Uvod u JavaScript

Srđan Pantić <srdjan@pantic.co.yu>

mart 1997. godine

Uvod u JavaScript

Šta je to JavaScript?

Javascript je novi skript jezik za korišćenje u programiranju WWW prezentacija. Uveden je od strane *Netscapea* u *Netscape Navigator*, počevši od verzije 2.0, uporedo sa uvođenjem podrške za Javu. Brzo je postigao veliku popularnost i raširenost, toliku da je i sam *Microsoft* pored uvođenja svog skript jezika - *Visual Basic Script*, koji je kompatibilan sa *Visual Basicom* i *Visual Basic for Aplication*, uveo svoju verziju JavaScripta - JScript. Jedini razlog zašto *Microsoft* nije licencirao *JavaScript* kao takav jeste što nije dobio licencu.

Javascript se uključuje u sadržaj HTML dokumenta i omogućava unapređenje HTML strana sa interesantnim efektima. Na primer, korišćenjem JavaScripta je moguće odgovarati na akcije korisnika u samom WWW čitaču. Ono što je bitno zapamtiti jeste da JavaScript program može da se izvrašava samo u okviru WWW čitača i nigde drugde.

Najbolji način za upoznavanje sa jednim Internet orjentisanim jezikom, kakav JavaScript jeste, je baš na Internetu. Veliki broj primera, dokumentacije i svega drugog vezanog za JavaScript možete naći na Gamelanu, WWW serveru posvećenom Javi i JavaScriptu, `http://www.gamelan.com`. Izvornu dokumentaciju samog *Netscapea* možete naći na `http://home.netscape.com`.

U ovom dodatku opisujemo JavaScript verziju 1.1, koja je podržana od strane *Netscape Navigatora 3.0* i kasnijih. Microsoftova verzija JScripta je kompatibilna sa JavaScriptom 1.0 i na razlike između njih ćemo ukazati u tekstu.

Koje su razlike između Jave i JavaScripta?

Bez obzira na nazive, Java i JavaScript su različiti jezici. Oni predstavljaju dve različite tehnike programiranja na Internetu. Java je programski jezik. JavaScript je (kako mu i samo ime kaže) skript jezik. Razlika je u tome što sa Javom možete da kreirate prave programe, koji mogu da se izvršavaju potpuno nezavisno od WWW čitača (poput programa napisanih sa C++), ili Java aplete koje možete da pozivate iz HTML dokumenta i koji se dovlače preko mreže i onda izvršavaju u okviru vašeg WWW čitača.

Ipak, najčešće vam za unapređenje HTML dokumenata zaista ne treba "teška artiljerija" poput Jave, jer želite da napravite neki efekat brzo, ne zamarajući se pravim programiranjem. Tu dolazimo na JavaScript, koji je jednostavan za razumevanje i upotrebu. JavaScript jeste zasnovan na Javi, ali postoje i brojne razlike.

Poređenje Jave i JavaScripta

Java	JavaScript
Prevodi se na serveru pre izvršavanja u klijentu.	Interpretira ga WWW čitač - ne prevodi se.
Objektno-orijentisan jezik. Postoji podrška za objekte, klase, nasleđivanje...	Objektno-baziran jezik. Postoje ugrađeni objekti.
Apleti su odvojeni od HTML dokumenata (iz kojih se pozivaju)	Integrisano u HTML dokument
Promenljive se moraju deklarisati (jaka tipizacija)	Promenljive se ne moraju deklarisati (slaba tipizacija)
Statičko povezivanje: reference objekata moraju postojati u vreme prevođenja.	Dinamičko povezivanje: reference objekata se proveravaju u vreme izvršavanja.

Sigurne: ne mogu da pišu na hard disk.	Sigurne: ne mogu da pišu na hard disk.

Kako pokrenuti program pisan u JavaScriptu?

JavaScript bez problema izvršavaju *Netscape Navigator* (od verzije 2.0 gde je i uveden i to na svim platformama) i *Internet Explorer* (od verzije 3.0), ali će bez sumnje biti podržan i od ostalih manje važnih WWW čitača.

Da bi programirali u *JavaScriptu* morate imati osnovno poznavanje HTML-a, a ne samo *JavaScripta*.

Pozivanje programa unutar HTML dokumenta

Ceo program se smešta unutar HTML dokumenta, unutar SCRIPT elementa. Atribut elementa SCRIPT je LANGUAGE, kojim se tačno određuje o kom se skript jeziku radi. Danas postoje dva skript jezika na WWW: JavaScript (Netsape) i VisualBasic Script (Microsoft).

Primer 1 Jednostavni JavaScript program

```
<HTML>
<HEAD>
<TITLE>
Jednostavan JavaScript
</TITLE>
</HEAD>
<BODY>
  Tekst prikazan na uobičajen način.
<BR>

      <SCRIPT LANGUAGE="JavaScript">
        document.write("<B>Tekst prikazan iz JavaScripta.</B>")
      </SCRIPT>
</BODY>
</HTML>
```

JavaScript se unutar HTML dokumenata uvek mora naći unutar SCRIPT elementa. Obratite pri tome pažnju da ako HTML dokument sa JavaScriptom pokušamo da pogledamo sa nekim WWW čitačom koji nema ugrađenu podršku za JavaScript, gornji dokument će se nepravilno prikazati.

Čitač će, prema specifikaciji HTML jezika ignorisati sve oznake unutar "<" i ">" koje ne ume da tumači. Problem koji ovde imamo je što skript zapravo nije unutar "<" i ">". Stariji WWW čitači bi prema tome sam skript prikazali kao tekst što je efekat koji sigurno ne želimo da izvedemo. Da bi to izbegli, potrebno je sam skript stavljati u oznake komentara "<!--" i "-->".

Pravilno napisan, primer 1 izgledaće ovako:

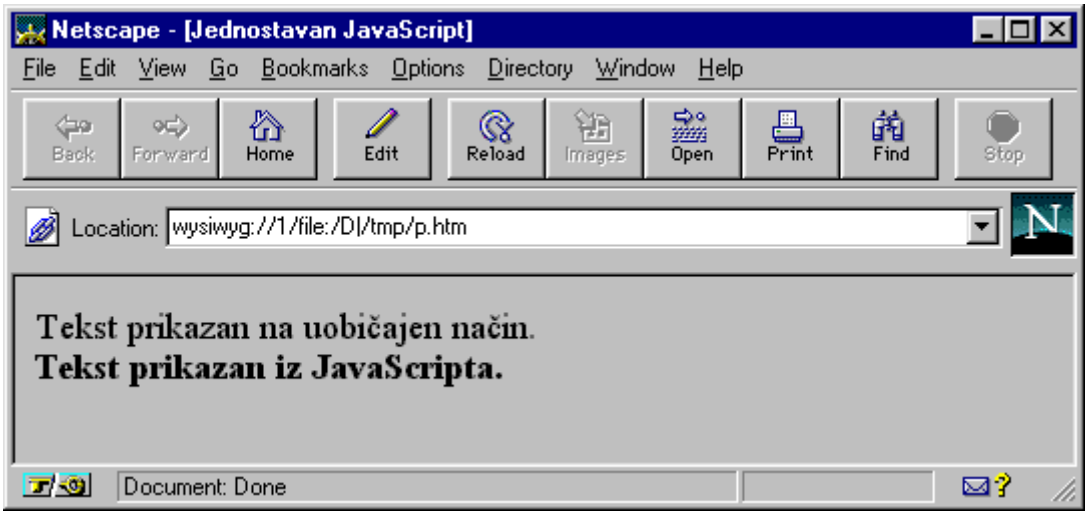
Primer 2 Jednostavni JavaScript program, koji neće smetati WWW čitačima koji ne podržavaju JavaScript

```
<HTML>
<HEAD>
```

```
<TITLE>
Jednostavan JavaScript
</TITLE>
</HEAD>
<BODY>
  Tekst prikazan na uobičajen način.
<BR>

      <SCRIPT LANGUAGE="JavaScript">
      <!-- Krije kod od starih WWW čitača

      document.write("<B>Tekst prikazan iz JavaScripta.</B>")

      // Kraj skrivanja koda -->
      </SCRIPT>
</BODY>
</HTML>
```

Komentar u sam skript stavljamo iza dvostruke kose crte (//), ili ih zatvaramo između "/*" i "*/".

Kada WWW čitač prikazuje HTML dokument on ide redom i kada naiđe na skript on će ga i izvršiti ako ima podršku za njega.



U gornjem primeru skript samo ispisuje u tekući dokument poruku. Objekat `document` ovde se odnosi na tekući dokument, a metodom `write` ispisujemo sam tekst.

Pozivanje eksternih skriptova

Uključivanje JavaScript programa direktno u HTML dokumenta je dobro rešenje za male skriptove i jednostavne HTML dokumente, ali lako može da izmakne kontroli kada je reč o velikim i kompleksnim skriptovima.

Zato je u takvim slučajevima najpogodnije rešenje čuvanje JavaScript programa u posebnom fajlu, odvojenom od HTML dokumenta i njegovom pozivanju od strane HTML dokumenta za koji je skript potreban. Na taj način isti skript možemo da koristimo u raznim HTML dokumentima, bez opterećivanja svih njih istim kodom.

Da bi to omogućili, uveden je atribut SCR za SCRIPT element. On je važeći od verzije 3.0 Netscape Navigatora. Fajl koji je naveden kao mesto u kome se nalazi JavaScript program se obavezno mora završavati sa ekstenzijom ".js".

```
<SCRIPT LANGUAGE="JavaScript" SRC="http://www.kombib.com/skript.js">
</SCRIPT>
```

Ako JavaScript program pozivate iz spoljnog fajla, najbolje mesto za SCRIPT element je u zaglavlju HTML dokumenta, kako bi se učitao odmah po preuzimanju HTML dokumenta.

Ova tehnika zahteva dodatni pristup HTTP serveru i dodatni njegov odgovor za svaki uključeni spoljni fajl, tako da predstavlja loše rešenje, ako se HTTP server nalazi na sporom linku ka Internetu i ako je veoma opterećen.

Tipovi podataka

JavaScript prepoznaje sledeće osnovne tipove podataka:

- *numbers*, celi i realni brojevi,
- *booleans*, mogu uzeti vrednost *true* ili *false*,
- *strings*, sadrže niz karaktera,
- *null*, koji predstavlja rezervisanu ključnu reč za null objekt i ne predstavlja zapravo tip podatka.

U JavaScriptu imena promenljivih se sastoje od sekvenci slova (a-z, A-Z), cifara (0-9) i donje crte (_). JavaScript razlikuje velika i mala slova.

U JavaScriptu ne moramo eksplicitno da deklarišemo promenljivu (mada je to dobra praksa) - JavaScript je slabo tipiziran jezik. Tipovi podataka će biti automatski konvertovani zavisno od mesta njihove upotrebe u programu. Tako na primer možemo definisati i inicijalizovati sledeću promenljivu

```
var promenljiva=42
```

a kasnije je možemo predefinisati, dodeljujući joj niz karaktera, na primer

```
promenljiva="Odgovor za sve tajne svemira je..."
```

Dobra je praksa definisati promenljivu sa `var`, kada je uvodimo.

Specijalna ključna reč **null** ukazuje da je promenljivoj dodeljena null vrednost. To ne znali da je promenljiva nedefinisana! Promenljiva je nedefinisana kada joj nije dodeljena nikakva vrednost i tada je ne možemo dodeliti drugoj promenljivoj ili koristiti u izrazima, a da ne dobijemo *run-time* grešku.

Primeri ispravnih promenljivih:

```
Broj_pogodaka

privremena90

_ime
```

Celi brojevi

Celi brojevi u JavaScriptu mogu biti predstavljeni u tri osnove: u decimalnom (baza 10), u oktalnom (baza 8) i heksadecimalnom (baza 16) formatu.

Decimalni celi brojevi se predstavljaju kao niz cifara (0-9) bez vodeće nule.

Oktalni celi brojevi se predstavljaju kao niz cifara (0-7) predvođen sa nulom (“0”).

Heksadecimalni celi brojevi se predstavljaju kao niz cifara (0-9) i slova (a-f i A-F) predvođen sa nulom koju sledi slovo x (“0x” ili “0X”).

Primer prestavljanja decimalnog celog broja deset (10) u tri brojna sistema:

- decimalnom: 10.
- oktalnom: 012.
- heksadecimalnom: 0xA.

Brojevi u pokretnom zarezu

Brojevi u pokretnom zarezu imaju sledeće delove: decimalni ceo broj, decimalnu tačku (“.”), deo iza decimalnog zareza (decimalni ceo broj), eksponent (“e” ili “E”, praćen decimalnim celim brojem).

Primeri brojeva u pokretnom zarezu:

- 3.1415
- .1E12
- -3.1E12
- 2E-10

Boolean

Promenljive ipa Boolean može da ima samo dve vrednosti: **true** (tačno) i **false** (netačno).

Stringovi

String je niz karaktera sačinjen od nula ili više karaktera zatvorenih u jednostrukim (‘) ili dvostrukim (“) navodnicima.

Primeri stringova:

- “string”
- ‘1234’
- “PRVA LINIJA \n druga linija”

Specijalni karakteri

U JavaScript stringovima možete koristiti sledeće specijalne znake:

- **\b**, *backspace*,
- **\f**, *form feed*,
- **\n**, *new line*,

- **\r**, *carriage return*,
- **\t**, *tab*.

