druge stranice , u tekućem prozoru ili u drugom okviru, osnovno je zadati svojstvo *location.href*. Do stranice na vlastitoj *Web* lokaciji moguće je doći navođenjem relativnog
URL-a (dakle, www.mySite.com), umesto kompletnog URL-a sa protokolom (http://www.mySite.com). Ako se stranica koja treba da se učita nalazi u drugom prozoru ili okviru, referenca prozora mora biti deo naredbe. Dakle, ako skript otvara novi prozor i dodeljuje njegovu referencu promenljivoj *newWindow*, naredba koja učitava stranicu u prozor biće:

newWindow.location= "http://www.mySite.com"

Znak tarabe- #, je URL konvencija koja usmerava *browser* na sidro koje se nalazi u dokumentu. Bilo koje ime da se dodeli sidru (pomoću oznaka ...) postaje deo URL-a posle tarabe. Kao što se postavljanjem svojstva *window.location* može otići do bilo kog URL-a, tako se može otići do drugog znaka tarabe u istom dokumentu postavljanjem samo svojstva *hash* za tu lokaciju, ali bez tarabe. Funkcija u skriptu izgledala bi ovako:

[*] metod za imenovanje datoteka i lokacija na internetu

```
function sledecaTaraba(gde) {
        window. location.hash = gde
}
```

Dalje u telu *HTML* stranice:

```
<form>
        <input type= button value next onclick= "sledecaTaraba(tar1)">
</form>
```

pa prethodno dugme vodi do:

```
<a name= "tar1">  prva taraba   </a>
```

Najčešće u skriptovima pozivano od svih svojstava objekta location je *hre*f- referenca hiperteksta. Svojstvo *location.href* obezbeđuje znakovni niz kompletnog URL-a za navedeni objekat window. U zavisnosti od *browser*-a, vrednosti svojstva *href* mogu biti i kodirana *ASCII* znakovima koji nisu alfanumerički. Najčešće kodirani znak je znak razmaka, %20. Najbezbednije je sve tako kodirane znakovne nizove propustiti kroz internu funkciju *JavaScript*-a *unescape()*; (za kodiranje je dostupna funkcija *escape(),* inverzna funkciji *unescape()*).

Mnogi elementi obrazaca zadržavaju svoje stanje na ekranu i posle pritiska na dugme *Reload* u *Navigator*-u tj. *Refresh* u *Internet Explorer*-u. Ovakvo ponovno učitavanje može se nazvati meko. Tvrdo ponovno učitavanje (*hard reload*) inicira metod *location. reload()*. Briše sve parametre dokumenta koje je *browser* možda sačuvao i ponovo otvara dokument. U skriptu se mogu koristiti obe vrste učitavanja: meko pozivanjem metoda *history.go()* a tvrdo pozivanjem metoda *location.reload().*

Objekat history je još jedan objekat koji nema fizičku reprezentaciju na stranici. Svaki prozor čuva spisak stranica koje je *browser*, odnosno korisnik nedavno posetio. Metodi objekta history omogućavaju kretanje kroz listu posećenih stranica, napred i nazad u odnosu na tekuću stranicu.

Jedna od primena objekta history i njegovih metoda *back()* ili *go()* jeste da obezbedi ekvivalent dugmetu *Back* u *HTML* dokumentu. To dugme aktivira skript koji proverava svaku stavku u istorijskoj listi , a zatim se vraća jednu stranicu unazad.

Pritiskom na dugmad *Back* i *Forward browser* učitava stranice posećene pre tj. posle tekuće. Da bi mogao da zna koje stranice da učita, pamti listu posećenih URL-ova. Zbog mogućnosti ugrožavanja privatnosti korisnika, history stabla su ograničena na stranice sa potpisanim skriptovima, čiji su posetiocii dozvolili pristup osetljivim podacima iz *browser*-a. Ne postoji ni svojstvo ni metod koji direktno otkrivaju vrednost trenutno učitanog URL-a, ali pomoću skripta moguće je dobiti tu informaciju poredeći vrednosti

svojstava *current*, *next* i *previous*, sa kompletnom listom. Ne preporučuje se korišćenje ovih mogućnosti osim ako korisnik nije prethodno upozoren.

Metod *history.go()* prihvata samo elemente koji već postoje u istorijskoj listi, tako da se ne može koristiti umesto *window.location*. Dakle, tekući URL ekvivalentan je *history.go(0)* (metod koji ponovo učitava prozor). Pozitivan ceo broj označava koliko stavki na listi *History* se preskače unapred. Pa je *history.go(-1)* isto što i *history.back()*, a *history.go(1)* isto što i *history.forward().*

4.4 Objekat *document*

Document sadrži stvarni sadržaj stranice i on je sve ono što postoji u oblasti prozora *browser*-a ili okvira prozora (naravno, izuzev *toolbar*-a, *statusbar*-a i sl.).
Svojstva i metodi ovog objekta utiču na izgled i sadržaj dokumenta koji se nalazi u prozoru. Mnoga od tih svojstava uspostavljaju se pomoću atributa iz oznaka *body*, i mnoga druga svojstva su nizovi drugih objekata u dokumentu. Svojstvima i metodima objekta dokumenta pristupa se direktno:

window.document.imeSvojstva
window.document.imeMetoda(parametri)

Referenca window je opciona kada se pristupa objektu dokumenta koji sadrži skript.

Ne zadaju se sva svojstva vezana za objekat dokumenta kao atributi oznaka *body*. Ako se stranica naslovi unutar oznaka *title* u zaglavlju, na taj naslov utiče svojstvo *document.title*. Naslov dokumenta je uglavnom kozmetičko podešavanje koje se prikazuje u *titlebar*-u *browser*-a tj. naslovnoj liniji. I naravno skriptovi unutar oznaka script su deo dokumenta.

Drugi način da se napravi dokument je metodom *document.write()* i da se jednostavno ceo sadržaj ili deo *HTML* stranice saspe u jedan prozor ili okvir. Metod *document.write()* se koristi u dve vrste skripta: u trenutnim koji prave sadržaj stranice dok se učitava, i u odloženim, koji prave novi sadržaj u postojećem ili novom prozoru. Dva najbezbednija načina za upotrebu metoda *document.write()* i *document. writeln ()* (razlikuje se od write () po tome što dodaje još i novi red) su sledeća:

a) ugrađivanje skripta u *HTML* dokument da bi se napisao deo ili ceo sadržaj stranice
b) slanje *HTML* koda, ili u novi prozor ili uzaseban okvir u prozoru sa više okvira

U prvom slučaju se prepliću delovi skripta i *HTML*-a; skript se pokreće kada se dokument učita, upisujući *HTML* sadržaj. U drugom slučaju, skript prihvata unos korisnika u jednom okviru i određuje izgled i sadržaj koji je namenjen drugom okviru.

Sledeći skript postavlja se unutar oznaka *body*, na željenom mestu, i omogućava prikaz pokretnog teksta- poruka *mesg* kretaće se levo- desno. Vidimo da se skript ugrađuje u *HTML* kod.

```
<table width=169 height=459>
<tr>
<td height=38 align=center width="163">
<script>
mesg = "welcome";
text = ("<i><font size=5 face=impact
color=#e09fce>"+mesg+"</font></i>")
document.write("<MARQUEE BEHAVIOR=ALTERNATE
DIRECTION=RIGHT>" +text+"</MARQUEE>")
</script>
</td>
</tr>
</table>
```

Svojstva *document*-a *alinkColor, vlinkColor, bgColor, fgColor, linkColor* tiču se podešavanje boje u dokumentu i svih pet je moguće podesiti pomoću skripta. Mogućnost promene nekog od njih zavisi od čitača i operativnog sistema. Vrednosti svih svojstava boje mogu biti uobičajenog *HTML* heksadecimalnog troznaka (npr. #0000ff) ili bilo koje od *Netscape*-ovih boja. *Internet Explorer* takođe prepoznaje ove oznake za boje.

4.4.1 *Cookies*

Svakako nešto sa čime se srećemo relativno često na *Web*-u jesu kolačići- *cookies*. Mehanizam *cookies* omogućava sladištenje male količine informacija na računaru na dosta bezbedan način. Cookie se često koristi za skladištenje korisnikovog imena i/ili lozinke na *Web* prezentacijama. Postoji nekoliko vrsta *cookies*-a:

- *Name cookie* - prvi put kada neko poseti *Web* prezentaciju koja sadrži skript za ovaj *cookie*, posetilac mora da unese neko korisničko ime. To ime se čuva u *cookie*-ju i sledeći put kada dati korisnik pristupi toj lokaciji, dobiće pozdravnu poruku tipa: "Zdravo Jim!". Ime se izdvaja iz *cookie*-ja.
- *Password cookie* - slično name *cookie*-ju, ali ovde se unosi lozinka koja se takođe čuva u *cookie*-ju.
- *Date cookie* - datum prve posete datog korisnika *Web* prezentaciji stavlja se u *cookie*. Pri sledećoj poseti moguće je ispisati poruku

korisniku tipa: "Poslednji put ste posetili ovu prezentaciju u utorak, 25.03. 2007."

Kada se prvi put takve informacije unesu u obrazac koji se šalje CGI[†] programu, CGI program preporučuje *browser*-u da tu informaciju zapiše na hard disk računara. Međutim, *browser* ne otvara svima direktorijum hard diska. Mehanizam *cookies* obezbeđuje pristup do posebne tekstualne datoteke na hard disku, koja se nalazi na mestu koje zavisi od operativnog sistema. U Windiws-ovoj verziji *Navigator*a datoteka se zove *cookies*.txt i smeštena je u direktorijumu *Navigator*a. *Internet Explorer* koristi malo drugačiji sistem: svaki *cookie* se snima u sopstvenu datoteku u direktorijumu *Cookies* koji je smešten među sistemskim direktorijumima. U poljima svakog zapisa *cookie* nalazi se sledeće:

- domen servera koji je napravio *cookie*
- informacija o tome da li je potrebna zaštićena HTTP veza da bi se pristupilo *cookie*-ju
- putanja URL-a koji mogu pristupiti *cookie*-ju
- ime *cookie*-ja
- znakovni nz koje je povezan sa *cookie*-jem

Pristup *cookies*-ima iz *JavaScript*-a ograničen je na postavljanje vrednosti (sa opcionim parametrima) i čitanje podataka iz njih (bez parametara). Da bi se podaci sa *cookie*-ja upisali u za njih predviđenu datoteku, koristi se *JavaScript* operator dodele i svojstvo *document.cookie*. Sintaksa za dodeljivanje vrednosti kolačiću je:

```
document. cookie =" cookieName= cookieData
        ; expires= timeInGMTString
        ; path= pathName
        ; domain= domainName
        ; secure "
```

Svaki *cookie* mora imati ime i znakovni niz kao vrednost (čak i prazan niz). Par ime- vrednost mora biti jedan niz bez tačke i zareza, zareza ili znaka razmaka. Za razmak između reči poželjno je koristiti funkciju *escape()* koja kodira *ASCII* razmak kao %20 (a zatim unescape() za dekodiranje na običan razmak za kasnije kada *cookie* bude pozvan). Na primer, ako se kolačiću userNmae dodaje znakovni niz *Jim*, to izgleda ovako:

```
document. cookie = "userName=Jim"
```

Ako se dodaje još parametara i oni svi moraju biti uključeni u jedan znakovni niz i pridruženi svojstvu *document.cookie*. Dakle, podaci koji se pozivaju iz *cookie*-ja pomoću *JavaScript*-a smeštaju se u jedan znakovni niz, uključujući i ceo par ime- podaci.

[†] CGI program- Common Gateway Interface' unete podatke šalje u bazu podataka ili ih proslešuje drugom programu na serveru

Kolačići se ne mogu tretirati kao objekti, već se iz znakovnog niza mora izdvojiti željeni podatak iz para ime- podaci koristeći metod *substring()*. Prvi parametar metoda *substring* sadrži znak jednakosti, kojim se razdvaja ime od podataka.

Najbolje rešenje je napraviti funkciju opšte namene koja može da radi i samo sa jednim i sa više *cookies*-a.

```
function getCookie(c_name) {
        if (document.cookie.length>0) {
                c_start=document.cookie.indexOf(c_name + "=")
                if (c_start!=-1) {
                        c_start=c_start + c_name.length+1
                        c_end=document.cookie.indexOf(";",c_start)
                        if (c_end==-1) c_end=document.cookie.length
                                return
unescape(document.cookie.substring(c_start,c_end))
                }
        }
        return ""
}
```

Pozivanjem ove funkcije predaje se ime *cookie*-ja kao parametar. Funkcija raščlanjuje ceo znakovni niz *cookie*-ja odbacujući sve unose koji se ne slažu sve dok ne pronađe ime *cookie*-ja.

