Programiranje Internet aplikacija
JavaScript

Zašto JavaScript

- Nedostatak HTML strana je nemogućnost dinamičke obrade unijetih podataka od strane korisnika.
- Zato se došlo do zaključka da HTML postaje ograničavajući faktor i da je potrebna nova tehnologija za realizaciju dinamičkih djelova aplikacije.

Prve tehnologije

- Prvi pokušaj je bio pomoću serverskih komponenti, od kojih je najpopularnija bila CGI (Common Gateway Interface).

  Ipak, problem je predstavljala česta klijent-server komunikacija. Sve akcije se obavljaju na serverskoj strani.

Istorijat

- Decembra 1995. godine, Netscape i Sun predstavili su jezik JavaScript 1.0, originalno nazvan LiveScript.
- Ovaj jezik je omogućio ne samo formatiranje podataka na klijentskoj strani, već i obradu i dinamičko izvršavanje stranica. Treba napomenuti da je implementiran dio jezika koji se izvršavao i na serverskoj strani, čime je omogućio da se ista tehnologija koristi na obje strane aplikacije, ali ovaj dio JavaScript jezika nije dostigao veću popularnost i neće se razmatrati.

Standardizacija

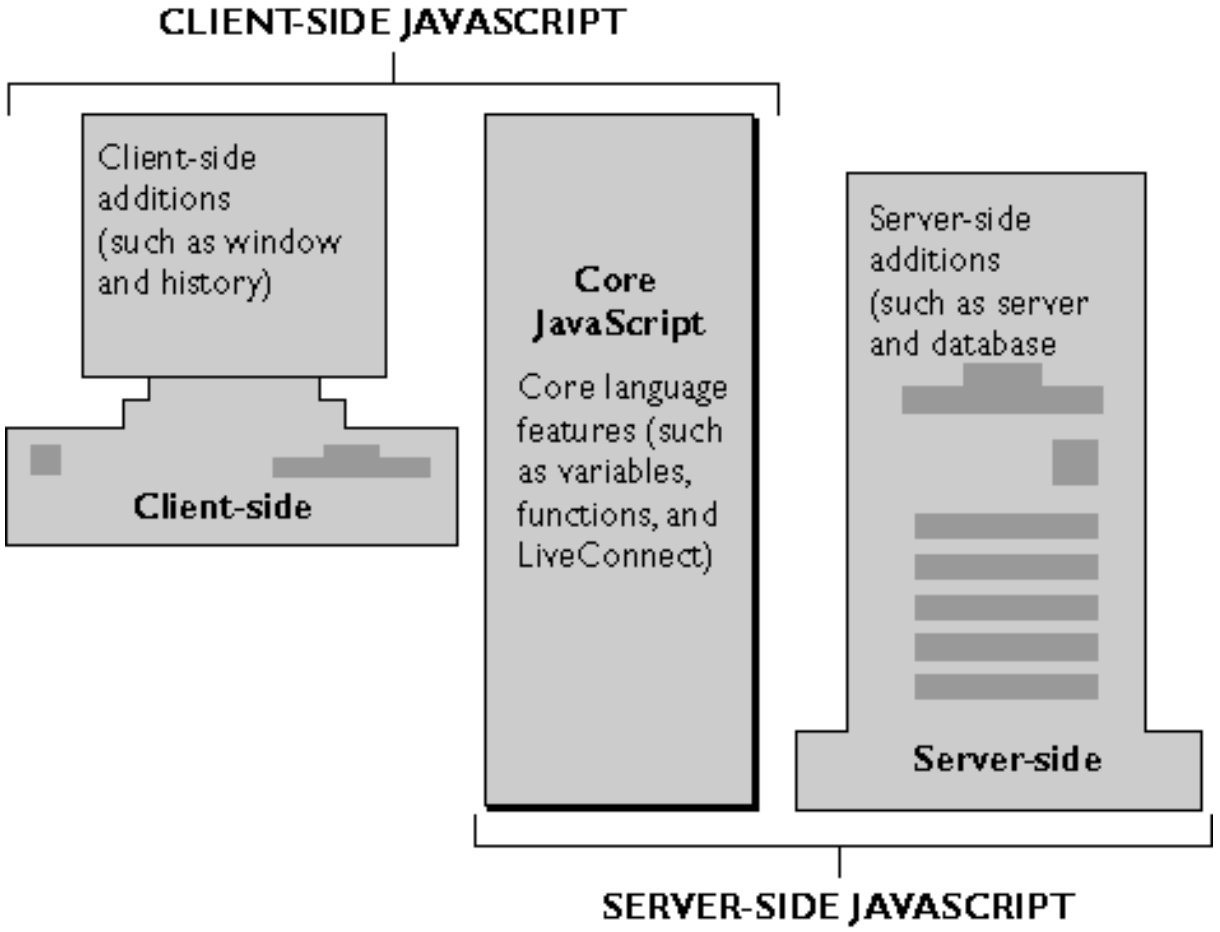
- Sljedeći korak u popularnosti JavaScript jezika je bila Microsoft-ova implementacija u okviru čitača Internet Explorer verzije 3, pri čemu je ova verzija od strane Microsoft-a nazvana JScript. JScript je bio baziran na javnoj dokumentaciji Netscape-a i bio je skoro identičan JavaScript jeziku.
- ECMA JavaScript verzija postao je Netscape-ova implementacija ovog standarda, a JScript Microsoft-ova. I danas obje verzije standarda su identične u preko 95% slučajeva.

- JavaScript je:
  - objektno baziran,
  - platformski neutralan,
  - višekorisnički jezik.
- JavaScript programeru omogućava mnogo veću funkcionalnost na klijentskoj strani.

Šta je objektno baziran?

- Svi koncepti objektno orijentisanih jezika nisu realizovani u ovom jeziku, da je veoma limitiran rad sa nasljeđivanjem, važenjem i funkcionalnošću samih objekata. Sa druge strane postoji hijerarhija ugrađenih objekata i oni se mogu koristiti, sa već definisanim metodama i osobinama (property).
- Ovakvim pristupom dobijeno je na jednostavnosti samog jezika, a pomoću ugrađenih objekata nije izgubljena potrebna funkcionalnost.

Opis JavaScript jezika



Platformski neutralan jezik

- Platformski neutralan jezik, kao i HTML, što znači da bi njegov kod (ako je pisan po standardu) trebalo da se izvršava u okviru čitača klijenta, bez obzira koja je hardverska mašina ili sofvtersko okruženje u pitanju.
- Veličina programa pisanih u ovom jeziku dovoljno je mala da može da se izvršava i na mašinama sa lošijim perfomansama.

JavaScript i HTML

- Još jedna od prednosti JavaScript jezika je njegova integrisanost sa HTML-om. U okviru jedne stranice je moguće na proizvoljan način kombinovati JavaScript i HTML kod.
- Takođe iz JavaScript-a moguće je generisati sam HTML kod, u zavisnosti od određene akcije korisnika.

JavaScript

Osnove jezika

Kako se uključuje programski kod?

- Programski kod ovog jezika se može uključuti u okviru HTML stranice na dva načina.
- Prvi je direktnim pisanjem koda u okviru stranice.
- **<script language="JavaScript">**
  ...neki JavaScript kod…
  </script>
- Nije neophodno da se navodi atribut **language= "JavaScript"**, jer on ima podrazumijevanu vrijednost **JavaScript**
- Primjer 1 - HelloJavaScript.html

JavaScript fajl (.js)

- Drugi način je poziv js dokumenta. U okviru taga se definiše spoljašnji dokument u okviru atributa src.

  Struktura ove vrste koda je:

```
<script language="JavaScript" src="JSkod.js">
</script>
```

  gdje je *JSkod.js* dokument koji sadrži željene JavaScript funkcije, na primjer:

```
document.write("Ovo je JavaScript eksterni
fajl!");
```

Odvajanje linija koda

- Podrazumijevani separator je novi red.
- Nije greška ako se koristi simbol ";".
- Jedini izuzetak, kada se obavezno mora koristiti tačka-zarez je ako se navodi više naredbi u istom redu. Tada se svaka pojedinačna naredba mora odvojiti sa tačkom-zarez.