Takođe je često neophodno koristiti navodnike unutar niza karaktera. To možete uraditi korišćenjem obrnute kose crte. Na primer:

```
var navod="<P>Sa radošću je čitao Tolkinovog \"Gospodara prstenova \". "
document.write(navod)
```

JavaScript izrazi i operatori

Izraz je ma koji, ispravan, skup konstanti, promenljivih, operatora i izraza koji se može dovesti do jedne vrednosti. Ta vrednost može biti broj (na primer, 7), string (`"Mikroračunari"`) ili logička vrednost (true ili false).

Operatori koje JavaScript podržava su poznati svima koji su imali prilike da rade u jeziku C ili C++. To su

- *standardni aritmetički operatori*: sabiranje (+), oduzimanje (-), množenje (*), delenje (/), moduo dva broja (deli15 % 5, vraća 2), inkrement (x++ i ++x), dekrement (x-- i --x) i unarna negacija (-x negira vrednost promenljive x).

- *logički operatori*: logičko I (a && a), logičko ILI (a || b), logička negacija (!a)

- *binarni operatori*: binarni logički operatori (I - a & b, ILI - a | b, ekskluzivno ili XOR - a ^ b) i binarni operatori pomeraja (levi pomeraj - `a << n`, desni pomeraj sa čuvanjem znaka - `a >> n` i desni pomeraj sa punjenjen nulama mesta najveće težine - `a >>> n`; pomeraj bitova promenljive a za *n* mesta ulevo ili udesno).

- *operatori poređenja*: jednako (`a==b`), nije jednako (`a!=b`), veće (`a>b`), veće ili jednako (`a>=b`), manje (`a<b`), manje ili jednako (`a<=b`).

- operatori stringova: konkatenacija, spajanje dva stringa (“ja” + “volim Tanju” nam vraća string ”ja volim Tanju”)

Dozvoljeni su i svi oblici skraćenog ispisa operacije i dodele poznati iz jezika C i C++:

- x+=y, što znači x=x+y
- x-=y, što znači x=x-y
- x*=y, što znači x=x*y
- x/=y, što znači x=x/y
- x%=y, što znači x=x%y
- x<<=y, što znači x=x<<y
- x>>=y, što znači x=x>>y
- x>>>=y, što znači x=x>>>y
- x&=y, što znači x=x&y
- x^=y, što znači x=x^y
- x|=y, što znači x=x|y

Kontrola izvršavanja programa

JavaScript podržava kontrolne strukture poznate većini programera iz drugih jezika, poput:

- *while* petlje, iz koje možemo izaći sa *break* (primer izračunava sumu svih brojeva od 0 zaključno sa 10)

```
{
  n=0;Suma=0
  while (n<=10)
      {
       n++;Suma+=n
      }
}
```

- *for* petlja (primer ispisuje svih šest nivoa naslova):

```
{
  for(i=1;i<=6;i++){
      document.write("<H"+i+"> Naslov nivoa " i "</H"+i+">")
  }
}
```

- *for .. in*, pomoću nje možemo proći kroz sve osobine (*properties*) nekog objekta. Koliko jedan objekat ima osobina, toliko puta će se izvršiti ova *for* petlja.

```
{
  for (prom in obj){
      proveri()
  }
}
```

- *break*, prekida izvršavanje *for* ili *while* petlje i nastavlja se izvršenje skripta posle petlje koja je prekinuta:

```
{
  while(i<5)
      {
      if(i==2){
            break}
      }
}
```

- *continue*, prekida izvršavanje bloka naredbi u petlji i nastavlja izvršavanje petlje u sledećoj iteraciji:

```
{
  while(i<5)
      {
      if(i==2){
            continue}
      }
}
```

- *if..else*, poznato svima odranije, može postojati samo u obliku *if(uslov){}* kao u ilustraciji naredbe *continue* ili *if(uslov){ } else { }*:

```
if (n>2){
```

```
        document.write("N je veće od 2")
}
else {
        document.write("N je manje ili jednako 2")
}
```

Rezervisane reči

Rezervisane reči iz ove liste ne mogu da se koriste kao nazivi promenljivih, funkcija, metoda ili objekata JavaScripta. Neke od ovih reči se koriste u JavaScriptu, dok su druge rezevisane za buduće korišćenje,

- abstract
- boolean
- break
- byte
- case
- catch
- char
- class
- const
- continue
- default
- do
- double
- else
- extend
- false
- final
- finally
- float
- for
- function
- goto
- if
- implements
- import
- in
- instanceof
- int
- interface
- long
- native
- new
- null
- package
- private
- protected
- public
- return
- short
- static
- super
- switch
- synchronized
- this
- throw
- throws
- transient
- true
- try
- var
- void
- while
- with

Osnove JavaScripta

Da bi smo mogli da počnemo sa pisanjem programa u JavaScriptu, potrebno je da smo upoznati sa osnovnom strukturom i sintaksom naredbi JavaScripta.

Naredbe i struktura JavaScripta veoma podsećaju na onu korišćenu u jezicima Java, C++ i C. JavaScript program je izgrađen iz funcija i naredbi, operatora i izraza. Osnovna jedinica je naredba ili izraz koji se završava sa tačkom-zarezom.

```
document.writeln("Ovo radi!<BR>");
```

Gornja komanda poziva `writeln()` metod, koji je deo `document` objekta. Tačka-zarez ukazuje na kraj komande. JavaScript komanda se može rasprostreti na više redova, sve dok se završava sa tačkom-zarezom. Isto tako, može se više naredbi naći u jednom redu, sve dok se završava sa tačkom-zarezom.

Komandni blokovi

Možemo grupisati naredbe u blokove naredbi, izdvojenih velikim zagradama:

```
{
  document.writeln("Da li ovo radi? ");
  document.writeln("Radi!<BR>");
}
```

Blokovi naredbi se koriste u definiciji funkcija i kontrolnim strukturama.

Izlaz iz JavaScripta

Jedna od osnovnih mogućnosti svakog programskog jezika je da pošalje tekst na izlaz. Ni JavaScript nije izuzetak. U JavaScriptu izlaz može biti preusmeren na nekoliko mesta uključujući trenutni prozor dokumenta i *pop-up* dijalog.

Osnovni izlaz je preusmeravanje teksta u prozor WWW klijenta, što se obavlja prosleđivanjem HTML koda. Rezultujući tekst će biti interpretiran kao HTML kod i prikazan u prozoru WWW klijenta. To ostvarujemo sa metodima `write` (šalje tekst u prozor WWW čitača bez pomeranja) i `writeln` (isto kao `write()`, s tim što se posle ispisa teksta kurzor pomera u sledeći red) objekta `document`:

```
document.write("Test");
document.writeln('Test');
```

Primer 3 Izlaz HTML elemenata iz JavaScripta

```
<HTML>
<HEAD>
<TITLE>Izlazni tekst</TITLE>
</HEAD>
<BODY>
Običan tekst.<BR>
<B>
<SCRIPT LANGUAGE="JavaScript">
<!--
    document.write("Boldiran tekst.</B>");
// -->
</SCRIPT>
</BODY>
</HTML>
```

Alert()

Jedna od velikih ograničenja HTML-a je što je dizajner sa HTML-om ograničen samo na jedan prozor za prikaz dokumenata. Ako bi iz čistog HTML-a želeo da pošalje potvrdu korisniku da je neki tekst ispravno unet u formi (ili slično) to bi zahtevalo prilično žongliranje.

Sa JavaScriptom je sve to veoma jednostavno.JavaScript ima mogućnost da kreira male dijaloge za ispis ili unos teksta koji su sadržajem potpuno nezavistan od sadržaja HTML dokumenta, a ničim isti ne remeti.

Najjednostavniji način da se prosledi kratka poruka korisniku je korišćenje `alert` metoda.

```
alert("Kliknite na OK za nastavak.");
```

Sigurno ste primetili da metod alert() nema ime objekta ispred sebe. Razlog za to je što pripada objektu windows, koji je u osnovi stabla objekata Navigatora i zato se uvek podrazumeva kada se negde ispred metoda naziv objekta ne navede.

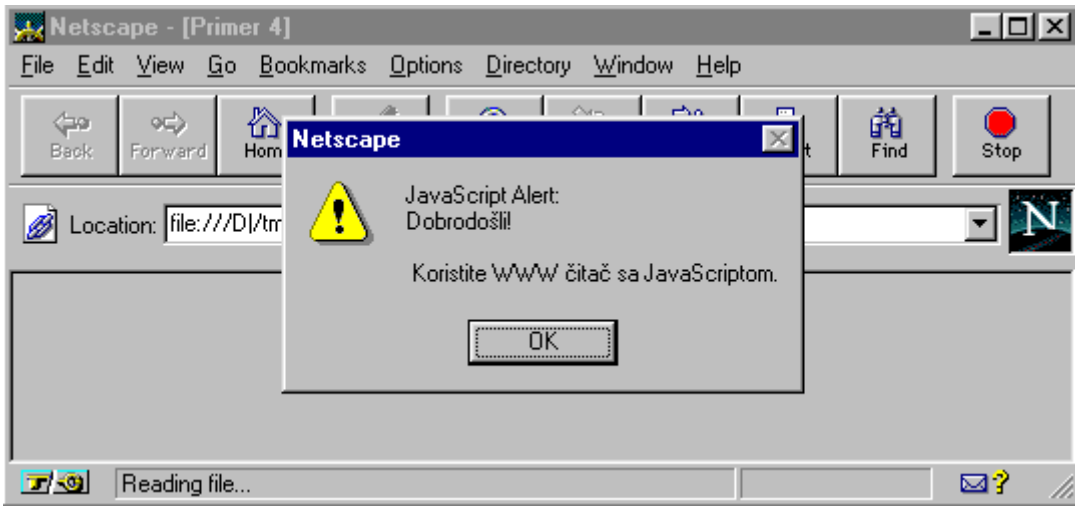
Znači, ”puno” ime za `alert()` metod bi bilo:

```
windows.alert("Kliknite na OK za nastavak.");
```

Generalno, `alert()` metod se koristi tačno za to - da upozori korisnika na nešto. Na primer, na netačno unet podatak u formu, pogrešan rezultat kalkulacije...

Primer 4 Prikaz poruke u dijalogu

```
<HTML>
<HEAD>
<TITLE>Primer 4</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
<!--
    alert("Dobrodošli!\n\n Koristite WWW čitač sa JavaScriptom.");
    document.write('<P>Sve najbolje</P>');
// -->
</SCRIPT>
</BODY>
</HTML>
```



Interakcija sa korisnikom

`Alert()` metoda nam omogućava da pošaljemo poruku korisniku, ali ne i da preuzmemo odgovor od njega.

Da bi smo omogućili jednostavnu interakciju sa korisnikom, koristimo metod `prompt()`. Poput `alert()` i `prompt()` kreira dijalog prozor u kome se može proslediti poruka korisniku, ali se formira i polje u koje korisnik može uneti proizvoljan tekst. Moguće je takođe da se predloži vrednost u tom polju, koju korisnik može prihvatiti ili ignorisati.

```
prompt("Unesite Vaše ime:","anonymous");
```

Ako gornji primer primenimo bez promene u našem programu, primetićemo da ne možemo da pristupimo informaciji u polju koju je uneo korisnik. Ta informacija se vraća kao rezultat metoda `prompt()`.

Da bi smo iskoristili tu informaciju moramo da dodelimo metod `prompt()` drugom metodu ili promenljivoj. Na primer:

```
document.write("Dobronam došli ");
document.writeln(prompt("Unesite Vaše ime:","anonymous"));
```

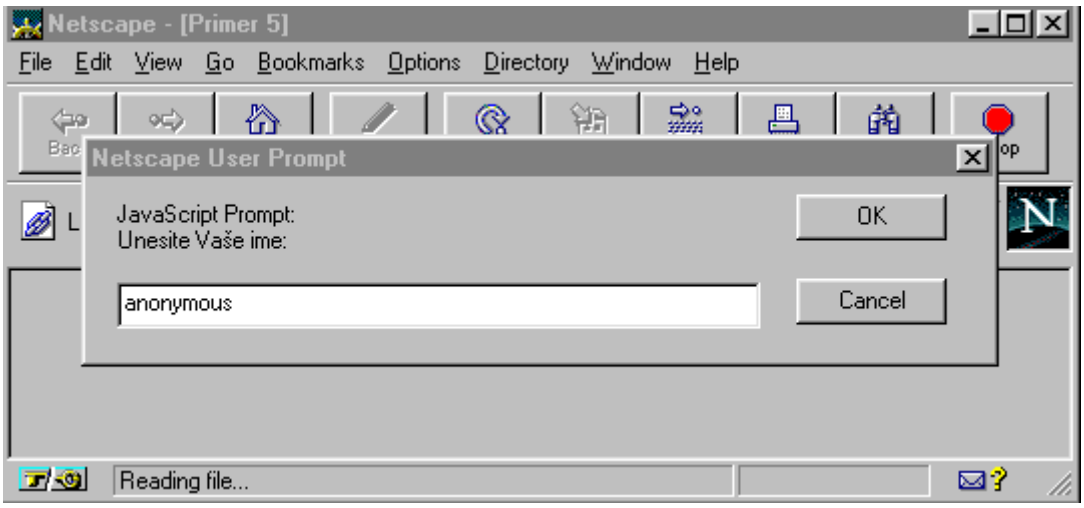
ili

```
ime_korisnika = prompt("Unesite Vaše ime:","anonymous");
```

Koristeći metod `prompt()` sad možemo da napišemo "ličniju" verziju programa iz primera 5.

