

JavaScript

UVOD

Zašto JavaScript

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

...prve tehnologije.. CGI(još uvek aktuelan)

- ❑ 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
-

- ❑ Decembra 1995.god, Netscape i Sun predstavili jezik JavaScript 1.0, originalno nazvan LiveScript. Kod pisan na tom jeziku je mogao da se izvršava u okviru Netscape Navigator 2 čitača.
 - ❑ Ovaj jezik je omogućio ne samo formatiranje podataka na klijentskoj strani, već i obradu i dinamičko izvršavanje stranica. Treba napomenuti da je implementiran deo jezika koji se izvršavao na serverskoj strani, čime je omogućeno da se ista tehnologija koristi na obe strane aplikacije, ali ovaj deo JavaScript jezika nije dostigao veću popularnost i neće se razmatrati u okviru ovog teksta.
-

standardizacija

- ❑ Sledeći korak u popularnosti JavaScript jezika je bila Microsoft-ova implementacija u okviru Internet Explorer 3 čitača, pri čemu je ova verzija od strane Microsoft-a nazvana JScript. JScript je bio baziran na javnoj dokumentaciji Netscape-a bio je skoro identičan JavaScript jeziku.
 - ❑ Konačno telo ECMA (European Computer Manufacturers Association) JavaScript postao Nestcape-ova implementacija ovog standarda, a JScript Microsoft-ova. I danas Netscape-ova i Microsoft-ova verzija standarda su identične u preko 95% slučajeva.
-

JavaScript je objektno

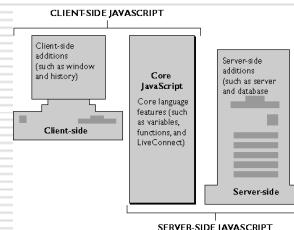
- bazirani,
- platformski neutralan,
- višekorisnički jezik

koji programeru omogućava mnogo veću funkcionalnost na klijentskoj strani.

Objektno baziran...

- ... znači da nisu svi koncepti objektno orijentisanih jezika realizovani u ovom jeziku, da je veoma limitiran rad sa nasleđivanjem, važenjem i funkcionalnošću samih objekata. Sa druge strane postoje 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 (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 softversko okruženje u pitanju. Zato je i veličina programa pisanih u ovom jeziku dovoljno mala da može da se izvršava i na mašinama sa lošijim performansama.

JavaScript i HTML

- Još jedna od prednosti JavaScript jezika je njegova integrisanost sa HTML-om. U okviru jedne stranice je moguće je 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 uključiti u okviru HTML stranice na dva načina.
 - Prvi je direktnim pisanjem koda u okviru stranice.
 - `<SCRIPT LANGUAGE="JavaScript">`
 - ...JavaScript kod...
 - `</SCRIPT>`
 - Primer - *HelloJavaScript.HTM*L

- 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>`gde je `JSkod.js` dokument koji sadrži željene JavaScript funkcije. Čitači podrazumevaju da je korišćeni skript jezik JavaScript

Odvajanje linija programskog koda

- Podrazumevani 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 primer:
 - `// komentar u jednoj liniji ...`
- Za komentarisanje više redova koriste se oznake `/*` za početak bloka pod komentarom i oznake `*/`

Prikaz HTML teksta

- HTML tekst se prikazuje pomoću JavaScript koda na stranici korišćenjem metoda `document.write("neki tekst koji se prikazuje na stranici")`. Argument ovog metoda je string koji može biti proizvoljan HTML kod. Na primer:
 - `<SCRIPT LANGUAGE="JavaScript">`
 - `document.write("Prvi red
<I>Drugi red</I>")`
 - `</SCRIPT>`
- Primer `HelloJSDrugiPut.HTML`

Nazivi promenljivih

- Imena promenljivih mogu da sadrže brojeve i slova engleske abecede, ali prvi znak mora da bude slovo engleske abecede ili simbol `_`.

Osetljiv na velika i mala slova

- JavaScript je case sensitive jezik, što znači da se velika i mala slova razlikuju, pa je promenljiva `Aaa` različita promenljiva od promenljive `AAA`. Takođe se ključne reči (`for`, `if`, `else`, `class`, `int`...) ne mogu da koristiti u imenu promenljivih.

Tipovi podataka...

- ...koji su podržani su
 - celobrojni brojevi,
 - racionalni brojevi,
 - stringovi i
 - logički tip.

Celobrojni brojevi

- ...se mogu 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 sme 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

- ...se mogu prikazati na dva načina.
 - pomoću decimalne tačke,
 - na primer 3.14,
 - ili pomoću eksponencijalne prezentacije
 - na primer 314E-2 ili 314e-2.

String

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

Specijalni karakteri (u stringu)

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

Logički tip

- ...podataka obuhvata dve vrednosti *true* (tačno) i *false* (netačno).
- Prilikom rada ako je potrebno može se izvršiti *konverzija* logičke vrednosti *true* u broj 1, odnosno vrednosti *false* u broj 0.

