Programiranje mrežnih aplikacija

Predavanje 3

Specijalni znakovi u HTML-u

- Neki znakovi imaju posebno značenje u HTML-u, poput znaka za manje (<) koji definira početak HTML oznake. Ako je potrebno prikazati znakove koji u HTML-u imaju posebno značenje potrebno ih je zamijeniti odgovarajućim entitetima u HTML dokumentu.
- Entitet se sastoji od tri dijela:
  - znaka i (&) – ampersand
  - naziva entiteta ili znaka # iza kojeg slijedi broj entiteta
  - točke-zareza na kraju (;)
- Kako bi se prikazalo oznaku manje < u pregledniku potrebno je upisati **<** ili **<**
- Prednost uporabe naziva entiteta nad brojem entiteta je očita. Jednostavnije je pamtiti naziv lt koje je skraćenica od less than (manje od). Problem uporabe naziva entiteta je različita podrška ovih naziva od strane različitih preglednika.
- Potrebno je napomenuti kako su nazivi entiteta osjetljivi na velika i mala slova.

Rezultat	Opis	Naziv entiteta	Broj entiteta
	Non-breaking space – razmak		
<	Less than - manje od	<	<
>	Greater then – veće od	>	>
&	Ampersand – i	&	&
“	Quotation mark – navodnici	"	"
‘	Apostrophe – apostrof		'
¢	Cent	¢	¢
£	Pound – funta	£	£
¥	Yen	¥	¥
§	Section – odjeljak	§	§
©	Copyright	©	©
®	Registrated trademark	®	®
×	Multiplication – množenje	×	×
÷	Division – dijeljenje	÷	÷

Linkovi ili poveznice

- Web stranice jedinstvene su upravo po tome što čitatelja kroz tekst ne vode linearno do informacije koja se uvijek prikazuje na isti, predvidljivi način.
- Čitatelj može ostvariti «skok» na drugi dio zemaljske kugle bez očitog prijelaza.
- Osnovni tekstualni link definira se oznakom <A> (anchor – sidro).

```
<a href="detalji.html">Detalji</a>
```

- Atribut href ukazuje na izvornu datoteku. Vrijednost ovog atributa je uvijek URL.
- Ako se datoteka nalazi u istom direktoriju kao i trenutni dokument moguće je koristiti URL skraćenice i napisati samo naziv datoteke koji se želi povezati.
- Ako se dokument nalazi na drugom serveru potrebno je navesti punu adresu do datoteke koja se vezuje.

Interni linkovi (poveznice)

- Linkovi ne moraju pokazivati na drugi dokument.
- Kod većih dokumenata vrlo je praktično dodijeliti linkove poglavljima koja se i dalje nalaze na istoj stranici. Primjer ovakve organizacije može biti telefonski imenik firme sa npr. 200 zaposlenih. Popis svih zaposlenih može se smjestiti na jednu stranicu, ali je vrlo zgodno organizirati linkove unutar dokumenta po abecedi. Kako bi mogli definirati interne linkove potrebno je koristiti atribut name koji se koristi u oznaci <a>.
- Ako se na početku nekog poglavlja izdefinira

  ```
  <a name="StraniDVD">
  ```
- onda se na početku dokumenta može definirati link na ovaj dio dokumenta uporabom posebne notacije:

  ```
  <a href="#StraniDVD">
  ```
- Znak # ukazuje na to da je to link na prazno sidro na trenutnoj stranici. Kako bi skočili na neko poglavlje udaljenog dokumenta može se kombinirati link i interni link pa onda imamo:

  ```
  <a href="ponudaDVD.html#StraniDVD">
  ```

Tablice – table element

- Tablice se definiraju oznakom **<table>…</table>**.
- Jedna od važnijih stvari koje treba imati na umu je činjenica kako svi najnoviji web preglednici ne mogu prikazati baš sve atribute i mogućnosti elementa table.
- **Width** – ova vrijednost pokazuje koliko široko treba prikazati tablicu unutar raspoloživog prostora na zaslonu. Mjeri se u postotku raspoloživog prostora (100% ne znači cijeli zaslon nego cijeli trenutno raspoloživ prostor koji zauzima preglednik).
- **Border** – ova vrijednost određuje širinu granica tablice, a mjeri se u pikselima. Vrijednost 0 ovog atributa znači kako se granice neće prikazati.
- **Cellspacing** – određuje prazan prostor između ćelija u tablici, mjereno u pikselima, a podrazumijevana vrijednost je 0.
- **Cellpadding** – određuje prazan prostor između zida ćelije i njenog sadržaja. Vrijednost ovog atributa izražava se u pikselima, a podrazumijevana vrijednost je 0.