Primer 5 Personalizovana verzija pozdravne poruke

```
<HTML>
<HEAD>
<TITLE>Primer 5</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
<!--

    document.write("<H1>Dobrodošli, ");
    document.writeln(prompt("Unesite Vaše ime:","anonymous"));
    document.write(" u svet JavaScripta!</H1>");
    document.write('<P>Čestitamo</P>');
// -->
</SCRIPT>
</BODY>
</HTML>
```



Funkcije

Funkcije predstavljaju jednu od osnova JavaScripta. Čim budete počeli da pišete složenije programe, brzo ćete otkriti potrebu da određene poslove i akcije izvršavate više puta tokom izvršavanja programa.

Tu do izražaja dolaze *funkcije*. Funkcije su slične metodima kod objekata (više o objektima možete naći kasnije u poglavlju, u delu *Objekti u JavaScriptu*), s tim što ne pripadaju nijednom objektu. Možete kreirati proizvoljan broj funkcija u programu.

Funkcija u JavaScriptu predstavlja skup naredbi koje obavljaju određen posao i koja može da vraća neku vrednost, bilo podatak ili objekt. Funcije mogu da imaju parametre koji se prenose po vrednosti.

Dopuštena je rekurzija. U HTML dokumentu možete koristiti sve funkcije koje su definisane u njegovom okviru.

Generalno se funkcija može smestiti bilo gde u okviru jednog HTML dokumenta, ali je dobra praksa grupisati sve funkcije u okviru zaglavlja HTML dokumenta, između elementa <HEAD> i </HEAD>. Tako će, kada korisnik učita HTML dokument, funkcije biti učitane prve. Unutar funkcije je moguće pozvati druge funkcije definisane unutar istog HTML dokumenta.

Funkcija se sastoji od rezervisane reči **function**, praćene sa imenom funcije, zatim listom parametara funkcije, odvojenih zarezima i zatvorenih u malu zagradu i skupom JavaScript naredbi koje čine telo funkcije zatvorenih u velike zagrade.

Tako možemo definisati funkciju koja neće vratiti nikakav podatak ili objekt, već će izvršiti određeni zadatak, u donjem slučaju odštampaće u HTML dokument neki tekst, čime se ostvaruje dinamičnost HTML dokumenta u opštem slučaju (ne mora biti isti svaki put kada ga pozovemo):

```
function stampaj(string){
      document.write("<P>" + string)
}
```

Prosleđivanje parametara

U gornjem primeru možete videti da funkcija `stampaj ()` ima argument, promenljivu `string`. Unutar funkcije, možemo pristupati toj promenljivi u kojoj je vrednost prosleđena funkciji.

Funkciji se po vrednosti mogu prosleđivati i promenljive i konstante.

Ukoliko se promenljiva prosledi funkciji, promena vrednosti parametra unutar funkcije ne menja vrednost promenljive koja je prosleđena funkciji.

Parametri postoje samo za života funkcije. Ukoliko zovete funkciju više puta, svaki put kada zovete funkcije imaćete "nove" parametre, odnosno vrednost koji će parametri imati na kraju prethodnog poziva funkcije neće biti sačuvana.

Na primer, kada funkciju `stampaj ()` pozovemo sa promenljivom `ime`:

```
var ime = "Tanja";
stampaj ();
```

kada funkcija počne da se izvršava, promenljiva `string` u funkciji će imati vrednost `"Tanja"`.

Isto tako, funkciju `stampaj ()` možemo da pozovemo sa konstantom:

```
stampaj ("Tanja");
```

kada funkcija počne da se izvršava, promenljiva `string` u funkciji će takođe imati vrednost `"Tanja"`.

Izvršavanje funkcija iz HTML dokumenta

To što smo definisali funkciju ne znači i da smo je izvršili. Moramo *pozvati* funkciju da bi smo je izvršili.

Pozivanje funkcije se može obaviti na više načina.

1. Iz druge funkcije ili JavaScript programa:

```
<SCRIPT LANGUAGE="JavaScript">
      stampaj("Ovo je tekst koji će se prikazati.")
</SCRIPT>
```

2. Kao reakciju na događaj:

```
<body OnLoad="stampaj("Ovo je tekst koji će se prikazati.")">
```

3. Izborom linka vezanog za događaj:

```
<a href="javascript:stampaj("Ovo je tekst koji će se
prikazati.");">Štampaj</a>
```

Vraćanje rezultata iz funkcije

Kao što je prethodno pomenuto, funkcija može da vraća rezultate.

Rezultat se iz funkcije vraća korišćenjem naredbe `return`. Koristi se za vraćanje bilo kog ispravnog izraza koji može da se izračuna u jednu vrednost. Na primer, u funciji `kub()`:

```
function kub(broj) {
  var kub = broj * broj * broj;
  return kub;
}
```

naredba `return` vraća vrednost promenljive `kub`. Istu funkciju smo mogli da napišemo i na sledeći način, preko vraćanja izraza:

```
function kub(broj) {
  return broj * broj * broj;
}
```

Izraz `broj * broj * broj` će se izračunati u jednu vrednost.

Prosleđivanje objekata funkciji

Parametri jedne funkcije nisu ograničeni samo na nizove karaktera i brojeve. Možete prosleđivati čitave objekte funkciji (više o objektima možete naći u sledećem delu poglavlja, u slučaju da ne razumete najbolje prosleđivanje objekata funkciji, pogledajte deo *Objekti u JavaScriptu*, pa se vratite ponovo ovom delu).

U gornjem primeru smo pozvali funkciju koja nije vratila nikakvu vrednost. U sledećem primeru možete videti upotrebu funcije kojoj se kao parametar prenosi čitav objekat, koja koristi *for.. in* petlju da bi prebrojala sva svojstva objekta i koja vraća kao rezultat niz karaktera u kome su navedena sva svojstva i njihove vrednosti.

Rezultat rada funkcije se vraća programu sa naredbom `return` čiji je parametar vrednost koja se vraća programu. Tako se u donjem slučaju vraća vrednost promenljive `rezultat`.

```
function vidi_svojstva(objekat, ime_objekta){
```

```
    var i, rezultat = ""

    for (i in objekat)
     rezultat += ime_objekta + ". " + i + " = " + objekat[i] + "\n"

  return rezultat
  }
```

Tako da ako funkciju pozovemo sa `vidi_svojstva(Kola, "Kola")`, gde su Kola objekat koji ima sledeća svojstva: proizvodjac, model i godina_proizvodnje (i oni imaju vrednosti `"Zastava", "YUGO 55" i 1990`) kao rezultat ćemo dobiti:

```
Kola.proizvodjac = Zastava
Kola.model = YUGO 55
Kola.godina_proizvodnje = 1990
```

Rekurzivnost funkcija

Funkcije mogu biti rekurzivne, odnosno mogu pozivati same sebe. Klasičan primer za demonstraciju rekurzije je naravno izračunavanje faktorijela nenegativnog broja:

```
function faktorijel(n){
   if ((n==0)||(n==1))
       return 1
   else {
       rezultat = (n * faktorijel(n-1))
       return rezultat
   }
}
```

Rekurzija funkcija je veoma moćna stvar, ali može biti i veoma opasna. Duboke rekurzije, čak iako nisu beskonačne, mogu prouzrokovati pucanjeWWW čitača.

Promenljive u funkciji

Primetite da smo u funciji `vidi_svojstva` definisali promenljive sa naredbom `var`. To otvara pitanje o dubini važenja promenljivih. Ako smo promenljivu definisali naredbom `var` u funkciji, oblast važnosti promenljive je ta funkcija.

Ako je promenljiva definisana van funcija, oblast njene važnosti je u celom skriptu, ona je globalna promenljiva.

Ako definišemo lokalnu promenljivu unutar funkcije sa istim imenom koju već ima neka globalna promenljiva, u toj funkciji će se kreirati nova verzija promenljive čija je oblast važenja funkcija, dok globalna promenljiva sa istim imenom neće promeniti vrednost.

Kada se argumenti prosleđuju funkciji, dva svojstva se kreiraju koja mogu biti korisna u radu sa argumentima: imefunkcije.arguments i imefunkcije.arguments.lenght. imefunkcije.arguments je niz čiji su elementi argumenti funkcije, a imefuncije.arguments.lenght je celobrojna promenljiva u kojoj se nalazi broj argumenata funkcije.

Ugrađene funkcije JavaScripta

JavaScript ima nekoliko funkcija ugrađenih u sam jezik. To su:

- `eval`,
- `parseInt`,
- `parseFloat`,
- `escape`,
- `unescape`,
- `isNaN`.

Eval

Ugrađena funkcija *eval* uzima string kao argument. String može da sadrži ma koji string koji predstavlja JavaScript izraz, naredbu ili sekvencu naredbi. Izraz može da uključuje promenljive i osobine postojećih objekata.

Ako argument predstavlja izraz, eval će izračunati izraz.

Ako argument predstavlja jednu ili više JavaScript naredbi, eval će izvršiti te naredbe.

Ova funkcija je korisna za transformisanje stringa koji predstavlja numerički izraz u broj. Na primer, ulaz u polje forme je uvek string, ali najčešće želimo da ga izarazimo kao broj, ako je to moguće.

ParseFloat

ParseFloat vraća vrednost izraženu kao broj u pokretnom zarezu. Sintaksa naredbe je

```
ParseFloat(string)
```

ParseFloat pokušava da parsira string i ako susretne ma koji karakter, osim znaka (+ ili -),decimalne tačke ili eksponenta, tada vraća vrednost koju je mogao da parsira do te tačke i ignoriše karakter na koji je naišao i sve karaktere koji ga slede. Ako prvi karakter nije mogao da se konvertuje u broj, vraća se *NaN* (0 na Windows platformi).

ParseInt

ParseInt vraća vrednost izraženu kao celobrojan broj. Sintaksa naredbe je

```
ParseInt(string, osnova)
```

ParseInt funkcija parsira svoj prvi argument i pokušava da vrati celobrojni broj u traženoj osnovi. Kada prvi put naleti na karakter koji nije cifra (cifra u traženoj osnovi, za osnovu 16, cifre su 0-9, A-F), vraća broj od prethodnih cifara i ignoriše karakter na koji je naletela i sve posle njega. Ako prvi karakter nije mogao da se konvertuje u broj, vraća se *NaN* (0 na Windows platformi).

Escape

Vraća string koji sadrži ASCII kod karaktera u obliku %xx (gde je xx numerički kod karaktera). Sintaksa mu je

```
escape(karakter)
```

isNaN

Testira vrednost da proveri da li je slučajno NaN. Funkcija je raspoloživa samo na UNIX paltformama, gde neke funkcije vraćaju NaN, ako im argument nije broj. Sintaksa je

```
isNaN(vrednost)
```

unescape

Vraća karakter zasnovan na ASCII kodu sadržanu u stringu. ASCII kod treba da uzme oblik "%integer" ili "heksadecimalnaVrednost". Sintaksa je

```
unescape(karakter)
```

Događaji u JavaScriptu

JavaScript programi su većinom pokretani događajima (*event-driven*). Događaji su akcije koje se pojavljuju, obično, kao rezultat nečega što korisnik uradi. Na primer, događaj je kliktanje mišem, kada elemenat forme dobije fokus i slično.

Koristeći obrađivače događaja (eng. *event handler*) ugrađene u JavaScript, mogu se pisati funkcije koje se aktiviraju kada se određeni događaj desi.

Obrađivači događaja su predstavljeni kao specijalni atributi koji modifikuju ponašanje HTML elemenata u okviru koji se nalaze. Na primer:

```
<body OnLoad="alert(Dobrodošli!)">
```

U gornjem primeru, oznaci početka tela HTML dokumenta dodat je atribut `OnLoad`, koji obrađuje događaj učitavanja HTML dokumenta. U konkretnom slučaju, kada se učitavanje HTML dokumenta završi, izvršiće se naredba JavaScripta dodeljena atributu `OnLoad`, u prozoru će se ispisati poruka “Dobrodošli”.

Generalno, sintaksa za upotrebu obrađivača događaja je:

```
<HTML_Oznaka Ostali_atributi eventHandler="JavaScript program">
```

Šta su to događaji?

Događaji su signali koji se generišu kada se odigra određena akcija. JavaScript može da detektuje te signale i mogu da se pišu programi koji reaguju na te događaje.

Događaji u WWW čitaču su kada korisnik klikne na hipertekst link, kada se promeni podatak u ulaznoj polji forme ili kada se završi učitavanje WWW dokumenta.