Konverzije podataka

- ... je jednostavan proces. Ako je potrebno, sam jezik automatski izvršava promenu jednog tipa u drugi, jer se **dozvoljava da promenljiva ima različite tipove podataka u različito vreme izvršavanja programa.**

- `a = 5;`
- `b = 8;`
- `b = "broj" + a;`

OPERATORI

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

Aritmetički operatori

- Koriste se za matematičke operacije.
- Ukoliko je jedan od operandi 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 uspe kao rezultat se dobija specijalna vrednost **NaN (Not A Number)**. (Izuzetak: kod sabiranja podatak koji nije tipa String konvertuje se u String i izvršava se sabiranje dva Stringa)

Operator	Opis	Operator	Opis
+	sabiranje	+=	sabiranje dodela
-	oduzimanje	-=	oduzimanje dodela
*	množenje	*=	množenje dodela
/	deljenje	/=	deljenje dodela
%	moduo	%=	Moduo dodela
++	inkrement	--	dekrement

Operatori na nivou bita

- Operatori iz ove grupe obavljaju operacije nad celobrojnim brojevima, i to dužine 32 bita.
- Ukoliko neki od operandi nije celobrojni broj dužine 32 bita, pokušaće se izvršiti konverzija u traženi tip, pa tek onda primeniti operaciju.

Tabela operatori na nivou bita

Operator		Opis
Logičko I (AND)	<code>a & b</code>	Rezultat se dobija 1, jedino ako su oba bita 1, u ostalim slučajevima rezultat je 0.
Logičko ILI (OR)	<code>a b</code>	Rezultat se dobija 0, jedino ako su oba bita 0, u ostalim slučajevima rezultat je 1.
Logičko ekkluzivno ILI (XOR)	<code>a ^ b</code>	Rezultat se dobija 1, ako bitovi imaju različite vrednosti, u slučaju da imaju iste vrednosti, rezultat je 0.
Logičko NE (NOT)	<code>~ a</code>	Komplementira bitove operand a.
Pomeranje ulevo	<code>a << b</code>	Pomera binarni sadržaj operand a za b mesta ulevo. Prazna mesta popunjava sa vrednošću 0.
Pomeranje udesno sa znakom	<code>a >> b</code>	Pomera binarni sadržaj operand a za b mesta udesno. Prazna mesta popunjavaju sa vrednošću najstarijeg bita.
Pomeranje udesno sa nulama	<code>a >>> b</code>	Pomera binarni sadržaj operand a za b mesta udesno. Prazna mesta popunjavaju sa vrednošću 0.

Logički operatori

- Deluju na operande logičkog tipa, koji mogu imati samo vrednosti **true** i **false**. Svi oni kombinuju dva operanda logičkog tipa i kao rezultat vraćaju vrednost logičkog tipa.
- Ovi operatori imaju veliku primenu u okviru kontrolama toka.

		Opis
I (&&)	<code>expr1 && expr2</code>	Rezultat se dobija true, jedino ako su oba operanda true, u ostalim slučajevima rezultat je false.
ILI ()	<code>expr1 expr2</code>	Rezultat se dobija false, jedino ako su oba operanda false, u ostalim slučajevima rezultat je true.
NE (!)	<code>!expr</code>	Rezultat se dobija komplement od vrednosti operanada. Ako je operand true, rezultat je false, ako je operand false, rezultat je true

Primer upotrebe navedenih operatora je:

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

Operatori poređenja

- Obavljaju poređenje dve vrednosti i kao rezultat vraćaju vrednost logičkog tipa **true** ili **false**.
- Svaki dozvoljeni tip podataka, celobrojan, racionalni, karakter, String i logičkitip mogu se upoređivati koristeći operatore **==** i **!=**.
 - Samo numerički tipovi koriste ostale operatore.

Tabela operatora za poređenje

Operator	Upotreba	Opis
Jednakost (==)	Rezultat je true ako su operandi jednaki	<code>x == y</code> rezultat je true ako su x i y jednaki.
Nejednakost (!=)	Rezultat je true ako su operandi različiti.	<code>x != y</code> rezultat je true ako su x i y različiti.
Veće (>)	Rezultat je true ako je levi operand veći od desnog operanda.	<code>x > y</code> rezultat je true ako je x veće od y.
Veće ili jednako (>=)	Rezultat je true ako je levi operand veći ili jednak desnom operandu	<code>x >= y</code> rezultat je true ako je x veće ili jednako y.
Manje (<)	Rezultat je true ako je levi operand manji od desnog operanda	<code>x < y</code> rezultat je true ako je x manje od y.
Manje ili jednako (<=)	Rezultat je true ako je levi operand manji ili jednak desnom operandu	<code>x <= y</code> rezultat je true ako je x manje ili jednako y.
Jednako bez konverzije tipova (===)	Rezultat je true ako su operandi jednaki bez konverzije podataka	<code>x === y</code> rezultat je true ako su x i y jednaki bez konverzije podataka
Različito bez konverzije tipova (!==)	Rezultat je true ako su operandi različiti bez konverzije podataka	<code>x !== y</code> rezultat je true ako su x i y različiti bez konverzije podataka