Cookies imaju i rok trajanja. Ako *cookie* treba da opstane i posle tekuće sesije, on ima i rok trajanja koji postavlja pisac *cookie*-ja. Datum roka trajanja mora biti predat kao znakovni niz koji označava vreme po Griniču (*GMT*).

Sledeća funkcija kao parametre ima ime *cookie*-ja, njegovu vrednost i broj dana preostalih do isteka roka *cookie*-ja. Prvo, broj dana se konvertuje u odgovarajući datum, zatim se dodaje broj dana posle kog ističe rok *cookie*-ja. Posle toga, pamti se ime, vrednost i datum isteka roka *cookie*-ja u *document*. *cookie* objektu.

```
function setCookie(c_name,value,expiredays){
        var exdate=new Date()
          exdate.setDate(exdate.getDate()+expiredays)
          document.cookie=c_name+ "=" +escape(value)+
                        ((expiredays==null) ? "" :
;expires="+exdate.toGMTString())
}
```

Konačno, funkcija koja ispisuje pozdravnu poruku nakon što prepozna korisnika po korisničkom imenu, ako je *cookie* postavljen. Ako *cookie* nije postavljen, od korisnika se traži da unese ime u *prompt box*.

```
function checkCookie() {
        username=getCookie('username')
        if (username!=null && username!="") {
                alert('Welcome again '+username+'!')}
        else {
                username=prompt('Please enter your name:',"")
                if (username!=null && username!="") {
                        setCookie('username',username,365)
        }}}
```

Ceo skript zajedno izgleda ovako:

```
<html>
<head>
<title>cookie</title>
<script language="javascript">
function setCookie(cname, value, expdays){
      var exdate= new Date();
      exdate.setDate(exdate.getDate() + expdays)
      document.cookie = cname+ "=" +escape(value)+ ((expdays = null)?"
":"; expires="+exdate.toGMTString())
}
function getCookie(cname){
      if (document.cookie.length>0){
            cstart=document.cookie.indexOf(cname+"=");
            if (cstart != -1){
                  cstart= cstart+cname.length+1;
                  cend= document.cookie.indexOf(";", cstart);
                  if (cend == -1){
                        cend= document.cookie.length;
                  }
                  return unescape(document.cookie.substring(cstart, cend));
            }
      }
      return "";
}
```

```
function checkCookie(){
	username= getCookie('username');
	if (username != null && username != ""){
		alert('welcome to shophopping '+username+'!');
	}
	else{
		username= prompt('WELCOME NEW VISITOR! Please enter your name:',"");
		if (username != null && username != ""){
			setCookie('username', username, 365);
		}
	}
}
</script>
</head>
<body bgcolor=#4d4029 onload= "checkCookie()">
</body>
```

Što se tiče *path* i *domain* parametara, najbolji izbor je izbor podrazumevane putanje tj. tekućeg direktorijuma za *cookies* na strani klijenta; i izostavljanje parametra *domain* ako se ne očekuje da se dokument ponovi na drugom serveru u tom domenu. *Navigator* automatski dodaje domen tekućeg dokumenta prilikom unosa *cookie*-ja.

Ako se ne navede parametar *secure* kada se snima *cookie*, omogućen je pristup podacima iz njega svakom domenu ili *CGI* programu sa date lokacije, u kome se slažu svojstva *domain* i *path*. Pri skriptovanju *cookie*-ja na strani klijenta treba izostaviti ovaj parametar pri snimanju kolačića.

4.5 Objekti *link* i *anchor*

Od svih objekata dokumenta u *JavaScript*-u, objekat link je taj koji povezuje. On po hijerarhiji pripada objektu *document*. Kada se dokument učita, *browser* pravi i čuva interni spisak svih veza definisanih u objektu, pa se referenciranje linkova vrši preko njegovog indeksa u nizu:

document.links[n]. imeSvojstva

Veza za *JavaScript* definiše se isto kaoi i za *HTML*, uz dodatak šest događaja koji se odnose na miša. U okruženju sa više okvira ili prozora važno je navesti atribut *target* sa imenom okvira ili prozora u kome će se pojaviti sadržaj. Postoje i prečice za reference prozora*: _top, _parent, _self, _blank*.

Veze se obično ne skriptuju ali postoji jedna važna komponenta ovih objekata u *JavaScript*-u. Kada je potrebno da se klikom miša na vezu izvrši skript, umesto da se ide na drugi URL, moguće je preusmeriti atribut *href* da

pozove skript funkciju. Ova tehnika uključuje pseudo-URL koji se zove *javascript:URL*. Sintaksa za ovu konstrukciju je sledeća:

* ... *

Osnovno svojstvo objekta link je *target*. Ova vrednost predstavlja ime prozora dodeljeno atributu target u definiciji veze. Svojstvu *target* obraća se preko indeksirane veze:

document.links[n]. target

Ponekad se može javiti potreba da se pre odlaska do određene veze tj. navigacije, izvrši skript. Ovo se postiže postavljanjem događaja *onClick*= i/ili *onDblClick*= u definiciju veze. Bilo koja funkcija il naredba koja se poziva kada se desi neki od ova dva događaja izvršava se pre navigacije. Događaj *onClick*= postoji u svim *browser*-ima i svim verzijama, dok događaj *onDblClick*= postoji samo u *Navigator*-u 4 i *Internet Explorer*-u 4.

Ako se u oznaci veze ne navede atribut *href*, umesto objekta *link* definiše se objekat *anchor* tj. sidro. Kao i za veze, *browser* čuva listu za sva sidra u dokumentu. Objektima *anchor* obraća se preko njihovih indeksa u svojstvu *document*.
anchors[index]

Objekat *link* pretvara se u objekat anchor dodavanjem atributa *name* u definiciju veze. Svojstvo *name* objekta anchor je znakovni niz dodeljen atributu *name* u definiciji sidra ili veze; može samo da se čita.

4.6 Objekti *image*, *area* i *layer*

Jedna od karaketristika generacije *browser*-a *Navigator*a 4 i *Internet Explorer*-a 4 je dinamički *HTML* ili D*HTML*- mogućnost da se u hodu promeni sadržaj kao odgovor na interakciju korisnika. Objekti *JavaScript*-a koji doprinose razvoju interaktivnosti *Web* stranica sa velikim vizuelnim efektom su *image, area* i *layer*.

4.6.1 *Image*

Objekat slike je jedan od objekata koji se sadrže u dokumentu. Ovaj objekat, međutim, nije dostupan svim skriptabilnim *browser*-ima. Moguće je koristiti ga prilikom skriptovanja u verziji *Navigator*-a 3 i novijim, ili u *Internet Explorer*-u 4 i novijim verzijama.

Kako dokument može imati više od jedne slike, reference objekata slika uskladištene su kao niz u modelu objekta. Prema tome, slika se može povezati pomoću indeksa niza ili pomoću imena:

document. images[n]
document. imeSlike

Svakom od atributa oznake *img* može se pristupiti kao svojstvu objekta slike. Ako želimo od slike da napravito element koji reaguje na pritisak tasterom miša, okružujemo ga vezom ili joj pridružujemo slikovnu mapu. Kombinacijom veze i slike pravi se dugme slike koje reaguje na pritisak tasterom miša.

Izlaskom *Navigator*-a 3 slika je postala objekat, i skript je mogao da zameni sliku u pravougaonom okviru u kome je već postojala slika Ovo je bio jedan od prvih primera dinamičke izmene sadržaja stranice, kada je stranica već bila učitana. Skript je krajnje jednostavan i zahteva da se svojstvu *src* slike pridruži novi URL. Dimenzije slike- *width* i *height*- definišu se u oznaci *img* i ne mogu se menjati kada se stranica učita, na šta treba obratit pažnju da prikaz slike ne bude deformisan.

Da bi se slike preuzele sa *Web* servera, obično je potrebno nekoliko sekundi, i ako se *Web* stranica menja kao odgovor na akcije korisnika, poželjno je da taj odgovor bude što je moguće brži. Rešenje je prethodno keširanje slika. Dakle, slika se prvo učita u keš *Web* čitača za slike kada se stranica prvobitno učitava. Što je mnogo "bezazlenije" čekanje za korisnika, od čekanja na samu sliku. Prethodno keširanje zahteva pravljenje objekta slike u memoriji. Objekti koji su samo u memoriji prave se pomoću skripta i ne vide se na stranici. Njihovo prisustvo u dokumentu primorava *browser* da zajedno sa stranicom učita i slike. Konstruktorska konstrukcija *JavaScript*-a za memorijski tip objekata slike je:

var mojaSlika = new Image(width, height)

Dimezije definisane ovde treba da se slažu sa dimenzijama definisanim u oznaci *img*. Kada objekat slike postoji u memoriji, svojstvu *src* tog objekta moguće je dodeliti ime datoteke ili URL:

mojaSlika.src= "slika. jpg"

Kada *JavaScript* naiđe na naredbu koja svojstvu *src* dodeljuje URL, naređuje *browser*-u da počne sa učitavanjem date slike. Dok se procedura *onLoad*= ne aktivira, sve slike generisane ovako čuvaju se u kešu za slike. Tada se svojstvo keširanih slika *src* ili aktuelni URL slike, može pridružiti svojstvu src slike dokumenta koje je definisano oznakom *img*:

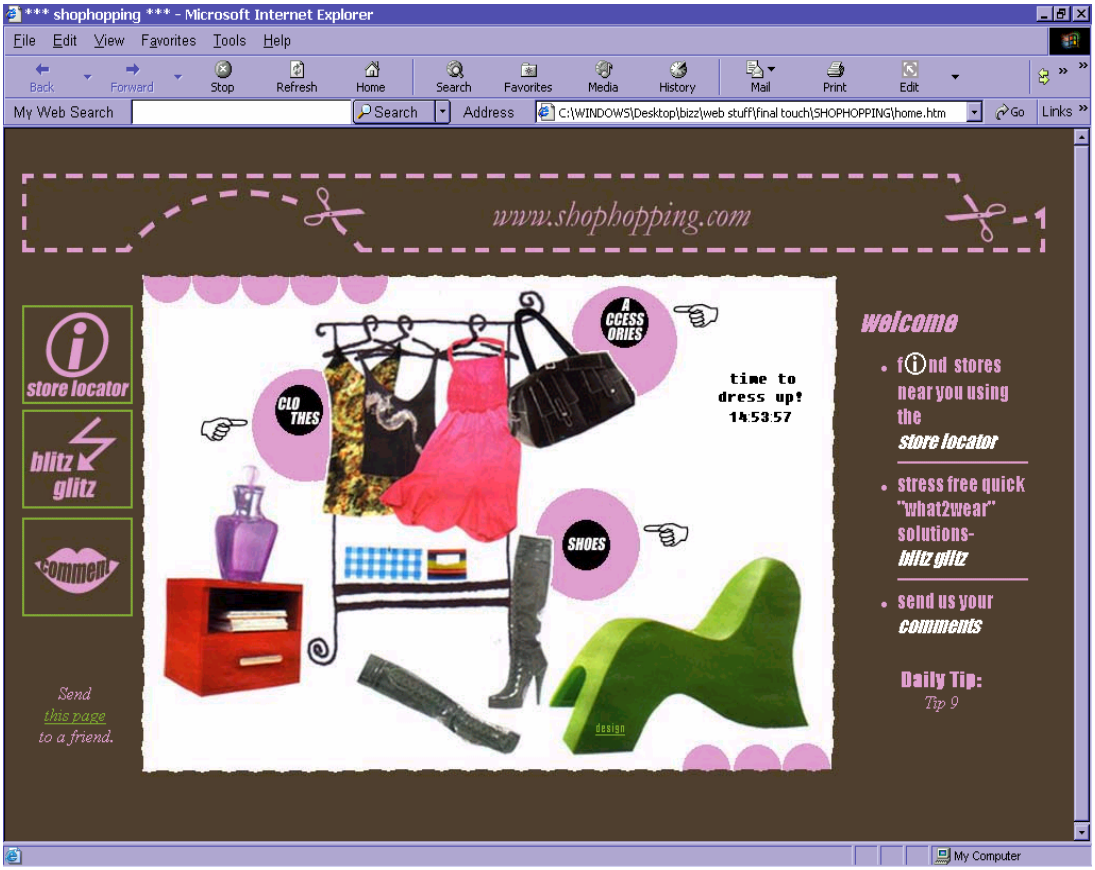
document. images[0].src= "mojaSlika.src"
document. images[0].src= "slika.jpg"
Promena slike je trenutna.

Najpopularnija tehnika za "oživljavanje" *Web* stranice je promena slike koje se ponašaju kao dugmad kada korisnik stane kursorom miša na njih.

