STANDARDNI KORISNIČKI INTERFEJSI

Predavanje broj: 08

Nastavna jedinica: JavaScript

Nastavne teme:

Uvod u JavaScript. Operatori. Naredbe. Objekti. Funkcije. Nizovi. Interakcija sa korisnikom. Kontrola izvršavanja programa. Obrađivači događaja. Menjanje stila. Provera ulaza. Write. Log. Redeklaracija promenljive. Podaci. null.

Predavač: prof. dr Perica S. Štrbac, dipl. ing.

Literatura:

J. D. Gauchat, "Integrisane tehnologije za izradu WEB strana", Mikroknjiga, Beograd, 2014.

W3C Tutorials, Internet, 2014

JavaScript

- Na klijentskoj strani koristićemo JavaScript skripte.
- Skripte klijentske strane su uglavnom „prikačene" u okviru HTML dokumenta, ali takođe mogu biti ubačene u zaseban fajl, koji je referenciran u HTML dokumentu.
- Odmah po zahtevu za određenom Web stranom koja sadrži skripte klijentske strane, neophodni fajlovi se šalju na klijentski računar od strane Web servera.
  - Nakon toga, klijentski Web browser izvršava skriptu i prikazuje dokument (Web page), sa svim vidljivim izlazima iz klijentske skripte.
  - Skripte klijentske strane takođe mogu sadržavati instrukcije koje browser treba da sledi ako korisnik reaguje na dokument na određeni način
    - npr. ako klikne na odgovarajuće dugme na dokumentu.
    - Takve instrukcije se mogu izvršavati bez komunikacije sa Web serverom.
- Ako korisnici pogledaju izvorni kod (source code) Web strane koja sadrži klijentsku skriptu, mogu videti i kod te skripte.
  - Mnogi Web programeri uče kako da pišu klijentske skripte istražujući izvorne kodove dokumenata drugih autora.
- Kod server – side skripti, korisnici ne mogu videti izvorni kod, jer se on izvršava na Web serveru i odmah se nakon toga prebacuje u HTML.

JavaScript

- Skripte klijentske strane imaju veći pristup informacijama i funkcijama koje se nalaze na klijentskim računarima, dok skripte serverske strane imaju veći pristup istim na serveru.
- Skripte serverske strane zahtevaju instaliran interpreter skriptnog jezika na serveru i proizvode isti output bez obzira na browser klijenta, operativni sistem koji klijent koristi i druge detalje klijentskog sistema.
- Skripte klijentske strane ne zahtevaju dodatno instaliran softver na serveru i tako postaju popularne autorima Web stranica koji imaju malo administrativnih ovlašćenja na serveru.
  - Međutim, skripte klijentske strane zahtevaju da klijentski Web browser „razume" skriptni jezik u kome su pisane.
- Autori Web strana koje sadrže kod skriptnog jezika trebaju pažljivo da analiziraju ponašanje tog koda na različitim platformama pre nego što ga stave u upotrebu.
- Skripte klijentske strane se upotrebljavalju:
  - Da bi se dobili podaci koji se prikazuju na ekranu klijenta
  - Za online računarske igre
  - Za personalizaciju prikaza (bez reloadovanje Web strane)
  - Za pred – procesiranje formi

JavaScript

- JavaScript je skript jezik koji je razvila kompanija Netscape Communications i uvela ga u Netscape Navigator, počevši od verzije 2.0, uporedo sa uvođenjem podrške za Javu.
- JavaScript je uveden radi povećanja interaktivnosti Web strana.
- Web browser - i (čitači) koji podržavaju JavaScript interpretiraju kod.
- JavaScript dodaje interaktivnost Web prezentaciji na strani klijenta.
  - Na primer, korišćenjem JavaScripta je moguće odgovarati na akcije korisnika u samom WWW čitaču.
  - Pri kreiranju JavaScript koda može se koristiti obični editor.
- Polazni naziv JavaScript jezika bilo je LiveScript, ali u saradnji Netscape-a i Sun-a, ime je izmenjeno u JavaScript, zbog namene da Java i JavaScript budu jezici Internet-a i internet programiranja.
- JavaScript služi kao programski jezik. Njen suštinski jezik (core language) sadrži osnovne elemente programskih jezika kao što su promenljive, tipovi podataka, naredbe za kontrolu toka programa, funkcije i objekte.
- JavaScript se koristi za aritmetičke operacije, manipulacije datumom i vremenom, radi sa nizovima, stringovima i objektima...