- Operatori **==** i **!=** obavljaju potrebnu konverziju podataka pre poređenja, ukoliko su operandi različitog tipa. Znači za ove operatore vrednosti `5` i `"5"` su iste, pa će posle njihovog poređenja rezultat sa operatorom `==` biti true, a sa operatorom `!=` false.
- S druge strane operatori **===** i **!==** ne obavljaju potrebnu konverziju podataka pre poređenja, ukoliko su operandi različitog tipa. Znači za ove operatore vrednosti `5` i `"5"` su različite, pa će posle njihovog poređenja rezultat sa operatorom `===` biti false, a sa operatorom `!==` true.
- Primer
 - `a = 4;`
 - `b = 1;`
 - `c = a < b;`
 - `d = a == b;`
 - `document.write(" c = " + c + "
");`
 - `document.write(" d = " + d);`
 - Rezultat izvršavanja prethodnog primera 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 delu može biti nova if-else konstrukcija. Primer upotrebe ove konstrukcije je
- ```
□ if (x == 8)
□ {
 y=x;
} else
□ {
 z=x;
 y=y*x
}
```

### Ternarni if-then-else operator

- Forma ovog operatora je:
  - expression ? statement1 : statement2
- gde je izraz expression bilo koji izraz čiji rezultat je vrednost logičkog tipa. Ako je rezultat izraza true, onda se izvršava statement1, u suprotnom statement2. Primer:

### switch

- if (mesec == 1)
- ime\_meseca = "Januar"
- else if (mesec == 2)
- ime\_meseca = "Februar"
- else if (mesec == 3)
- ime\_meseca = "Mart"
- else if (mesec == 4)
- ime\_meseca = "Maj"
- else
- ....
- else if (mesec == 12)
- ime\_meseca = "Decembar"

- ```
□ switch(mesec) {
□ case 1: ime_meseca = "Januar"; break;
□ case 3: ime_meseca = "Mart"; break;
□ case 5: ime_meseca = "Maj"; break;
□ case 7: ime_meseca = "Jul"; break;
□ case 8: ime_meseca = "Avgust"; break;
□ case 10: ime_meseca = "Oktobar"; break;
□ case 12: ime_meseca = "Decembar"; break;
□ case 4: ime_meseca = "April"; break;
□ case 6: ime_meseca = "Jun"; break;
□ case 9: ime_meseca = "Septembar"; break;
□ case 11: ime_meseca = "Novembar"; break;
□ case 2: ime_meseca = "Februar ";
□ }
□ Ukoliko se vrednost izraza izraz ne nalazi medju vrednostima vr1,..., vrN 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){
■ telo_petlje;
■ }
```
- Jednostavan primer:

```
■ i=1
■ while(i<=10){
■ document.writeln(i);
■ i=i+1;
■ }
```
  - Nakon izvršavanja ovog primera dobiće se prikazani brojevi od 1 do 10.
  - Treba napomenuti da će se u slučaju da uslov petlje nije ispunjen kada se prvi put pristupi petlji telo petlje neće izvršiti nijednom. Znači 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, telo petlje će se sigurno izvršiti bar jednom.

```
do {
 telo_petlje
 [iteracija]
} while (uslov);
```
- ```
i=1
do {
    document.writeln(i);
    i=i+1;
} while(i<=10)
```

for petlja

Opšti oblik for petlje izgleda ovako:

- `for(inicijalizacija; uslov; iteracija){`
- `telo_petlje;`
- `}`
- `for(i=0; i<10; i++){`
- `document.writeln(i);`
- `}`
- Takođe, u vredna pažnje je i deklaracija promenljive `i` u zaglavlju petlje. Ta promenljiva je privremena promenljiva a blok u kome je definisana je blok u kome se nalazi for petlja.

break

- Ona se koristi za skok na kraj bloka koji je označen labelom uz break ili na kraj bloka u kome se i 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 primer, sledeći kod:

```
a: {
b: {
c: {
    document.writeln("pre break-a");
    break b;
    document.writeln("ovo neće biti prikazano");
}
}
document.writeln("posle break-a");
}
```
- Nakon izvršavanja navedenog primer dobija se:
pre break-a
posle break-a

return

- return se koristi za povratak iz funkcije na mesto poziva. Ukoliko funkcija vraća neku vrednost tada return mora slediti 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 od 5 je " + x)
```
- Kao rezultat poziva funkcije dobija se:
- Kvadrat od 5 je 25

continue

- Prelaz na sledeću iteraciju petlje a da se deo koda pre njenog kraja ne izvrši. Za takve situacije se koristi continue.