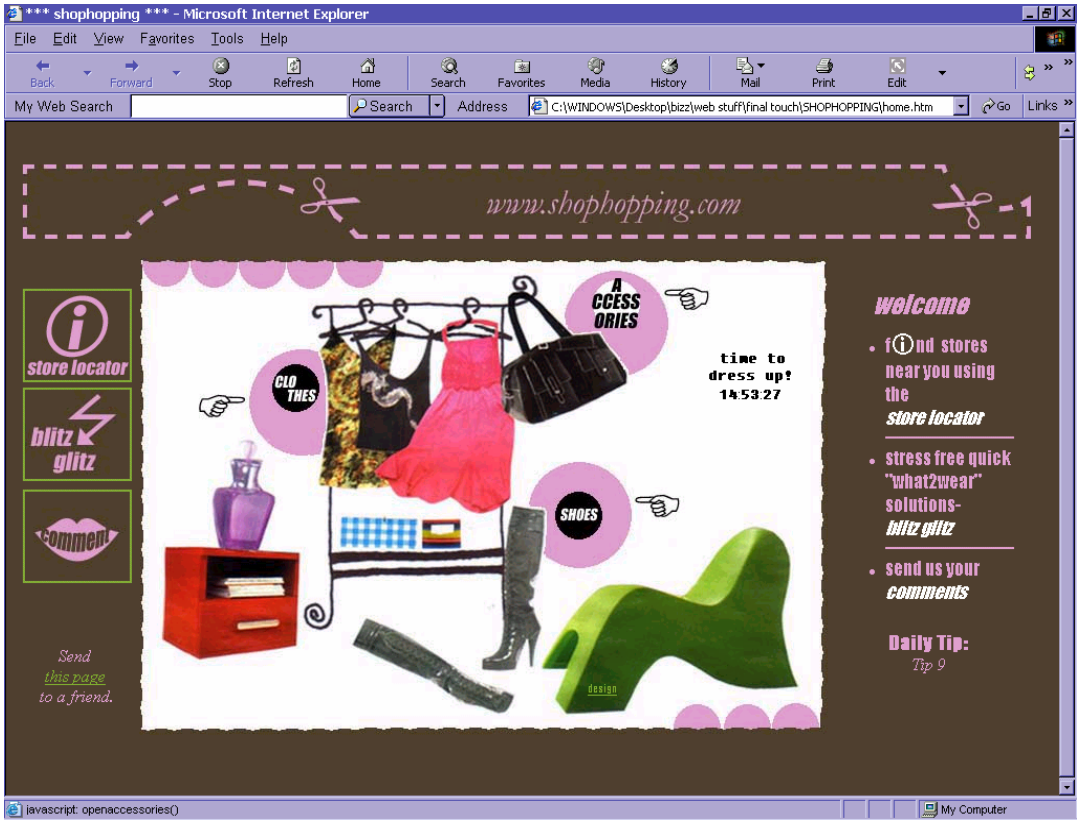
Efekat koj će biti prikazan je pitanje ukusa- promena boje, pojavljivanje okvira, blago zatamnjenje- koji god efekat da je u pitanju skript je isti.

Najjednostavnije je napraviti dve funkcije u skriptu: jednu koja će biti pozvana procedurom za obradu događaja *onmouseove=r*, i druga koja će biti pozvana procedurom *onmouseout=*.

```
<head>
<title>promena slike</title>
<script language="JavaScript">
function alterOver1(){
        document.a1.src="preko.jpg" }
function alterOut1(){
        document.a1.src="nazad.jpg" }
</script>
<body>
<img src="linka.jpg" name="a1" height=50 width=50
    border=0 onmouseover="alterOver1()"
onmouseout="alterOut1()">
</body>
```



Primetimo promenu sličice "accessories".



Sledi primer skripta za prikaz slika u formi slajdova. Moguće je pregledati slike manuelno, klićući na "*next*" i "*previous*", ili auto prelistavanje- "*start/stop*".

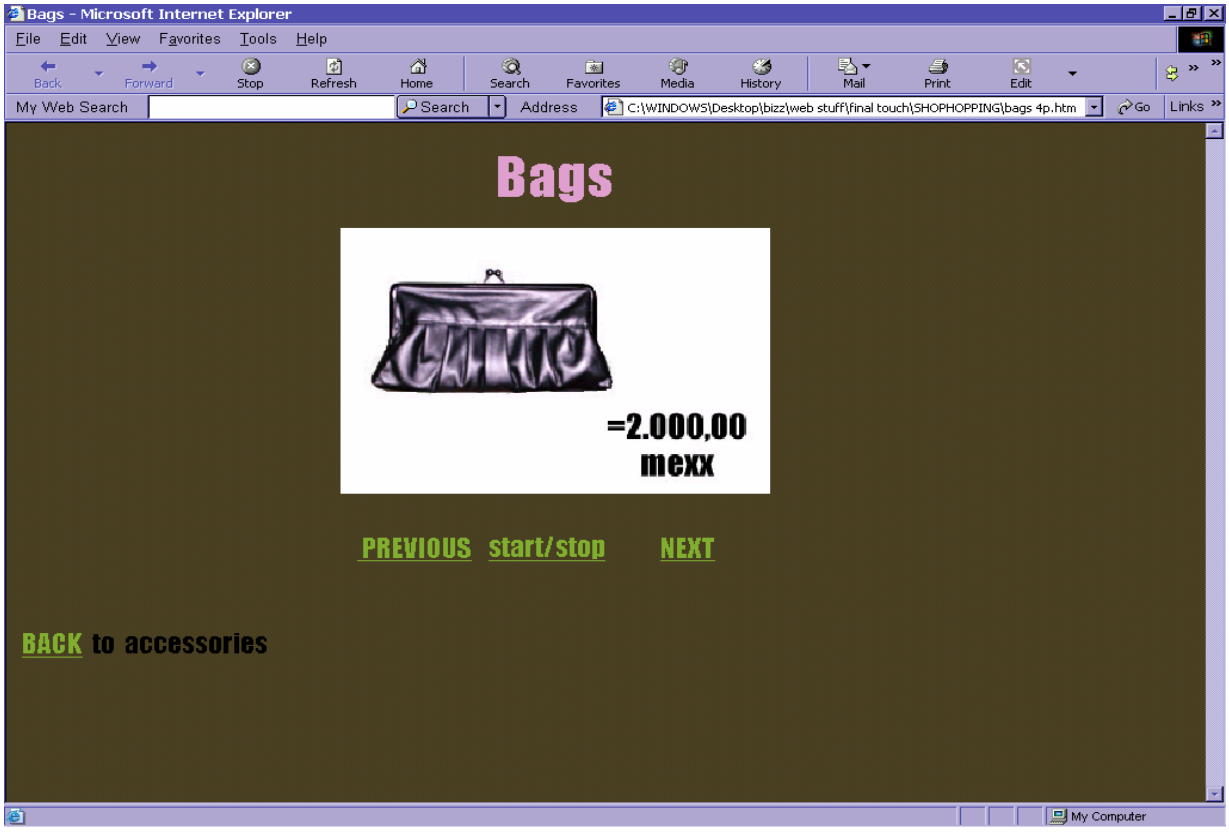
```
<html>
<head>
<title>Bags</title>
<script language="JavaScript">
NewImg= new Array( "bag1.jpg", "bag2.jpg", "bag3.jpg");
var ImgNum = 0;
var ImgLength = NewImg.length - 1;
var delay = 3000;
var lock = false;
var run;
function chgImg(direction) {
        if (document.images) {
        ImgNum = ImgNum + direction;
                if (ImgNum > ImgLength) {  ImgNum = 0;      }
                if (ImgNum < 0) {   ImgNum = ImgLength;     }
                document.slideshow.src = NewImg[ImgNum];
        }
}
```

```
function auto() {
        if (lock == true) {
        lock = false;
        window.clearInterval(run);
        }
        else if (lock == false) {
        lock = true;
        run = setInterval("chgImg(1)", delay);
}       }
</script>
</head>
<body bgcolor=#4d4029 link=#89ae3b alink=#89ae3b vlink=#89ae3b>
<table width=400 height=380>
        <tr>     <td height=38 width=400 align=center>
                 <font face="impact" size="6"
color="#E09FCE">Bags</font></td>
        </tr>
        <tr>     <td height=261 width=400 align=center valign=middle>

                <img src="bag1.jpg" height=200 width=300
name="slideshow">
                <table width="307">
                      <tr>
                      <td align="center" width="85">
                        <a href="javascript:chgImg(-1)">
                    <FONT face=impact size=4 >< 
PREVIOUS</font></a>
                      </td>
                      <td align="center" width="120">
                        <a href="javascript:auto()">
                    <FONT face=impact size=2 >start/stop</font></a>
                        </td>
                        <td align="center" width="88">
                          <a href="javascript:chgImg(1)">
                    <FONT face=impact size=4 >NEXT
 ></font></a>
                      </td>
                      </tr>
                </table>
                </td>
        </tr>
        <tr>
           <td height=36 width=100>
           <FONT face=impact size=3>
           <a href="accessories.htm">BACK</a></font>
          <FONT face=impact size=2>accessories</font>
          </td>
          </tr>
```

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

Prikaz u *browser*-u:



4.6.2 *Area*

Pored elemenata koji se prave oznakama *img* kao prve grupe objekata koji se koriste u skriptovima da povećaju interaktivnost, tu su i elementi koji se prave pomoću oznaka *area* i služe za pravljenje slikovnih mapa. *JavaScript* tretira objekat mape kao jedan od objekata veze (*link, anchor*) u dokumentu jer pritisak miša na oblast mape vodi korisnika do drugog dokumenta ili sidra na istoj stranici, što je karakteristika hiperveze. *HTML* definicije veza i oblasti mapa su različite ali su im svojstva i događaji skoro isti. na osnovu koordinata i oblika moguće je definisati bilo koji broj oblasti unutar slike, a uz pomoć grafičkih alata lako se određuju koordinate slike.

Navedeni delovi koda pišu se unutar oznaka *body* na željenom mestu, a predstavljaju slikovnu mapu u kombinaciji sa *JavaScript*-om (primetimo da se koristi i promena slike kotrljanjem miša, kao i procedura *onclick*=)

```
<img src="aCircle.jpg" height=50 width=50 border=0 usemap="#acc
onmouseover="alterOver1()" onmouseout="alterOut1()">

<map id="acc" name="acc">
    <area shape="circle" coords="25,25,23"
onclick="openaccessories()">
</map>
```

4.6.3 *Layer*

Sa svakom novom generacijom *browser*-a skriptovi su sve moćniji kada se radi o promeni sadržaja. *Navigator* uvodi nove objekte i oznake. *Netscape*-ov doprinos D*HTML*-u, oznaka <*layer*> označava objekat unutar dokumenta koji može da sadrži vlastiti dokument, da bude postavljen i da ima određene dimenzije, bilo gde u dokumentu. *Internet Explorer* 4 nema objekat *layer*.

Svaki *layer* u *Navigator*u sadrži neki *HTML* koji postoji u vlastitoj ravni iznad glavnog dokumenta koji se učitava u prozor. Sadržaj pojedinačnog sloja može promeniti položaj, veličinu ili biti sakriven pomoću skripta. U tom slučaju, bilo koja promena koja utiče na osnovni sloj, utiče i na sve slojeve u njemu.

Formiranje referenci u *JavaScript*-u za slojeve i njihove objekte prilično je slično postavljanju okvira. Dakle, pre referenciranja mora se znati odnos između dokumenta koji sadrži skript i cilja reference. Recimo da skript u zaglavlju sadrži naredbu koja u sloju *sloj1 (layer name="sloj1")* menja svojstvo *bgColor*:

document.sloj1.bgColor = "red"

Kako svaki sloj sadrži objekat *document*, kada bi se ispitivalo svojstvo *lastModified* iz *HTML* dokumenta, iz određenog sloja, naredba bi bila:

document. sloj1. document. lastModified

Svaki sloj ima svoj fizički sloj. Širina i visina se po pravilu predstavljaju promenljivama x i y, treća dimenzija- položaj sloja u skupu slojeva- naziva se z redosled. Redosled se automatski određuje po redosledu učitavanje, gde najviši *layer* ima najveći broj. Najviši je onaj koji je najbliži posmatraču kada gleda stranicu na monitoru. Svojstva *above, below, siblingAbove, siblingLeft*, definišu položaj datog *layer*-a u odnosu na one oko njega. *background* i *bgColor* tiču se pozadine i boje pozadine sloja. Moguće je i definisati položaj i

veličinu pravougaone oblasti sloja koja je vidljiva za korisnika pomoću objekta svojstva *layer.clip*.