Tablice

- Važna inovacija u HTML-u 4.0 je koncept grupiranja ćelija unutar tablice.
- I redovi i stupci mogu se grupirati.
- Uvedena su tri nova elementa u redcima:
  - TABLE HEAD **<thead>**,
  - TABLE FOOT **<tfoot>** i
  - TABLE BODY **<tbody>.**
- Svaka grupa mora sadržavati barem jedan red, koji je definiran pomoću elemenata TABLE ROW **<tr>.**
- Elementi TABLE HEAD i TABLE FOOT su namijenjeni opisnim informacijama o stupcima tablice koje će se prikazati u samom tijelu tablice.
- Pravi sadržaj tablice može biti prikazan samo u tijelu tablice.

Tablice

```
<thead>
<tr>Sadrzaj thead elementa</tr>
</thead>
<tfoot>
<tr>Sadrzaj tfoot elementa</tr>
</tfoot>
<tbody>
<tr>Sadrzaj prvog reda tablice</tr>
<tr>Sadrzaj drugog reda tablice</tr>
</tbody>
</table>
```

- Ovaj primjer pokazuje samo početak strukture HTML tablice. Ova tablica daleko je od potpune tablice i ako je probate učitati u preglednik nećete vidjeti ništa.

Tablice

- Stupci imaju samo jedan element koji kontrolira njihovo grupiranje <colgroup>. Ovaj element ima dva važna atributa: **span** i **width**.
- **Span** – atribut definira broj stupaca koji će se naći unutar grupe. Vrijednost mora biti cijeli broj veći od nule, a podrazumijevana vrijednost je 1.
- **Width** – atribut definira podrazumijevanu širinu za svaki stupac unutar grupe stupaca (za razliku od širine cijele grupe). Vrijednosti mogu biti iskazane u pikselima, postocima i relativnim vrijednostima. Neka je primerice tablica napravljena tako da prikazuje podatke za izračunavanje i neka može imati 10 stupaca širine od po 50 piksela.

  <colgroup span="20" width="50">...</colgroup>
- Definicija grupe stupaca navodi se unutar elementa <table>, ali prije bilo kojeg elementa head ili foot i prije elementa row.

Ćelija tablice

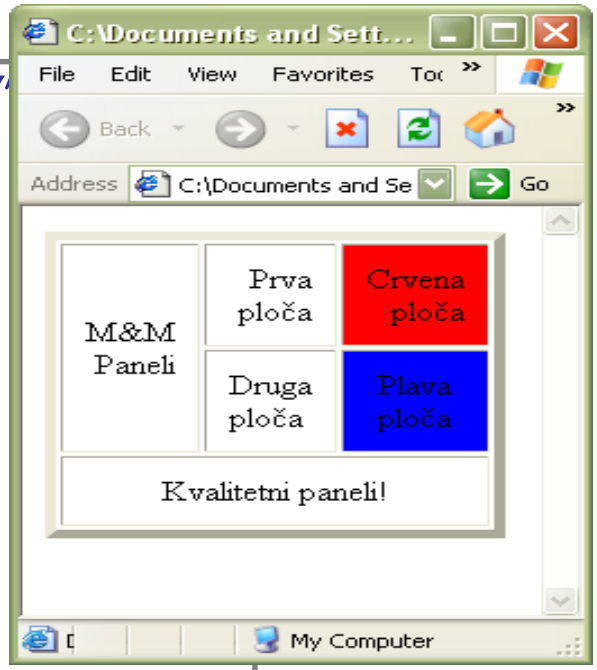
- Za definiranje ćelija unutar redova tablice raspolaganju su nam dva elementa:
  - Table header cell <th> i
  - Table data cell <td>
- Atributi:

Rowspan – atribut daje broj redova spojenih sa trenutnom ćelijom, a podrazumijevana vrijednost mu je 1. Vrijednost nula, iako to nije očito, ukazuje na to da se ćelija spaja ne samo sa trenutnim redom nego i sa svim ostalim redovima u tablici.