```
for( int i=0; i<10; i++)
{
    document.write(i+ " ");
    if (i%2 ==0)
        continue;
    document.writeln(" ");
}
```
- Zahvaljujući continue naredbi nakon izvršavanja ovog primera dobija se:
0 1
2 3
4 5
6 7
8 9

Specijalne naredbe

for...in

- Izvršava iteraciju po specifičnoj promenljivoj za svaku osobinu (property) u okviru određenog objekta. Znači za svaku definisanu osobinu u okviru nekog objekta izvršava se niz naredbi definisan u okviru tela ove petlje.
- Primer:

```
function dump_props(obj, objName)
{
  var result = "";
  for (var i in obj)
  {
    result += objName + "," + i + " = " + obj[i] + "<BR>";
  }
  result += "<HR>";
  return result;
}
```

function

- Deklariše JavaScript funkciju sa specificiranim parametrima. 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 dodeljuje specifične vrednosti za određene osobine objekta. Na primer, matematičkim funkcijama mora prethoditi objekat Math. Sledeći primer 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);
}
```

var

- Deklariše promenljivu. Opciono moguće je izvršiti i njenu inicijalizaciju.
- `var imePromenljive [= vrednost]
[...imePromenljive2 [= vrednost2]]`

OBJEKTI Date i String

Date objekat

- Ovaj objekat se koristi kada je potrebno primeniti određene operacije u kojima se koriste vremenske promenljive.
- 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. Na primer argument 5000 će kreirati datum koji predstavlja 5 sekundi posle ponoći 1/1/1970.

- ❑ U programu kreiranje promenljive od ovog objekta se postiže na jedan od sledećih 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)`

- ❑ `today = new Date()//trenutno vreme i datum`
- ❑ `birthday = new Date("December 17, 1995 03:24:00")`
- ❑ `birthday = new Date(95,12,17)`
- ❑ `birthday = new Date(95,12,17,3,24,0)`

- ❑ **Date.parse(datum)** Ovaj metod vraća broj milisekundi do navedenog datuma po lokalnom vremenu (od 1.1.1970 00:00:00). Primer: `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 (GMT). Primer: `gmtDatum = new Date(Date.UTC(96, 11, 1, 0, 0, 0))`
- ❑ **datum.getDate()** Ovaj metod vraća dan u mesecu (1-31) za navedeni datum.
- ❑ Primer: `datum = new Date("December 25, 2001 23:15:00");`
- ❑ `dan = datum.getDate()`
- ❑ Nakon izvršavanja primera promenljiva `dan` dobija vrednost 25.
- ❑ **datum.getDay()** Ovaj metod vraća dan u nedelji (0-ned, 1-pon ... 6-sub) za navedeni datum.
- ❑ Primer: `datum = new Date("December 25, 2001 23:15:00");`
- ❑ `dan = datum.getDay()`
- ❑ Nakon izvršavanja primera promenljiva `dan` dobija vrednost 2, jer je 25.12.2005.god., bio utorak.
- ❑ **datum.getHours()** Ovaj metod vraća sat za navedeni datum, moguće vrednosti su brojevi u opsegu od 0 do 23. Primer:
- ❑ `datum = new Date("December 25, 2001 23:15:00");`
- ❑ `sati = datum.getHours()`
- ❑ Nakon izvršavanja primera promenljiva `sati` dobija vrednost 23.
- ❑ **datum.getMinutes()** Ovaj metod vraća minute za navedeni datum, moguće vrednosti su brojevi u opsegu od 0 do 59. Primer:
- ❑ `datum = new Date("December 25, 2001 23:15:00");`
- ❑ `minuti = datum.getMinutes()`
- ❑ Nakon izvršavanja primera promenljiva `minuti` dobija vrednost 15.
- ❑ **datum.getMonth()** Ovaj metod vraća mesec za navedeni datum (0-januar, 1-februar, ... 11-december). Primer:
- ❑ `datum = new Date("December 25, 2001 23:15:00");`
- ❑ `mesec = datum.getMonth()`
- ❑ Nakon izvršavanja primera promenljiva `mesec` dobija vrednost 11.