Sloj prepoznaje kada kursor uđe u njegovu oblast ili iz nje izađe. Objekat *layer* ima događaje *onMouseOver*= i *onMouseOut*= koji omogućavaju da se obave bilo koje operacije kao reakcija na aktivnosti korisnika.

4.7 Obrasci- objekat *form*, dugmad i objekat *select*

Interakcija između *Web* stranice i korisnika odvija se uglavnom unutar obrasca čiji elementi predstavljaju dvosmerne kanale informacija između skriptova i korisnika. Elementi obrasca su jedini način da korisnik unosi tekstualne podatke ili bira između ponuđenih opcija prikazanih kao polja za potvrdu, uzajamno isključiva radio dugmad ili stavke u listi.

Informacije unete u obrazac od strane korisnika šalju se serveru, a na koji način će biti obrađene zavisi od *CGI* programa koji se nalaze na serveru. Ako je server pod kontrolom kreatora *Web* stranice, svi programi za prikupljanje i pretraživanje podataka su dozvoljeni. Međutim, ako je za smeštanje *HTML* datoteka zadužen *ISP*[*], na raspolaganju su samo oni *CGI* programi dostupni svim korisnicima usluga.

Složen *HTML* dokument može imati više obrazaca, gde svaki par oznaka *<form>* i *</form>* definiše po jedan.

Na objekat obrasca može da se ukaže na osnovu njegovog položaja u nizu obrasca koji sadrži dokument, ili na osnovu imena (ukoliko je imenovan unutar oznaka *form*). Čak iako postoji samo jedan obrazac, on pripada nizu obrasca od jednog elementa, a njegova referenca je sledeća:

*document.*forms*[0]*

Ako je obrascu dodeljeno ime, u referencu se, umesto *forms[0]* uključuje ti ime.

Većina *ISP*-a koji omogućavaju postavljanje *Web* prezentacija na svoje servere imaju standardne *CGI* programe za slanje podataka na željene *e-mail* adrese. Ovako poslati podaci najčešće ne sadrže *e-mail* adresu osobe koja ga je poslala osim ako se na obrascu ne nalazi polje u koje se ta informacija može upisati. Tri atributa se moraju koristiti ako želimo da podatke iz obrasca pošaljemo *e-mail*-om. Prvi je method. On se mora postaviti na *post*. Zatim, *action*. Da bi se uspostavila veza do neke datoteke ili *CGI* programa, koristi senaredba *mailto: URL*. Tako, dolazi se do sledećeg:

action = "mailto: jim@jimsnet.com ? subject= comments"

[*] ISP- Internet Service Provider

Poslednji važan atribut je *enctype*. Ovaj atribut je poželjno postaviti na "*text/plain*" iz razloga što će, ako se izostavi ovaj atribut, *browser* podatke iz obrasca poslati nečitko i u jednom redu. Dakle:

ime=Jim+Jones&adresa=Elm+Street+5&zanimanje=slikar

ili sa *enctype="text/plain":*

ime=Jim Jones
adresa=Elm Street 5
zanimanje=slikar

Svi ovi atributi su i svojstva objekta obrasca i može im se pristupiti korišćenjem samo malih slova u nazivu.

*document.*forms*[0].action*
document.imeObrasca.action

Bilo koja od ovih vrednosti može se promeniti jednostavnim dodeljivanjem nove vrednosti.

Ako ne želimo da jednostavno dodamo klasično *Submit* dugme koje šalje obrazac na server, već bismo radije neku lepšu sliku, oznaka koja tu sliku povezuje moraće da koristi *javascript:URL* da bi pozvala skript koji će predati obrazac. Skriptovani ekvivalent predaje obrazac je metod objekta *form.submit().*

*document.*forms*[0]. submit()*

Isto važi i za dugme za resetovanje tj. postavljanje svih elemenata na početne vrednosti, metodom *reset().*

Da bi se olakšalo praćenje tipa za svaki elemenat u obrascu, *browser* obezbeđuje listu svih elemenata. Ova lista je poseban niz, sa elementima koji su navedeni po redosledu pojavljivanja njihovih *HTML* oznaka u programskom kodu. Mnogo je efikasnije napraviti vezu do elemenata direktno, koristeći njihova imena.

Svaki od četiri elementa obrasca koji se odnose na tekst- *text, texarea, password* i *hidden*- predstavljaju objekte koji mogu sadržati tekst. Svi osim *hidden* objekta se prikazuju na stranici, omogućavajući korisniku da unese informaciju.

Procedure za obradu događaja se aktiviraju pomoću mnogih akcija korisnika, kao što su fokusiranje polja ili menjanje teksta. Većina polja za tekst aktivira se procedurom *onChange=.* Najčešće korišćeno svojstvo je *value*. Ovo svojstvo predstavlja tekući sadržaj elementa teksta. Skript može

pozvati i podesiti njegov sadržaj u svakom trenutku. Sadržaj svojstva *value* uvek je znakovni niz.

Pre slanja serveru, možda je potrebno proveriti neke od unetih podataka u obrazac. Ovo može da se obavi u funkciji koju poziva procedura za obradu događaja *onSubmit=. onSubmit=* vraća *true* i predaja se nastavlja, odnosno *false* da bi prekinula predaju. Kada se poziv obrade događaja *onSubmit=* definiše kao atribut u definiciji *form*, *JavaScript* aktivira događaj *submit* neposredno pre nego što se podaci pošalju na server. Zbog toga se bilo koja funkcija navedena u atributu *onSubmit=* izvršava pre nego šeto se podaci pošalju. Ovaj događaj se aktivira samo ako se slanje podataka na server vrši klasičnim *Submit* dugmetom a ne metodom *form.submit().*

Pogledajmo primer skripta sa funkcijom koja proverava da li su u obrazac uneti svi zahtevani podaci.

```
<html>
<head>
<title></title>
<script>
function checkform(){
      var form=document.forms[0]
      for (var i=0; i<form.elements.length; i++){
          if (form.elements[i].value==""){
              alert("please fill out all the fields")
              break
          }
      }
      return true
}
</script>
</head>

<body>
<form onsubmit="checkform()">
      <p align=center>
          <font face=impact size=4 color=#e09fce>name</font>
          <input  type=text maxlength=20 size=30>
      </p>
      <p align=center>
          <font face=impact size=4 color=#e09fce>e- mail</font>
          <input  type=text maxlength=20 size=30>
      </p>
      <p align=center>
          <font face=impact size=4 color=#e09fce>message</font>
          <textarea id=unos rows=10 cols=40> </textarea>
      </p>
```

```
        <p align=center>
            <input id=Submit type=submit value=send></font>
        </p>
</form>
</body>
</html>
```

4.7.1 Dugmad

Dugme ili *button*, je jedan od najjednostavnijih objekata za skriptovanje. Ono ima samo nekoliko svojstava kojima se retko pristupa ili se retko modifikuju. Vizuelni efekat dugmeta ne proizvodi *HTML* ili skript, već operativni sistem ili *browser* koje koristi posetilac stranice. Najkorisnija procedura za obradu događaja objekta *button* je *onClick=.* Ona se aktivira, kako i ime kaže, svaki put kad korisnik tasterom miša pritisne dugme.

Checkbox tj. polje za potvrdu je takođe jednostavan objekat obrasca. Svojstvo *value checkbox*-a je tekst koji želite da povežete sa tim objektom. Ovaj tekst se ne pojavljuje na stranici u bilo kom obliku, ali ovo svojstvo može biti važno skriptu koji želi da zna više o *checkbox*-u. Ključno svojstvo *checkbox*-a jeste da li je ono povrđeno ili nije. Svojstvo *checked* je logička vrednost: *true* ako je polje potvrđeno, *false* ako nije.

Što se tiče objekata tipa *radio* dugme tj. radio, situacija je malo drugačija. Naime, da bi se *browser*-u omogućilo da upravlja osvetljavanjem i zatamnjivanjem odgovarajuće grupe dugmadi, potrebno je svakom dugmetu u grupi dodeliti isto ime. Moguće je imati više grupa u obrascu, ali svaki član jedne grupe mora imati isto ime. Dakle, *browser* upravlja nizom objekata sa istim imenom, pa ime dodeljeno grupi postaje ime niza. Neka svojstva se primenjuju na grupu kao celinu, druga svojstva se primenjuju na pojedinu dugmad u grupi i moraju se adresirati pomoću indeksa niza. Na primer, broj dugmadi u grupi govori svojstvo *length*:

document. forms*[0].imeGrupe.length*

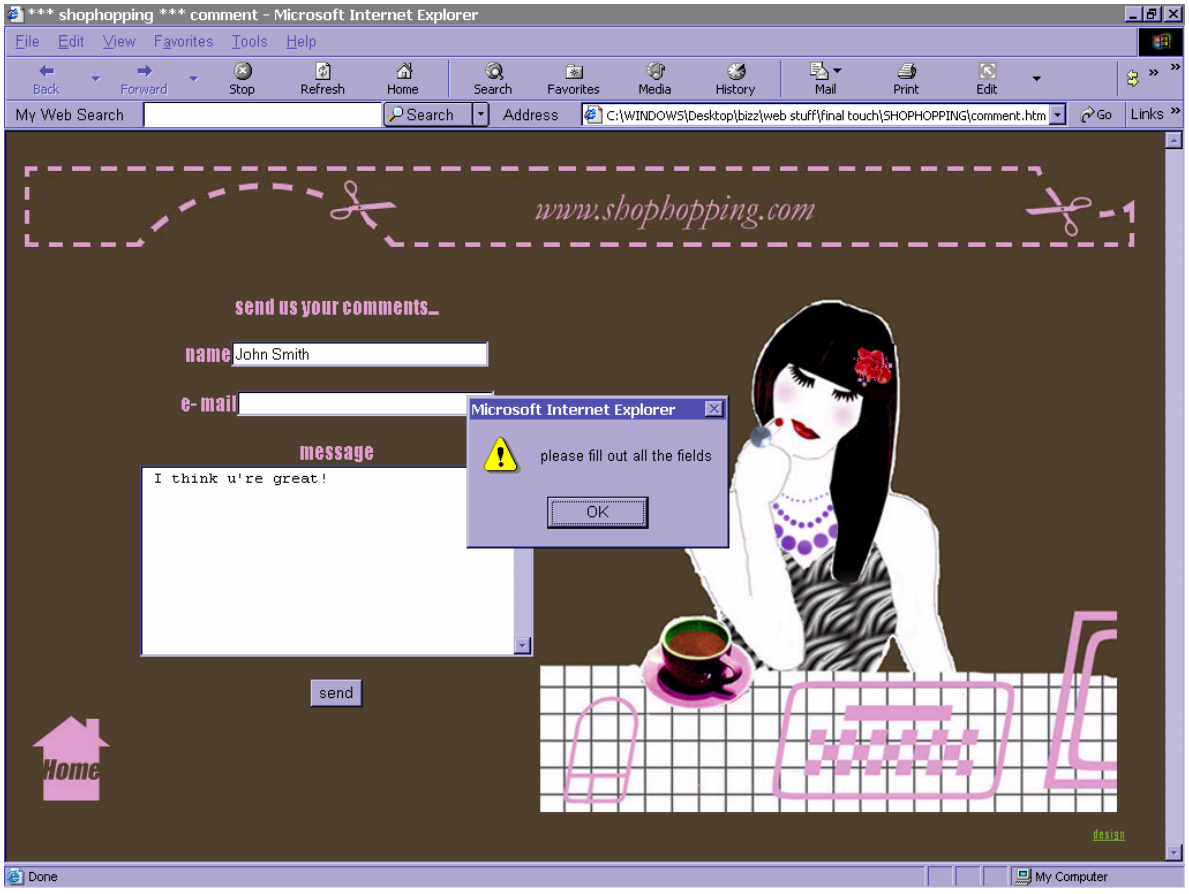
Ako je, na primer, potrebno proveriti da li je određeno dugme trenutno izabrano, mora se individualno pristupiti tom dugmetu i očitati svojstvo *checked*:

document. forms*[0]. imeGrupe[0]. checked*

Realizovanje provere svojstva *checked* dugmeta checkbox krajnje je jednostavno.

```
<html>
<head>
<title> provera svojstva checked </title>
<script language="javascript">
        function inspectbox(form){
                if (form. checkbutton.checked){
                                alert('polje je potvrdjeno!')
                } else {
                                alert('polje trenutno nije potvrdjeno')
                }
        }
</script>
</head>
<body>
<form>
        <input type="checkbox" name ="checkbutton"> potvrdite ovde <br>
        <input type="button" name ="boxchecker" onClick=
"inspectbox(this.form)" value="stanje">
</form>
</body>
</html>
```

Skript javlja da nisu sva polja popunjena.