JavaScript

- Kada browser korisnika primi Web stranu koja sadrži JavaScript kod, kod se šalje JavaScript interpreteru, koji izvršava skriptu.
- Budući da svaki browser ima svoj interpreter, često postoje razlike u načinu na koji se skripta izvršava.
- Java i JavaScript su različiti jezici i predstavljaju dve različite tehnike programiranja na Internetu.
  - Java je programski jezik. Javom se mogu kreirati programi koji mogu da se izvršavaju potpuno nezavisno od WWW čitača (pomenimo i tehnologiju koja nestaje - Java aplete, koji se mogu pozivati iz HTML dokumenta i koji se učitavaju preko mreže i onda izvršavaju u okviru WWW čitača (browsera)).
  - JavaScript je skript jezik.
- JavaScript se uključuje u HTML dokument na sledeće načine:
  - u sekciji <head>... </head> unutar oznaka <script> ... </script>
  - u sekciji <body>... </body> unutar oznaka <script> ... </script>
  - u sekciji <body>... </body> ili <form>... </form> unutar HTML elementa.
  - u datotekama sa ekstenzijom .js, uz uključenje u tekući HTML dokument:

    <script src="source.js"></script>

JavaScript

- U zavisnosti od toga, u kom segmentu HTML dokumenta je JavaScript uključen, postoje tri načina izvršenja JavaScript koda:
  - ako je skript uključen u zaglavlju, ignorisan je do poziva;
  - ako je skript uključen u telu, rezultat se prikazuje na Web strani;
  - ako je skript vezan za događaj, skript će se izvršiti kad se desi događaj.

- **Tipovi podataka, literali, promenljive, operatori**
- JavaScript prepoznaje sledeće osnovne tipove podataka:
  - numbers, celi i realni brojevi,
  - boolean, mogu uzeti vrednost true ili false,
  - strings, sadrže niz karaktera,
    - null, predstavlja rezervisanu ključnu reč za null objekat, suštinski nije 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 (CASE SENSITIVE).

JavaScript

- U JavaScriptu ne moramo eksplicitno da deklarišemo promenljivu (mada je to dobra praksa i paziti gde se to radi).
- U stringovima je dozvoljeno koristiti sledeće specijalne karaktere:
  - \b = pomeraj za jedno mesto ulevo (backspace)
  - \f = pomeraj jedan red dole (form feed)
  - \n = na početak novog reda (new line character)
  - \r = return (carriage return)
  - \t = tabulator (tab).
- JavaScript je slabo tipiziran jezik.
  - Tipovi podataka će biti automatski konvertovani zavisno od mesta njihove upotrebe u programu.
    - može se definisati i inicijalizovati sledeća promenljiva:

      var promenljiva=11
    - a kasnije se može predefinisati, dodeljujući joj niz karaktera, na primer

      promenljiva="Evo stringa"

JavaScript

- U kombinaciji broja i stringa, Java Script konvertuje broj u string.

```
x = "Primer za kombinaciju " + 11
y = 11 + " jos jedan primer."
```

- Za konverziju stringa u broj, koriste se funkcije:
  - eval(str) - ocenjuje string i ako je moguće pretvara ga u broj (izračunava se);
  - parseInt(str, baza) - konvertuje string (po bazi) u integer, ako je moguće;
  - parseFloat(str) - konvertuje string u floating-point broj, ako je moguće.
- Dobra je praksa definisati promenljivu sa var (paziti gde)
- Specijalna ključna reč null ukazuje da je promenljivoj dodeljena null vrednost.
  - To ne znači da je promenljiva nedefinisana.
  - Promenljiva je nedefinisana kada joj nije dodeljena vrednost i tada je ne možemo dodeliti drugoj promenljivoj ili koristiti u izrazima (run-time error)
- Celi brojevi u JavaScriptu mogu biti predstavljeni u tri osnove:
  - u oktalnom (baza 8),
  - u decimalnom (baza 10),
  - heksadecimalnom (baza 16) formatu.

JavaScript

- Decimalni celi brojevi se predstavljaju kao niz cifara (0-9) bez vodeće nule.
- Oktalni celi brojevi se predstavljaju kao niz cifara (0-7) predvođen nulom ("0").
- Heksadecimalni celi brojevi se predstavljaju kao niz cifara (0-9) i slova (a-f i A-F) predvođen nulom koju sledi slovo x ("0x" ili "0X").
- Primer prestavljanja celog broja deset (10) u tri brojna sistema:
  - decimalnom: 10
  - oktalnom: 012
  - heksadecimalnom: 0xA
- 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:

1.1234 .1E23 -1.1E12 2E-10

JavaScript

- Promenljive tipa Boolean može da ima samo jednu od dve vrednosti:
  - true (tačno)
  - false (netačno).
- String je niz karaktera sačinjen od nijednog ili više karaktera zatvorenih u jednostrukim (') ili dvostrukim (") navodnicima.
- Primeri stringova:
  - "prvi"
  - '1234'
  - "PRVA LINIJA \n druga linija"
- Takođe je često neophodno koristiti navodnike unutar niza karaktera. To se može uraditi korišćenjem obrnute kose crte. Na primer:

```
var tekst="<P>On je isao u skolu \"Naziv skole \".</p> ";
document.write(tekst);
```

- Prilikom deklarisanja promenljivih trebate obratiti pažnju na sledeće :
  - Pokušajte da deklarišete sve varijable, koje ćete koristi, na početku programskog koda, čak i ako još nemate konkretne vrednosti za njih.
  - Koristite ispravna imena varijabli. Ne koristite rezervisane reči i reči koje su predugačke ili teške za pamćenje (ali da budu dovoljno asocijativne). Nemojte davati slična imena različitim varijablama.
  - Imena varijabli su osetljiva na velika i mala slova (case sensitive),
  - Ključnu reč var koristiti ali paziti na lokalnost.