Komentar

- Za komentar jedne linije koda se koristi oznaka **//**, na primjer:

```
// komentar u jednoj liniji
```

- Za komentarisanje više redova koriste se oznake **/*** za početak bloka pod komentarom

  i oznake ***/** za kraj bloka pod komentarom

  Primjer:

```
/* ovo je komentar
u vise linija */
```

Prikaz HTML teksta

- HTML tekst se prikazuje pomoću JavaScript koda na stranici korišćenjem metoda

```
document.write("neki tekst koji se prikazuje");
```

- Argument ovog metoda je string koji može biti proizvoljan HTML kod. Na primjer:

```
<script language="JavaScript">
    document.write("<b>Prvi red</b><br><i>Drugi
red</i>")
</script>
```

Nazivi promjenljivih

- Imena promjenljivih mogu da sadrže brojeve i slova engleske abecede, ali prvi znak mora da bude slovo engleske abecede ili simbol "_" .
- Ne mogu se koristiti prazna mjesta u okviru imena.
- Ne mogu se koristiti rezervisane riječi kao imena promjenljivih.

var

- Koristi se za deklarisanje promjenljive.
  (deklaracija je kreiranje promjenljive, a definicija znači i inicijalizaciju - postavljanje početne vrijednosti)

```
var imePromjenljive;
```

- Opciono moguće je izvršiti i njenu inicijalizaciju.

```
var imePromjenljive = vrijednost;
var ime promjenljive1 =
    vrijednost1,  ime
    promjenljive2 = vrijednost2,
    ...;
var euro; //deklaracija promjenljiva
var dolat = 95; //definicija int promjenljive
```

var

- Nije neophodno deklarisati promjenljivu prije prve dodjele vrijednosti (automatski će se izvršiti deklarisanje).
- Predeklarisanje je dozvoljeno.

var

- Sljedeće četiri sekvence imaju isti efekat:
  - ```
    var x;
    x=8;
    ```
  - ```
    x=8;
    var x;
    ```
  - ```
    var x=8;
    ```
  - ```
    x=8;
    ```

Case sensitive

- JavaScript je **case sensitive jezik**, što znači da se velika i mala slova razlikuju, pa je promjenljiva Aaa različita promjenljiva od promjenljive AAA.

  Također se ključne riječi (for, if, else, class, int,...) ne mogu koristiti u imenu promjenljivih.

Tipovi podataka

Informacija koja se sadrži u promjenljivoj.

Postoje:

- cjelobrojni brojevi,
- racionalni brojevi,
- stringovi (niz karaktera),
- logički tip (true/false).

Tip podataka definiše i vrste operacija koje se mogu izvršiti sa tom promjenljivom.

Cjelobrojni brojevi

- Mogu se koristiti sa brojnom osnovom 10, sa osnovom 8 i osnovom 16.
- Uobičajena je predstava pomoću osnove 10. Ovakvi brojevi imaju cifre od 0 - 9, s tim da početna cifra ne smije biti 0.
- Brojevi prikazani u oktalnom brojnom sistemu sa osnovom 8 moraju počinjati sa cifrom 0, a ostale cifre su od 0 -7.
- Brojevi prikazani u heksadecimalnom brojnom sistemu sa osnovom 16 moraju počinjati sa 0x ili 0X,
  a ostale cifre su od 0 - 15, s tim da se cifre 10 - 15 prikazuju slovima A - F.

Racionalni brojevi

- Mogu se prikazati na dva načina:
  - pomoću decimalne tačke,
    na primjer 3.14
  - pomoću eksponencijalne prezentacije,
    na primjer 314E-2 ili 314e-2

String

- String predstavlja proizvoljan niz karaktera između navodnika ("neki tekst") ili između apostrofa ('neki tekst').
- U stringovima se mogu koristiti i specijalni karakteri.

Specijalni karakteri

- \b = jedno mjesto lijevo (backspace)
- \f = jedan red nadolje (form feed)
- \n = početak novog reda (new line character)
- \r = return (carriage return)
- \t = tabulator (tab)

Konverzija u string - primjer

```
<script>
  x = 2 + 4;
  document.write(x); document.write("<br>");
  x = "2" + "4";
  document.write(x); document.write("<br>");
  x = 2 + "4";
  document.write(x); document.write("<br>");
  x = "2" + 4;
  document.write(x); document.write("<br>");
</script>
```

Konverzija u string

- Rješenje:

```
6
24
24
24
```

- Zaključak:

  Integer se uvek konvertuje u string pri konkatenaciji sa stringom.

Logički tip

- Logički tip podataka obuhvata dvije vrijednosti **true (tačno)** i **false (netačno)**.
- Prilikom rada ako je potrebno može se izvršiti konverzija logičke vrijednosti **true** u broj **1** i vrijednosti **false** u broj **0**.

Konverzije podataka

- JavaScript je jezik koji automatski izvršava promjenu jednog tipa u drugi, jer se dozvoljava da promjenljiva ima različite tipove podataka u različito vrijeme izvršavanja programa.
- Primjer:

```
a = 5; //a je sada cjelobrojni
podatak     b = 8; //b je sada
cjelobrojni podatak
b = "broj " + a;
/* b je sada string podatak, zato što
   se na string "broj" nadovezuje cio
   broj, pa se dobija string! */
```

Null vrijednost

- Vrijednost `null` je
  - tip podataka/vrijednost koja se može dodijeliti promjenljivoj
  - promjenljiva koja nema vrijednost
  - dodijeljena promjenljivoj kada želimo da definišemo da promjenljiva ne sadrži nikakav podatak

OPERATORI

- Operatori su specijalni karakteri, koji definišu operaciju koja treba da se izvrši nad operandima, koji mogu biti promjenljive, izrazi ili konstante.

Aritmetički operatori

- Koriste se za matematičke operacije.
- Ukoliko je jedan od operanada tipa String za sve operatore, osim za sabiranje, pokušaće se da se izvede konverzija Stringa u broj i da se tako izvrši definisana operacija.
  Ako se ne uspije kao rezultat se dobija specijalna vrijednost NaN (Not A Number).
- Izuzetak kod sabiranja: podatak koji nije tipa String konvertuje se u String i izvršava se sabiranje dva Stringa.
- `a=24; b="broj " + a; //dobija se da je b: broj 24`

Aritmetički operatori - pregled

Operator	Opis	Operator	Opis
+	sabiranje	+=	sabiranje dodjela
-	oduzimanje	-=	oduzimanje dodjela
*	množenje	*=	množenje dodjela
/	dijeljenje	/=	dijeljenje dodjela
%	moduo	%=	moduo dodjela
++	inkrement (x=x+1)	--	dekrement (x=x-1)

Operatori na nivou bita

- Operatori iz ove grupe obavljaju operacije nad cjelobrojnim brojevima, i to dužine 32 bita.
- Ukoliko neki od operanada nije cjelobrojni broj dužine 32 bita, pokušaće se izvršiti konverzija u traženi tip, pa tek onda primijeniti operacija.

Tabela operatori na nivou bita

Operator		Opis
Logičko I (and)	a & b	Rezultat je 1, samo ako su oba bita 1.
Logičko ILI (or)	a | b	Rezultat je 0, samo ako su oba bita 0.
Logičko eksluzivno ILI (xor)	a ^ b	Rezultat je 1, samo ako je jedan bit 1, a drugi 0.
Logičko NE (not)	~ a	Komplementira bit 0->1, 1->0.
Pomijeranje ulijevo	a << b	Pomijera binarni sadržaj operanda a za b mjesta ulijevo. Prazna mjesta popunjava nulama.
Pomijeranje udesno sa znakom	a >> b	Pomijera binarni sadržaj operanda a za b mjesta udesno. Prazna mjesta popunjava vrijednošću najstarijeg bita.
Pomijeranje udesno sa nulama	a >>> b	Pomijera binarni sadržaj operanda a za b mjesta udesno. Prazna mjesta popunjava sa vrijednošću 0.

Primjeri

- 13 & 8 daje 8 (1101 & 1000 = 1000)
- 13 | 8 daje 13 (1101 | 1000 = 1101)
- 13 ^ 8 daje 5 (1101 ^ 1000 = 0101)

Logički operatori

- Imaju vrijednosti:
  - true
  - false
- Ovi operatori imaju veliku primjenu u kontrolama toka.

Logički operatori - pregled

Operator		Opis
I (&&)	izraz1 && izraz2	Rezultat je TRUE, jedino ako su oba izraza TRUE, u ostalim slučajevima FALSE.
ILI (||)	izraz1 || izraz2	Rezultat je TRUE, ako je bar jedan izraz TRUE, ako su oba FALSE, rezultat je FALSE.
NE (!)	! izraz	Rezultat daje komplement: ako je izraz TRUE rezultat je FALSE i obrnuto.

Primjer upotrebe navedenih operatora

```
a = true;
b = false;
c = a || b;
d = a && b;
f = (!a && b) || (a && !b);
g = !a;
document.write(" a = " + a + "<BR>");
document.write(" b = " + b + "<BR>");
document.write(" c = " + c + "<BR>");
document.write(" d = " + d + "<BR>");
document.write(" f = " + f + "<BR>");
document.write(" g = " + g);
```

Operatori poređenja

- Obavljaju poređenje dvije vrijednosti i kao rezultat vraćaju vrijednost logičkog tipa true ili false.
- Svaki dozvoljeni tip podataka, cjelobrojan, racionalni, karakter, String i logički tip može se upoređivati koristeći operatore == i !=.
- Samo numerički tipovi koriste ostale operatore.