4.7.2 Objekat *select*

Najsloženiji objekat za skriptovanje je objekat *select*. *Select* sadrži niz opcionih objekata, a formira se u *HTML*-u da bi se prikazao kao padajući meni (*dropdown menu*) ili lista sa *scrollbar*-om tj. trakom za pomeranje. Ako neki skript treba da odredi koju stavku je odabrao korisnik, moraju se koristiti dva svojstva: svojstvo izabranog objekta i svojstvo izabrane opcije tog objekta.

Najvažnije svojstvo objekta select je *selectedIndex*. Kada korisnik odabere opciju na listi, svojstvu *selectedIndex* se dodeljuje broj te opcije. Prva opcija ima vrednost nula.Svojstvu *selectedIndex* pristupa se ovako:

*document.*forms*[0].selectName.selectedIndex*

Informacija koju sadrži *selectedIndex* može se koristiti kao prečica do svojstava izabrane opcije. Umesto prolaska kroz petlju i ispitivanja svake opcije u listi, zgodno je u referenci navesti svojstvo *selectedIndex* čija vrednost daje broj izabrane opcije.

Dva važna svojstva stavke *options* su *text* i *value*.

*document.*forms*[0].selectName.options[0]. text*
*document.*forms*[0].selectName.options[0]. value*

Svojstvo *text* je znakovni niz koji se pojavljuje na ekranu u objektu *select*. Unutar *HTML* oznake *option* može se postaviti atribut *value* omogućava povezivanje nekih skrivenih informacija u obliku niza znakova sa svakim vidljivim unosom na listi.

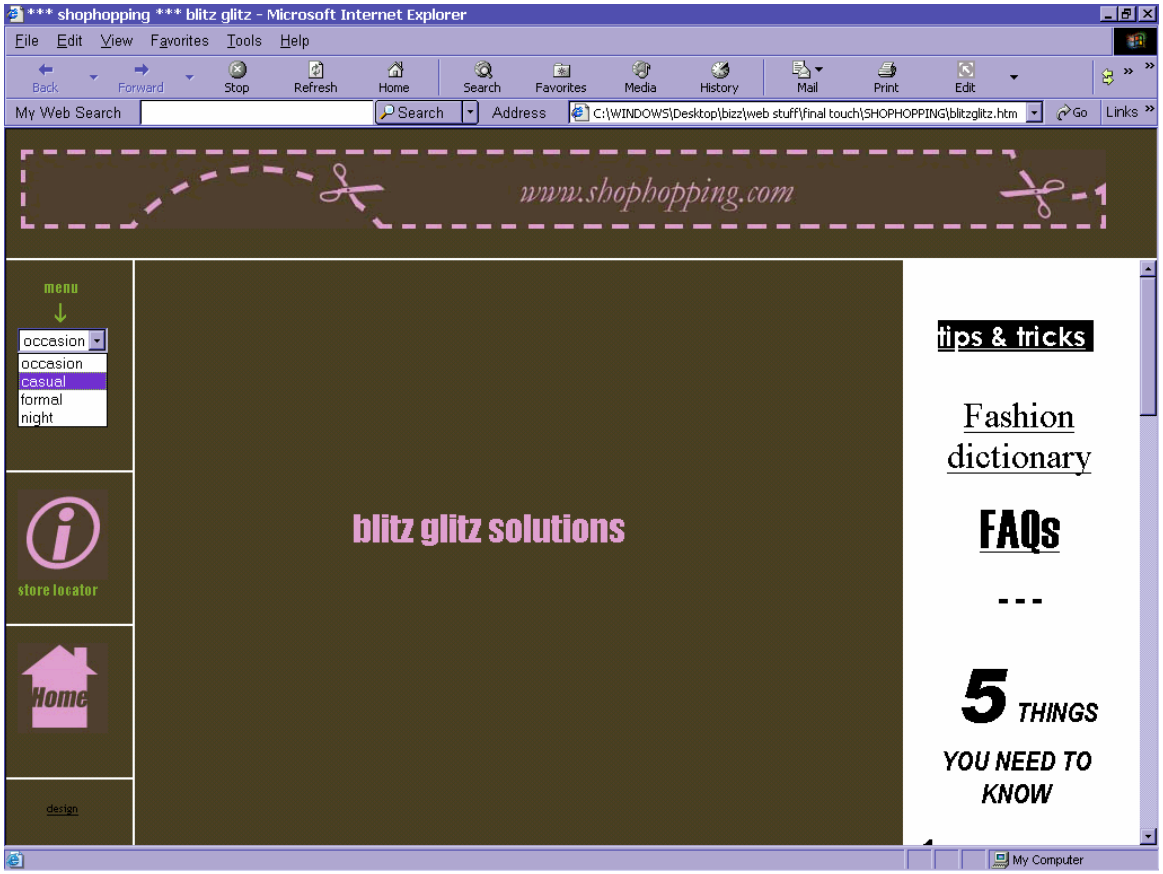
Za pokretanje objekta *select* koristi se procedura za obradu događaja onChange=. Čim korisnik izabere novu stavku u listi, ova procedura pokreće skript koji je povezan sa njom, skript uzima svojstvo value izabrane opcije i dodeljuje tu vrednost objektu lokacije da bi omogućio navigaciju.

Najbolji primer je svakako *dropdown* meni.

```
<html>
<head>
<script>
      function DropDownMenu(entered){
          with (entered){
             ref=options[selectedIndex].value;
             splitcharacter=ref.lastIndexOf("&");
             if (splitcharacter!=-1) {
                loc=ref.substring(0,splitcharacter);
                 target=ref.substring(splitcharacter+1,1000).toLowerCase();
             }
```

```
                else {
                        loc=ref;
                        target="_self";
                }
                lowloc=loc.toLowerCase();
                if (lowloc=="false") {return;}
                parent.main.location=loc;
            }
        }
</script>
</head>
<body bgcolor=#4d4029>
<p align=center>
        <font size="2" face="Impact" color="#89ae3b"> menu </font><br>
        <select onChange="DropDownMenu(this)">
                <option value="false"> occasion </option>
                <option value="casual.htm"> casual </option>
                <option value="formal.htm"> formal </option>
                <option value="night.htm"> night </option>
        </select>
</p>
</body>
</html>
```

Okvir gore levo sadrži padajući meni.



4.8 Objekat niza znakova, matematika i datum

Znakovni niz tj. *string*, je svaki tekst između znakova navoda. Znakovi navoda mogu biti celi ili polunavodnici.To omogućava da jedan niz bude ugnežđen unutar drugoga. *JavaScript* ne nameće nikakva ograničenja u broju znakova u nizu znakova. Mnogi *browser*-i međutim ograničavaju dužinu naredbe skripta na 255 znakova. Ova granica se ponekad prekorači ako skript sadrži dugačak znakovni niz koji treba da postane skriptovani sadržaj stranice. Takve redove treba podeliti na manje.

Postoje dva načina kako promenljivoj dodeliti vrednost niza znakova. Najjednostavniji je naredbom dodele:

var mojString = "Zdravo"

Objekat *string* može se napraviti i pomoću kondtrukcije sa ključnom reči *new*:

var mojString = new String("Zdravo")

Promenljiva inicijalizovana na bilo koji od ova dva načina moćiće da odgovori na sve metode objekta *string*.

Spajanje dva stringa u jedan je tzv. povezivanje stringova ili konkatenacija, i zahteva jedan od dva operatora *JavaScript*-a. Prvi od njih je operator sabiranja koji se koristi i za tekst koji se dinamički prikazuje na *Web* stranici:

document.write("moj browser je b" +navigator.appName+"/b .") ,

ali i da bi se već postojećem tekstu dodalo još teksta:

```
var poruka= "Zdravo"
poruka = poruka + "prijatelju"
poruka = poruka + "moj!"
```

Drugi operator nudi prečicu. Smisao je da se sadržaj sa desne strane operatora, doda na kraj sadržaja sa leve strane operatora.

```
var poruka = "Zdravo"
poruka += "prijatelju"
poruka += "moj!"
```

Od svih objekata *JavaScript*-a, objekat *string* ima najraznovrsnije metode. Pri korišćenju metoda niza znakova, string na koji se metod primenjuje postaje deo reference, iza čega sledi ime metoda.

string.imeMetoda()

Nekoliko metoda konvertuje ceo string u mala ili velika slova:

var velika = string.toUpperCase()
var mala= string.toLowerCase()

Za proveru sadržavanja jednog stringa u drugom koristi se metod *string.indexOf().* Ovaj metod vraća broj koji je vrednost indeksa znaka u većem nizu znakova, gde počinje manji niz. Ako se traženo ne pronađe, povratna vrednost je -1.

Metod *charAt()* izdvaja jedan znak sa poznatog mesta u stringu. Parametar metoda je broj indeksa znaka koji treba izdvojiti. Još jedan metod- *string.substring()*- omogućava izdvajanje neprekidne sekvence znakova, a parametri su početna i krajnja pozicija podniza koji je potrebno izdvojiti. Važno je znati da znak na završnoj poziciji neće biti izdvojen; dakle, svi znaci do tog ali ne i taj.

Kao i slične funkcije drugih programskih jezika, metod *JavaScript*-a *indexOf()* pruža skriptu mogućnost da otkrije lokaciju u glavnom znakovnom nizu gde počinje znakovni niz koji se traži:

string. indexOf (trazeniString, startIndex)

Opciono je zadavanje početnog indeksa (*startIndex*) odlakle bi trebalo da počne traženje.

Dodavanje pojedinih znakova u JavaScriptu je otežano načinom na koji su definisani znakovni nizovi. Dodavanje znaka navoda, znaka za novi red, apostrofa i sl. *JavaScript* međutim ima mehanizam za unos takvih znakova u obične znakovne nizove. To se postiže upisivanjem kose crte (*slash*) a zatim željenog znaka. Za tzv. nevidljive znakove, u *JavaScript*-u postoji poseban skup slova iza kose crte. Najčešće se susreće sa sledećim:

\"	navodnik
\'	apostrof
\\	back slash (obrnuta kosa crta)
\b	backspace (brisanje unazad)
\t	tabulator (uvlačenje)
\n	novi red
\r	carriege return (prelazak u novi red)

Ove kombinacije oznaka pišu se unutar znakovnih nizova u navodnicima da *JavaScript* može da ih prepozna. Na primer:

poruka= "It \' s a nice day today!";
poruka= "Ovo je prvi red. \n Ovo je drugi red.";
poruka= document.title +"\n"+" ima " +document.links. length+" linkova."

4.8.1 Matematički objekat *Math*

JavaScript obezbeđuje mnoštvo matematičkih mogućnosti. Sve ove mogućnosti sadržane su u matemetičkom objektu *Math*. Ovaj objekat se razlikuje od drugih objekata *JavaScript*-a po tome što se za njegovu upotrebu ne pravi njeova kopija. Skriptovi direktno pozivaju svojstva i metode objekta *Math* i on je deo reference.

var brojPi = Math.PI
var drugiKoren = Math.SQRT2

Metodi pokrivaju širok dijapazon trigonometrijskih i drugih matematičkih funkcija koje rade sa brojnim vrednostima koje su prethodno definisane u skriptu.

var kojiJeVeci = Math. max(broj1, broj2)
var naStepen = Math. pow (broj, stepen)
var zaokruzi = Math.round(broj)

4.8.2 Datumski objekat *Date*

Rad sa datumom počinje pozivanjem konstruktora objekta *Date* da bi se dobo jedan objekat klase *Date* povezan sa određenim vremenom i datumom.

var danas = new Date()

Objekat *Date* uzima snimak *PC*-jevog internog sata i vraća datumski objekata za taj momenat. Interno, vrednost datumskog objekta u nekom momentu je vreme u milisekundama počev od nula časova 1. januara 1970. po Griniču (*GMT*)- svetskoj referentnoj tački za sve vremenske konverzije. Tako datumski objekat sadrži informacije i o datumu i o vremenu. Moguće je definisati objekat *Date* za neki drugi trenutak, navodeći tu informaciju kai parametar u knstruktorskoj funkciji *Date*:

var nekiDatum = new Date ("Month dd, yyyy hh:mm:ss")
var nekiDatum = new Date ("Month dd, yyyy")
var nekiDatum = new Date (yy, mm, dd)