- ❑ **datum.getSeconds()** Ovaj metod vraća sekunde za navedeni datum, moguće vrednosti su brojevi u opsegu od 0 do 59. Primer:
- ❑ `datum = new Date("December 25, 2001 23:15:08");`
- ❑ `sekunde = datum.getSeconds()`
- ❑ Nakon izvršavanja primera promenljiva `sekunde` dobija vrednost 8.
- ❑ **datum.getTime()** Ovaj metod vraća vreme do navedenog datuma u milisekundama (od 1.1.1970 00:00:00). Primer:
- ❑ `datum = new Date("December 25, 2001 23:15:00");`
- ❑ `protkalo = datum.getTime()`
- ❑ Nakon izvršavanja primera promenljiva `protkalo` dobija vrednost 1009318500000, koja odgovara broju milisekundi od 1.1.1970 00:00:00 do 25.12.2001. 23:15:00.
- ❑ **datum.getTimezoneOffset()** Ovaj metod vraća razliku lokalnog vremena i GMT u minutama. Primer:
- ❑ `datum = new Date();`
- ❑ `razlikaSati = datum.getTimezoneOffset()/60`
- ❑ Nakon izvršavanja primera promenljiva `razlikaSati` dobija vrednost -1.
- ❑ **datum.getYear()** Ovaj metod vraća godinu iz navedenog datuma (dvocifreno, od 1900 do 1999, u ostalim slučajevima 4 cifre). Primer:
- ❑ `datum = new Date();`
- ❑ `godina = datum.getYear()`
- ❑ Nakon izvršavanja primera promenljiva `godina` dobija vrednost 2006.
- ❑ **datum.setDate(brojDana)** Ovaj metod postavlja dan u mesecu za navedeni datum. Argument metoda je broj u opsegu od 1 do 31. Primer:
- ❑ `datum = new Date("July 27, 1960 23:30:00");`
- ❑ `datum.setDate(24)`
- ❑ Nakon izvršavanja primera promenljiva `datum` dobija vrednost 24.7.1960 23:30:00.
- ❑ **datum.setHours(brojSata)** Ovaj metod postavlja broj sati za navedeni datum. Argument metoda je broj u opsegu od 0 do 23. Primer:
- ❑ `datum = new Date("July 27, 1960 23:30:00");`
- ❑ `datum.setHours(7)`
- ❑ Nakon izvršavanja primera promenljiva `datum` dobija vrednost 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. Primer:
- ❑ `datum = new Date("July 27, 1960 23:30:00");`
- ❑ `datum.setMinutes(35)`
- ❑ Nakon izvršavanja primera promenljiva `datum` dobija vrednost 27.7.1960 23:35:00.
- ❑ **datum.setMonth(brojMeseca)** Ovaj metod postavlja dan u mesecu za navedeni datum. Argument metoda je broj u opsegu od 0 do 11. Primer:
- ❑ `datum = new Date("July 27, 1960 23:30:00");`
- ❑ `datum.setMonth(35)`
- ❑ Nakon izvršavanja primera promenljiva `datum` dobija vrednost 27.7.1960 23:30:35.
- ❑ **datum.setTime(vreme)** Ovaj metod definiše novi datum. Argument metoda je broj milisekundi od 1.1.1970 00:00:00 do željenog datuma.
- ❑ `datum.setTime(1009318500000)`
- ❑ Nakon izvršavanja primera promenljiva `datum` dobija vrednost 25.12.2001. 23:15:00.
- ❑ **datum.setYear(brojGodine)** Ovaj metod postavlja godinu za navedeni datum. Argument metoda je broj u opsegu od 0 do 99 za godine koje počinju sa 19, za ostale je 4 cifre.
- ❑ `datum = new Date("July 27, 1960 23:30:00");`
- ❑ `datum.setYear(2010)`
- ❑ Nakon izvršavanja primera promenljiva `datum` dobija vrednost 27.7.2010 23:30:00.
- ❑ **datum.toGMTString()** Ovaj metod vrši konverziju datuma u GMT string iz lokalne vremenske zone. Primer:
- ❑ `datum = new Date("December 25, 2001 23:15:00");`
- ❑ `datum.toGMTString()`
- ❑ Nakon izvršavanja primera promenljiva `datum` dobija vrednost "Tue, 25 Dec 2001 22:15:00 UTC"
- ❑ **datum.toLocaleString()** Ovaj metod vrši konverziju datuma u lokalni datum string iz GMT. Primer:
- ❑ `datum.toLocaleString()`

Skrivanje JavaScript za starije pretraživače

- ❑ Ovo se izvodi postavljajući JavaScript u HTML komentar. Ovo će sakriti JavaScript ukoliko pretraživač ne poznaje JavaScript
 - `<SCRIPT language=JAVASCRIPT>`
 - `<!--`
 - `document.write("I lost a buttonhole. ");`
 - `-->`
 - `</SCRIPT>`
- ❑ Ipak, neki pretraživači ni ovo nemogu prepoznati korektno pa će prikazati kod.
- ❑ Najbolji način je da se koristi tag `<SCRIPT SRC>`.
- ❑ Takođe, rešenje je koristiti `<NOSCRIPT>` koji će izazvati da pretraživač ignoriše JavaScript kod.

String objekat

Čemu služi?

- ❑ 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 definiše 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 kôdove karaktera u okviru argumenta. Primer: `y = escape("!#")`
- ❑ *Nakon izvršavanja primera promenljiva y dobija vrednost „%21%23“, jer su ASCII kôdovi za simbole ! i # 21 i 23.*

eval(izraz)

Ova funkcija izračunava vrednost izraza koji je definisan kao argument funkcije.