Operacija	Objašnjenje
X + Y	- sabiranje
X - Y	- oduzimanje
X * Y	- množenje
X / Y	- deljenje
X % Y	- ostatak deljenja X sa Y
X++	- postfiksno uvećanje za 1
++X	- prefiksno uvećanje za 1
X--	- postfiksno umanjenje za 1
--X	- prefiksno umanjenje za 1
X**Y	- stepenovanje

JavaScript

- Relacijski operatori

Operacija	Objašnjenje
X > Y	- X veće od Y
X < Y	- X manje od Y
X >= Y	- X veće ili jednako Y
X <= Y	- X manje ili jednako Y
X == Y, X === Y	- X jednako Y, X identično Y
X != Y, X !=== Y	- suprotno prethodnom

- Operatori uslova

Operacija	Objašnjenje
X && Y	- konjukcija (logičko "I")
X || Y	- disjunkcija (logičko "ILI")
!X	- negacija (logičko "NE")

- Operatori pridruživanja

Operacija	Objašnjenje
X = Y	promenljivoj X pridružuje se vrednost Y
X += Y, X -= Y	X = X + Y, X = X - Y
X *= Y, X /= Y	X = X * Y, X = X / Y
X %= Y	X = X % Y (ostatak pri deljenju)
X**=Y	X = X ** Y (stepenovanje)

JavaScript

- Bitwise operatori su operatori na nivou bita

&	AND	rezultantni bit je 1 ako su oba bita 1
|	OR	rezultantni bit je 1 ako je bar jedan bit 1
^	XOR	rezultantni bit je 1 ako su oba bita različita
~	NOT	invertuje bit
<<	Zero fill left shift	pomeranje u levo sa popunjavanjem nulama
>>	Signed right shift	pomeranje u desno sa očuvanjem znaka
>>>	Zero fill right shift	pomeranje u desno sa popunjavanjem nulama

Primeri (uzeta su samo 4 niža bita za pozitivne, a za negativne uzmite 32 bita):

`5 & 3 =`	`1`	`0101 & 0011`	`0001`
`5 | 3 =`	`7`	`0101 | 0011`	`0111`
`~ 5 =`	`-6`	`~0101`	`1...1010`
`2 << 1 =`	`4`	`0010 << 1`	`0100`
`5 ^ 1 =`	`4`	`0101 ^ 0001`	`0100`
`-5 >> 1 =`	`-3`	`1 1011 >> 1`	`1...1101`
`-5 >>> 1 =`	`2147483645`	`1 1011 >>> 1`	`01...10101`

Osnovna struktura i sintaksa naredbi

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

  document.writeln("Pocetak!
");

  – Gornja komanda poziva writeln() metod, koji je deo document objekta. Tačka-zarez ukazuje na kraj komande. JavaScript komanda se može prostirati na više redova, sve dok se završava sa tačkom-zarezom. Isto tako, može se više naredbi naći u jednom redu, sve dok se svaka završava sa tačkom-zarezom.
- Možemo grupisati naredbe u blokove naredbi, izdvojene velikim zagradama:

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

- Blokovi naredbi se koriste u definiciji funkcija i kontrolnim strukturama.

Osnovna struktura i sintaksa naredbi

- Jedna od osnovnih mogućnosti svakog programskog jezika je da pošalje tekst na izlaz.
- 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 se ostvaruje metodama:
    - write (šalje tekst u prozor WWW čitača bez pomeranja)
    - writeln (isto kao write() samo što još posle ispisa teksta dodaje \n , setiti se kako će reagovati HTML)

    objekta document:

```
document.write("Test");
document.writeln('Hm,probajte');
```

Objekti

- JavaScript podržava objekte. Objekti sadrže:
  - podatke, osobine objekta (*properties*), koji su JavaScript promenljive.
  - funkcije objekta su poznate kao metode objekta (*methods*).
- JavaScript ima određen broj ugrađenih funkcija.
  - Skoro svaki HTML element je dodeljen nekom JavaScript objektu.
  - Ti objekti su razvrstani u logičkoj hijerarhiji koja ima oblik stabla.
- JavaScript objekt ima osobine koje mu pripadaju. Njima se može pristupiti preko sledećeg modela:

  NazivObjekta.NazivOsobine

- I u nazivu objekta i u nazivu osobine objekta razlikujemo mala i velika slova. Osobinu objekta definišemo tako što joj dodelimo vrednost.
- JavaScript ima određeni broj ugrađenih objekata, ali programer može kreirati svoje sopstvene objekte. Kreiranje objekta se odvija u dva koraka:
  - 1. Definisanje tipa objekta preko pisanja funkcije,
  - 2. Kreiranje instance objekta sa new.