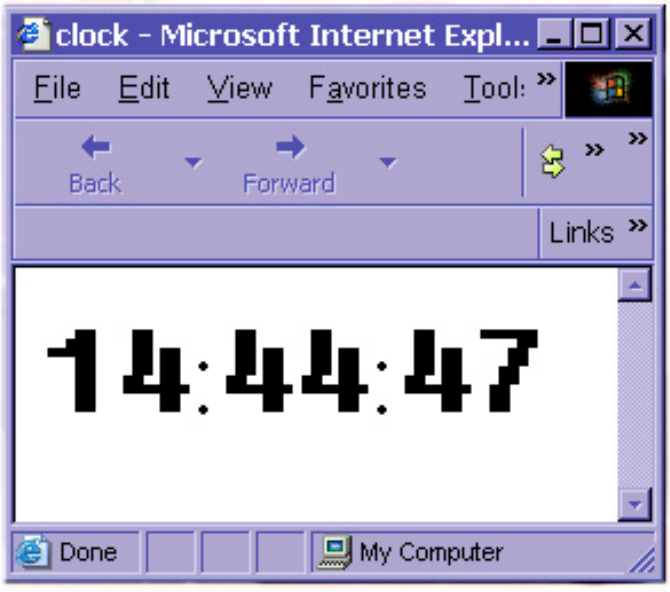
Komponente datumskog objekta mogu se izdvojiti pomoću niza metoda koji se mogu primeniti na trenutni datumski objekat, kao što su: *getTime(), getYear(), getHours(), ..., setMinutes(val), setSeconds(val).*

Sledi primer skripta za prikaz trenutnog vremena.

```
<html>
<head>
<script type="text/javascript">
	function startTime() {
	var today=new Date()
	var h=today.getHours()
	var m=today.getMinutes()
	var s=today.getSeconds()
	m=checkTime(m)
	s=checkTime(s)
	document.getElementById('txt').innerHTML=h+":"+m+":"+s
	t=setTimeout('startTime()',500)
	}
	// ispred brojeva manjih od 10 dodaje se nula
	function checkTime(i) {
		if (i<10) {  i="0" + i  }
		return i
	}
</script>
</head>

<body onload="startTime()">
<div id="txt"></div>
</body>
</html>
```

Browser to prikazuje ovako:



Poglavlje 5

DHTML i kompatibilnost platformi

5.1 DHTML- Dynamic HyperText Markup Language

Browser-i četvrtog nivoa, *Navigator* 4 i Interet Explorer 4, udružili su najnovije WWW tehnologije za prikaz i kontrolu sadržaja *Web* stranice i sve obuhvatili imenom dinamički *HTML* (D*HTML*).

D*HTML* ne eliminiše tehnologije koje su se i pre koristile *Web design*-u, već kreatorima *Web* stranica pruža više mogućnosti da ulepšaju svoje aplikacije. Ključne tehnologije koje objedinjuje D*HTML* su :

- *Cascading Style Sheets (CSS1)*
- *Cascading Style Sheets- Positioning (CSS- P)*
- *Document Object Model (DOM)*
- upotreba jezika za skriptovanje.

Kaskadne liste stilova ili *CSS1* (broj je oznaka nivoa; postoji i 2), definišu karakteristike za prikazivanje elemenata dokumenta. Bilo koji element koji sadrži *HTML* oznaku može biti vezan za jednu ili više postavki koje se donose na način prikaza tog sadržaja. Kombinacija *HTML* oznake i definicije stila naziva se pravilo (*rule*). Delovi pravila su selektor (identifikator oznake, npr. h1) i deklaracija. Deklaracija ima dve komponente: ime svojstva i vrednost, odvojene dvotačkom.

```
<style type="text/css">
p {color: green}
</style>
```

Dakle, tip stila je definisan kao tip *CSS*. Pravilo određuje da svi blokovi između oznaka i (selektor *p*), imaju tekst u zelenoj boji (svojstvo je *color*, a vrednost *green*). Moguće je dodati još deklaracija i one u tom slučaju moraju biti odvojene jedna od druge tačkom i zarezom.

```
<style type="text/css">
p {color: greenč font' size: 10pt}
</style>
```

Ako je neki pasus definisan ovim pravilom, on može imati sopstvenu boju:

```
p style= "color: blue"
```

Takođe, *CSS1* nudi i objedinjavanje stilova povezivanjem sa spoljašnjim dokumentima koji sadrže pravila stila, kao *JavaScript* sa spoljašnjom bibliotekom skriptova.

Klase, odnosno podskupovi stilova su takođe jedna od mogućnosti: svakom stilu se dodeli ime klase u atributu *class*, tako *browser* prepoznaje koje pravilo treba da primeni.

Navigator 4 i *Internet Explorer* 4 podržavaju *CSS1*. *Netscape* je još dodao jedan tip stila *Navigator*u- *JavaScript Syle Sheets (JSS)* čija sintaksa dosta liči na *JavaScript*.

Pozicioniranje pomoću kaskadnih lista stilova (*CSS-P*), je više od običnog definisanja tačke na stranici gde određeni element treba da počne da se prikazuje. U specifikaciji postoje još i svojstva koja definišu da li će element biti vidljiv ili skriven, i koji je element vizuelno najbliži korisniku. Sintaksa postavlja ova svojstva dok se stranica učitava i u kombinaciji sa *JavaScript*-om ona postaju dinamička.

Model objekata dokumenta (*DOM*), možda je i najvažniji deo D*HTML*-a, jer se pisci skriptova oslanjaju na *DOM* u svim pitanjima različitosti platformi. *Internet Explorer* 3, na primer, nema objekat *image*, što je uzrokovalo probleme- skripte za promenu slike kotrljanjem miša u *Navigator*u 3, izazivale su greške korisnicima *IE* 3. Kako *Navigator* i *Internet Explorer* imaju različite razvojne puteve *DOM*-a, biće teško usaglasiti standard *DOM*-a za skriptabilne objekte. Jer, što je više platforma uložila u instaliranu osnovu svojih postojećih objekata *DOM*-a, to će teže biti usaglasiti standard zbog mogućnosti potrebe degradiranja zarad pronalaženja zajedničkog imenioca tj. svojstava koja se poklapaju da bi skript imao efekta na obema platformama.

Jezici za pisanje skriptova, kao što je *JavaScript*, su veza između autora stranice i njenih elemenata koji se mogu kontrolisati. *Navigator* podržava samo *JavaScript*, dok *Internet Explorer* podržava i vlastitu verziju *JavaScript*-a (*JScript*) i *VBScript* (derivat *Visual Basic*-a). *JavaScript*, zbog postojanja podrške na više platformi, danas je dominantan skript jezik. D*HTML*, čija je svrha pisanje skriptova za elemente stranice, prvenstveno služi kao osnovna podrška *JavaScript*-u koji se bavi i modelom objekata.

I *Netscape* i *Microsoft* razvili su proširenja standarda koja se često međusobno isključuju i onemogućavaju pisanje skriptova pogodnih za oba *browser*-a, osim ako pisci nisu upoznati sa specifičnostima svake implementacije. *Netscape* je predložio novu *HTML* oznaku za pozicioniranje elemenata- *layer*. Organizacija *W3C* koja je nadležna za *HTML* specifikacije nije prihvatila ovu oznaku, ali je *Netscape* već izgradio D*HTML* oko ovog objekta. *Internet Explorer* ignoriše oznaku *layer* tako da sve definisano oznakom *layer* u dokumentu neće biti prikazano u *Internet Explorer*u. *Microsoft*umesto toga ima drugačiju tehniku rada sa elementima koji mogu da se pozicioniraju. Naime,u definiciju stila tj. u oznake *style* ili u atribut *style*,

dodaje se svojstvo *position* koje ima dve moguće vrednosti: *absolute* i *relative* (ovo svojstvo postoji i u *CSS-P*).

Zajednički imenilac elemenata koji se mogu pozicionirati je *CSS-P* sintaksa koja definiše stil sa svojstvom position. Koristeći oznaku *style* elementima se dodeljuju imena:

```
<html>
<head>
<style type="text/css">
#element1ime    {position: absolute}
#element2ime    {position: absolute}
</style>
</head>
...
```

Kasnije u dokumentu, atributu *id* dodeljuje se ime elementa unutar oznake koja sadrži to sve:

```
<div id="element1ime"> … </div>
```

Alternativna sintaksa je:

```
<div id="element1ime" style="position: absolute"> … </div>
```

U oba slučaja moguće je definisati još svojstava: *top, left, visibility*. itd.

5.2 Kompatibilnost platformi

Kada se govori o izbegavanju nekompatibilnosti, suština je u prevazilaženju nekompatibilnih referenci objekata i ponekad imena svojstava. Skriptovi nude nekoliko načina da se ovaj problem izbegne. Tri osnovne tehnike su:

a. grananje
b. jednakost platformi
c. namenski *API (Application Program Interface)*.

a. Grananje podrazumeva pisanje koda u dve grane, za svaki *browser*. Prethodno je potrebno definisati dve globalne promenljive koje služe kao logički indikatori za konstrukcije *if.. else*.

```
var Nav4, IE4
if (navigator. appVersion. charAt(0) == "4"){
        if (navigator. appName == "Netscape"){
        Nav4  true
        }
        else if (navigator. appVersion. indexOf("MSIE") != -1) {
        IE4 = true
        }
}
```

(ovaj primer pristup omogućava samo *browser*-ima verzije 4)

Globalne promenljive definisane na početku mogu se koristiti i u *if.. else* konstrukcijama koje se odnose na reference odgovarajuće za svaku platformu:

if (Nav4) { document. instructions. visibility = "hidden" }
else { document. instructions. style. visibility = "hidden" }

b. Reference objekata svake platforme sadrže referencu prema dokumentu i imenu objekta. Sintaksa *Internet Explorer*a sadrži još i reči *all* i *style*. Suština tehnike jednakosti platformi je to da se ova dodatna imena svojstava dodele promenljivama za *Internet Explorer* 4 i izostave za *Navigator* 4. Tako se u jednoj naredbi može sastaviti referenca za obe platforme.

```
var range = ""
var styleObj = ""
if (navigator. appVersion. charAt(0) == "4"){
        if (navigator. appVersion. indexOf("MSIE") != -1) {
                range = "all."
                styleObj = ". style"
        }
}
```

Alternativno, funkcija *eval()* pruža mogućnost dodele svih svojstava jednom naredbom:

eval("document. "+ range + "instructions" +styleObj+ " .visibility = 'hidden'")

Metod jednakosti platformi, međutim, nema efekta ako je objekat *layer Navigator*a ugnežđen unutar drugog *layer*-a, jer jednakost platformi podrazumeva da se objektu može prići iz spoljašnjeg dokumenta.

c. Namenski *API (Application Program Interface) JavaScript*-a je funkcija koju definiše pisac skripta a u ulozi je posrednika između skriptova i ostalih celina. Na primer, postavljanje svojstva *zIndex* objekta čije ime je prosleđeno kao parametar:

```
function setZindex(objName, zOrder) {
	var theObj
	if (Nav4) {
		theObj = eval("document. " objName)
	}
	else {
		theObj = eval ("document. all"+ objName + ". style")
	}
	if (theObject) {
		theObj.zIndex = zOrder
	}
}
```

Tako pozivanjem ove funkcije, skript može da pozove *zIndex()* i da ne misli o platformi. *API* se može koristiti u svim dokumentima a takođe i kao spoljašnja biblioteka.

5.3 Rad sa *browser*-ima koji ne podržavaju *DHTML*

Još jedan od potencijalnih problema je rad sa *browser*-ima koji ne podržavaju D*HTML*. Bez D*HTML*-a, *browser* prikazuje sadržaj po pravilima klasičnog *HTML*-a: elementima nije dozvoljeno preklapanje, svaka oznaka se prikazuje na levoj margini osim ako nije eksplicitno zadata druga oznaka za poravnavanje, element koji je u D*HTML*-u skriven pojaviće se, itd. Stoga, stranice pisane u D*HTML*-u uvek je preporučljivo proveriti i u starijim *browser*-ima. Sve, ipak, nije izgubljeno. Ako je stranica čitljiva u *browser*-u koji ne podržava D*HTML*, to ponekad može biti dovoljno. Inače, potrebno je napisati poseban sadržaj za obe vrste *browser*-a.

Poglavlje 6

Otkrivanje i sprečavanje problema

Kada poruke o grešci ne upućuju pravo na problem, od koristi su neke od sledećih tehnika za otkrivanje grešaka:

- provera *HTML* oznaka- pre pregleda koda *JavaScript*-a, pažljivo pregledati sve *HTML* oznake. Dakle, da li sve oznake imaju par uglastih zagrada i da li svi parovi oznaka imaju usaglašene oznake otvaranja i zatvaranja, takođe da li svi atributi čije se vrednosti pišu pod navodnicima imaju zatvorene navodnike, i sl.

- *source*- sa menija *View* odabirom opcije source moguće je videti svaki *HTML* kod koji skript formira. Tako je *HTML* kod napisan u JavaScriptu moguće štampati, pregledati i čuvati.