Događaji u JavaScriptu

Događaj	Opis	Obrađivač događaja
blur	Događa se kada se ulazni fokus prebaci sa elementa forme (kada korisnik klikne van polja forme).	`onBlur`
click	Događa se kada korisnik klikne na formu ili na link.	`onClick`
change	Događa se kada korisnik promeni vrednost u *text*, *textarea* ili *select* elementima forme.	`OnChange`
focus	Događa se kada polje forme dobije ulazni fokus.	`OnFocus`
load	Događa se kada se učitava HTML dokument u WWW čitač.	`OnLoad`
mouseover	Događa se kada korisnik pomera miša preko objekta. JavaScrip 1.1, JSCript nema ovaj događaj.	`OnMouseOver`
mouseout	Događa se kada miš pređe preko objekta.	`OnMouseOut`
select	Događa se kada korisnik izabere ulazno polje forme.	`OnSelect`
submit	Događa se kad je forma prosleđena HTTP serveru (kada korisnik klikne na dugme za slanje sadržaja forme).	`OnSubmit`
unload	Događa se kada korisnik napušta HTML dokument, pre nego što počne sa učitavanjem novog HTML dokumenta.	`OnUnload`

Događaje vezujemo za sledeće HTML elemente:

- Focus, Blur, Change događaji: tekst polja forme, textarea i selekcije,
- Click događaj: dugmad, linkove, polja za izbor, polja za selekciju,
- Select događaj: tekst polja forme, textarea,
- MouseOver događaj: linkovi.

Objekti JavaScripta i događaji vezani za njih

Objekt	Raspoloživ obrađivač događaja
Lista za izbor	onBlur, onChange, onFocus
Text element	onBlur, onChange, onFocus, onSelect
Textarea element	onBlur, onChange, onFocus, onSelect
Button element	onClick
Checkbox	onClick

Radio Button	onClick
Hypertext Link	onClick, onMouseOver
Reset Button	onClick
Submit Button	onClick
Document	onLoad, onUnload
Window	onLoad, onUnload
Form	onSubmit

Emuliranje događaje

Dosta objekata ima metode koji emuliraju događaje. Na primer, dugmad imaju click metod koji emulira dugme koje se pritisne. On neće "okinuti" `onClick` obrađivač događaja. Naravno, uvek možete direktno pozvati `onClick` iz programa.

To može biti posebno korisno ako želimo da prosledimo sadržaj forme bez da tražimo da korisnik klikne na dugme za prenos (*submit button*).

U JavaScriptu su raspoloživi sledeći emulatori događaja:

- `blur()`
- `click()`
- `focus()`
- `select()`
- `submit()`

JavaScript obrađivači događaja

JavaScript 1.1 trenutno podržava deset događaja da bi program mogao da odgovori na aktivnosti sistema i na akcije korisnika. Obrađivači događaja se mogu podeliti na dve kategorije: sistemski događaji i događaji uzrokovani akcijama miša.

Sistemski događaji

Sistemski događaji ne zahtevaju nikakvu akciju korisnika da bi se aktivirali. Na primer, to je signal da se HTML dokument učitao, da će trenutni HTML dokument biti izbačen, ili da je protekao određeni period vremena.

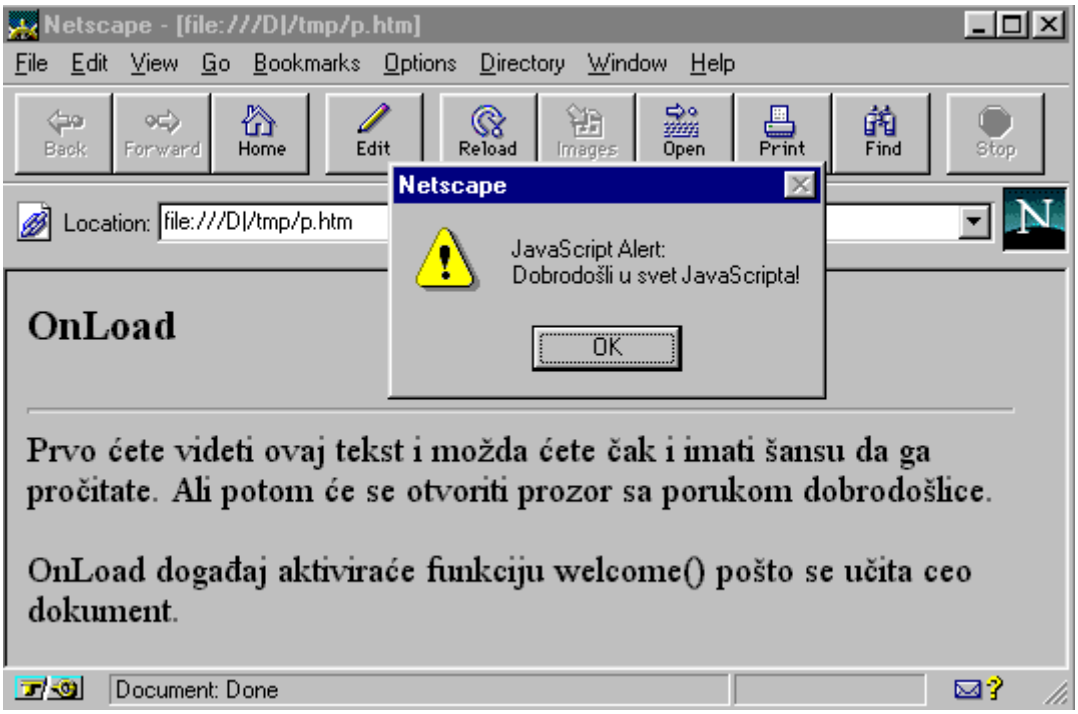
OnLoad

Ovaj događaje se aktivira pošto se tražen HTML dokument (i svi njegovi delovi) kompletno učita. Dodeljen je elementima `<BODY>` i `<FRAMESET>`. Kada radite sa okvirima, često je neophodno da budete sigurni da su neki okviri učitani, pre nego što druge obnovite ili promenite (na primer, u slučaju kada jedan okvir sadrži formu, čije elemente referencira drugi okvir).

S obzirom na prirodu Internteta, ne postoje garancije kada će se i u kom roku učitati neki dokument (opterećenje linka može biti veliko, ili isti može biti u prekidu), ali ako koristite OnLoad, možete biti sigurni da su svi ostali okviri učitani.

Primer 6 Primer upotrebe događaja OnLoad

```
<html>
<head>
<script language="JavaScript">
<!--
function welcome(){
      window.alert("Dobrodošli u svet JavaScripta!");
}
// -->
</script>
</head>
<body OnLoad="welcome()">
<h3>OnLoad</h3>
<hr>
Prvo ćete videti ovaj tekst i možda ćete čak i imati šansu da ga pročitate. Ali potom će se otvoriti prozor sa porukom dobrodošlice.
<p>
OnLoad događaj aktiviraće funkciju welcome() pošto se učita ceo dokument.
</body>
</html>
```



OnUnload

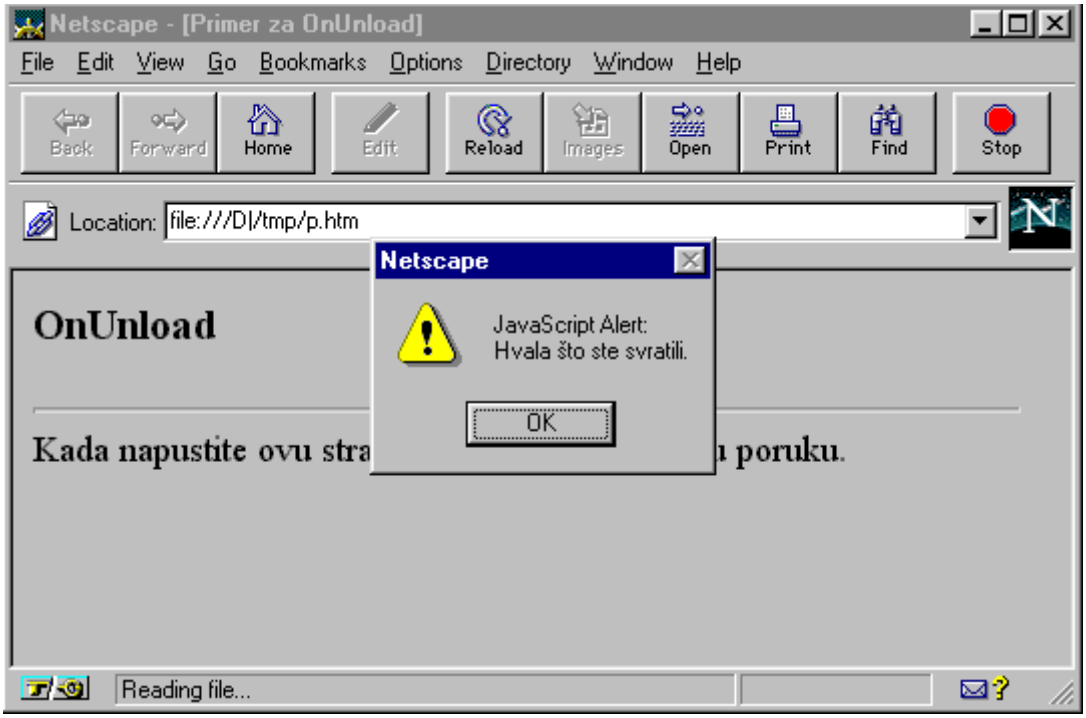
Događaj `OnUnload` je koristan za "čišćenje" posle posete nekoj prezentaciji. Tokom posete nekoj prezentaciji, korisniku se može desiti da ima otvoreno nekoliko prozora kao rezultat rada programa u JavaScriptu. Njih treba zatvoriti, a to se najlakše obalja obrađivanjem događaja `OnUnload`.

Primer 7 Primer upotrebe događaja OnUnload

```
<html>
```

```
<head>
<title>Primer za OnUnload</title>
<script language="JavaScript">
<!--
function dovidjenja(){
window.alert("Hvala što ste svratili.");
}
// -->
</script>
</head>
<body OnUnload="dovidjenja()">
<h3>OnUnload</h3>
<hr>
Kada napustite ovu stranicu dobićete pozdravnu poruku.
</body>
</html>
```

U ovom primeru, kada korisnik napušta HTML dokument (kliknuvši na link ili koristeći opcije iz WWW čitača, poput dugmadi Forward ili Back), OnUnload događaj aktiviraće `dovidjenja()` funkciju.



Događaji uzrokovani akcijama miša

Događaji uzrokovani akcijama miša zahtevaju akciju korisnika (rad sa mišem) da bi se aktivirali. Ovo su događaji koji se često koriste.

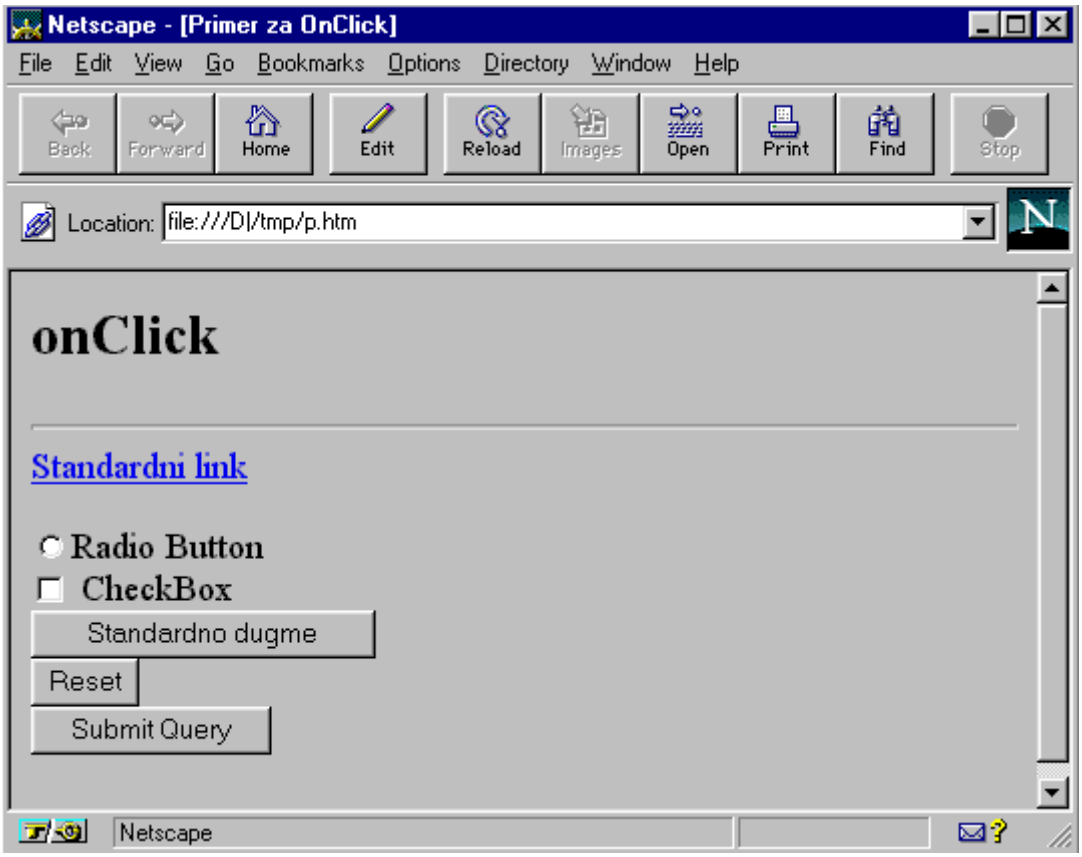
OnClick

Najčešće korišćeni događaj uzrokovan akcijom miša je OnClick. Ovaj događaj se aktivira uvek kada korisnik klikne na objekat koji prihvata takav događaj. Objekti koji prihvataju OnClick događaj su linkovi, *check box*-ovi i dugmad (uključujući *submit*, *reset* i *radio buttons*).