Objekti

- Da bi definisali tip objekta mora se napisati funkcija koja određuje njegovo ime, njegove osobine i metode.
- Na primer, neka se kreira tip objekta za studente.
  - Neka se objekat zove student i neka ima osobine ime, srednje_ime, prezime i indeks.
  - Potrebno je napisati sledeću funkciju:

```
function student(ime, srednje_ime, prezime, indeks) {
  this.ime=ime;
  this.srednje_ime = srednje_ime;
  this.prezime=prezime;
  this.indeks=indeks;
}
```

- Sada možemo da kreiramo objekat student1 pozivajući funkciju student korišćenjem new:

```
student1=new student("Pera", "Mitar","Perić", "100/99");
```

Objekti

- Objektu možemo da dodamo i metode. Recimo, funkcija prikaziProfil() koja ispisuje sve podatke o studentu:

```
function prikaziProfil() {
  document.write("Ime: " + this.ime + "<br>");
  document.write("Srednje ime: " + this.srednje_ime + "<br>");
  document.write("Prezime: " + this.prezime + "<br>");
  document.write("Indeks: " + this.indeks + "<br>");
}
```

- Sada ćemo ponovo napisati funkciju student() preko koje definišemo objekat.

```
function student(ime, srednje_ime, prezime, indeks) {
  this.ime=ime;
  this.srednje_ime = srednje_ime;
  this.prezime=prezime;
  this.indeks=indeks;
  this.prikaziProfil=prikaziProfil;
}
```

- Kada definišemo objekat student1, funkciji prikaziProfil() se pristupa preko:

```
student1.prikaziProfil();
```

- Objekat može da ima osobinu koja je i sama objekat.
  - Na primer, objekat tipa student, može u sebe uključivati osobinu smer koja i sama može biti objekat.
- Objektu Student možemo dodeliti osobine pod nazivima ime, srednje_ime, prezime i indeks na sledeći način:

```
Student.ime="Pera";
Student.srednje_ime="Petar";
Student.prezime="Perić";
Student.indeks="100/99";
```

- 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 Student možemo pristupiti i na sledeći način, potpuno ekvivalentan prethodnom primeru:

```
Student["ime"]="Pera";
Student["srednje_ime"]="Petar";
Student["prezime"]="Perić";
Student["indeks"]="100/99";
```

Objekti

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

```
Student[0]="Pera";
Student[1]="Petar";
Student[2]="Perić";
Student[3]="100/99";
```

- Opšti slučaj metode objekta
  - Metoda je funkcija pridružena objektu.
  - Programer definiše metod na isti način kao što definiše funkciju.
  - Potom tu funkciju pridružuje objektu na sledeći način:

```
NazivObjekta.NazivMetode = NazivFunkcije;
```

gde su:

NazivObjekta postojeći objekat,
NazivMetode je naziv koji dodeljujemo metodi,
NazivFunkcije je naziv funkcije koju povezujemo sa objektom.

Potom možemo pozivati metodu u kontekstu objekta kao:

```
NazivObjekta.NazivMetode(parametri);
```

Objekti

- JavaScript ima specijalnu rezervisanu reč, this, koja se koristi za referenciranje na trenutni objekt.
- Na primer, ako funkcija Proveri() proverava da li se vrednost osobine objekta nalazi u propisanim granicama:

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

- Tada se može pozvati funkcija 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)">
```

Objekti

- Svaka web stranica poseduje objekte:
  - window: top-level objekt; sadrži svojstva primenljiva na celokupan prozor,
  - location: sadrži svojstva tekuće URL lokacije
  - history: sadrži svojstva prethodno posećenih URL
  - document: sadrži svojstva sadržaja tekućeg dokumenta, kao što su naziv (title), boja pozadine (bgcolor), forme
- Primer svojstava:
  - location.href = "http://www.mape.rs/index.html" //lokacija dokumenta
  - document.title = "Probni dokument" //naziv dokumenta (title)
  - document.fgColor = #000000 //boja slova
  - document.bgColor = #FFFFFF //boja podloge
  - history.length = 5 //koliko poslednjih dokumenta da "pamti" u history-ju
- Browser može kreirati objekte bazirane na sadržaju stranice, npr.:

```
document.mojaforma              //forma
document.mojaforma.Check1       //check polje na formi
document.mojaforma.Button1      //taster na formi
```

Objekti

- Prethodni objekti mogu imati svojstva kao što su:
  - document.mojaforma.action = "http://www.mape.rs/index.html"
  - document.mojaforma.method = get
  - document.mojaforma.Button1.name = DUGME1
  - document.mojaforma.text1.name = TekstPolje1
  - document.mojaforma.text1.value = "sadržaj tekst polja"
  - document.mojaforma.Check1.defaultChecked = true
  - document.mojaforma.Check1.value = on
  - itd.
- Mnogi objekti imaju metode koje emuliraju događaje.
- Npr, button objekt ima click metodu koja emulira klik miša na tasteru.

Objekti

- Lista događaja i pripadnih obrađivača događaja

Event (događaj)	Nastaje kada korisnik:	Event Handler
blur	izađe iz fokusa elementa forme	**onBlur**
click	klikne na element forme ili link	**onClick**
change	podesi/promeni vrednost "text", "textarea" ili izabranog elementa	**onChange**
focus	uđe u fokus nekog elementa forme	**onFocus**
load	učita stranicu u browser	**onLoad**
mouseover	pomera pokazivač miša preko linka ili "anchora"	**onMouseOver**
select	izabere "input" polje elementa forme	**onSelect**
submit	izvrši "submit" (slanje) forme	**onSubmit**
unload	"napusti" stranicu	**onUnload**

Prosleđivanje objekata funkcijama

- Parametri jedne funkcije nisu ograničeni samo na nizove karaktera i brojeve, već se mogu prosleđivati čitavi objekti.
- U sledećem primeru data je upotreba funkcije 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.
  - Naredba return vraća programu na mesto poziva rezultat "rada funkcije"

```
function vidi_svojstva(objekat, ime_objekta){
  var i, rezultat = "";
  for (i in objekat)
    rezultat += ime_objekta + "." + i + " = " + objekat[i] + "\n";
  return rezultat;
}
```

- Ako sada funkciju pozovemo sa vidi_svojstva(Avion, "Avionce"), gde je Avion objekat koji ima sledeća svojstva: proizvodjac, model i brzina (i oni imaju vrednosti "Boing", "747" i 950) kao rezultat ćemo dobiti:

```
Avionce.proizvodjac = Boing
Avionce.model = 747
Avionce.brzina = 950
```

Nizovi

- 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;}
}
```

- Ova funkcija definiše niz tako da prva osobina, duzina , predstavlja broj elemenata niza.
  - Preostale osobine imaju celobrojni indeks, od jedan pa više. Možemo takođe kreirati niz tako što ga pozovemo, kao u sledećem primeru, određujući broj članova niza:

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

- Ovo kreira naš niz niz1 sa 10 elemenata i sve ih inicijalizuje na nulu.
- Nizove radimo detaljno kasnije.

Interakcija sa korisnikom

- „Programi vole da pričaju sa korisnikom, postavljaju pitanja, daju odgovore i odgovaraju na poziv“. Metod predstavlja način na koji se radi nešto, a JavaScript metodi su akcione reči (action words) JavaScript jezika. Metodi su pokretači dešavanja.
- JavaScript koristi dijalog prozore (dialog boxes) za interkaciju sa korisnikom. Dijalog prozori se kreiraju pomoću tri metoda:
  - Alert()
  - Prompt()
  - Confirm()
- Videli smo da se write() i writeln() metode koriste za slanje outputa na Web stranu.
- Drugi način za slanje output–a browser–u je alert() metoda.
  - Alert() metoda pravi mali, nezavisan prozor – nazvan dijalog prozor, prozor za dijalog.
  - Prikazuje se personalizovana poruka korisnika i OK dugme.
  - Kada se pojavi dijalog prozor, svako izvršavanje koda je stopirano dok korisnik ne pritisne OK dugme.

Interakcija sa korisnikom

- Izled dijalog prozora može varirati od browser–a do browser–a, ali u svakom browser–u funkcioniše na isti način.
- Primer alert() metode.

```
<!DOCTYPE html>
<html>
<head><title>Dijalog prozor</title></head>
<body bgcolor="yellow" text="blue">
<b>Testiranje alert metode</b><br>
<script>
  document.write("It's a bird, <br>");
  document.write("It's a plane, <br>");
  alert("It's Superman!");
</script>
</body>
</html>
```

- Primer alert metode

```
<script>
window.alert(5 + 6);
</script>
```

Interakcija sa korisnikom

- Budući da JavaScript ne obezbeđuje jednostavan metod za prihvatanje korisničkih inputa, za ove potrebe, koriste se HTML forme i prozori za dijalog (*prompt dialog boxes*).
- Prompt dialog box se pojavljuje sa jednostavnim poljem za tekst (*textfield*). Prompt dialog box koristi dva argumenta: string (tekst koji obično predstavlja pitanje postavljeno korisniku) i još jedan string koji obično predstavlja default odgovor koji je napisan u dialog box-u. Ako korisnik pritisne OK dugme, ceo tekst iz dijalog prozora se vraća.
- Primer prompt() metode:

```
<!DOCTYPE html>
 <html><head><title>Korišćenje prompt box-a</title></head>
<body><script>
var ime=prompt("Koje je vaše ime? ");
document.write("<br>Dobrodošli " + ime + "<br>");
var godine=prompt("Recite nam Vaše godine.", "Age");
if (godine == null){    // Ako korisnik pritisne cancel dugme
  alert("Nećete da nam kažete koliko godina imate!");
}else{alert(godine + " godina imate");}
alert(prompt("Gde stanujete ", ""));
</script></body></html>
```