Tabela operatora za poređenje

Operator	Upotreba	Opis
Jednakost	x == y	Rezultat je TRUE, ako su operandi x i y jednaki
Nejednakost	x != y	Rezultat je TRUE, ako su operandi x i y različiti
Veće	x > y	Rezultat je TRUE, ako je x veće od y
Veće ili jednako	x >= y	Rezultat je TRUE, ako je x veće ili jednako y
Manje	x < y	Rezultat je TRUE, ako je x manje od y
Manje ili jednako	x <= y	Rezultat je TRUE, ako je x manje ili jednako y
Jednakost (bez konverzije tipova)	x === y	Rezultat je TRUE, ako su x i y jednaki, ali bez konverzije tipova (moraju biti istog tipa!)
Različito (bez konverzije tipova)	x !== y	Rezultat je TRUE, ako su x i y različiti, ali bez konverzije tipova

Razlika između == i ===

- Operatori == i != obavljaju potrebnu konverziju podataka prije poređenja, ukoliko su operandi različitog tipa. Znači za ove operatore vrijednosti 5 (integer) i "5" (string) su iste, pa će poslije njihovog poređenja rezultat sa operatorom == biti TRUE, a sa operatorom != FALSE.
- S druge strane operatori === i !== ne obavljaju potrebnu konverziju podataka prije poređenja, ukoliko su operandi različitog tipa. Znači za ove operatore vrijednosti 5 (cio broj) i "5" (string) su različite, pa će poslije njihovog poređenja rezultat sa operatorom === biti FALSE, a sa operatorom !== TRUE.

- Primjer:

```
a = 4;
b = 1;
c = a < b;
d = a == b;
document.write(" c = " + c + "<BR>");
document.write(" d = " + d);
```

- Rezultat izvršavanja prethodnog primjera je

```
c = false
d = false
```

Kontrole toka

if - else

- konstrukcija omogućava izvršenje određenog bloka instrukcija ako je uslov konstrukcije ispunjen. Opšti oblik konstrukcije je:

```
if (boolean_izraz) blok1;
[else blok2;]
```

svaki od blokova, bilo u if ili u else dijelu može biti nova if-else konstrukcija. Primjer upotrebe ove konstrukcije je:

```
if (x == 8) {
    y = x;
} else {
    z = x;
    y = y * x
}
```

if - then - else

- Forma ovog operatora je:

```
expression ? statement1 : statement2
```

  gdje je izraz expression bilo koji izraz čiji rezultat je vrijednost logičkog tipa (na primjer: a>b)

- Ako je rezultat izraza true, onda se izvršava *statement1*, u suprotnom *statement2*.
- Primjer:

```
(x%2==0) ? document.write("paran broj") :
           document.write("neparan broj");
```

Složena if - else konstrukcija

```
if (mjesec == 1)
  ime_mjeseca = "Januar"
else if (mjesec == 2)
  ime_mjeseca = "Februar"
else if (mjesec == 3)
  ime_mjeseca = "Mart"
else if (mjesec == 4)
  ime_mjeseca = "Maj"  else
  ....
else if (mjesec == 12)
  ime_mjeseca = "Decembar"
```

switch

```
switch(mjesec) {
case 1: ime_mjeseca = "Januar"; break;
case 3: ime_mjeseca = " Mart"; break;
case 5: ime_mjeseca = "Maj"; break;
case 7: ime_mjeseca = "Jul"; break;
case 8: ime_mjeseca = "Avgust"; break;
case 10: ime_mjeseca = "Oktobar"; break;
case 12: ime_mjeseca = "Decembar"; reak;
case 4: ime_mjeseca = " April"; break;
case 6: ime_mjeseca = "Jun"; break;
case  9:  ime_mjeseca  =  "Septembar";  break;
case  11:  ime_mjeseca  =  "Novembar";  break;
case 2: ime_mjeseca = " Februar";
default: ime_mjeseca = " Nije naveden mjesec";
}
```

- Ukoliko se vrijednost izraza *mjesec* ne nalazi medju vrijednostima case 1,..., N, tada se izvršava blok naredbi **default**;

while petlja

- while petlja funkcioniše na taj način što se blok instrukcija unutar nje ponovljeno izvršava sve dok je uslov za ostanak u petlji, koji se nalazi na ulasku u petlju, ispunjen. Opšti oblik petlje izgleda ovako:

- ```
  while(uslov_ostanka) {
      tijelo_petlje;
   }
  ```
- Jednostavan primjer:

```
i=1
while(i<=10){
    document.writeln(i);
    i=i+1;
}
```
```

Izvršavanje while petlje

- Nakon izvršavanja ovog primjera dobiće se prikazani brojevi od 1 do 10.
- Treba napomenuti da će se u slučaju da uslov petlje nije ispunjen kada se prvi put ispituje uslov petlje, tijelo petlje neće izvršiti nijednom.
- Dakle, ovo je petlja koja se izvršava nijednom, jednom ili više puta.

do - while petlja

- Za razliku od prethodne petlje koja je imala uslov na svom početku, do-while petlja ima uslov na kraju. Prema tome, tijelo petlje će se sigurno izvršiti bar jednom.

```
do {
    tijelo_petlje
    [iteracija]
} while (uslov);


i=1
do {
    document.writeln(i);
    i++;   //i=i+1
} while(i<=10)
```

for petlja

- Opšti oblik for petlje izgleda ovako:

```
for( inicijalizacija; uslov; iteracija){
    tijelo_petlje;
}

for(i=0; i<10; i++){
    document.writeln(i);
}
```

- Promjenljiva *i* je privremena promjenljiva,
  a blok u kome je definisana je blok u kome se nalazi for petlja.

break naredba

- BREAK se koristi za skok na kraj bloka koji je označen labelom uz break ili na kraj bloka u kome se break nalazi, ako break stoji bez labele.
- Labele, pomoću kojih se označavaju blokovi, se formiraju kao i svi ostali identifikatori s tim što iza njih mora stajati dvotačka (:). Na primjer, sljedeći kod:

```
a: {
b: {
    c: {
      document.writeln("prije break-a"); //ovo se
      izvrsava!  break b;
      document.writeln("ovo nece biti prikazano"); //ovo se
      ne izvrsava!
      }
     } // ovde izlazi iz bloka kada uradi break b!
    document.writeln("poslije break-a"); //ovo se
    izvrsava!
}
```

return

- return se koristi za povratak iz funkcije na mjesto poziva. Ukoliko funkcija vraća neku vrijednost tada return mora slijediti izraz čiji je tip kompatibilan sa povratnim tipom funkcije. U suprotnom return izjava može stajati sama.

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

- Kao rezultat poziva funkcije dobija se:

  Kvadrat broja 5 je 25

continue

- Prelaz na sljedeću iteraciju petlje a da se dio koda prije njenog kraja ne izvrši. Za takve situacije se koristi continue.

```
for(i = 0; i < 10; i++) {
    document.write(i + " ");
    if (i % 2 == 0) continue; /*kada je broj paran
                                preskace sve naredbe
                                do kraja petlje */
    document.writeln("<br>");
}
```

- Zahvaljujući continue naredbi nakon izvršavanja ovog primjera dobija se:

```
01
23
45
67
89
```

Specijalne naredbe

for .. in

- Izvršava iteraciju po specifičnoj promjenljivoj za svaku osobinu (property) u okviru određenog objekta. Znači za svaku definisanu osobinu u oviru nekog objekta izvršava se niz naredbi definisan u okviru tijela ove petlje. Primjer:

```
niz = new Array ("Federer", "Djokovic", "Nadal")
for (var i in niz) {
    document.write(niz[i] + "<BR>");
}
```

function

- Deklariše JavaScript funkciju sa specificiranim parameterrima. Tipovi podataka mogućih parametara obuhvataju stringove, brojevi i objekte.

```
function ime([param1] [, param2] [...,paramN])
{
  //izrazi
}
```

with

- Definiše tip objekta za niz izraza. U okviru izraza dodjeljuje specifične vrijednosti za određene osobine objekta. Na primjer, matematičkim funkcijama mora prethoditi objekat Math. Sljedeći primjer podrazumeva Math ispred PI, COS() i SIN():

```
var a, x, y;
var r=10;
with (Math) {
      a  = PI * r * r;
      x  = r * cos(PI);
      y  = r * sin(PI/2);
}
```

Metode objekta Math

- round(0.60)
- ceil(0.49)
- floor(-4.60)
- random()
- min(-3,2)
- max(5,7)
- sqrt(25)
- abs(-3)
- PI, E
- sin(3.5), cos(2.7), tan(5)

Nizovi

Nizovi

- Sadrže skup podataka definisanih u jednoj promjenljivoj.
- Da bi se kreirao niz koristi se objekat **Array()**.
- Poziva se konstruktor, specijalni tip funkcije koja se koristi za kreiranje instance promjenljive i vraća referencu na kreiranu promjenljivu.

Array()

- Niz se kreira pomoću riječi new i konstruktora Array() na sljedeći način:

  ```
  var arrayName = new Array();
  ```

  ili ovako inicijalizovan elementima:

  ```
  arrayObjectName = new Array(element0, element1,..., elementN);
  ```

- Svaki podatak u nizu se naziva element.

Pozicija u nizu

- Indeks je numerička pozicija u nizu.
- Brojanje elemenata u okviru niza počinje sa indeksom nula (0).
- Pojedinačnom elementu se pristupa tako što se navodi njegov indeks u srednjim zagradama.
- Dodjeljivanje vrijednosti pojedinačnom članu niza se navodi isto kao kod promjenljive, samo se navodi i indeks elementa, na primjer:
  `niz[3] = "IP"`
- Veličina niza se može dinamički mijenjati.

Primjer sa definisanjem niza (1)