U primeru 8 je dat primer jednostavne forme koja odgovara na kliktaj jedne od komponenti forme.

Primer 8 Primer upotrebe događaja OnClick

```
<html>
<head>
<title>Primer za OnClick</title>
<script language="JavaScript">
<!--
function ziv_sam(){
window.alert("Aktiviran je onClick dogadjaj!");
}
// -->
</script>
</head>
<body>
<h2>onClick</h2>
<hr>
<a href="javascript:ziv_sam()">Standardni link</a>
<form method="POST" enctype=application/x-www-form-urlencoded>
<input name="radio1" type="RADIO" OnClick="ziv_sam()">Radio
Button<br>
<input name="checkbox1" type="CHECKBOX" OnClick="ziv_sam()">
CheckBox<br>
<input name="button1" type="BUTTON" OnClick="ziv_sam()"
value="Standardno dugme"><br>
<input type="RESET" OnClick="ziv_sam()"><br>
<input type="SUBMIT" OnClick="ziv_sam()"><br>
</form>
</body>
</html>
```



Primetite da se OnClick događaj koristi na ulaznom polju forme. Prozor sa porukom se aktivira svaki put kada korisnik klikne na stanradno dugme, *rese, submit* dugmad, radio dugmad, *checkbox*.

OnFocus

Događaj *Focus* se pojavljuje kada objekat postane predmet u fokusu, ili uprošćeno rečeno "dođe u centar pažnje". To se događa kada korisnik klikne mišem na određeni objekat, ili tamo dođe pritiskajući taster **tab**.

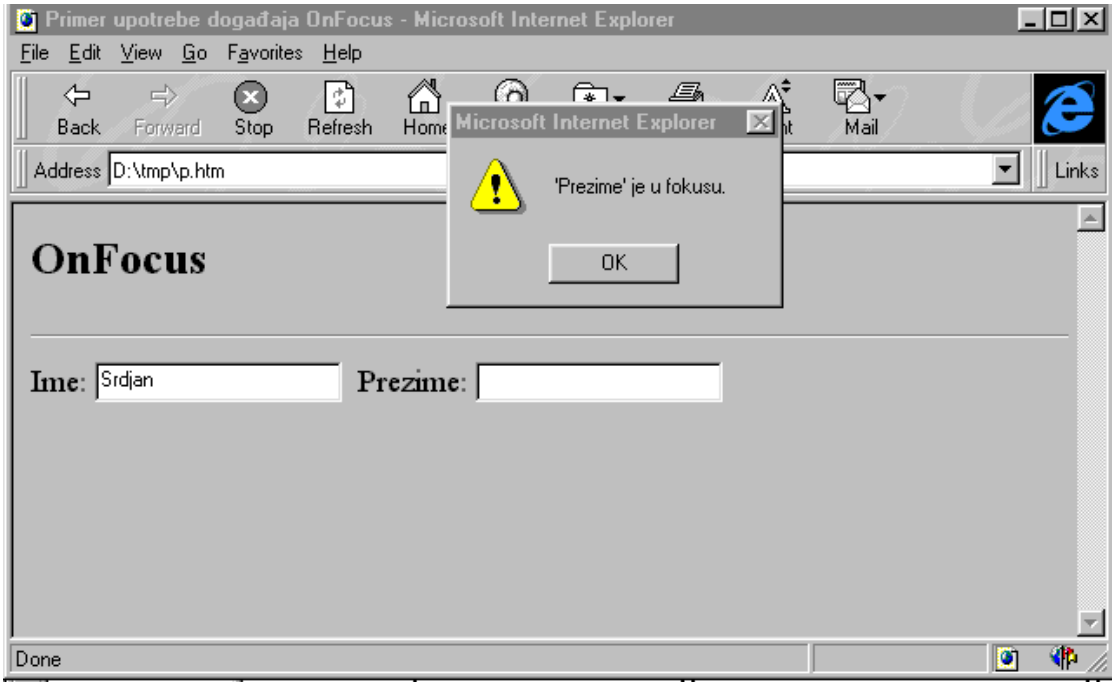
Ako korisnik može da unese podatke u objekat (ili da promeni trenutni izbor za slučaj lista izbora), tada objekat ima fokus.

Obrađivač događaja OnFocus se koristi samo sa objektima `text`, `textarea`, `password` i `select`.

U primeru 9 prikazujemo upotrebu događaja OnFocus sa jednostavnom formom koju čine ulazna polja Ime i Prezime. Događaj OnFocus je povezan da poljem Prezime i prikazuje prozor sa porukom kada korisnik uđe u to polje. Kako to izgleda u Netscape Navigatoru 3.0 možete videti na slici 7.

Primer 9 Primer upotrebe događaja OnFocus

```
<html>
<title>Primer upotrebe događaja OnFocus</title>
<head>
<script language="JavaScript">
<!--
function ima_fokus(){
window.alert("'Prezime' je u fokusu.");
}
// -->
</script>
</head>
<body>
<h2>OnFocus</h2>
<hr>
<form method="POST" enctype=application/x-www-form-urlencoded>
Ime: <input name="tekst1" type="TEXT">
Prezime: <input name="tekst2" type="TEXT" OnFocus="ima_fokus()">
</form>
</body>
</html>
```



Prozor sa porukom se pojavljuje kada polje Prezime dođe u fokus.

Primetite da se obrađivač događaja OnFocus nalazi u `<input>` oznaci za polje `tekst2`. Kada objekt dobije fokus, aktiviraće se funkcija `ima_fokus()`

OnBlur

Događaj *Blur* se pojavljuje kada objekt nije više u fokusu. To se može desiti sa prebacivanjem u drugi prozor ili aplikaciju, ili tako što kliknemom mišem na drugi objekt, ili što pomerimo fokus pritiskom na taster **tab**.

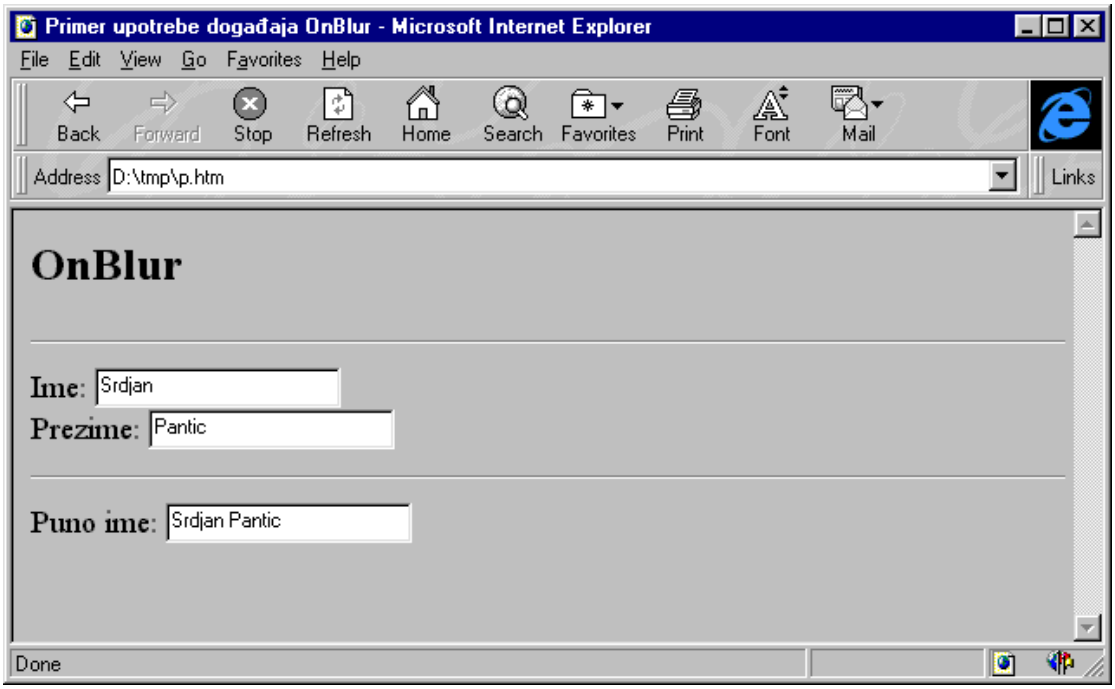
Obrađivač događaja OnBlur se koristi samo sa objektima `text`, `textarea`, `password` i `select`, baš kao i OnFocus.

Primer 10 Primer upotrebe događaja OnBlur

```
<html>
<title>Primer upotrebe događaja OnBlur</title>
<head>
<script language="JavaScript">
<!--
function dodaj(){
document.forma1.puno.value = (document.forma1.ime.value + " " +
   document.forma1.prezime.value);
}
// -->
</script>
</head>
<body>
<h2>OnBlur</h2>
<hr>
<form name="forma1" method="POST"
enctype=application/x-www-form-urlencoded>
Ime: <input name="ime" type="TEXT" OnBlur="dodaj()"><br>
```

```
Prezime: <input name="prezime" type="TEXT" OnBlur="dodaj()"><br>
<hr>
Puno ime: <input name="puno" type="TEXT">
</form>
</body>
</html>
```

U primeru 10 da bi ste formirali puno ime, potrebno je da unesete ime i prezime. Po unosu bilo imena, bilo prezimena, kada pomerite fokus na drugi objekat, promeniće se sadržaj polja "Puno ime",.



OnChange

Događaj `OnChange` se aktivira uvek kada objekat izgubi fokus i ako se njegova vrednost promenila.

Obrađivač događaja OnChange se koristi samo sa objektima `text`, `textarea`, `password` i `select`.

OnChange može biti veoma koristan kada želite da izračunate sumu unetih vrednosti na primer, ili kada na osnovu unetih podataka želite da date profakturu za neku robu, na primer.

Primer 11 Primer upotrebe događaja OnChange

```
<html>
<head>
<script language="JavaScript">
<!--
function kalkulacija(){
window.alert("Izračunavanje podataka.");
document.imena.total.value = document.imena.ime1.value + " " +
 document.imena.ime2.value + " " + document.imena.ime3.value
}
// -->
</script>
</head>
```

```
<body>
<h3>OnChange</h3>
<form  name="imena"  method="POST"  enctype=application/x-www-form-
 urlencoded>
Ime:
<input name="ime1" type="TEXT" size="10,1"
 OnChange="kalkulacija()">
<br>
Srednje ime:
<input name="ime2" type="TEXT" size="10,1"
  OnChange="kalkulacija()">
<br>
Prezime:
<input name="ime3" type="TEXT" size="10,1"
  OnChange="kalkulacija()">
<br>
<hr>
Total: <input name="total" type="TEXT" size="35,1">
</form>
</body>
</html>
```

Uvek kada se test promeni u jednom od ulaznih polja, OnChange događaj uzrokuje da se pokrene funcija `kalkulacija()` koja izračuna puno ime korisnika i upiše ga u polje `total`.

OnSelect

Događaj OnSelect se odnosi samo na `text`, `textarea` i `password` objekte i izvršava se kada korisnik označi deo teksta u jednom od nabrojanih tipova objekata.

Sintaksa bi mu bila:

```
<input name="mojobjekt" Type="TEXT" OnSelect="funkcija()">
```

OnSubmit

Događaj OnSubmit je povezan sa objektom forme. Najčešće je potrebno da se podaci uneti u polja forme, provere ili iskoriste pre prosleđivanja sadržaja forme HTTP serveru. Na taj način, moguće je smanjiti opterećenje samog HTTP servera, pošto se provera greški vrši na strani klijenta, te je redukovan broj izveštaja o greški koje vraća server, da se samo vreme obrade greški ni ne računa.

OnSubmit omogućava da se zaustavi prosleđivanje sadržaja forme, ako je to neophodno. Prihvata povratnu informaciju kao parametar. Ako JavaScript funkcija vrati vrednost *false*, sadržaj forme neće biti prosleđen HTTP serveru.

Primer 12 Primer upotrebe događaja OnSubmit

```
<html>
<head>
<script language="JavaScript">
<!--
function proveri(){
```

```
if (document.num_form.num1.value.length == 10) {
window.alert("Hvala što ste uneli samo 10 karaktera.");
return true;
} else {
window.alert("Unesite samo 10 karaktera, molim Vas.");
return false;
}
}
// -->
</script>
</head>
<body>
<h3>OnSubmit</h3>
<hr>
<form name="num_form" method="POST"
enctype=application/x-www-form-urlencoded
OnSubmit="return proveri(this)">
Unesite ime od pet karaktera:
<input name="num1" type="TEXT" size="10,1">
<br><hr>
<INPUT TYPE=SUBMIT value="Pošalji podatke!">
</form>
</body>
</html>
```

U ovom primeru proveravamo samo osnovne vrednosti, dužinu unetih podataka. Koristeći String metode možemo uraditi mnogo složenije provere sadržaja polja forme pre njenog slanja. Ako unete vrednosti u polje ne bi zadovoljili proveru, sadržaj polja neće biti prosleđen HTTP serveru.