- isprekidani skriptovi- na primer, dugmad se ne pokreću na događaj *onClick*= ako se stranica ne učita ponovo što rezultuje time da skript jednom radi a jednom ne. Razlog ovakvog ponašanja skripta je zahtev *Navigator*a da sve oznake *img* imaju postavljene atribute *height* i *width* čak iako slike nisu u skriptu.

- tabele- za skriptove u *Navigator*u tabele predstavljaju problem; oznaka script unutar iznaka za ćeliju tabele <*td*> izaziva poteškoće. Rešenje je da se oznaka *script* koristi van ćelije tabele i da se za definisanje oznake <*td*> i njenog sadržaja koristi *document.write().* Ovo je neophodno samo kada su naredbe koje se izvršavaju potrebne za pravljenje sadržaja ćelije.

- ponovno učitavanje- ako je u dokumentu napravljena neka izmena s ciljem ispravke problema, ali problem je i dalje prisutan, rešenje je ponovno učitavanje datoteke iz menija *File*. Novo otvaranje datoteke potpuno briše staru datoteku iz memorije računara i učitava najnoviju verziju datoteke.

- ispravan rad- kada poruka o grešci ne daje pravu lokaciju problema u toku izvršavanja ili se jave greške u nepoznatoj tački važno je otkriti šta radi pravilno. Preporučuje se uklanjanje svih skriptova osim onog koji poziva *onLoad=.* Tako je moguće utvrditi da li je problem možda samo *HTML*. Drugo rešenje je postavljanje *alert box*-a na početku svih grupa naredbi koje se izvršavaju (npr. *alert*("ovde- 1")). Ova upozorenja dalje postavljati sve dublje dok se ne desi greška.

- provera izračunavanja- za proveru izračunavanja najbolji alat je *JavaScript Debugger*[*]. Pored *Debugger* postoje i druge tehnike. Kako većina problema sa izvršavanjem potiče iz funkcija, pretragu treba početi od vrha funkcije. Pre svake dodele svojstva objekta promenljivoj ili poziva metoda objekta *string, Math* ili *Date* ubaciti *alert box*. Kada treba proveriti vrednost izraza u toku izvršenja petlje ili kroz dugačak niz naredbi skripta, bolje je usput ispisivati vrednosti. Sa dobro smeštenim pozivima metoda

- *trace (flag, label, value),* problem će biti brzo lociran. Prvi parametar, *flag*, logička vrednost koja određuje da li pretragu treba nastaviti (*true*) ili ne. Zatim, *label*, znakovni niz koji služi za prepoznavanje vrednosti koja se prati. A treći parametar, *value*, je vrednost koju treba prikazati.

I pored tehnika za otklanjanje grešaka i pored alata za izradu *Web* stranica, ipak je najvažnije izbeći greške.

Problemi koji se javljaju u ranoj fazi razvijanja *Web* stranice obično su strukturalne prirode. Stoga, dovoljno pažnje treba posvetiti *HTML* kodu stranice, zajedno sa definicijama obrazaca ako ih ima, pre nego što se umetne *JavaScript* kod.

U dugačkom kodu poruke o grešci mogu ukazati na red koji je dosta udaljen od onog u kome se greška zapravo nalazi. Radi uštede vremena koje bi se potrošilo na lociranje problema, zgodno je isprobati kod posle svake napisane celine.

Što se tiče izraza i povratnih vrednosti, bolje je ispitati ga u kontrolisanom i izolovanom okruženju, u zasebnom dokumentu. Takođe i funkcije- zahvalno je ispitati ih u zasebnom dokumentu sa minimalnim brojem elemenata korisničkog interfejsa.

Web stranice obogaćene *JavaScript*-om svakako zahtevaju više pažnje od strane njihovih kreatora od običnih *HTML* stranica. Samo dizajniranje *HTML* srtanice podrazumeva i predviđanje određenih akcija korisnika i, uopšte, ispitivanje sposobnosti da se korisnik uspešno kreće po datoj *Web* lokaciji. Ako sam dizajner, programer, zna ispravnu sekvencu unosa podataka i kog tipa podataka treba da su unete informacije, ne mora da znači da će korisnik baš to i uraditi. Na primer, ako skript očekuje samo numeričku vrednost, a korisnik unese slovo ili ne unese ništa; ili ako korisnik prekine učitavanje dokumenta usred preuzimanja sa servera, kako će skript reagovati? Ovakve greške su nekad pripadale grupi "korisničkih grešaka", ali danas sve takve greške *Web* korisnici pripisuju programerima. Korisnici očekuju jednostavan rad i kretanje po *Web* stranici, tako da je za jednu *Web* lokaciju, a i njenog programera- dizajnera, svakako dobro preduhitriti potencijalne probleme.

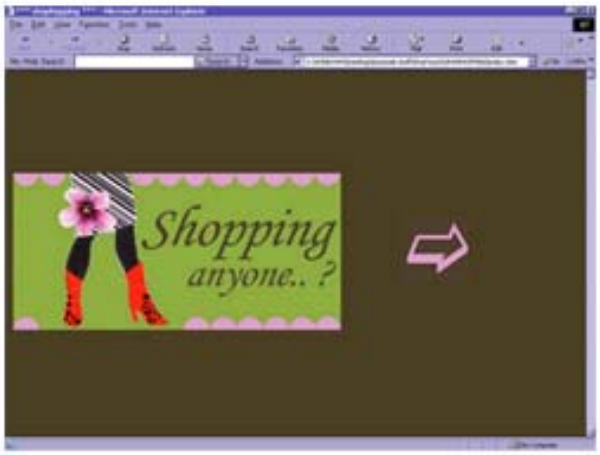
[*] JavaScript Debugger može se preuzeti sa http://developer.netscape.com

Poglavlje 7

Primer upotrebe *JavaScript*-a

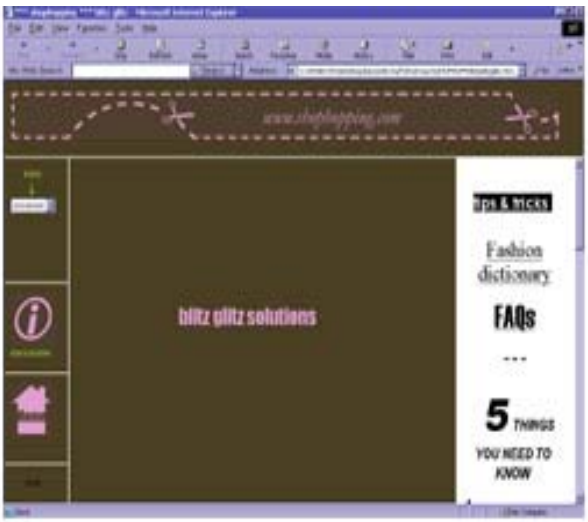
Polazna stranica *Web* prezentacije pored skripta za promenu slike kotrljanjem miša sadrži i skript za pravljenje *cookie*-ja. Od korisnika se traži da unese ime, i pri sledećoj poseti dočekuje ga pozdravna poruka.

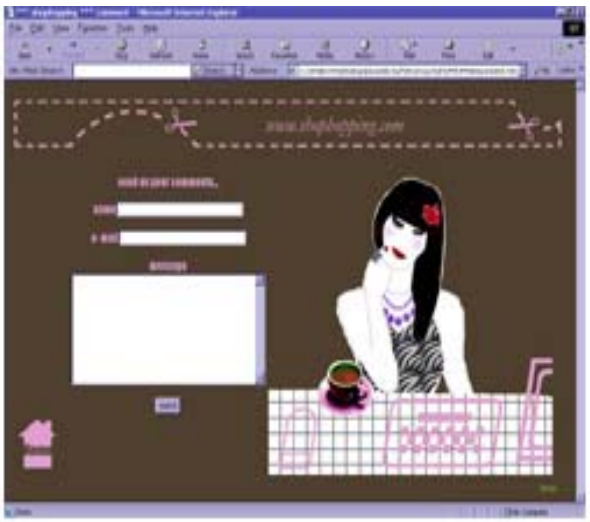
Stranica *Home* osmišljena je kao glavna stranica prezentacije i raskrsnica svih ostalih sadržajnih celina.



Sa *Home* stranice, pritiskom na neki linkova *clothes*, *accessories* ili *shoes*, dobija se ovakav prozor sa potpodelom. A pritiskom na dugme određeno za željenu podgrupu, ulazi se u zaseban *slide show* (*Clothes/Jackets*).

Celina *BlitzGlitz* je stranica koja se sastoji od mnoštva okvira koji međusobno komuniciraju. Stranica odabrana iz padajućeg menija otvara se u centralnom okviru. Stranica predviđena za korisnike tj. njihov odziv i komentare- *Comments*, sadrži klasičnu formu obrasca kao i skript za proveru unetih podataka.





Sa stranica Home i BlitzGlitz moguće je otvoriti prozor StoreLocator koji sadrži informacije o lokacijama određenih butika u gradu, a klikom na simbol oka otvara se i mapa grada.

Poglavlje 8

Zaključak

JavaScript postoji već dugi niz godina, u različitim oblicima i formama. Najveći doprinos *JavaScript*-a *Web design*-u je dinamičnost i veća interaktivnost između *Web* stranica i njihovih posetilaca. Skriptovi omogućavaju sadržaj da "reaguje" na određene događaje i "ponaša" se u skladu sa različitim skriptovima u različitim scenarijima.

Sledeći korak, nakon savladavanja *JavaScript*-a, svakako je *HTML DOM* i već pomenuti D*HTML*, a ako postoje aspiracije ka serverskom programiranju, tada i *ASP- Active Server Pages*.

HTML DOM definiše standardan način za pristup i manipulaciju *HTML* dokumenata, ne zavisi od platforme i jezika tako da ga može koristiti bilo koji programski jezik kao što su Java, *JavaScript*.

D*HTML* predstavlja kombinaciju *HTML*-a, *CSS*-a i *JavaScript*-a, i kako mu i samo ime kaže- Dynamic *HTML*- mogućnost ubacivanja dinamike u običan *HTML* sadržaj je njegova ključna osobina.

Dok se skriptovi u *HTML* dokumentu izvršavaju na klijentu tj. *browser*-u, skriptovi pisani u *ASP*-u izvršavaju se na serveru. *ASP* omogućava dinamičko editovanje, menjanje i dodavanje sadržaja *Web* stranice, zatim odgovor na podatke predate u *HTML* obrascima, pristup bilo kojim podacima ili bazama podataka i povraćaj rezultata *browser*-u, takođe podešavanje *Web* stranice individualnim potrebama korisnika. *ASP* dokumenti se vode kao *plain HTML*, tako da se mogu pregledati u svakom *browser*-u.

Web se bori za gledaočevu pažnju svim vrstama informacija stvorenih pomoću računara. To uključuje sve što se na ekranu pojavi u obliku interaktivnog multimedijskog sadržaja.

Osvrćući se na sadržaj ovog rada, može se reći da je *JavaScript* jedan moćan instrument na polju *Web* dizajna, i zaključiti da milioni *Web* stranica sadrže Javascript iz prostog razloga što omogućava:

- bolji dizajn
- čitanje i pisanje *HTML* elemenata
- proveru unetih podataka pre slanja istih serveru
- prepoznavanje *browser*-a kao i samog korisnika, itd.

Kao takav, *JavaScript* je najpopularniji *scripting* jezik na internetu i, jednom savladan, piscima skriptova pruža neograničenu kreativnost.