Colspan – atribut je sličan prethodnom. Podrazumijevana vrijednost mu je 1, a nula ukazuje na spajanje trenutne ćelije i ostatka tablice.

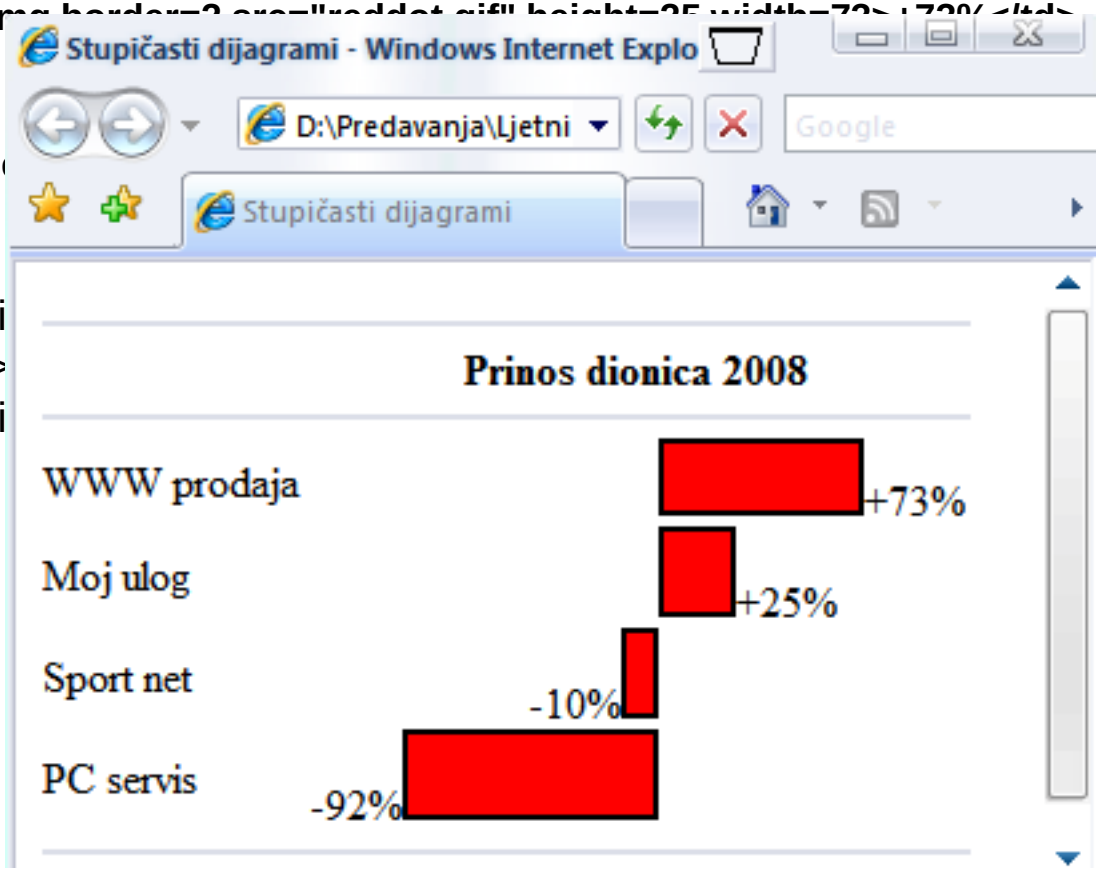
Ćelija tablice

```
<table width="75%" border="5"
   cellpadding="10">
<tr align="right">
   <td rowspan="2">M&M Paneli
   </td>
   <td>Prva ploča</td>
   <td bgcolor="red">
        Crvena ploča
   </td>
</tr>
<tr>
   <td>Druga ploča</td>
   <td bgcolor="blue" align="center">
   Plava ploča
</td>
</tr>
<tr><td colspan="3" align="center">
Kvalitetni paneli!</td></tr>
</table>
```