Primetite da je u primeru korišćena naredba this. Ona se odnosi na tekući objekat i biće detaljnije objašnjena u poglavlju *Objekti u JavaScriptu*.

OnMouseOver

OnMouseOver događaj **ne zahteva** da se klikne mišem na neki objekat da bi se aktivirao. Kada je pointer miša iznad nekog objekta, događaj je okinut.

OnMouseOver se može upotrebiti za objašnjenje linkova ili dugmadi. Pogledajte kako u primeru 13. OnMouseOver događaj menja tekst u prozoru dobijenom primenom metoda `alert()`.

Primer 13 Primer upotrebe događaja OnMouseOver

```
<html>
<head>
<script language="JavaScript">
<!--
function objasni1(){
window.alert("Ovo je link ka EUnet Jugoslaviji.");
}
function objasni2(){
window.alert("Ovo je link ka matičnoj strani Mikroračunara.");
}
// -->
</script>
</head>
<body>
<h3>OnMouseOver</h3>
<hr>
<a href="http://www.EUnet.yu" OnMouseOver="objasni1()">
EUnet Jugoslavija</a><br><br>
```

```
<a href="http://www. Mikroracunara.co.yu"
OnMouseOver="objasni2()">
Mikroračunara </a><br><br>
</body>
</html>
```

Zavisno od toga preko kog linka pređemo mišem, prozor upozorenja će prikazati drugačiji podatak.

OnMouseOut

OnMouseOut događaj **ne zahteva** da se klikne mišem na neki objekat da bi se aktivirao. Kada pointer miša izađe iz područja nekog objekta, događaj je okinut.

OnMouseOut se često koristi u sprezi sa događajem OnMouseOver.

Primer 14 Primer upotrebe događaja OnMouseOut

```
<html>
<head>
<script language="JavaScript">
<!--
function objasni1(){
window.status("Ovo je link ka EUnet Jugoslaviji.");
}
function objasni2(){
window.status("Ovo je link ka matičnoj strani Mikroračunara.");
}
function izadji(){
window.status("");
}
// -->
</script>
</head>
<body>
<h3>OnMouseOut</h3>
<hr>
<a href="http://www.EUnet.yu" OnMouseOver="objasni1()"
OnMouseOut="izadji()">
EUnet Jugoslavija</a><br><br>
<a href="http://www.Mikroracunara.co.yu" OnMouseOver="objasni2()"
OnMouseOut="izadji()">
Mikroračunari</a><br><br>
</body>
</html>
```

Objekti u JavaScriptu

Objektno orijentisano programiranje (eng. *Object Oriented Programming*, skraćeno OOP) je, u osnovi, stil programiranjakod koga su povezani koncepti grupisani zajedno. Ukoliko imate, recimo, pet podataka i tri funkcije koje manipulišu nad tim podacima, tada te podatke i funkcije grupišete zajedno u generički kontejner poznat pod imenom *objekt*.

JavaScript takođe podržava objekte. Objekti sadrže podatke, *osobine* objekta (eng. *properties*), koji su JavaScript promenljive. Funkcije objekta su poznate kao metodi objekta (eng. *methods*).

JavaScript ima takođe određen broj ugrađenih funkcija.

Takođe, skoro svaki HTML element je dodeljen nekom JavaScript objektu. Ti objekti su razvrstani u logičkoj hijerarhiji koja ima oblik stabla direktorijuma.

Programer takođe može definisati svoje objekte.

Objekti i osobine

JavaScript objekt ima osobine koje mu pripadaju. Njima možete pristupiti preko sledećeg modela:

```
NazivObjekta.NazivOsobine
```

I u nazivu objekta i nazivu osobine objekta razlikujemo mala i velika slova.

Osobinu objekta definišemo tako što joj dodelimo vrednost. Na primer, pretpostavimo da imamo objekat po imenu `Gradjanin` (način kreiranja objekata biće objašnjen malo kasnije, sada samo pretpostavite da objekat već postoji).

Objektu `Gradjanin` možete dodeliti osobine pod nazivima `ime`, `ime_roditelja`, `prezime` i `maticni_broj` na sledeći način:

```
Gradjanin.ime=”Pera”;
Gradjanin.ime_roditelja=”Peroljub”;
Gradjanin.prezime=”Perić”;
Gradjanin.maticni_broj=”0311937710191”;
```

Ovo nije jedini način pristupa osobinama objekta.

Nizovi su skup vrednosti organizovan po brojevima kome je dodeljeno jedno ime promenljive. Osobine objekta i nizovi su u JavaScriptu direktno povezani, oni zapravo predstavljaju različit način pristupa istoj strukturi podataka.

Tako, na primer, osobinama objekta `Gradjanin` možete pristupiti i na sledeći način, potpuno ekvivalentan prethodnom primeru:

```
Gradjanin[”ime”]=”Pera”;
Gradjanin[”ime_roditelja”]=”Peroljub”;
Gradjanin[”prezime”]=”Perić”;
Gradjanin[”maticni_broj”]=”0311937710191”;
```

Takođe je moguće prići svakoj od osobina niza i preko indeksa. Tako je potpuno ekvivalentno gornjim načinima ekvivalentan i pristup preko indeksa:

```
Gradjanin[0]=”Pera”;
Gradjanin[1]=”Peroljub”;
Gradjanin[2]=”Perić”;
Gradjanin[3]=”0311937710191”;
```

Ovakav tip niza je poznat pod nazivom asocijativni niz, pošto je svakom indeksu niza dodeljen i tekstualni podatak.

Objeti i metodi

Metod je funkcija pridružena objektu. Programer definiše metod na isti način kao što definiše funkciju. Potom tu funkciju pridruži objektu na sledeći način:

```
NazivObjekta.NazivMetoda = NazivFunkcije
```

gde je *NazivObjekta* postojeći objekat, *NazivMetoda* je naziv koji dodeljujemo metodu, a *NazivFunkcije* je naziv funkcoje koju povezujemo sa objektom.

Potom možemo pozivati metod u kontekstu objekta kao:

```
NazivObjekta.NazivMetoda(parametri);
```

This

JavaScript ima specijalnu rezervisanu reč, **this**, koja se koristi za referenciranje na trenutni objekt.

Na primer, pretpostavimo da imamo funkciju `Proveri()` koja proverava da li se vrednost osobine objekta nalazi u propisanim granicama:

```
function Poveri(obj, donja, gornja){
      if((obj.value < donja)||(obj.value > gornja))
            alert("Pogrešna vrednost")
}
```

Tada možemo pozvati funkciju `Proveri()` u svakom elementu forme na obrađivač događaja `OnChange`, koristeći **this** da prosledimo element forme, kao u sledećem primeru:

```
<INPUT TYPE="text" NAME="godine" SIZE=3 OnChange="Proveri(this,18,77)">
```

Kreiranje novog objekta

JavaScript ima određeni broj ugrađenih objekata (o kojima će malo kasnije biti više reči), ali programer može kreirati svoje sopstvene objekte.

Kreiranje objekta se odvija u dva koraka:

1.Definisanje tipa objekta preko pisanja funkcije,
2.Kreirajući instancu objekta sa **new**.

Da bi definisali tip objekta moramo napisati funkciju koja određuje njegovo ime, njegove osobine i metode. Na primer, pretpostavimo da želimo da kreiramo tip objekta za građanine, koji smo već koristili na početku dela *Objekti u JavaScriptu*. Neka se objekat zove `osoba` i neka ima osobine `Ime`, `Srednje_ime`, `Prezime` i `Maticni_broj`.

Da bi to uradili potrebno je da napišete sledeću funkciju:

```
function osoba(ime, srednje_ime, prezime, MatBroj) {
    this.ime=ime;
    this.ime_roditelja = srednje_ime;
    this.prezime=prezime;
    this.maticni_broj=MatBroj;
}
```

Sada možemo da kreiramo objekat *osoba1* pozivajuči funkciju `osoba` preko **new**:

```
osoba1=new osoba("Pera", "Peroljub","Perić", "0311937710191");
```

Objektu možemo da dodamo i metode. Prvo ćemo napisati funkciju koju želimo da dodamo objektu. Recimo, funkcija `prikaziProfil()` koja ispisuje sve podatke o osobi:

```
function prikaziProfil() {
  document.write("Ime: " + this.ime + "<BR>");
  document.write("Srednje ime: " + this.ime_roditelja + "<BR>");
  document.write("Prezime: " + this.prezime + "<BR>");
  document.write("Matični broj: " + this.maticni_broj + "<BR>");
}
```

Sada ćemo ponovo napisati funkciju `osoba()` preko koje definišemo objekt.

```
function osoba(ime, srednje_ime, prezime, MatBroj) {
    this.ime=ime;
    this.ime_roditelja = srednje_ime;
    this.prezime=prezime;
    this.maticni_broj=MatBroj;
    this.prikaziProfil=prikaziProfil;
}
```

Kada definišemo objekat `osoba1`, funkciji `prikaziProfil()` se pristupa preko:

```
osoba1.prikaziProfil();
```

Objekat može da ima osobinu koja je sama objekat. Na primer, objekat tipa *osoba*, može u sebe uključivati osobine *otac* i *majka*, koje su i same objekat. I tako redom.

Primer 14 Primer upotrebe korisničkih objekata u JavaScriptu

```
<HTML>
<HEAD>
<TITLE>Primer 14</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
//Definicija metoda

function prikaziPodatke() {
  document.write("<H1>Profil zaposlenog: " + this.ime + "</H1><HR><PRE>");
  document.writeln("Broj zaposlenog: " + this.broj);
  document.writeln("Matični broj: " + this.matbroj);
  document.writeln("Trenutna plata: " + this.plata);
  document.write("</PRE>");
}

//Definicija objekta

function zaposleni() {
  this.ime=prompt("Unesite ime zaposlenog","Ime");
  this.broj=prompt("Unesite broj radnika za " + this.ime,"000-000");
  this.matbroj=prompt("Unesite matični broj za " +
     this.ime,"0000000000000");
  this.plata=prompt("Unesi trenutnu platu za" + this.ime,"$00,000");
  this.prikaziPodatke=prikaziPodatke;
}

noviZaposleni=new zaposleni();
// -->
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
<!--
```

```
noviZaposleni.prikaziPodatke();
// -->
</SCRIPT>
</BODY>
</HTML>
```

Ugrađeni objekti JavaScripta

JavaScript jezik sadrži sledeće ugrađene objekte:

- anchor
- button
- checkbox
- Date
- document
- *elements* array
- form (*forms* array)
- frame (*frames* array)
- hidden
- history
- link (*links* array)
- location
- Math
- navigator
- password
- radio
- reset
- select (*options* array)
- string
- submit
- text
- textarea
- window

Ti objekti i njihove osobine i metodi su ugrađeni u jezik. Programeri mogu pristupati tim objektima.

Ugrađeni objekti imaju osobine i metode.

Kreiranje nizova u JavaScriptu

JavaScript nema zapravo niz kao eksplicitan tip podataka, ali zbog veze izme]u nizova i osobina objekata o kojoj je već pre bilo reči, veoma je lako kreirati objekte u JavaScriptu.

Programer može da definiše niz kao objekat, na sledeći način:

```
function NapraviNiz(n) {
    this.duzina=n;
    for(var i=1;i<=n;i++){
          this[i]=0;}
    return this;
}
```

ova funkcija definiše niz tako da prva osobina, *duzina* (indeks nula), predstavlja broj elemenata niza. Preostale osobine imaju celobrojni indeks, od jedan pa više.

Možete zatim kreirati niz tako što ga pozovete, kao u sledećem primeru, određujući broj članova niza:

```
niz1= new NapraviNiz(10);
```

ovo kreira niz `niz1` sa 10 elemenata i inicijalizuje ih sve na nulu.

Referentni pregled objekata JavaScripta

anchor

Pogledajte *anchors* osobinu *document* objekta.

button

Objekat *button* se u JavaScriptu odnosi na dugme *push* iz HTML forme.

Osobine

- name Sadrži naziv *button* elementa.
- value Sadrži vrednost *button* elementa.

Metodi

- click() Emulira akciju kliktanja mišem na dugme.

Obrađivači događaja

- onClick Određuje JavaScript kod koji će se izvršiti kada bude kliknuto na dugme.

checkbox

Objekat *checkbox* omogućava da na nam *check box* element HTML forme bude dostupan iz JavaScripta.

Osobine

- checked Boolean vrednost koja ukazuje da li je *check box* element uključen.
- defaultChecked Boolean vrednost koja ukazuje da li je predloženo stanje *check box* elementa da je uključen (odnosi se na CHECKED atribut).
- name Sadrži naziv *check box* elementa.
- value Sadrži vrednost *check box* elementa.

Metodi

- click() Emulira akciju kliktanja mišem na *check box*.

Obrađivači događaja

- onClick Određuje JavaScript kod koji će se izvršiti kada bude kliknuto na *check box*.

Date

Objekat *Date* daje mehanizam za rad sa datumima i vremenom u JavaScript. Instance objekta se mogu kreirati sa

```
noviDatum = new Date(PodaciODatumu)
```

u ovom primeru, `PodaciODatumu` su opciona specifikacija određenog datuma i može biti u sledećim oblicima:

- mesec dan, godina čas:minut:sekunda
- godina, mesec, dan
- godina, mesec, dan, čas, minut, sekunda

U slučaju da `PodaciODatumu` nisu specificirani, novi objekt će biti predstavljen sa trenutnim datumom i vremenom.