Interakcija sa korisnikom

- Confirm box (prozor) se koristi da bi se potvrdio odgovor korisnika na pitanje. Znak pitanja će se pojaviti u prozoru zajedno sa OK i Cancel dugmetom.
  - Ako korisnik pritisne OK dugme, vraća se vrednost true, a ako pritisne Cancel, vraća se vrednost false.
  - Ova metoda koristi samo jedan argument, pitanje koje je postavljeno korisniku.
- Primer Confirm() metode

```
<!DOCTYPE html>
<html><head><title>Korišćenje confirm box – a </title>
</head><body>
<script>
document.clear;   // Briše sadržaj strane
if(confirm("Da li ste zaista dobro?")== true){
alert("Onda možemo da nastavimo");
}
else{
alert("Pokušaćemo kada Vam bude bolje");
}
</script>
</body></html>
```

Kontrola izvršavanja programa, iskazi i petlje

- JavaScript podržava kontrolne strukture poznate iz drugih jezika:
  - while petlja, (primer izračunava sumu svih brojeva od 0 zaključno sa 10)

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

  - for petlja (primer ispisuje tri nivoa naslova):

```
{
  for(i=1;i<=3;i++){
    document.write("<h"+i+"> Naslov na nivou "+i+"</h"+i+">")
  }
}
```

  - for .. in, pomoću nje možemo proći npr. kroz sve osobine (properties) nekog objekta.

```
{
  for (pro in obj){ ispisi(pro); }
}
```

Kontrola izvršavanja programa, iskazi i petlje

```
<!DOCTYPE html>
<html><body><p id="demo"></p>
<script>
var day;
switch (new Date().getDay()) {
    case 0:         day = "Sunday";         break;
    case 1:         day = "Monday";         break;
    case 2:         day = "Tuesday";        break;
    case 3:         day = "Wednesday";      break;
    case 4:         day = "Thursday";       break;
    case 5:         day = "Friday";         break;
    case 6:         day = "Saturday";       break;
}
document.getElementById("demo").innerHTML = "Today is " + day;
</script>
</body></html>
```

- Naredba do while

```
do {
    text += "The number is " + i;    i++;
}
while (i < 10);
```

Kontrola izvršavanja programa, iskazi i petlje

- if..else, može postojati u obliku if(uslov){} ili if(uslov){ } else { }:

```
if (n>10){
  document.write("N je veće od 10");
}
else {
  document.write("N je manje ili jednako 10");
}
```

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

```
i=3;
while(i<10){
  if(i==5){ break;} i++; document.write(i);
}
```

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

```
i=3;
while(i<10){
 i++; if(i==5){continue;} document.write(i);
}
```

Kontrola izvršavanja programa, iskazi i petlje

- function i return iskaz: primer funkcije koja izračunava kvadrat broja:

```
function kvadratBroja( x ) {
 return x * x ;
}
---
x = kvadratBroja(5);   /* poziv funkcije */
document.write("Kvadrat broja 5 je " + x);
---
Kvadrat broja 5 je 25
```

- new iskaz: definiše novi objekt. Donji primer promenljivoj datum pridružuje tekući datum:

```
var datum = new Date();
```

- this iskaz: referenca na tekući objekt. Donji primer definiše da se klikom na objekt poziva funkcija slucajniBroj() a kao parametar prenosi se naziv tekuće forme (this.form):

```
onClick="slucajniBroj(this.form)";
```

- var iskaz: definisanje promenljivih (opciono):

```
var broj1 = 10, tekst1="test";
```

JavaScript obrađivači događaja (event handlers)

- 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.
  - primeri događaja u WWW čitaču: korisnik klikne na hipertekst link, kada se promeni podatak u ulaznom polju forme, kada se završi učitavanje WWW dokumenta.
- 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 događaj se aktivira pošto se traženi HTML dokument (i svi njegovi delovi) kompletno učita. Dodeljen je elementu <body>.
  - Prema prirodi Interneta, ne postoje garancije kada će se i u kom roku učitati neki dokument
    - opterećenje linka može biti veliko
    - link može biti u prekidu

    ali ako koristimo OnLoad, možemo biti sigurni da je ceo html dokument učitan.

JavaScript obrađivači događaja (event handlers)

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 obavlja obrađivanjem događaja OnUnload.

Događaji uzrokovani akcijama miša

- Događaji uzrokovani akcijama miša zahtevaju akciju korisnika (rad mišem) da bi se aktivirali.

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.

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.

JavaScript obrađivači događaja (event handlers)

OnClick

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

OnFocus

- Događaj Focus se pojavljuje kada objekat postane predmet u fokusu. To se događa kada korisnik klikne mišem na određeni objekat (ili pritiskanjem tab-a).
  - Ako korisnik može da unese podatke u objekat (ili da promeni trenutni izbor za slučaj lista izbora), tada objekat ima fokus.
  - Obrađivač događaja OnFocus obično se koristi sa objektima text, textarea, password i select.