Poglavlje 9

DODACI

9.1 Dodatak A - Rezervisane reči *JavaScript*-a

abstract	boolean	break	byte
case	catch	char	class
const	comtinue	debugger	default
delete	do	double	else
enum	export	extends	false
final	finallz	float	for
function	goto	if	implements
import	in	instanceof	int
inerface	long	native	new
null	package	private	protected
public	return	short	static
super	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
while	with		

9.2 Dodatak B - svojstva, metodi i procedure za obradu događaja objekata *JavaScript*-a

Objekat *window*

Svojstva	Metodi	Procedure za obradu događaja
closed	alert()	onBlur=
defaultStatus	back()	onDragDrop=
document	blur()	onFocus=
frames[]	captureEvents()	onLoad=
history	clearInterval()	onMove=
innerHeight	clearTimeout()	onResize=
innerWidth	close()	onUnload=
location	confirm()	
locationbar	disableExternalCapture()	
menubar	enableExternalCapture()	
name	find()	
onerror	handleEvent()	
opener	forward()	
outerHeight	home()	
outerWidth	moveBy()	
pageXOffset	moneTo()	
pageYOffset	focus()	
parent	open()	
personalbar	print()	
scrollbars	prompt()	
self	releaseEvents()	
status	resizeBy()	
statusbar	resizeTo()	
toolbar	routeEvent()	
top	scroll()	
window	scrollBy()	
	scrollTo()	
	setTimeout()	
	stop()	

Objekat *location*

Svojstva	Metodi	Procedure za obradu događaja
hash	assign()	(nema)
host	reload()	
hostname	replace()	
href		
pathname		
port		
protocol		
search		

Objekat *history*

Svojstva	Metodi	Procedure za obradu događaja
current	back()	(nema)
length	forward()	
next	go()	
previous		

Objekat *document*

Svojstva	Metodi	Procedure za obradu događaja
alinkColor	captureEvents()	(nema)
anchors[]	clear()	
applets[]	close()	
bgColor	getSelection()	
cookie	handleEvent()	
domain	open()	
embeds	releaseEvents()	
fgColor	routeEvent()	
forms[]	write()	
images[]	writeLn()	
lastModified		
layers[]		
linkColor		
links[]		
location		
referrer		
title		
URL		
vlinkColor		

Objekat *link*

Svojstva	Metodi	Procedure za obradu događaja
Target	(nema)	onClick=
Text		onDblClick=
X		onMouseDown=
Y		onMouseOut=
[svojstva objekta location]		onMouseOver=
		onMouseUp=

Objekat *anchor*

Svojstva	Metodi	Procedure za obradu događaja
name	(nema)	(nema)
text		
x		
y		

Objekat *image*

Svojstva	Metodi	Procedure za obradu događaja
border	(nema)	OnAbort=
complete		OnError=
height		OnLoad=
hspace		
lowsrc		
name		
src		
vspace		
width		
x		
y		

Objekat *area*

Svojstva	Metodi	Procedure za obradu događaja
Target	(nema)	onClick=
[svojstva objekta location]		onMouseOut=
		onMouseOver=

Objekat *layer*

Svojstva	Metodi	Procedure za obradu događaja
above	load()	onBlur=
background	moveAbove()	onFocus=
below	moveBelow()	onLoad=
bgcolor	moveBy()	onMouseOut=
clip. bottom	moveTo()	onMouseOver=
clip. left	moveToAbsolute()	
clip. right	resizeBy()	
clip. top	resizeTo()	
document		
left		
name		
pageX		
pageY		
parentLayer		
siblingAbove		
siblingBelow		
src		
top		
visibility		
zIndex		

Objekat *form*

Svojstva	Metodi	Procedure za obradu događaja
action	handleEvent()	onReset=
elements[]	reset()	onSubmit=
encoding	submit()	
length		
method		
name		
target		

Objekat *text*, *password* i *textarea*

Svojstva	Metodi	Procedure za obradu događaja
defaultValue	blur()	onBlur=
form	focus()	onChange=
name	handleEvent()	onFocus=
type	select()	onKeyDown=
value		onKeyPress=
		onKeyUp=
		onSelect=

Objekat *button*, *submit* i *reset*

Svojstva	Metodi	Procedure za obradu događaja
name	handleEvent()	onClick=
type	click()	onMouseDown=
value		onMouseUp=

Objekat *select*

Svojstva	Metodi	Procedure za obradu događaja
length	blur()	onChange=
name	focus()	onFocus=
options[]	handleEvent()	onBlur=
selectedIndex[]		
options[i].defaultSelected		
options[i].index		
options[i].selected		
options[i].text		
options[i].value		
type		

Objekat *navigator*

Svojstva	Metodi	Procedure za obradu događaja
appName	javaEnabled()	(nema)
appCodeName	preferance()	
appVersion	taintEnabled()	
language		
mimeTypes[]		
platform		
plugins[]		
userAgent		

Objekat *string*

Svojstva	Metodi	Procedure za obradu događaja
length	anchor()	(nema)
prototype	big()	
	blink()	
	bold()	
	charAt()	
	charCodeAt()	
	concat()	
	fixed()	
	fontcolor()	
	fontsize()	
	fromCharCode()	
	indexOf()	
	italics()	
	lastIndexOf()	
	link()	
	match()	
	replace()	
	search()	
	slice()	
	small()	
	split()	
	strike()	
	sub()	
	substr()	
	substring()	
	sup()	
	toLowerCase()	
	toUpperCase()	

Objekat *Math*

Svojstva	**opis**	**Metodi**	**opis**
Math. E	Ojlerova konstanta	Math. abs(vr)	apsolutana vrednost
Math. LN2	prirodni algoritam broja 2	Math. acos(vr)	arkus kosinus u rad*
Math. LN10	prirodni algoritam broja 10	Math. asin(vr)	arkus sinus u rad
Math. LOG2	logaritam ode osnoce 2	Math. atan(vr)	arkus tangens u rad
Math. LOG10E	logaritam od e osnove 10	Math. atan2(vr1,vr2)	ugao polarnih koordinata
Math. PI	pi	Math. ceil(vr)	sledeci ceo jednak ili veci
Math. SQRT1_2	kvadratni koren broja 0.5	Math. cos(vr)	kosinus
Math. SQRT2	kvadratni koren broja 2	Math. exp(vr)	Ojlerova konst. na stepen
		Math. floor(vr)	sledeci ceo manji ili jednak
		Math. log(vr)	prirodni algoritam osn. e
		Math. max(vr1,vr2)	veca od vr1 ili vr2
		Math. min(vr1,vr2)	manja od vr1 ili vr2
		Math. pow(vr1,vr2)	vr1 na stepen vr2
		Math. random()	slucajni broj izmedu 0 i 1
		Math. round(vr)	n+1 kada je vr≥5; inace n
		Math. sin(vr)	sinus u rad
		Math. sqrt(vr)	kvadratni koren
		Math. tan(vr)	tangens u rad

* radijani

Objekat *Date*

Metod	opseg	opis
dateObj. getTime()	0 - ...	Milisekunde od 1/1/70 00:00:00 GMT
dateObj. getYear()	70 - ...	Naznačena godina minus 1900
dateObj. getMonth()	0 - 11	Mesec unutar godine
dateObj. getDate()	1 - 31	Dan unutar meseca
dateObj. getHours()	0 - 6	Dan u nedelji (nedelja = 0)
dateObj. getMinutes()	0 - 23	Sat dana u 24-časovnom vremenu
dateObj. getSeconds()	0 - 59	Minuti naznačenog dana
dateObj. setTime(val)	0 - 59	Sekunde u naznačenom minutu
dateObj. setYear(val)	0 - ...	Milisekunde od 1/1/70 00:00:00 GMT
dateObj. setMonth(val)	70 - ...	Naznačena godina minus 1900
dateObj. setDate(val)	0 - 11	Mesec unutar godine
dateObj. setDay(val)	1 - 31	Dan unutar meseca
dateObj. setHours(val)	0 - 6	Dan u nedelji (nedelja = 0)
dateObj. setMinutes(val)	0 - 23	Sat dana u 24-časovnom vremenu
dateObj. setSeconds(val)	0 - 59	Minuti naznačenog dana
dateObj. getTimezoneOffset()	0 - ...	Ostupanje minuta od GMT/UTC
dateObj. toGMTString()		Znakovni niz datuma u univerzalom formatu
dateObj. toLocalString()		Znakovni niz datuma u sistemskom formatu
Date. parse('dateString')		Pretvara znakovni niz u milisekunde
Date. UTC(date values)		Generiše datumsku vrednost iz GMT vrednosti

9.3 Dodatak C - spisak skriptova

Priložena su još dva skripta koja su deo primera *Web* prezentacije, a nisu prikazana u tekstu rada.

Skript koji prikazuje poruku dana- "*tip of the day*", a nalazi se na stranici Home:
Ovaj deo se piše u okviru *head* oznaka:

```
var msg = new Array();
dat = new Date();
today = dat.getDate();
msg[1] = "Tip 1";
msg[2] = "Tip 2";
msg[3] = "Tip 3";
msg[4] = "Tip 4";
msg[5] = "Tip 5";
msg[6] = "Tip 6";
msg[7] = "Tip 7";
msg[8] = "Tip 8";
msg[9] = "Tip 9";
msg[10] = "Tip 10";
msg[11] = "Tip 11";
msg[12] = "Tip 12";
msg[13] = "Tip 13";
msg[14] = "Tip 14";
msg[15] = "Tip 15";
msg[16] = "Tip 16";
msg[17] = "Tip 17";
msg[18] = "Tip 18";
msg[19] = "Tip 19";
msg[20] = "Tip 20";
msg[21] = "Tip 21";
msg[22] = "Tip 22";
msg[23] = "Tip 23";
msg[24] = "Tip 24";
msg[25] = "Tip 25";
msg[26] = "Tip 26";
msg[27] = "Tip 27";
msg[28] = "Tip 28";
msg[29] = "Tip 29";
msg[30] = "Tip 30";
msg[31] = "Tip 31";

function writeTip() {
      var msgf=("<i><font size=3 face=monotype corsiva color= #e09fce>"

                                    +msg[today]+"</font></i>")
      document.write(msgf);
}
```

a zatim na mestu gde želimo da poruka bude prikazana (u oznakama *body*):

```
<script>writeTip()</script>
```

Drugi skript je popularni "*send this page to a friend*", i stavlja se u odeljak *body* na željenu poziciju:

```
<script language= "javascript">
    var subject='Hey, take a look at what I found, '+top.document.title;
var text='You can see this page at: '+top.location.href;
var message=('Send <br>
<A HREF="mailto:?SUBJECT='+(subject)+'&BODY='+(text)+'">this
page</A> <br>
to a friend.')
mes=("<i><font size=3 face=monotype corsiva color=#e09fce>"
+message+ "</font></i>")
document.write(mes);
</script>
```

Literatura

- Goodman, D., *JavaScript* Biblija- prevod trećeg izdanja, Mikroknjiga, Beograd, 2000.
- Negrino, t., Smith, D., *JavaScript* For The *WWW*
- Watt, A., Watt, J., Teach Yourself *JavaScript* in 21 Days, Sams Publishing, Indianapolis, IN 2002.

Biografija

Milica Perišić je rođena 21.septembra 1981. godine u Novom Sadu, od oca Đorđa i majke Ljubice. Školovanje je započela u osnovnoj školi "Jovan Popović" a nastavila u gimnaziji "Laza Kostić".

Smer daljeg školovanja odredila je potreba za ličnom identifikacijom i slobodnim izrazom svakog mladog čoveka, kao i saznanje o rastućem značaju informatike u svim oblastima ljudskog postojanja i delovanja. Tako, 2000. godine započinje studije na Prirodno-matematičkom fakultetu u Novom Sadu, smer diplomirani informatičar na Depratmanu za matematiku i informatiku.

UNIVERZITET U NOVOM SADU
PRIRODNO MATEMATIČKI FAKULTET
KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj:
RBR
Identifikacioni broj:
IBR
Tip dokumentacije: Monografska dokumentacija
TD
Tip zapisa: Tekstualni štampani materijal
TZ
Vrsta rada: Diplomski rad
VR

Autor: Milica Perišić
AU
Mentor: dr Zoran Budimac
MN
Naslov rada: *JavaScript* u *Web design*-u
NR

Jezik publikacije: srpski (latinica)
JP
Jezik izvoda: s/en
JI
Zamlja publikovanja: Republika Srbija
ZP
Uže geografsko područje: Vojvodina
UGP
Godina: 2007
GO

Izdavač: autorski reprint
IZ
Mesto i adresa: Novi Sad, Trg D. Obradovića 4
MA
Fizički opis rada: (9/82/0/23/20/0/0)
(broj poglavlja/strana/lit.citata/tabela/slika/grafika/priloga)
FO
Naučna oblast: Računarske nauke
NO
Naučna disciplina: *Web design*
ND
Predmetne odrednica, Ključne reči: *JavaScript*, *Web design*
PO
UDK

Čuva se:
ČU
Važna napomena: nema
VN
Izvod:
IZ

JavaScript je jezik opšte namene i najveće primene u *Web design*-u. Ugrađuje se u servere, razvojne alate i druge *browser*-e. *JavaScript* postoji već dugi niz godina, u različitim oblicima i formama. Najveći doprinos *JavaScript*-a *Web design*-u je dinamičnost i veća interaktivnost između *Web* stranica i njihovih posetilaca.
JavaScript je najpopularniji *scripting* jezik na internetu i, jednom savladan, piscima skriptova pruža neograničenu kreativnost.

Datum prihvatanja teme od strane NN veća: januar 2007.

DP
Datum odbrane: 3.04.2007.
DO
Članovi komisije:
(Naučni stepen/ime i prezime/zvanje/fakultet)
KO

Predsednik: dr Mirjana Ivanović, vanredni profesor Prirodno-matematičkog fakulteta u Novom Sadu
Član: dr Zoran Budimac, docent Prirodno-matematičkog fakulteta u Novom Sadu
Član: dr Miloš Racković, vanredni profesor Prirodno-matematičkog fakulteta u Novom Sadu