```
var auto = new Array(); //definisanje niza
auto[0] = "Audi";
auto[1] = "Volvo";
auto[2] = "BMW";

for (i = 0; i < auto.length; i++)
{
  document.write(auto[i] + " ");
}
```

Izlaz: Audi Volvo BMW

Primjer sa definisanjem niza (2)

```
var auto = new Array(2); //niz od 2 elementa
auto[0] = "Fiat";
auto[1] = "Peugeot";
auto[2] = "Citroen"; //niz ce da se prosiri
auto[3] = "Skoda";   //dinamicki

for (i = 0; i < auto.length; i++)
{
  document.write(auto[i] + " ");
}
```

Izlaz: Fiat, Peugeot, Citroen, Skoda

Primjer sa definisanjem niza (3)

```
var auto = new Array("Volkswagen", "Ford",
  "Mercedes");  //definisemo niz od 3 elementa
auto[1] = "Opel"; //mijenjamo drugi element
niza

for (i = 0; i < auto.length; i++)
{
  document.write(auto[i] + " ");
}
```

Izlaz: Volkswagen, Opel, Mercedes

Funkcija **sort()**

- Ova metoda uređuje (sortira) elemente niza direktno u izvornom nizu i vraća tako uređen niz.
- Kada se metoda sort() pozove bez argumenata, sortira elemente niza po abecednom redosljedu.
- Ako niz sadrži nedefinisane elemente, oni se stavljaju na kraj niza.

```
var niz = new Array("Marko", "Vesna", "Ana",
  "Stefan", "Darija", "Ivan");
document.write(niz + "<br />")
document.write(niz.sort() + "<br />")
```

Izlaz:

```
Marko,Vesna,Ana,Stefan,Darija,Ivan
Ana,Darija,Ivan,Marko,Stefan,Vesna
```

Funkcija **sort()** po numeričkom redu

- Da bi sortirali niz po redosljedu koji nije abecedni, moratmo metodi sort() proslijediti kao argument neku funkciju za poređenje.

```
function sortNumber(a, b){
   return a - b;
} //vraca vrijednost <0, 0 ili >0, zavisno od redosljeda

var numeric = new Array[3, 44, 1111, 222];
document.write(numeric.sort() + "<br />");
document.write(numeric.sort(sortNumber));
//prvo je abecedno, drugo numericko sortiranje!
```

Izlaz:

1111, 222, 3, 44

3, 44, 222, 1111

Funkcija **reverse()**

- Ova metoda obrće redosljed elemenata niza i vraća niz sa obrnuto raspoređenim elementima. Da bi to uradila, ne pravi novi niz s preuređenim elementima, već mijenja redosljed direktno u postojećem nizu.
- a[0] postaje a[n], a[1] postaje a[n-1],...

```
var niz = new Array("Marko", "Vesna", "Ana",
  "Stefan", "Darija", "Ivan");
document.write(niz + "<br />")
document.write(niz.reverse() + "<br />")
```

Izlaz:

```
Marko,Vesna,Ana,Stefan,Darija,Ivan
Ivan,Darija,Stefan,Ana,Vesna,Marko
```

Funkcija **concat()**

- Metoda concat() pravi i vraća nov niz koji sadrži elemente izvornog niza, s pridodatim argumentima te funkcije.
- Ako je neki od ovih argumenata niz, on se razlaže na svoje elemente koji se zasebno pridodaju rezultujećem nizu.

```
var brojevi = [1,2,3];
brojevi.concat(4,5);          //Rezultat: 1,2,3,4,5
brojevi.concat([4,5]);        //Rezultat: 1,2,3,4,5
brojevi.concat([4,5],[6,7]);  //Rezultat: 1,2,3,4,5,6,7
brojevi.concat(4, [5,[6,7]]); //Rezultat: 1,2,3,4,5,6,7
```

Funkcija **join()**

- Metoda join() konvertuje sve elemente niza u znakovne nizove i nadovezuje ih.
- Ukoliko se ne navede nijedan graničnik u obliku znakovnog niza, za razdvajanje se koristi zarez.

```
var brojevi = [1,2,3]; //Pravi novi niz sa ova 3 elem.
var s = brojevi.join();//Rezultat:  s=1,2,3
s = brojevi.join(" | ");    //Rezultat: s = 1 | 2 | 3
s = brojevi.join("#");  //Rezultat: s = 1#2#3
```

Funkcija **slice()**

- Metoda slice() vraća isječak, odnosno podniz navedenog niza. Ima dva argumenta koja određuju početak i kraj isječka koji se dobija.
- Rezultujući niz sadrži element određen prvim argumentom, i sve naredne elemente sve do elementa (ali ne i njega) određenog drugim argumentom.
- Ako je naveden samo jedan argument, rezultujući niz sadrži sve elemente počev od onog predviđenog tim argumentom, do kraja niza. Ako je negativan, gleda se od posljednjeg.

```
var brojevi = [1,2,3,4,5];
brojevi.slice(0,3);    //Rezultat: 1,2,3
brojevi.slice(3);      //Rezultat: 4,5
brojevi.slice(1, -1);  //Rezultat: 2,3,4
```

Funkcije **push()** i **pop()**

- Metode push() i pop() omogućavaju da se s nizovima radi kao da su stekovi.
- Metoda push() dodaje jedan ili više elemenata na kraj niza i vraća novu dužinu niza.
- Metoda pop() radi suprotno: briše posljednji element niza, skraćuje niz za jedan i vraća uklonjenu vrijednost.
- Oba metoda mijenjaju izvorni niz umjesto da prave izmijenjenu kopiju niza.

Primjeri za stek

```
var stek = new Array();  //prazan stek[]
stek.push(1,2); //stek[1,2]   Rezultat je 2
stek.pop();     //stek[1]     Rezultat je 2
stek.push(3);   //stek[1,3]   Rezultat je 2
stek.pop();     //stek[1]     Rezultat je 3
stek.push([4,5]); //stek [1,[4,5]] Rez. je 2
stek.pop();     //stek[1]     Rezutat je [4,5]
stek.pop();     //stek[]      Rezultat je 1
```

Funkcija **toString()**

- Nizovi imaju metodu toString(), koja konvertuje svaki element niza u znakovni niz i kao rezultat prikazuje listu tako dobijenih znakovnih nizova razdvojenih zarezima.
- Rezultat ne sadrži uglaste zagrade ili bilo koju drugu vrstu graničnika oko vrijednosti iz niza.
- Primjer:

```
[1,2,3].toString()          //Rezultat je '1,2,3'
["a", "b", "c"].toString()  //Rezultat je 'a,b,c'
[1, [2, 'c']].toString()    //Rezultat je '1,2,c'
```

Objekat Date

Date objekat

- Ovaj objekat se koristi kada je potrebno primijeniti određene operacije u kojima se koriste vremenske promjenljive.
- Svaki datum koji se pojavi u okviru nekog JavaScript programa se pamti kao broj koji predstavlja broj milisekundi između dobijenog datuma i ponoći 1. Januara 1970. god. po UTC vremenu.

Kreiranje Date objekta

- U programu kreiranje promjenljive ovog objekta se postiže na jedan od sljedećih načina:

```
dateObjectIme = new Date()
dateObjectIme = new Date("month day, year
   hours:minutes:seconds")
dateObjectIme = new Date(year, month, day)
dateObjectIme = new Date(year, month, day,
   hours, minutes, seconds)
```

Primjeri nekih datuma

- `today = new Date() //trenutno vrijeme i datum`
- `birthday = new Date("December 17, 1995 03:24:00")`
- `birthday = new Date(95, 11, 17)`
- `birthday = new Date(95, 11, 17, 3, 24, 0)`

Metode sa datumom (1)

- **Date.parse(datum)**
  Ovaj metod vraća broj milisekundi do navedenog datuma po lokalnom vremenu (od 1.1.1970 00:00:00).
  Primjer: datum.setTime(Date.parse("Aug 9, 2005")
- **Date.UTC(gg,mm,dd [,hh][,mh][,sec])**
  Vraća broj milisekundi od 1.1.1970 00:00:00 do datuma, prema Universal Coordinate Time (UCT).
  Primjer: gmtDatum = new Date(Date.UTC(96, 11, 1, 0, 0, 0))
- **datum.getDate()**
  Ovaj metod vraća dan u mjesecu (1-31) za navedeni datum. Primjer: datum = new Date("December 25, 2001 23:15:00"); dan = datum.getDate()
  Nakon izvršavanja primjera promjenljiva dan dobija vrijednost 25.
- **datum.getDay()**
  Metod vraća dan u nedelji (0-nedelja, 1-ponedeljak ... 6-subota) za navedeni datum.
  Primjer: datum = new Date("November 14, 2009 23:15:00");
  dan = datum.getDay()
  Nakon izvršavanja primjera promjenljiva dan dobija vrijednost 6, jer je 14.11.2009.god., bila subota.

Metode sa datumom (2)

- **datum.getHours()**
  Ovaj metod vraća sat za navedeni datum, moguće vrijednosti su brojevi u opsegu od 0 do 23.
  Primjer: datum = new Date("November 14, 2009 23:15:00"); sati = datum.getHours()
  Nakon izvršavanja primjera promjenljiva sati dobija vrijednost 23.
- **datum.getMinutes()**
  Ovaj metod vraća minute za navedeni datum, moguće vrijednosti su brojevi u opsegu od 0 do 59.
  Primjer: datum = new Date("November 14, 2009 23:15:00"); minuti = datum.getMinutes()
  Nakon izvršavanja primjera promjenljiva minuti dobija vrijednost 15.
- **datum.getMonth()**
  Ovaj metod vraća mjesec za navedeni datum (0-januar, 1-februar, ... 11-decembar).
  Primjer: datum = new Date("November 14, 2009 23:15:00");
  mjesec = datum.getMonth()
  Nakon izvršavanja primjera promjenljiva mjesec dobija vrijednost 10 (jer je januar 0!!!)

Metode sa datumom (3)

- **datum.getSeconds()**
  Ovaj metod vraća sekunde za navedeni datum, moguće vrijednosti su brojevi u opsegu od 0 do 59.
  Primjer: datum = new Date("November 14, 2009 23:15:08"); sekunde = datum.getSeconds()
  Nakon izvršavanja primjera promjenljiva sekunde dobija vrijednost 8.
- **datum.getTime()**
  Ovaj metod vraća vrijeme do navedenog datuma u milisekundama (od 1.1.1970 00:00:00).
  Primjer: datum = new Date("November 14, 2009 23:15:00"); proteklo = datum.getTime()
  Nakon izvršavanja primjera promjenljiva proteklo dobija vrijednost koja odgovara broju milisekundi od 1.1.1970 00:00:00 do 14.11.2009. 23:15:00.
- **datum.getTimezoneOffset()**
  Ovaj metod vraća razliku lokalnog vremena i GMT u minutima.
  Primjer: datum = new Date();
  razlikaSati = datum.getTimezoneOffset() / 60
  Nakon izvršavanja primjera promjenljiva razlikaSati dobija vrijednost -1.

Metode sa datumom (4)

- **datum.getFullYear()**
  Ovaj metod vraća godinu iz navedenog datuma (4 cifre).
  Primjer: datum = new Date("November 14, 2009 23:15:00"); godina = datum.getYear()
  Nakon izvršavanja primjera promjenljiva godina dobija vrijednost 2009.
- **datum.setDate(brojDana)**
  Ovaj metod postavlja dan u mjesecu za navedeni datum. Argument metoda je broj u opsegu od 1 do 31.
  Primjer: datum = new Date("July 27, 1960 23:30:00"); datum.setDate(24)
  Nakon izvršavanja primjera promjenljiva datum dobija vrijednost 24.7.1960 23:30:00.
- **datum.setMonth(brojMjeseca)**
  Ova metoda postavlja mjesec za navedeni datum.
  Argument je broj od 0 do 11 (0-januar, 1-februar, …)
  Primjer: datum = new Date ("July 27, 1960 23:30:00"); datum.setMonth(1)
  Nakon izvršavanja primjera promjenljiva datum dobija vrijednost 24.2.1960 23:30:00.

Metode sa datumom (5)

- **datum.setHours(brojSata)**
  Ovaj metod postavlja broj sati za navedeni datum.
  Argument metoda je broj u opsegu od 0 do 23.
  Primjer: datum = new Date("July 27, 1960 23:30:00"); datum.setHours(7)
  Nakon izvršavanja primjera promjenljiva datum dobija vrijednost 27.7.1960 07:30:00.
- **datum.setMinutes(brojMinuta)**
  Ovaj metod postavlja broj minuta za navedeni datum.
  Argument metoda je broj u opsegu od 0 do 59.
  Primjer: datum = new Date("July 27, 1960 23:30:00"); datum.setMinutes(35)
  Nakon izvršavanja primjera promjenljiva datum dobija vrijednost 27.7.1960 23:35:00.
- **datum.setSeconds(brojSekundi)**
  Ovaj metod postavlja sekunde za navedeni datum.
  Argument metoda je broj u opsegu od 0 do 59.
  Primjer: datum = new Date("July 27, 1960 23:30:00"); datum.setSeconds(35)
  Nakon izvršavanja primjera promjenljiva datum dobija vrijednost 27.7.1960 23:30:35.

Metode sa datumom (6)

- **datum.setFullYear(brojGodine)**
  Ovaj metod postavlja godinu za navedeni datum.
  Argument metoda je broj u opsegu od 0 do 9999.
  datum = new Date("July 27, 1999 23:30:00");
  datum.setYear(2017)
  Nakon izvršavanja primjera promjenljiva datum dobija vrijednost 27.7.2017 23:30:00.
- **datum.setTime(vrijeme)**
  Ovaj metod definiše novi datum. Argument metoda je broj milisekundi od 1.1.1970 00:00:00
- **datum.toGMTString()**
  Ovaj metod vrši konverziju datuma u GMT string iz lokalne vremenske zone.
  Primjer: datum = new Date("December 25, 2001 23:15:00"); datum.toGMTString()
  Nakon izvršavanja primjera promjenljiva datum dobija vrijednost "Tue, 25 Dec 2001 22:15:00 UTC"
- **datum.toLocaleString()**
  Ovaj metod vrši konverziju datuma u lokalni datum string iz GMT.
  Primjer: datum.toLocaleString()
  Nakon izvršavanja primjera promjenljiva datum dobija vrijednost u zavisnosti od podešavanja računara na kome se pokreće.

String objekat

Šta je String?

- Ovaj objekat se koristi da bi se efikasnije obradio niz karaktera, što objekat tipa Sting u suštini i jeste.
  U okviru JavaScript jezika String se defniše kao niz karaktera između apostrofa ili između dvostrukih navodnika: "neki String" ili 'neki String'. I u okviru ovog objekta postoje dostupni metodi koji se mogu koristiti.

escape ("string")

- Ova funkcija kao rezultat vraća ASCII kodove karaktera u okviru argumenta.

  Primjer:

  ```
  y = escape("!#")
  ```

- Nakon izvršavanja primjera promjenljiva y dobija vrijednost "%21%23", jer su ASCII kodovi za simbole ! i # 21 i 23.

eval ("izraz")

- Ova funkcija izračunava vrijednost izraza koji je definisan kao argument funkcije.

  Primjer:

```
var x = eval("4+5-8")
```

  Nakon izvršavanja primjera promjenljiva x dobija vrijednost 1.

linkTekst.**link**(linkURL)

- Ovaj metod kreira tekst linkTekst koji predstavlja HTML link na neku drugu stranicu, čiji je adresa definisana sa argumentom linkURL (dejstvo kao i HTML taga <a href="…">). Primjer:

```
var naziv = "ETF sajt";
var URL = "http://www.ucg.ac.me/etf";
document.write("Ovo je " + naziv.link(URL))
```

- Nakon izvršavanja primjera na stranici će se pojaviti tekst "Ovo je ETF sajt", koji će predstavljati vezu ka stranici www.ucg.ac.me/etf.

parseInt(StringBroj [,osnova])

- Ova funkcija kao rezultat vraća cio broj dobijen konverzijom argumenta stringBroj koji je tipa String u brojnom sistemu sa osnovom koju definiše argument osnova.
- Ovaj argument je opcioni i ako se ne navede podrazumijeva se osnova 10, tj. dekadni brojni sistem. Primjer:

```
x = parseInt("17", 8);
y = parseInt("15", 10);
```

Nakon izvršavanja primjera i promjenljiva x i promjenljiva y dobija vrijednost 15.

string.**big()**

- Ovaj metod prikazuje string sa uvećanim slovima (veća veličina i boldovan font).
- Primjer:

```
"Dobar dan!".big();
```

string.**bold()**

- Ovaj metod prikazuje podebljan string (ima isto dejstvo kao HTML tag <b>).
- Primjer:

```
"Dobar dan!".bold();
```

string.**italics()**

- Ovaj metod prikazuje string kurziv stilom (ima isto dejstvo kao HTML tag <i>).
- Primjer:

```
"Dobar dan!".italics();
```

string.**fontcolor()**

- Ovaj metod prikazuje string u određenoj boji (ima isto dejstvo kao HTML tag <font color=...>).
- Primjer:

```
"Dobar dan!".fontcolor("blue");
```

string.**fontsize()**

- Ovaj metod prikazuje string u određenoj veličini (ima isto dejstvo kao HTML tag <font size=...>).
- Primjer:

```
"Dobar dan!".fontsize(7);
```

string.**charAt(**broj**)**

- Ovaj metod kao rezultat vraća znak na navedenoj poziciji. Pozicije unutar stringa se računaju sa lijeve na desnu stranu i prva pozicija ima indeks 0. U okviru svakog objekta tipa String postoji i osobina (property) length koja je jednaka broju karaktera u posmatranom stringu. Korišćenjem ovog podatka može se odrediti i indeks posljednjeg karakera u stringu, a to je vrijednost string.length-1. Primjer:

```
x= "Dobar dan!".charAt(4);
y= "Dobar dan!".charAt(6);
```

Nakon izvršavanja primjera promjenljiva x dobija vrijednost ’r’, a promjenljiva y je ’d’.

string.**indexOf**(traziString, [odPozicije])

- Ovaj metod vraća broj pozicije na kojoj je prvi put pronađen argument tipa String traziString.
  U slučaju da se trazeni string ne nalazi u početnom stringu kao rezultat se vraća vrijednost -1. Ako postoji i drugi argument odPozicije, tada će se pretraga izvršavati od zadate pozicije. Primjer:

```
x ="Dobar dan!".indexOf("r")
y ="Dobar dan!".indexOf("a",4)
```

  Nakon izvršavanja primjera promjenljiva x dobija vrijednost 4, a promjenljiva y je 7.

string.**lastIndexOf**(traziString, [doPozicije])

- Ovaj metod vraća broj pozicije na kojoj se posljednji put pojavljuje argument tipa String traziString. U slučaju da se traženi string ne nalazi u početnom stringu kao rezultat se vraća vrijednost -1. Ako postoji i drugi argument doPozicije, tada će se pretraga izvršavati do zadate pozicije. Primjer:

```
x = "Dobar dan!".lastIndexOf("a")
y = "Dobar dan!".lastIndexOf("a", 6)
```

Nakon izvršavanja primjera promjenljiva x dobija vrijednost 7, jer je to posljednje pojavljivanje stringa "a", a promjenljiva y je 3, jer je to posljednje pojavljivanje stringa "a" do pozicji 6.

string.**strike()**

- Ovaj metod prikazuje string koji je precrtan (ima isto dejstvo kao HTML tag <strike>).
- Primjer:

```
"Dobar dan!".strike();
```

string.**sub()**

- Ovaj metod prikazuje string koji je prikazan kao indeks (ima isto dejstvo kao HTML tag <sub>).
- Primjer:

```
"Zdravo".sub();
```

string.**sup()**

- Ovaj metod prikazuje string koji je prikazan kao eksponent (ima isto dejstvo kao HTML tag <sup>).
- Primjer:

```
s="x"+"2".sup()
document.write(s)
```

string.**substring(**prvi, posljednji**)**

- Ovaj metod vraća dio stringa počev od pozicije prvi do pozicije posljednji, tj. uzima redom karaktere na pozicijama prvi, prvi + 1, prvi + 2, ..., posljednji -2, posljednji – 1.

```
x = "Dobar dan!".substring(6,9)
```

- Nakon izvršavanja primjera promjenljiva x dobija vrijednost "dan", jer su to karakteri na pozicijama 6, 7 i 8.

substring i substr - razlike

- Razlikuju se u drugom argumentu!
- **substring (prvi_karakter, posljednji_karakter)**

```
x = "Internet".substring(1,3)
```

- Ova funkcija vraća: nt

- **substr (prvi_karakter, dužina)**

```
y = "Internet".substr(1,3)
```

- Ova funkcija vraća: nte

string.**toLowerCase()**

- Ovaj metod izvrši konverzija svih karaktera u okviru stringa u mala slova. Primjer:

```
x = "Dobar dan!".toLowerCase()
```

- Nakon izvršavanja primjera promjenljiva x dobija vrijednost "dobar dan!", jer je izvršena konverzija svih karaktera u mala slova.

string.**toUpperCase()**

- Ovaj metod izvrši konverzija svih karaktera u okviru stringa u velika slova. Primjer:

```
x = "Dobar dan!".toUpperCase()
```

- Nakon izvršavanja primjera promjenljiva x dobija vrijednost "DOBAR DAN!", jer je izvršena konverzija svih karaktera u velika slova.

unescape("kodovi"**)**

- Ova funkcija kao rezultat vraća ASCII znakove navedenih kodova u okviru argumenta funkcije. Primjer:

```
x = unescape("%21%23")
```

- Nakon izvršavanja primjera promjenljiva x dobija vrijednost /"!#"/, jer su simboli ! i # kodovani sa ASCII kodovima 21 i 23.

Rad sa uzorcima

Pattern Matching

Definisanje uzorka (1)

- JavaScript funkcije se često upotrebljavaju za provjeru unijetih podataka od strane klijenta.

  JavaScript ima razvijenu podršku za razne vrste provjera i one se obavljaju na klijentskoj strani.
- Uzorak se još naziva i regularni izraz (regular expression) i može se definisati na dva načina:
  - `var ime_uzorka = new RegExp("primjer")`
  - `var ime_uzorka = /primjer/`
- Na oba načina se formira objekat uzorka koji se naziva `ime_uzorka` i kome odgovara svaki string koji u sebi sadrži podstring primjer.

Definisanje uzorka (2)

- `var uzorak = new RegExp("HTML")`
- `var uzorak = /HTML/`

- Prvim se poziva RegExp konstruktor, a u drugom se sadržaj uzorka piše između početnog i krajnjeg znaka / (slash)

- `var uzorak = new RegExp("s$")`
- `var uzorak = /s$/`

- Simbol $ označava kraj stringa. Sada promjenljiva uzorak odgovara bilo kom stringu koji se završava sa s.

Karakteri koji se koriste u uzorku

Karakter	Predstavlja
alfanumerički znak	sebe
\d	Bilo koja cifra od 0 do 9
\D	Bilo koji karakter koji nije cifra
\w	Bilo koji karakter (slova a-z, A-Z, 0-9 i _)
\W	Neki specijalni karakteri (na primjer: @)
\s	Neki bijeli karakter (tab, nova linija, …)
\S	Neki karakter koji nije bijeli
.	Bilo koji karakter (osim nove linije)
[…]	Bilo koji karakter naveden između []
[^…]	Bilo koji karakter koji nije naveden između []
[\b]	Brisanje unazad

Znakovi za ponavljanje

Karakter	Predstavlja
{n,m}	Prethodni element se ponavlja najmanje n puta i ne više od m puta
{n,}	Prethodni element se ponavlja n ili više puta
{n}	Prethodni element se ponavlja TAČNO n puta
?	Prethodni element se ne pojavljuje ili se pojavljuje samo jednom. Ekvivalentno izrazu {0,1}
+	Prethodni element se ponavlja jednom ili više puta. Ekvivalentno izrazu {1,}
*	Prethodni element se ne pojavljuje ili se ponavlja više puta. Ekvivalentno izrazu {0,}

Primjeri (1)

- `/[abc]/`
  - predstavlja jedno pojavljivanje slova a ili slova b ili slova c. String "c" ispunjava uslove definisane uzorkom, ali string "s" ne ispunjava definisane uslove.
- `/^[abc]/`
  - predstavlja karakter koji nije slovo a ili slovo b ili slovo c

Primjeri (2)

- Primjer za petocifreni poštanski broj
  - `/\d\d\d\d\d/`
  - `/\d{5}/`
- `/\d{2,4}/`
  - uzorak koji označava 2, 3 ili 4 pojavljivanja cifara
- `/\w{3}\d?/`
  - uzorak koji označava tačno 3 pojavljivanja slova i opciono jedne cifre
  - primjer: web8, ana, iva

Primjeri (3)

- `/\s+Internet\s+/`
  - uzorak koji označava string "Internet" sa jednim ili više prostora prije ili poslije stringa.
- `/[a-z]+\d+/`
  - uzorak koji označava jedno ili više malih slova praćenih jednom ili više cifara.

Znakovi za alternativu, grupisanje i sidrenje

Karakter	Predstavlja
|	Alternative. Pojavljuje se ili samo desni ili samo lijevi dio uzorka u stringu.
(. . .)	Grupisanje simbola u jedan objekat nad kojim se mogu koristiti *, +, ?, |
^	Pretragu uzorka na početku znakovnog niza
$	Pretragu uzorka na kraju znakovnog niza

Primjeri (4)

- `/ab|cd|ef/`
  - uzorak koji označava pojavljivanje ab ili cd ili ef
- `/\d{3}|[A-Z]{4}/`
  - uzorak koji označava pojavljivanje 3 cifre ili 4 velika slova
- `/java(script)/`
  - uzorak koji označava pojavljivanje stringa "java" ili stringa "javascript"

Primjeri (5)

- `/(ab|cd)+|ef/`
  - uzorak koji označava pojavljivanje stringa "ef"
    ili pojavljivanje jednom ili više puta stringa "ab"
    ili pojavljivanje jednom ili više puta stringa "cd"

Atributi

Atribut	Značenje
i	case-insensitive ispitivanje
g	globalno izvršavanje (pronalaženje svih pojavljivanja definisanog uzorka)
M	rad sa više linija

Ispitivanje uzoraka pomoću metoda

- `search()` - traži određeni uzorak u tekstu
- `replace()` - traži određeni uzorak u tekstu i zamjenjuje ga nekim stringom
- `match()` - formira niz koji sadrži samo traženi uzorak
- `split()` - dijeli string određenim uzorkom (uzorak je kao separator)

search() metod

- Ispituje da li u stringu postoji definisani uzorak
- Rezultat je pozicija prvog pojavljivanja uzorka ili -1, ako ne pronađe uzorak
- Primjer1:

```
x = /Script/i
y = "JavaScript".search(x);
```

- Kao rezultat izvršavanja ovog primjera

  promjenljiva y će dobiti vrijednost 4
- Ovaj metod ne podržava globalnu pretragu, tj. ignoriše upotrebu atributa g u okviru definicije uzorka

replace() metod

- Ispituje da li u stringu postoji uzorak i ako postoji zamijeni uzorak unutar stringa nekim drugim stringom
- Metod ima dva argumenta, prvi je uzorak, a drugi je string koji treba da zamijeni uzorak
- Primjer2:

```
"html: HTML se uci na IP".replace(/HTML/,"JAVA")
```

- Ovaj metod podržava globalnu primjenu, pa ako se u okviru uzorka navede i g atribut, ovaj metod će izvršiti zamjenu svakog uzorka koji pronađe u okviru stringa

match() metod

- Vrlo sličan search() metodu, samo umjesto pozicije vraća niz elemenata sa svim pojavljivanjima definisanog uzorka, ako je definisan atribut g.
- Primjer:
  - `"1 plus 2 jednako je 3".match(/\d+/g)`
  - Rezultat:

    ["1", "2", "3"], jer je uzorak definisan kao pojavljivanje cifre, jednom ili više puta, u cijelom stringu

split() metod

- Ima jedan argument - uzorak!
- Rezultat je niz koji se dobija kada se string podijeli argumentom (uzorkom) kao separatorom
- Primjer:
  - `"123, 456, 2009   ,   3141”.split(/\s*,\s*/)`
  - Rezultat je ["123", "456", "2009", "3141"], jer je uzorak definisan sa određenim brojem blanko znakova prije i poslije zareza, uključujući zarez

Metodi objekta RegExp

- `exec()`
- `test()`

exec()

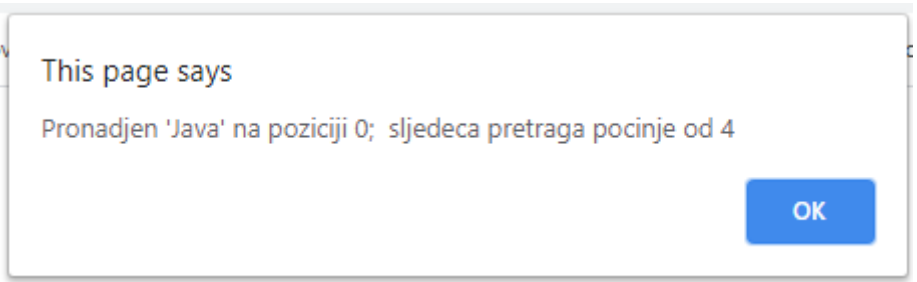
- Ovaj metod je sličan string metodu match(). Razlika je u tome što kod ovog metoda argument je string, a primenjuje se na uzorku, dok je kod match() obrnuto.
- Rezultat izvršavanja exec() je niz koji sadrži rezultate ispitivanja, definisane na isti način kao i metod match().
- Za razliku od match() metoda exec() vraća isti rezultat ako postoji atribut g i ako ne postoji.

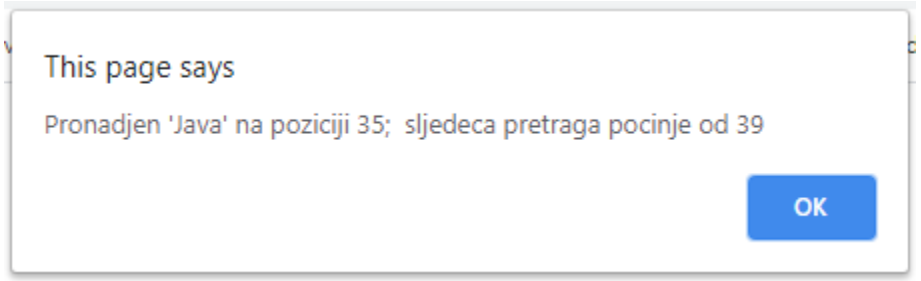
lastIndex

- Ako se metodi exec() proslijedi regularni izraz sa indikatorom g, u svojstvo lastIndex objekta klase Regex upisuje se **pozicija prvog znaka poslije odgovarajućeg podniza**.
- Kada se metoda exec() ponovo pozove za isti regularni izraz, počinje pretraživanje od pozicije zadate vrijednošću svojstva lastIndex.
- Ovo ponašanje omogućava da ponovljene pozive metode exec() izvršavamo kroz petlju, kako bi se pristupilo svim podnizovima u znakovnom nizu podudarnim sa regularnim izrazom.

Primjer

```
var pattern = /Java/g;
var text = "JavaScript je mnogo zabavniji nego
   Java!";
var result;
while((result = pattern.exec(text)) != null)
{
   alert("Pronadjen '" + result[0] + "'" +
   " na poziciji " + result.index +";
   sljedeca pretraga pocinje od " +
   pattern.lastIndex);
}
```





test()

- Ova metoda se ponaša kao exec() tj. vraća vrijednost true, ako njen rezultat nije null.
- Počinje da pretražuje znakovni niz počevši od pozicije zadate svojstvom lastIndex (isto kao exec()!!!) i ako nađe odgovarajući podniz, zadaje tom svojstvu vrijednost pozicije prvog znaka neposredno poslije nađenog podniza.
- Svojstvo lastIndex postoji samo ako regularni izraz ima indikator g, u suprotnom metode exec() i test() zanemaruju svojstvo lastIndex bez indikatora g.

JavaScript i forme

- Rad sa događajima
- Rad sa više prozora
- Cookie
- Rad sa pauzama i intervalima

HTML i JavaScript

- Programski jezik JavaScript je svoju popularnost stekao **mogućnošću da pristupa elementima forme, čita njihove vrijednosti, obrađuje ih i definiše nove vrijednosti elemenata**. Takođe iskorišćena je i osobina HTML jezika da prepozna korisnikovu akciju i reaguje na nju.

- **Čitač može da prepozna svaku akciju korisnika, bilo da ona potiče od miša ili tastature.**

Rad sa događajima

Događaj	Nastaje kada korisnik...	Kod
blur	izađe iz fokusa elementa forme	onBlur
click	klikne na element forme ili link	onClick
change	promijeni vrijednost izabranog elementa forme	onChange
focus	uđe u fokus nekog elementa forme	onFocus
load	učita stranicu u browser	onLoad
mouseover	pređe pokazivačem miša preko linka	onMouseOver
mouseout	izađe pokazivačem miša sa određene površine ili linka	onMouseOut
select	izabere polje elementa forme	onSelect
submit	izvrši slanje forme	onSubmit
unload	napusti stranicu	onUnload
reset	resetuje sadržaj forme	onReset
error	dobije grešku prilikom učitavanja slike ili stranice	onError
abort	prekine učitavanje slike ili stranice	onAbort

Primjer - Događaji miša

- Napisati JavaScript program koji registruje:
  - prelazak pokazivača miša preko linka,
  - odlazak pokazivača miša sa nekog linka,
  - broj prelazaka pokazivača miša preko nekog dugmeta (realizovati JS funkcijom),
  - dvostruki klik miša koji će zatvoriti prozor.

Primjer - Događaji miša (1)

```
<script language="JavaScript">
var counter=0;
function closeWindow(){
   alert("Gotovo je!");
   window.close();
}
function mouseOverCounter(){
   counter++;
   if(counter==1){
        alert(counter + " prelazak preko dugmeta!");
   }
   else{
        alert(counter + " prelaska preko dugmeta!");
   }
}
</script>
```

Primjer - Događaji miša (2)

```
<body onDblClick=" closeWindow()";>
<p><font face="arial" size=3>Dva puta kliknite da bi ste
   zatvorili prozor!
<p>Registruje se prelazak misa preko linka.
<a href="#"
   onMouseOver="alert('Dogadjaj:onMouseOver');">onMouseOver</a>
<p>Registruje se odlazak misa sa linka.<a href="#"
   onMouseOut="alert('Dogadja:onMouseOut');">onMouseOut</a>
<p>Kada se mis pozicionira na dugme i pomjeri poziva se
   funkcija<br>koja broji koliko puta se desio ovakav dogadjaj.
<form>
<input type="button" value="onMouseMove"
onMouseMove="mouseOverCounter();">
</form>
</body>
```

Vrijednosti elementa forme

- JavaScript može i da pročita vrijednost proizvoljnog elementa forme. Vrijednosti elementa forme se prilazi u opštem slučaju na sljedeći način:

  document.imeForme.imeElementa.value

  gdje je document službena riječ,
  imeForme ime forme u okviru koje se nalazi element, čijoj se vrijednosti pristupa,
  imeElementa ime elementa (name ili id) i value službena riječ za vrijednost tog elementa.

Primjer - Sabiranje dva broja

- Napisati JavaScript program kojim možete da unesete dva broja u dva tekstualna polja, a zatim klikom na dugme "SABERI” JavaScript funkcija izračuna zbir ta dva broja i taj rezultat ispiše u trećem tekstualnom polju.

Primjer - Sabiranje dva broja (1)