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 focus pritiskom na taster tab.
  - Obrađivač događaja OnBlur obično se koristi sa istim objektima kao i OnFocus.

JavaScript obrađivači događaja (event handlers)

OnChange

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

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.

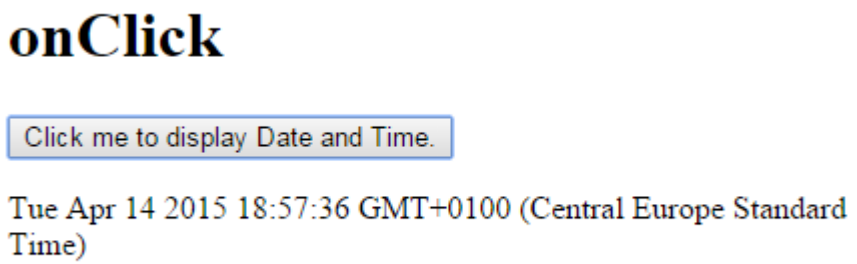
OnSubmit

- 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.
- 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: onClick

- Primer reagovanja na onClick dogadjaj

```
<!DOCTYPE html>
<html>
<body>
<h1>onClick</h1>
<button type="button"
    onClick="document.getElementById('demo').innerHTML = Date()">
    Click me to display Date and Time.
</button>

<p id="demo"></p>

</body>
</html>
```



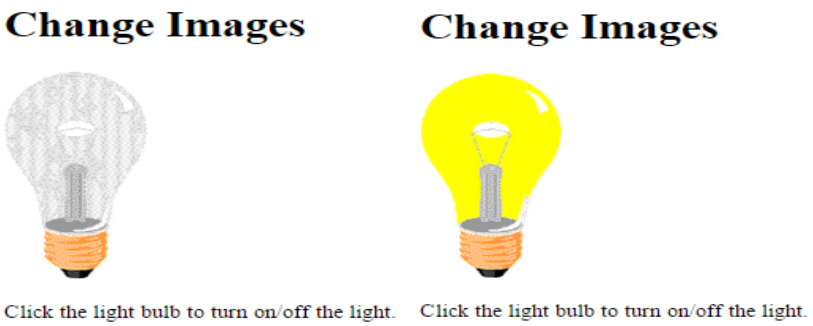
Paljenje i gašenje sijalice

- Paljenje i gašenje sijalice.


```
<!DOCTYPE html>
<html>
<body>
<script>
function light(sw) {
    var pic;
    if (sw == 0) {  pic = "pic_bulboff.gif" }
    else         {  pic = "pic_bulbon.gif"  }
    document.getElementById('myImage').src = pic;
}
</script>

<img id="myImage" src="pic_bulboff.gif" width="100" height="180">

<p>
<button type="button" onclick="light(1)">Light On</button>
<button type="button" onclick="light(0)">Light Off</button>
</p>

</body></html>
```

Paljenje i gašenje sijalice

```
<!DOCTYPE html>
<html>
<body>
<h1>Change Images</h1>
```




```
<img id="myImage" onclick="changeImage()" src="pic_bulboff.gif"
width="100" height="180">

<p>Click the light bulb to turn on/off the light.</p>

<script>
function changeImage() {
    var image = document.getElementById('myImage');
    if (image.src.match("bulbon")) {
        image.src = "pic_bulboff.gif";
    } else {
        image.src = "pic_bulbon.gif";
    }
}
</script></body></html>
```

Menjanje stila

```
<!DOCTYPE html>
<html>
<body>

<h1>Menjanje stila</h1>

<p id="demo">JavaScript can change the style of an HTML
element.</p>

<script>
function myFunction() {
    var x = document.getElementById("demo");
    x.style.fontSize = "25px";
    x.style.color = "red";
}
</script>
<button type="button" onclick="myFunction()">Click Me!</button>
</body>
</html>
```

Provera ulaza

```
<!DOCTYPE html>
<html><body>
<h1>Provera ulaza</h1>
<p>Please input a number between 1 and 10:</p>
<input id="numb" type="number">
<button type="button" onclick="myFunction()">Proveri</button>
<p id="demo"></p>
<script>
function myFunction() {
    var x, text;
    // Get the value of the input field with id="numb"
    x = document.getElementById("numb").value;
    // If x is Not a Number or less than one or greater than 10
    if (isNaN(x) || x < 1 || x > 10) {
        text = "Input not valid";
    } else {
        text = "Input OK";
    }
    document.getElementById("demo").innerHTML = text;
}
</script></body></html>
```

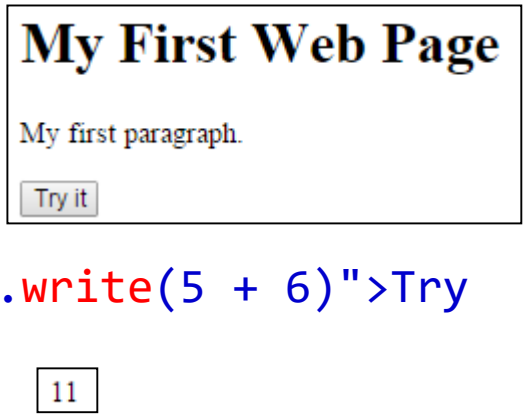
document.write()