Metodi

- getDate() Vraća dan u mesecu za trenutni Date objekt kao celobrojni broj od 1 do 31.
- getDay() Vraća dan u nedelji za trenutni Date objekt kao celobrojni broj od 0 do 6 (gde je 0 nedelja, 1 je ponedeljak i tako redom).
- getHours() Vraća sat iz vremena za trenutni Date objekt kao celobrojni broj od 0 do 23.
- getMinutes() Vraća minute iz vremena za trenutni Date objekt kao celobrojni broj od 0 do 59.
- getMonth() Vraća mesec iz datuma za trenutni Date objekt kao celobrojni broj od 0 do 11 (gde je 0 januar, 1 je februar i tako redom).
- getSeconds() Vraća sekunde iz vremena za trenutni Date objekt kao celobrojni broj od 0 do 59.
- getTime() Vraća vreme iz trenutnog Date objekta kao celobrojni broj koji predstavlja broj milisekundi proteklih od 1 januara 1970. godine od 00:00:00.
- getTimezoneOffset() Vraća razliku između lokalnog vremena i Griniča (GMT) kao celobrojni broj, koji predstavlja broj minuta.
- getYear() Predstavlja godinu iz trenutnog Date objekta kao dvocifarski celobrojni broj koji predstavlja godinu minus 1900.
- parse(dateString) Vraća broj milisekundi između 1. januara 1970. godine u 00:00:00 i vremena specificiranog u `dateString`. `dateString` može da uzme format

  Day, DD Mon YYYY HH:MM:SS TZN
  Mon DD, YYYY
- setDate(dateValue) Postavlja dan u mesecu u trenutni Date objekt. `dateValue` je celobrojni broj između 1 i 31.
- setHours(hoursValue) Postavlja sat u vreme trenutnog Date objekta. `hoursValue` je celobrojni broj između 0 i 23.
- setMinutes(minutesValue) Postavlja minute u vreme trenutnog Date objekta. `minutesValue` je celobrojni broj između 0 i 59.

- setMonth(monthValue) — Postavlja mesec za trenutni Date objekt. `monthValue` je celobrojni broj između 0 i 11 (gde je 0 januar, 1 je februar i tako redom).
- setSeconds(secondsValue) — Postavlja sekunde u vreme za trenutni Date objekt. `secondsValue` je celobrojni broj između 0 i 59.
- setTime(timeValue) — Postavlja vrednost u vreme za trenutni Date objekt. `timeValue` je celobrojni broj koji predstavlja broj milisekundi između 1. januara 1970. godine u 00:00:00.
- setYear(yearValue) — Postavlja godine za trenutni Date objekt. `yearValue` is je celobrojni broj veći od 1900.
- toGMTString() — Vraća vrednost trenutnog Date objekta u vremenu Griniča (GMT) kao string, koristeći Internet konvenciju u sledećoj formi:

  Day, DD Mon YYYY HH:MM:SS GMT
- toLocaleString() — Vraća vrednost trenutnog Date objekta u lokalnom vremenu koristeći lokalne konvencije.
- UTC(yearValue, monthValue, dateValue, hoursValue, minutesValue, secondsValue) — Vraća broj milisekundi proteklih od 1. januara 1970. godine u 00:00:00 o Griniču (GMT).

  - `yearValue` je celobrojna vrednost veća od 1900.
  - `monthValue` je celobrojna vrednost između 0 i 11.
  - `dateValue` je celobrojna vrednost između 1 i 31.
  - `hoursValue` je celobrojna vrednost između 0 i 23.
  - `minutesValue` i `secondsValue` je celobrojna vrednost između 0 i 59.

  `hoursValue`, `minutesValue` i `secondsValue` su opcioni.

document

Objekt *document* omogućuje rad sa atributima HTML dokumenta u JavaScriptu.

Osobine

- alinkColor — Boja aktivnog linka kao string ili heksadecimalni triplet.
- anchors — Niz *anchor* objekata po redu pojavljivanja u HTML dokumentu. Koristite `anchors.length` da dobijete broj linkova (*anchors*) u dokumentu.
- bgColor — Boja pozadine dokumenta.
- cookie — Sadrži *cookie* vrednost za trenutni dokument dokument.
- fgColor — Boja prednje strane dokumenta (*foreground*).
- forms — Niz objekata formi u redu pojavljivanja formi u HTML dokumentu. Koristite `forms.length` da dobijete broj formi u dokumentu.
- lastModified — Sadrži datum posledenje modifikacije dokumenta.
- linkColor — Boja linkova kao string ili heksadecimalni triplet.
- links — Niz objekata veza (*links*) u redu pojavljivanja veza u HTML dokumentu. Koristite `links.length` da dobijete broj veza u dokumentu. Linkovi i anchor objekti nisu jedno isto!
- location — Sadrži URL trenutnog dokumenta.
- referrer — Sadrži URL dokumenta iz koga je pozvan trenutni dokument, kada korisnik sledi link.
- title — Sadrži naslov trenutnog dokumenta.
- vlinkColor — Boja posećenih linkova kao string ili heksadecimalni triplet.

Metodi

- clear() Briše prozor za prikaz dokumenta.
- close() Zatvara trenutni ispis u prozor.
- open(mimeType) Otvara upis u prozor, dozvoljavajući *write()* i *writeln()* metode za upis u prozor dokumenta. `mimeType` je opcioni string koji određuje tip dokumenta podržan od strane WWW čitača ili plug-in (na primer, text/html, image/gif i tako redom).
- write() Ispisuje tekst i HTML kod u dokument.
- writeln() Ispisuje tekst i HTML kod u dokument praćeno znakom za novi red.

form

Objekat *form* omogućava rad sa HTML formama iz JavaScripta. Svaka HTML forma u dokumentuje predstavljena različitim pojavljivanje *form* objekta.

Osobine

- action Sadrži URL na koju se prosleđuje sadržaj (podaci) forme.
- elements Niz objekata za svaki element forme u redosledu pojavljivanje u formi.
- encoding Sadrži MIME kodiranje za formu koje je dodeljeno ENCTYPE atributu forme.
- method Sadrži metod prenosa podataka iz forme serveru.
- target Sadrži naziv prozora WWW čitača za koji su odgovori na prosleđivanje podataka iz forme upućeni.

Metodi

- submit() Prosleđuje sadržaj forme. Simulira događaj OnSubmit.

Obrađivači događaja

- onSubmit Određuje JavaScript kod koji se izvršava neposredno pre slanja sadržaja forme. Može da se vrati *true*, što dozvoljava da se sadržaj forme prosledi. Ako je vraćeno *false*, sprečava se slanje sadržaja forme.

.

frame

Objekat *frame* omogućava rad sa HTML okvirima iz JavaScripta.

Osobine

- frames — Niz objekata, svaki član niza je po jedan okvir u prozoru. Okviri su u nizu u redosledu pojavljivanja u izvornom kodu HTML dokumenta.
- parent — Sadrži naziv prozora koji sadrži definiciju okvira (*frameset*).
- self — Alternetiva za naziv trenutnog prozora.
- top — Alternetiva za naziv *top-most* prozora.
- window — Alternetiva za naziv trenutnog prozora.

.

Metodi

- alert — Prikazuje poruku u dijalog prozoru.
- close — Zatvara prozor.
- confirm(message) — Prikazuje poruku u dijalog prozoru sa OK i CANCEL dugmadima. Vraća *true* odnosno *false* zavisno od toga koje je dugme pritisnuto.
- open(url,name,features) — Otvara HTML dokument sa URL-a u prozoru čiji je naziv dat (name). Ako `name` ne postoji, otvara se novi prozor sa tim imenom. `features` je opcioni argument koji sadrži a listu osobina novog prozora. List osobina sadrži ma koji od sledećih parova naziv/vrednost odvojenih zarezima i bez dodatnih praznih mesta između:
  - toolbar=[yes,no,1,0]: ukazuje da li prozor treba da ima *toolbar*.
  - location=[yes,no,1,0]: ukazuje da li prozor treba da ima polje za prikazivanje trenutnog URL-a.
  - directories=[yes,no,1,0]: ukazuje da li prozor treba da ima dugmad direktorijuma.
  - status=[yes,no,1,0]: ukazuje da li prozor treba da ima *status bar*.
  - menubar=[yes,no,1,0]: ukazuje da li prozor treba da ima menije.
  - scrollbars=[yes,no,1,0]: ukazuje da li prozor treba da ima klizače prozora (*scroll bars*).
  - resizable=[yes,no,1,0]: ukazuje da li prozor treba da može da promeni veličinu.
  - width=pixels: daje širinu prozora u pikselima.
  - height=pixels: daje visinu prozora u pikselima.
- prompt(message,response) — Prikazuje poruku (*message*) u dijalog prozoru sa ulaznim tekst poljem sa predloženom vrednošću odgovora (*response*). Korisnikov odgovor je sadržaj tekst ulaznog polja koje se vraća kao string.
- setTimeout(expression,time) — Izvršava narednu posle određenog vremena, izraženog u milisekundama i sadržanog u `time`. Izlaz može da bude sledećeg oblika structure:

  name = setTimeOut(expression,time)
- clearTimeout(name) — Prekida brojač sa datim imenom.

.

hidden

Objekat *hidden* se odnosi na rad sa sakrivenim (*hidden*) poljem u HTML formi iz JavaScripta.

Osobine

- name — Sadrži naziv *hidden* elementa.
- value — Sadrži vrednost *hidden* elementa.

history

Objekat *history* omogućava skriptu da radi sa listom posećenih lokacija (*history list*) WWW čitača iz JavaScripta. Iz razloga bezbednosti i privatnosti, stvaran sadržaj liste nije dostupan iz JavaScripta.

Osobine

- length — An integer representing the number of items on the history list.

Metodi

- back() — Vraća se nazad na prethodni dokument u *history* listi.
- forward — Idemo napred na sledeći dokument u *history* listi.
- go(location) — Idemo na dokument u *history* listi određen sa `location`. `location` može biti string ili celobrojna vrednost. Ako je string, onda predstavlja deo ili celi URL iz *history* liste. Ako je celobrojna vrednost, `location` predstavlja relativnu poziciju dokumenta u *history* listi. Kao celobrojna vrednost, `location` može biti pozitivan ili negativan.

link

Objekat *link* odnosi se na hipertekst link u telu HTML dokumenta.

Osobine

- target — Sadrži naziv prozora ili okvira određenog u TARGET atributu.

.

Obrađivač događaja

- onClick — Određuje JavaScript kod koji će se izvršiti kada bude kliknuto na link.
- onMouseOver — Određuje JavaScript kod koji će se izvršiti kada miš pređe preko hipertekst linka.

location

Objekat *location* se odnosi na informacije o trenutnom URL-u.

Osobine

- hash — Sadrži *anchor* naziv u URL-u.
- host — Sadrži naziv računara i broja porta iz URL-a.
- hostname — Sadrži adresu računara (ili numeričku IP adresu) iz URL-a.
- href — A Sadrži ceo URL.
- pathname — Sadrži putanju iz URL-a.
- port — Sadrži broj porta iz URL-a.
- protocol — Sadrži protokol iz URL-a (uključujući i dvotačku, ali ne i crtu).
- search — Sadrži informacije koje se prosleđuju pri GET CGI-BIN pozivu (sve informacije posle znaka pitanja).

Math

Objekat *Math* obezbeđuje osobine i metode za napredna matematička izračunavanja.

Osobine

- E — Vrednost Ojlerove konstante (približno 2,718) koja se koristi kao baza prirodnih algoritama.
- LN10 — Vrednost prirodnog algoritma od 10 (približno 2,302).
- LN2 — Vrednost prirodnog algoritma od 2 (približno 0,693).
- PI — Vrednost PI (približno 3,1415).
- SQRT1_2 — Vrednost kvadratnog korena od jedne polovine (približno 0,707).
- SQRT2 — Vrednost kvadratnog korena od dva (približno 1,414).

Metodi

- abs(number) — Vraća apsolutnu vrednost broja.
- acos(number) — Vraća vrednost arkus kosinu broja u radijanima.
- asin(number) — Vraća vrednost arkus sinus broja u radijanima.
- atan(number) — Vraća vrednost arkus tangens broja u radijanima..
- ceil(number) — Vraća sledeći celobrojni broj veći od navedenog broja. Drugim rečima, vrši se zaokruživanje broja na sledeći celobrojni broj.
- cos(number) — Vraća kosinus ugla koji je dat u radijanima.
- exp(number) — Vraća vrednost eksponenta datog broja.
- floor(number) — Vraća prethodni celobrojni broj manji od navedenog broja. Drugim rečima, vrši se zaokruživanje broja na prethodni celobrojni broj.
- log(number) — Vraća se prirodni logaritam datog broja.
- max(number1,number2) — Vraća veći broj od dva data.

- min(number1,number2) Vraća manji broj od dva data.
- pow(number1,number2) Vraća vrednost `number1` dignut na `number2`.
- random() Vraća slučajan broj između nule i jedan.
- round(number) Vraća najbliži celi broj.
- sin(number) Vraća sinus ugla koji je dat u radijanima.
- sqrt(number) Vraća kvadratni koren broja.
- tan(number) Vraća tangens ugla koji je dat u radijanima.

navigator

Objekat *navigator* se odnosi na informacije o WWW čitaču koji se koristi.