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

Nakon izvršavanja primera promenljiva x dobija vrednost 1.

linkTekst.link(linkURL)

- ❑ Ovaj metod kreira tekst linkTekst koji predstavlja HTML link na neku drugu stranicu, čiji je adresa definisana sa argumentom linkURL (desetvo kao i HTML taga <A HREF...>), Primer
`var naziv = „link ka prezentaciji Univerziteta Singidunum“;`
`var URL = "http://www.singidunum.ac.yu";`
`document.write("Ovo je " + naziv.link(URL))`

Nakon izvršavanja primera na stranici će se pojaviti tekst "Ovo je [link ka prezentaciji Univerziteta Singidunum](http://www.singidunum.ac.yu)", koji će predstavljati vezu ka stranici www.singidunum.ac.yu.

parseInt(stringBroj [,osnova])

- ❑ Ova funkcija kao rezultat vraća ceo 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 podrazumeva se osnova 10, t.j. dekadni brojni sistem. Primer:
 - ❑ `x = parseInt("17", 8);`
 - ❑ `y = parseInt("15", 10)`
 - ❑ Nakon izvršavanja primera i promenljiva x i promenljiva y dobija vrednost 15.

string.big()

- ❑ Ovaj metod prikazuje string sa uvećanim slovima (isto dejstvo kao HTML tag <BIG>). Primer:
- ❑ "Dobar dan!".big()

string.blink()

Ovaj metod prikazuje string sa blinkovanjem (isto dejstvo kao HTML tag <BLINK>). Primer:

- ❑ "Dobar dan!".blink()

string.bold()

- ❑ Ovaj metod prikazuje string boldovano (isto dejstvo kao HTML tag). Primer:
- ❑ "Dobar dan!".bold()

string.charAt(broj)

- ❑ Ovaj metod kao rezultat vraća znak na navedenoj poziciji. Pozicije unutar stringa se računaju sa leve 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 poslednjeg karaktera u stringu, a to je vrednost string.length. - 1. Primer:
- ❑ x= "Dobar dan!".charAt(4)
- ❑ y= "Dobar dan!".charAt(6)
- ❑ Nakon izvršavanja primera promenljiva x dobija vrednost 'r', a promenljiva y je 'd'.

string.fontcolor("boja")

- ❑ Ovaj metod prikazuje string u zadanoj boji (isto dejstvo kao HTML tag). Primer:
- ❑ "Dobar dan!".fontcolor("blue")

string.fontsize(broj)

- ❑ Ovaj metod prikazuje string u zadatoj veličini (isto dejstvo kao HTML tag)
- ❑ "Dobar dan!".fontsize(7)

string.indexOf(traziString, [odPozicije])

- ❑ Ovaj metod vraća broj pozicije na kojoj je prvi put pronađen argument tipa String traziString. U slučaju da se traženi string ne nalazi u početnom stringu kao rezultat se vraća vrednost -1. Ako postoji i drugi argument odPozicije, tada će se pretraga izvršavati od zadate pozicije. Primer:
- ❑ `x = "Dobar dan!".indexOf("r")`
- ❑ `y = "Dobar dan!".indexOf("a", 4)`
- ❑ Nakon izvršavanja primera promenljiva x dobija vrednost 4, a promenljiva y je 7.

string.italics()

- ❑ Ovaj metod prikazuje string sa italik stilom (isto dejstvo kao HTML tag <I>). Primer:
- ❑ `"Dobar dan!".italics()`

string.lastIndexOf(traziString, [doPozicije])

- ❑ Ovaj metod vraća broj pozicije na kojoj se poslednji put pojavljuje argument tipa String traziString. U slučaju da se traženi string ne nalazi u početnom stringu kao rezultat se vraća vrednost -1. Ako postoji i drugi argument doPozicije, tada će se pretraga izvršavati do zadate pozicije. Primer:
- ❑ `x = "Dobar dan!".lastIndexOf("a")`
- ❑ `y = "Dobar dan!".lastIndexOf("a", 6)`
- ❑ Nakon izvršavanja primera promenljiva x dobija vrednost 7, jer je to poslednje pojavljivanje stringa "a", a promenljiva y je 3, jer je to poslednje pojavljivanje stringa "a" do pozicije 6.

string.strike()

- ❑ Ovaj metod prikazuje precrtani string (isto dejstvo kao HTML tag <STRIKE>). Primer:
- ❑ `"Dobar dan!".strike()`

string.sub()

- ❑ Ovaj metod prikazuje string kao subscript (isto dejstvo kao HTML tag <SUB>). Primer:
- ❑ `"Hej!".sub()`

string.substring(prvi, poslednji)

- ❑ Ovaj metod vraća deo stringa počev od pozicije prvi do pozicije poslednji, t.j. uzima redom karaktere na pozicijama prvi, prvi + 1, prvi + 2, ..., poslednji - 2, poslednji - 1.
- ❑ `x = "Dobar dan!".substring(6,9)`
- ❑ Nakon izvršavanja primera promenljiva x dobija vrednost "dan", jer su to karakteri na pozicijama 6, 7 i 8.

string.sup()

- ❑ Ovaj metod prikazuje string kao superscript (isto dejstvo kao HTML tag <SUP>). Primer:
- ❑ "Hej!".sup()

string.toLowerCase()

- ❑ Ovaj metod izvrši konverzija svih karaktera u okviru stringa u mala slova. Primer:
- ❑ x = "Dobar dan!".toLowerCase()
- ❑ Nakon izvršavanja primera promenljiva x dobija vrednost "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. Primer:
- ❑ x = "Dobar dan!".toUpperCase()
- ❑ Nakon izvršavanja primera promenljiva x dobija vrednost "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. Primer:
- ❑ x = unescape("%21%23")
- ❑ Nakon izvršavanja primera promenljiva x dobija vrednost "!#/", jer su simboli ! i # kodovani sa ASCII kodovima 21 i 23.