```
<SCRIPT LANGUAGE="JavaScript">
   function Saberi() {
       var  br1  =  document.mojaforma.X.value  -  0;
       var  br2  =  document.mojaforma.Y.value  -  0;
       var ukupno = br1 + br2;
   //sabiranje br1 + br2
   // i smijestanje rezultata u promjenljivu ukupno
       document.mojaforma.zbir.value = ukupno;
   }
</SCRIPT>
```

Primjer - Sabiranje dva broja (2)

```
<FORM METHOD="post" NAME="mojaforma">
   X   =  
    <INPUT TYPE="text" NAME="X" SIZE=5> <br>
    Y   =  
    <INPUT TYPE="text" NAME="Y" SIZE=5> <br><br>
    <INPUT TYPE="button" VALUE="SABERI" NAME="dugme"
      onClick="Saberi()"> <br><br>
    <hr>
    REZULTAT   =  
    <INPUT TYPE="text" NAME="zbir" SIZE=5>
    <br>
</FORM>
```

Rad sa više prozora

- JavaScript omogućava da se iz jednog prozora formira, kontroliše ili mijenja sadržaj u okviru drugog prozora.

Poruke upozorenja (alerti)

- Alerti se koriste unutar HTML stranice kada se želi prikazati određeno obavještenje - novi manji prozor

```
<form action="">
<input type="button" value="Pritisni me" onClick="alert()" />
</form>
<script type="javascript">
function alert()
{
alert ("Prvi red "+"i ovde je prvi red - \n Drugi red!");
}
</script>
```

- U okviru alerta korišćena je oznaka za prelazak u novi red : "\n"

window

- U ovaj objekat su uključene window metode za manipulaciju sa istim
- `window.open()` otvara novi prozor pretraživača
  - `WindowName=window.open("URL","WindowName","Feature List");` WindowName je promjenljiva. Koristeći ovu promjenljivu možemo pozivati funkcije ili pristupati elementima istog
  - "URL" je url za novi prozor. Ako je prazan ništa neće biti učitano.
  - "WindowName" je ime prozora koje se koristi pri pozivu nekih funkcija
  - "Feature List" je opcioni parametar. Čuva listu parametara odvojenih zarezima.
- Ovaj metod uključuje **width**, **height**, zatim nekoliko veličina koje mogu biti **yes(1)** ili **no(0)**
- **toolbar**, **location**, **directories**, **status**, **menubar**, **scrollbars**, **resizable**.
- `SmallWin = window.open("","small", "width=100,height=120,toolbar=0,status=0");`

Primjer - Novi prozor (1)

```
<script language="javascript">
 function prozor( page, width, height, top, left ) {
   var yes         = 1;
   var no          = 0;
   var  menubar   =  no;
   var scrollbars =  no;
   var locationbar = no;
   var directories = no;
   var resizable   = no;
   var statusbar   = no;
   var toolbar     = yes;
   features = "" +
        "width=" + width + "," +
        "height=" + height + "," +
        "top=" + top + "," +
        "left=" + left + "";
```

Primjer - Novi prozor (2)

```
features += "" +
        (menubar ? ",menubars" : "") +
        (scrollbars ? ",scrollbars" : "") +
        (locationbar ? ",location" : "") +
        (directories ? ",directories" : "") +
        (resizable ? ",resizable" : "") +
        (statusbar ? ",status" : "") +
        (toolbar ? ",toolbar" : "");

   var reftt = window.open( page, 'fullPopup', features );
}
</script>
```

Kolačići

- **Kolačić (Cookie)** je mali imenovani segment podataka koji Web čitač pamti i koji je povezan sa određenom Web stranom ili Web lokacijom.
- Obično se koristi da bi se podaci unijeti na jednoj strani koristili na drugoj, tj. da bi čitač mogao da ponovi korisničke parametre ili druge promjenljive stanja kada korisnik napusti stranu i kasnije se vrati.
- U JS se koristi svojstvo ***cookie*** objekta tipa Document

Cookie

- Format koji cookie fajl mora da zadovolji je:
  - *ime = vrijednost* [;EXPIRES=*datum*] [;DOMAIN=*imeDomena*] [;PATH=*putanja*] [;SECURE]
  - *ime* - ime koje definiše upisani cookie;
  - *vrijednost* - informacija koja se želi zapamtiti;
  - *datum* - datum koji definiše do kada cookie ostaje upisan na klijentskoj mašini;
  - *imeDomena* - definiše jedini domen sa kog cookie može da se čita i
  - da mu se mijenja vrijednost;
  - *putanja* - definiše jedinu putanju sa koje cookie može da se čita i da mu se mijenja vrijednost;
  - SECURE - upis i čitanje cookie se izvršava preko posebnih, bezbjednijih linija;
  - Opcije EXPIRES, DOMAIN, PATH, SECURE su opcione i nije bitan
  - redosljed u kom se pojavljuju;

Cookie

Čitanje vrijednosti:

var citamCookie = document.cookie

Upis na klijentskoj strani:

```
document.cookie = "primjerCookie="
+  vrijednostKojuPamtim + ";secure"
```

Cookie

```
<html>
<head>
<script language="javascript">
function postavljanjeCookie(){
      document.cookie = 'Cookie je='+document.forma1.imeCookie.value;
}
function prikazCookie(){
      alert(document.cookie);
}
</script>
</head>
<body>
<h1>Cookie 1</h1>
<h2>Postavljanje i pregled cookie</h2>
<form name="forma1">
  <p>
     <input name="imeCookie" type="text" id="imeCookie" size="20">
  </p>
  <p>
     <input type="button" value="Upisite ime" name="B1"
onClick="postavljanjeCookie()">
    <input  type="button" value="Prikazi cookie" name="B2"
onClick="prikazCookie()">
  </p>
</form>
</body>
</html>
```

Rad sa pauzama i intervalima

- Korišćenjem metoda objekta Window može se realizovati kod koji izvršava automatski.

setTimeout()

- Koristi se u okviru JavaScript-a za izvršavanje određenog koda nakon specificiranog vremenskog intervala;
- Kod koji se definiše u okviru setTimeout() metoda, izvršava se samo jednom;

- Sintaksa upotrebe setTimeout() metoda:

```
var variable = setTimeout("funkcija()",
  brojMiliSekundi);
```

clearTimeout ()

- Metod se koristi da bi se prekinuo metod setTimeout() prije nego što se izvrši
- clearTimeout() sadrži jedan argument:
  - Promjenljivu koja predstavlja poziv metoda setTimeout()

- Druga dva metoda JavaScripta koji automatski izvršavaju određeni kod su:
  - setInterval() metod
  - clearInterval() metod

setInterval()

- setInterval():
  - sličan metodu setTimeout(),
    OSIM što ponavlja izvršavanje istog koda!

clearInterval()

- clearInterval():
  - Koristi se da bi prekinuo izvršavanje metoda setInterval() na isti način kao što metod clearTimeout() poništava poziv metoda setTimeout().

Objekat History

- U okviru web čitača održava se interna lista (poznata pod imenom history list) svih dokumenata koji su bili otvarani tokom trenutne sesije Web čitača.

- Svaki prozor Web čitača i frejm sadrže svoj sopstveni objekat History, koji predstavlja internu listu dokumenata.

URL i History

- U okviru history list ne mogu se vidjeti posjećeni URL-ovi, ali se može napisati script koji koristeći ovu listu prolazi kroz Web stranice koje su bile otvarane tokom sesije Web čitača.

Neka ograničenja

- U okviru Internet Explorera, može se koristiti JavaScript kod da bi se pretraživala history list
  - Jedino ako se trenutna Web page stranica nalazi na istom domenu kao i Web stranica koja sadrži JavaScript kod koji pokušava da pretraži listu

Objekat Location

- Dozvoljava da se promijeni adresa nove Web stranice pomoću JavaScript koda
- Jedan razlog za ovu promjenu je i mogućnost
  - Da se izvrši redirekcija korisnika stranice na drugu stranicu ili drugi URL
- Kada se koristi metod ili property objekta Location mora se
  - Uključiti i referenca na sam Location objekat

Objekat Navigator

- Koristi se da bi se dobile informacije o trenutnom Web čitaču.
- Netscape i Internet Explorer sadrže jedinstvene metode i properties objekta Navigator koje se ne mogu koristiti sa ostalim čitačima.
- Najviše se koriste metodi pomoću kojih se prepoznaje tip web čitača koji se koristi.

Frame i Target

- Atribut Target definiše koji frejm ili prozor Web čitača će prikazati dokument:

  – Bazira se na vrijednosti prikazanoj u okviru target atributa

  <a> elementa ili vrijednosti u okviru atributa name

  <frame> elementa

<base>

- Atribut target se koristi i sa <base> elementom
  - da specificira default target za sve linkove u okviru dokumenta
    - Koristi data imena prozora ili frejma