- Ako se koristi u fazi formiranja dokumenta

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<script>
  document.write(5 + 6);
</script>
</body>
</html>
```

- Ako se koristi u fazi kada je dokument kreiran, obrisaće html dokument

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<button type="button" onclick="document.write(5 + 6)">Try
it</button>
</body></html>
```



console.log()

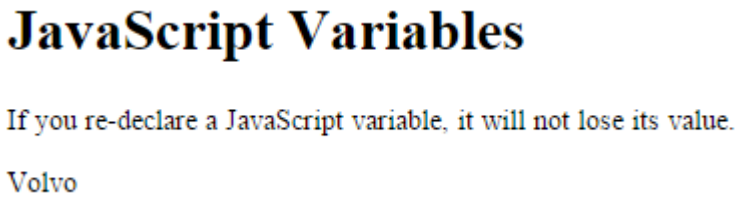
- Za prikazivanje podataka u browseru koristi se console.log() metoda.
- Konzola browser-a se aktivira sa F12, i selekcijom "Console" u debager meniju.
- Primer

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

Redeklaracija promenljive

- Redeklaracija promenljive zadržava njenu prethodnu vrednost

```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Variables</h1>

<p>If you re-declare a JavaScript variable, it will not lose its
value.</p>

<p id="demo"></p>

<script>
var carName = "Volvo";
var carName;
document.getElementById("demo").innerHTML = carName;
</script>

</body>
</html>
```



Podaci

```
var length = 16;                                // number
var lastName = "Johnson";                       // string
var cars = ["Saab", "Volvo", "BMW"];            // array
var x = {firstName:"John", lastName:"Doe"};     // object
var x = 16 + "Volvo";          //16Volvo
var x = 16 + 4 + "Volvo";      //20Volvo
var x = "Volvo" + 16 + 4;      //Volvo164
```

- Promena tipa

```
var x;               // x is undefined
var x = 5;           // x is a Number
var x = "John";      // x is a String
```

- Stringovi

```
var carName = "Volvo XC60";
var carName = 'Volvo XC60';
```

Podaci

- Razne verzije

```
var answer = "It's alright";
var answer = "He is called 'Johnny'";
var answer = 'He is called "Johnny"';
(2147483645).toString(2); //01111111111111111111111111111101
```

- Brojevi

```
var x1 = 34.00;     // sa decimalama
var x2 = 34;        // bez decimala
var y = 123e5;      // 12300000
var z = 123e-5;     // 0.00123
```

- Bool podaci

```
var x = true;
var y = false;
```

- Polja

```
var cars = ["Saab", "Volvo", "BMW"];
document.write(cars); //Saab,Volvo,BMW
```

Podaci

- Objekti

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>

<script>
var person = {
    firstName : "John",
    lastName  : "Doe",
    age       : 50,
    eyeColor  : "blue"
};

document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>

</body>
</html>
```

Podaci

- Tip podatka

```
typeof "John"                 // string
typeof 3.14                   // number
typeof false                  // boolean
typeof [1,2,3,4]              // object
typeof {name:'John', age:34} // object
```

```
<!DOCTYPE html>
<html><body>
<p>The typeof operator returns the type of a variable or an
expression.</p>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML =
typeof "john" + "<br>" +
typeof 3.14 + "<br>" +
typeof false + "<br>" +
typeof [1,2,3,4] + "<br>" +
typeof {name:'John', age:34};
</script></body></html>
```

Podaci

- Postavljanje na nedefinisanu vrednost i tip.

```
<!DOCTYPE html>
<html><body>
<p>Variables can be emptied by setting the value to
<b>undefined</b>.</p>
<p id="demo"></p>
<script>
var person = {firstName:"John", lastName:"Doe", age:50,
eyeColor:"blue"};
var person = undefined;
document.getElementById("demo").innerHTML =
person + "<br>" + typeof person;
</script>
</body></html>
```

- Nedefinisani su i vrednost i tip:

```
var person;
```

- Definisani su i vrednost i tip:

```
var car = "";
```

Variables can be emptied by setting the value to **undefined**.

undefined
undefined

null

- Postavljanje na null, ostaje definisan tip

```
<!DOCTYPE html>
<html>
<body>
<p>Objects can be emptied by setting the value to <b>null</b>.</p>
<p id="demo"></p>
<script>
  var person = {firstName:"John", lastName:"Doe", age:50,
                eyeColor:"blue"};
  var person = null;
  document.getElementById("demo").innerHTML = typeof person;
</script>
</body>
</html>
```

- Razlika između undefined i null

```
typeof undefined              // undefined
typeof null                   // object
null === undefined            // false
null == undefined             // true
```