RAD SA UZORCIMA

(Pattern Matching)

Definisanje uzorka

- ❑ Česta upotreba JavaScript funkcija je provera unetih podataka od strane klijenta. Zato JavaScript ima razvijenu podršku za razne vrste provera i one se obavljaju na klijentskoj strani.
 - ❑ Uzorak se još naziva i regularni izraz (regular expression) i može se definisati na dva načina
- ```
var uPrimer = new RegExp(„HTML“)
ili
var uPrimer = /HTML/
```

- Na oba načina se formira objekat uzorka koji se naziva uPrimer i kome odgovara svaki string koji u sebi sadrži podstring HTML
- `var uPrimer = new RegExp(„s$“)`
- ili
- `var uPrimer = /s$/`
- Sada je promenljiva uPrimer uzorak koji odgovara bilo kom stringu koji se završava sa s. Ovakva vrednost je dobijena, jer u okviru uzorka simbol s predstavlja samog sebe, a simbol \$ predstavlja metakarakter koji označava kraj stringa.

## Karakteristi koji se mogu koristiti u okviru uzorka

| Karakter            | Predstavlja                                                                                                                                     |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Slovo ili broj      | Istu vrednost                                                                                                                                   |
| <code>\0</code>     | Specijalni NUL karakter                                                                                                                         |
| <code>\t</code>     | Tab znak                                                                                                                                        |
| <code>\n</code>     | Nova linija                                                                                                                                     |
| <code>\w</code>     | Vertikalni tab znak                                                                                                                             |
| <code>\f</code>     | Form feed                                                                                                                                       |
| <code>\r</code>     | Carriage return                                                                                                                                 |
| <code>\uxxxx</code> | Unicode karakter definisan pomoću heksadecimalnog boja <code>xxxx</code> , na primer, <code>\u0009</code> ima isti efekat kao i <code>\t</code> |

## Specijalni simboli sa posebnim značenjem

| Karakter            | Predstavlja pojavljivanje                                |
|---------------------|----------------------------------------------------------|
| <code>[...]</code>  | Bilo kog karaktera od onih koji su navedeni između [ i ] |
| <code>[^...]</code> | Bilo kog karaktera koji nije naveden između [ i ]        |
| <code>.</code>      | Bilo kog karaktera osim nove linije                      |
| <code>\w</code>     | Bilo kog ASCII definisanog slova                         |
| <code>\W</code>     | Bilo kog karaktera koji nije ASCII definisano slovo      |
| <code>\d</code>     | Bilo koje ASCII definisane cifre                         |
| <code>\D</code>     | Bilo kog karaktera koji nije ASCII definisana cifra      |
| <code>\b</code>     | Blanko znak                                              |

### □ `/[abc]/`

- predstavlja jedno pojavljivanje simbola a ili jedno pojavljivanje simbola b ili jedno pojavljivanje simbola c. Tako da string "c" ispunjava uslove definisane uzorkom, a string "s" ne ispunjava definisane uslove. Na sličan način uzorak

### □ `/[^abc]/`

- predstavlja karakter koji nije simbol a ili simbol b ili simbol c.

- Primer za poštanski broj koji je petocifren bi izgledao

■ `/\d\d\d\d\d/`

- Uzorak bi bio još nepregledniji da je potrebno definisati broj koji se sastoji od 18 ili 28 cifara. Za rad sa ovakvom vrstom uzoraka postoje i specijalne oznake koje su date u sledećoj tabeli.

| Oznaka             | Značenje                                                                                                                                       |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>{n,m}</code> | Ponavljanje prethodne grupe najmanje <i>n</i> puta, ali najviše <i>m</i> puta.                                                                 |
| <code>{n}</code>   | Ponavljanje prethodne grupe <i>n</i> ili više puta.                                                                                            |
| <code>{n}</code>   | Ponavljanje prethodne grupe tačno <i>n</i> puta.                                                                                               |
| <code>?</code>     | Ponavljanje prethodne grupe jednom ili nijednom. Isto dejstvo kao i <code>{0,1}</code> .                                                       |
| <code>+</code>     | Ponavljanje prethodne grupe jednom ili više puta. Isto dejstvo kao i <code>{1,}</code> .                                                       |
| <code>*</code>     | Ponavljanje prethodne grupe nijednom ili više puta. Isto dejstvo kao i <code>{0,}</code> .                                                     |
| <code> </code>     | Alternative. Pojavljivanje dela izraza sa desne ili pojavljivanje izraza sa leve strane.                                                       |
| <code>(...)</code> | Grupisanje simbola u jedan objekat nad kojim se mogu koristiti oznake <code>*</code> , <code>+</code> , <code>?</code> , <code> </code> , itd. |
| <code>^</code>     | Pretraga uzorka se obavlja na početku stringa                                                                                                  |
| <code>\$</code>    | Pretraga uzorka se obavlja na kraju stringa                                                                                                    |