Osobine

- appCodeName Sadrži kodni naziv klijenta (na primer, "Mozilla" za Netscape Navigator).
- appName Sadrži naziv klijenta (na primer, "Netscape" za Netscape Navigator).
- appVersion Sadrži informaciju o verziji klijenta u formi `versionNumber` (`platforma`; zemlja). Na primer, Navigator 2.0, beta 6 za Windows 95 (međunarodna verzija), ima `appVersion` osobinu "2.0b6 (Win32; I)".
- userAgent Sadrži kompletnu vrednost `user-agent` zaglavlja poslatog u HTTP zahtevu. Sadrži sve informacije koje se nalze i u `appCodeName` i `appVersion`: na primer, `Mozilla`/2.`0b6` (`Win32; I`)

password

Objekat *password* odnosi se na pristup tekst polju sa lozinkom HTML forme iz JavaScripta.

Osobine

- defaultValue Sadrži predloženu vrednost za *password* element (odnosno vrednost njegovog VALUE atributa).
- name Sadrži naziv *password* tekst polja forme.
- value Sadrži vrednost *password* tekst polja forme.

Metodi

- focus() Emulira akciju fokusiranja na *textarea* polje.
- blur() Emulira akciju uklanjanja fokusa sa *textarea* polja.
- select() Emulira akciju selektovanja teksta u *textarea* polju.

radio

The radio object reflects a set of radio buttons from an HTML form in JavaScript. To access individual radio buttons, use numeric indexes starting at zero. For instance, individual buttons in a set of radio buttons named testRadio could be referenced by testRadio[0], testRadio[1], and so on.

Osobine

- checked — Boolean vrednost koja ukazuje da li je određeno dugme uključeno. Može se koristiti za selektovanje ili deselektovanje dugmeta.
- defaultChecked — Boolean vrednost koja ukazuje da li je predloženo stanje određenog dugmeta da je uključen (odnosi se na CHECKED atribut).
- length — Celobrojna vrednost koja ukazuje na broj *radio* dugmadi u skupu.
- name — Sadrži naziv skup *radio buttons* elemenata.
- value — Sadrži vrednost *radio* dugmeta u skupu (odnosno VALUE atributa).

Metodi

- click() — Emulira akciju kliktanja mišem na *radio* dugme.

Obrađivači događaja

- onClick — Određuje JavaScript kod koji će se izvršiti kada bude kliknuto na *radio button*.

.

reset

Objekat *reset* se koristi za rad sa *reset* dugmetom u HTML formi iz JavaScripta.

Osobine

- name — Sadrži naziv *reset* dugmeta.
- value — Sadrži vrednost *reset* dugmeta (tekst koji se ispisuje na *reset* dugmetu).

Metodi

- click() — Emulira akciju kliktanja mišem na *reset* dugme.

Obrađivači događaja

- onClick Određuje JavaScript kod koji će se izvršiti kada bude kliknuto na *reset* dugme.

select

Objekat *select* se odnosi na rad sa selekcionim listama u HTML formi iz JavaScripta.

Osobine

- length Sadrži celobrojnu vrednost koja predstavlja broj opcija u listi izbora.
- name Sadrži naziv izbora sa liste.
- options Niz čiji su elementi svaka od opcija sa liste izbora poredu pojavljivanja u listi. Ova osobina ima svoje sopstvene osobine:
  - defaultSelected: Boolean vrednost koja ukazuje koja od opcija je predložena (odnosno odražava se SELECTED atribut).
  - index: Celobrojna vrednost koja označava indeks opcije.
  - length: Beoj opcija u listi izbora
  - name: Sadrži naziv liste izbora.
  - options: Sadrži puni HTML kod za listu izbora
  - selected: Boolean vrednost koja ukazuje da li je opcija izabrana. Može da se koristi za selekciju ili deselekciju opcije.
  - selectedIndex: Celobrojna vrednost koja predstavlja indeks trenutno izabrane opcije.
  - text: Sadrži tekst koji se prikazuje u listi izbora za određenu opciju.
  - value: Sadrži vrednost unetu za specificiranu opciju (odražava VALUE atribut).
- selectedIndex Odražava indeks trenutno izabrane opcije u listi izbora.

Obrađivači događaja

- onBlur Određuje JavaScript kod koji se izvršava kada lista izbora izgubi fokus.
- onFocus Određuje JavaScript kod koji se izvršava kada lista izbora dobije fokus.
- onChange Određuje JavaScript kod koji se izvršava kada se selektovana opcija promeni.

string

Objekat *string* daje osobine i metode za rad sa string konstantama i promenljivima.

Osobine

- length Celobrojna vrednost koja sadrži broj karaktera u stringu.

Metodi

• anchor(name)	Vraća string koji sadrži vrednost string objekta okruženog sa `A` oznakom sa NAME atributom.
• big()	Vraća string koji sadrži vrednost string objekta okruženog sa BIG oznakom.
• blink()	Vraća string koji sadrži vrednost string objekta okruženog sa BLINK oznakom.
• bold()	Vraća string koji sadrži vrednost string objekta okruženog sa B oznakom.
• charAt(index)	Vraća karakter sa lokacije određene indeksom.
• fixed()	Vraća string koji sadrži vrednost string objekta okruženog sa FIXED oznakom.
• fontColor(color)	Vraća string koji sadrži vrednost string objekta okruženog sa FONT oznakom sa COLOR atributom koji postavlja boju, gde je `color` naziv boje ili RGB triplet.
• fontSize(size)	Vraća string koji sadrži vrednost string objekta okruženog sa FONTSIZE oznakom sa veličinom na koju je postavljen.
• indexOf (findString,startingIndex)	Vraća mesto prvog pojavljivanja `findString`, koji počinje da pretražuje string od mesta određenog sa `startingIndex` gde je to samo opcija, ukoliko ne postoji, pretraživanje počinje od početka stringa.
• italics()	Vraća string koji sadrži vrednost string objekta okruženog sa I oznakom.
• lastIndexOf (findString,startingIndex)	Vraća mesto poslednjeg pojavljivanja `findString`, koji počinje da pretražuje string od mesta određenog sa `startingIndex` unazad gde je to samo opcija, ukoliko ne postoji, pretraživanje počinje od kraja stringa.
• link(href)	Vraća string koji sadrži vrednost string objekta okruženog sa A oznakom sa HREF atributom postavljenim na `href`.
• small()	Vraća string koji sadrži vrednost string objekta okruženog sa SMALL oznakom.
• strike()	Vraća string koji sadrži vrednost string objekta okruženog sa STRIKE oznakom.
• sub()	Vraća string koji sadrži vrednost string objekta okruženog sa SUB oznakom.
• substring (firstIndex, lastIndex)	Vraća podstring koji počinje od pozicije `firstIndex` i koji se završava sa jdnim karakterom pre `lastIndex`. Ako je `firstIndex` veće od `lastIndex`, izabrani podstring počinje od `lastIndex` i završava se karakter pre `firstIndex`.
• sup()	Vraća string koji sadrži vrednost string objekta okruženog sa SUP oznakom.
• toLowerCase()	Vraća string koji sadrži vrednost string objekta sa svim karakterima konvertovanim u mala slova.
• toUpperCase()	Vraća string koji sadrži vrednost string objekta sa svim karakterima konvertovanim u velika slova.

submit

Objekat *submit* omogućava rad sa *submit* dugmetom u HTML formi iz JavaScripta.

Osobine

- name — Sadrži naziv *submit* dugmeta.
- value — Sadrži vrednost *submit* dugmeta, odnosno tekst koji se ispisuje na dugmetu.

Metodi

- click() — Emulira akciju kliktanja mišem na *submit* dugme.

Obrađivači događaja

- onClick — Određuje JavaScript kod koji će se izvršiti kada bude kliknuto na *submit* dugme.

text

Objekat *text* omogućava rad sa text poljem HTML forme iz JavaScripta.

Osobine

- defaultValue — Sadrži predloženu vrednost za text element (odnosno sadrži vrednost VALUE atributa).
- name — Sadrži naziv *text* elementa.
- value — Sadrži vrednost *text* elementa.

Metodi

- focus() — Emulira akciju fokusiranja na *text* polje.
- blur() — Emulira akciju uklanjanja fokusa sa *text* polja.
- select() — Emulira akciju selektovanja teksta u *text* polju.

Obrađivači događaja

- onBlur — Određuje JavaScript kod koji se izvršava kada polje izgubi fokus.
- onChange — Određuje JavaScript kod koji se izvršava kada sadržaj polja promeni.
- onFocus — Određuje JavaScript kod koji se izvršava kada polje dobije fokus.
- onSelect — Određuje JavaScript kod koji se izvršava kada korisnik selektuje deo ili ceo tekst u polju.

textarea

Objekat *textarea* omogućava rad sa višelinijskim tekst poljem u HTML formi iz JavaScripta.

Osobine

- defaultValue — Sadrži predloženu vrednost za *textarea* element (odnosno sadrži vrednost VALUE atributa).
- name — Sadrži naziv *textarea* elementa.
- value — Sadrži vrednost *textarea* elementa.

Metodi

- focus() — Emulira akciju fokusiranja na *textarea* polje.
- blur() — Emulira akciju uklanjanja fokusa sa *textarea* polja.
- select() — Emulira akciju selektovanja teksta u *textarea* polju.

Obrađivači događaja

- onBlur — Određuje JavaScript kod koji se izvršava kada polje izgubi fokus.
- onChange — Određuje JavaScript kod koji se izvršava kada sadržaj polja promeni.
- onFocus — Određuje JavaScript kod koji se izvršava kada polje dobije fokus.
- onSelect — Određuje JavaScript kod koji se izvršava kada korisnik selektuje deo ili ceo tekst u polju.

window

Objekat *window* je *top-level* objekat za svaki prozor ili okvir i roditeljski je objekat za `document`, `location` i `history` objekte.

Osobine

- defaultStatus — Sadrži predloženu vrednost prikazanu u *status bar*-u.
- frames — Niz objekata, gde jedan element niza odgovara jednom okviru u prozoru. Okviri se pojavljuju u nizu onim redom kojim se pojavljuju u HTML izvornom kodu.
- length — Broj okvira u roditeljskom prozoru.
- name — Sadrži naziv prozora ili okvira.
- parent — Sadrži naziv prozora u kome se nalazi dokument sa definicijom okvira.
- self — Alternativa za naziv trenutnog prozora.
- status — Koristi se za prikazivanje poruke u *status bar*-u (u donjem delu WWW čitača je prostor u kome se isposuju poruke pri radu). To se obavlja prostom dodelom vrednosti ovoj osobini.
- top — Alternativa za naziv *top-most* prozora.
- window — Alternativa za naziv trenutnog prozora.

Metodi

- alert(message) — Prikazuje poruku u prozoru dijaloga (*dialog box*).
- close() — Zatvara prozor.
- confirm(message) — Prikazuje poruku u prozoru dijaloga (*dialog box*) sa OK i CANCEL dugmadima. Vraća *true* ili *false* zavisno od toga na koje je dugme kliknuo korisnik.
- open(url,name,features) — Otvara dokument sa datog URL-a u prozoru određenim sa `name`. Ako `name` ne postoji, otvara se novi prozor sa tim imenom. `features` je opcioni argument koji sadrži listu osobina za novi prozor.

  List osobina sadrži ma koji od sledećih parova naziv/vrednost odvojenih zarezima i bez dodatnih praznih mesta između:

  - toolbar=[yes,no,1,0]: ukazuje da li prozor treba da ima *toolbar*.
  - location=[yes,no,1,0]: ukazuje da li prozor treba da ima polje za prikazivanje trenutnog URL-a.
  - directories=[yes,no,1,0]: ukazuje da li prozor treba da ima dugmad direktorijuma.
  - status=[yes,no,1,0]: ukazuje da li prozor treba da ima *status bar*.
  - menubar=[yes,no,1,0]: ukazuje da li prozor treba da ima menije.
  - scrollbars=[yes,no,1,0]: ukazuje da li prozor treba da ima klizače prozora (*scroll bars*).
  - resizable=[yes,no,1,0]: ukazuje da li prozor treba da može da promeni veličinu.
  - width=pixels: daje širinu prozora u pikselima.
  - height=pixels: daje visinu prozora u pikselima.
- prompt(message,response) — Prikazuje poruku (*message*) u dijalog prozoru sa ulaznim tekst poljem sa predloženom vrednošću odgovora (*response*). Korisnikov odgovor je sadržaj tekst ulaznog polja koje se vraća kao string.
- setTimeout(expression,time) — Izvršava narednu posle određenog vremena, izraženog u milisekundama i sadržanog u `time`. Izlaz može da bude sledećeg oblika structure:

  name = setTimeOut(expression,time)
- clearTimeout(name) — Prekida brojač sa datim imenom.

Obrađivači događaja

- onLoad — Određuje JavaScript kod koji se izvršava kada se HTML dokument potpuno učita u prozor ili okvir.
- onUnload — Određuje JavaScript kod koji se izvršava kada se napušta trenutni HTML dokument u prozoru ili okviru.