- `/\d\d\d\d\d/` se može zapisati i kao `/\d{5}/`
- `/\d{2,4}/` // uzorak koji označava 2, 3 ili 4 pojavljivanje cifara
- `/\w{3}\d?/` // uzorak koji označava tačno tri pojavljivanja slova i opcionalno jedne cifre. Na primer string koji odgovara ovom uzorku je „abc8” ili „qqq”.
- `/\s+java\s+/` // uzorak koji označava string „java” sa jednim ili više prostora pre ili posle stringa
- `/[^\s]*/` // uzorak koji označava nula ili više pojavljivanje karaktera navoda
- `/ab|cd|ef/` // uzorak koji označava pojavljivanje ab ili pojavljivanje cd ili pojavljivanje ef
- `/\d{3}[a-z]{4}/` // uzorak koji označava pojavljivanje tri cifre ili 4 mala slova
- `/java(script)?/` // uzorak koji označava pojavljivanje stringa „java” ili stringa „javascript”
- `/((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”

## Dodatna ispitivanja

| Atribut | Značenje                                                                                                                                                 |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| i       | Izvršavanje case-insensitive ispitivanja.                                                                                                                |
| g       | Izvršava globalno ispitivanje, znači pronaći će se sva pojavljivanja definisanog uzorka, a neće se ispitivanje zaustaviti posle prvog pronalaska uzorka. |
| M       | Rad sa više linija. ^ označava početak linije ili stringa, a \$ predstavlja kraj linije ili stringa.                                                     |

## Ispitivanje uzoraka pomoću metoda objekta tipa String

- **search( )**. Ovaj metod ispituje da li u okviru stringa postoji definisani uzorak, i kao rezultat vraća poziciju njegovog prvog pojavljivanja, ili -1 ako ne pronađe uzorak. Primer:
  - `x = /script/`
  - `y = "JavaScript".search(x,i);`
 Kao rezultat izvršavanja ovog primera promenljiva y će dobiti vrednost 4. Ovaj metod ne podržava globalnu pretragu, znači on ignoriše upotrebu atributa g u okviru definicije uzorka.

- **replace( )**. Obavlja ispitivanje dali u stringu postoji uzorak i ako postoji zamenu uzorka unutar stringa sa nekom drugom vrednošću. Ima dva argumenta, prvi je uzorak, a drugi je string koji treba da zameni uzorak. Ako se u okviru uzorka navede i g atribut, ovaj metod će izvršiti zamenu svakog uzorka koji pronađe u okviru stringa.

```
str = "Lana ima 5 pomorandzi i 135 limuna"
promena3u5 = new RegExp("[3-5]", "g")
str.replace(promena3u5, "9")
```

- Nakon izvršavanja ovog primera promenljiva str će imati vrednost "Lana ima 9 pomorandzi i 199 limuna " , jer je uzorkom definisano da se svako pojavljivanje cifri 3, 4 i 5 zameni sa cifrom 9.

- **match( )** Ima samo argument tipa uzorka. Rezultat njegovog izvršavanja je niz koji sadrži rezultate ispitivanja. Ako je u okviru uzorka definisan atribut g, rezultat je niz sa svim pojavljivanjem definisanog uzorka.
- Na primer:
 

```
"1 plus 2 equals 3".match(/\d+/g)
```

 Rezultat izvršavanja metoda match() u primeru je niz ["1", "2", "3"], jer je uzorak definisan kao pojavljivanje 1 ili više puta cifre, i to u celom stringu



- 
- **split( )**. Ovaj metod ima jedan argument i to tipa uzorka. Rezultat izvršavanja ovog metoda je niz koji se dobija kada se string podeli argumentum uzorkom kao separatorom.

*"123,456,789".split(",");*

*Rezultat izvršavanja ovog primera je niz ["123","456","789"], jer je string podeljen pomoću separatora ",".*

---

- 
- *"1,2, 3 , 4 ,5".split(/\s\*,\s\*/);*
  - *Rezultat ovog primera je niz ["1","2","3","4","5"], jer je uzorak definisan sa određenim brojem blanko znakova pre i posle zareza, uključujući i zarez*
- 

## **Metodi objekta RegExp**

---

### **exec( )**

---

- Ovaj metod je veoma sličan String metodu match( ). Razlike je u tome što je kod ovog metoda argument string, a primenjuje se na uzorka, dok je kod metoda match( ) argument bio uzorak, a primenjivao se na String. Znači rezultat izvršavanje metoda exec( ) je niz koji sadrži rezultate ispitivanja, definisane na isti način kao i metod match( ).
  - Za razliku od metoda match( ) metod exec( ) vraća isti rezultat ako postoji atribut g i ako ne postoji, i to uvek prvo poklapanje i sve relevantne informacije o njemu
- 

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

### **test( )**

---

- Njegov argument je string, a rezultat true ako string odgovara uzorku.

*var pattern = /java/i;*  
*pattern.test("JavaScript");*

---

## JavaScript i FORME