Primjer stupičasti dijagrami

```
<table width=95% border=0 cellspacing=0 cellpadding=0>
  <tr><td colspan=3><hr noshade></td></tr>
  <tr><td> </td><th colspan=2>Prinos dionica 2008</th></tr>
  <tr><td colspan=3><hr noshade></td></tr>
  <tr><td nowrap>WWW prodaja</td><td> </td>
  <td align=left nowrap><img border=2 src="reddot.gif" height=25 width=73> +73%</td>
  </tr><tr>
  <td>Moj ulog</td><td>
  <td align=left><img bord
  </tr><tr>
  <td>Sport net</td>
  <td align=right>-10%<i
  <td> </td></tr><tr><td>
  <td align=right>-92%<i
  </td><td></td></tr>
  <tr><td colspan=3><hr
</table>
```



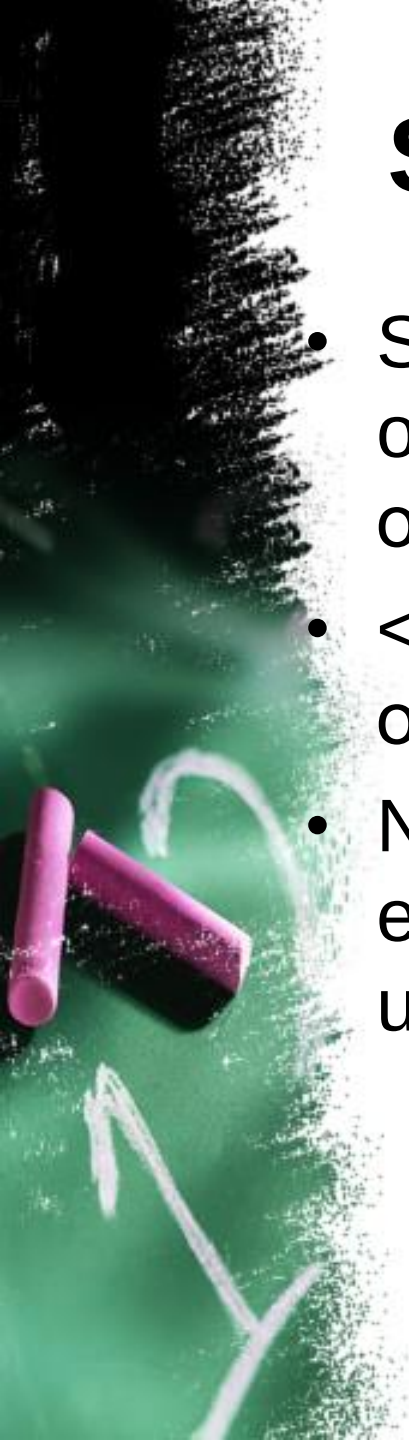
HTML resursi

- Kako web stranica ne bi izgledala suhoparno sa čistim textom dobro

GIF ili JPEG?

GIF format (.gif) podržava prekrivanje, prozirnu podlogu slike i animirane slike. Mana mu je ograničen broj boja koje može prikazati (256). Za razliku od GIF formata, JPEG (.jpeg) može prikazati desetke tisuća boja, stoga je pogodan za prikaz fotografija. JPEG format koristi algoritam koji može postići znatnu kompresiju slike. Na primjer, 200KB GIF se može reducirati na 30KB JPEG. Stoga, format GIF je bolji za prikaz slika u kojima nema puno boja kao što su crteži i ikone, dok je JPEG bolji za slike s puno boja i fotografije.

različitim formatima (gif, animated gif, jpg, jpeg, png, ico)



Slike

- Sliku se u HTML postavlja upotrebom oznake <img> koja ima samo početnu oznaku. Sintaksa je sljedeća:
- <img src="url">, gdje URL u atributu *src* označava lokaciju gdje se slika nalazi.
- Npr. <img src="ime_slike.ext"> HTML je element koji bi prikazao sliku *ime_slike.ext* u pregledniku.

Slike

- Npr. relativni URL

<img src="../ime_direktorija/ime_slike.ext">.

- Slika može biti i na webu:

<img src="http://www.ffst.hr/ime_direktorija/ime_slike.ext">

bi prikazalo sliku koja se nalazi na navedenom URL-u.

Ovaj način se ne preporuča ako autor web stranice nije vlasnik slike koju prikazuje jer uklanjanjem slike s tog URL-a nestaje i slika s web stranice.

Obrazac ili forma

- Vjerojatno ste često puta surfajući po internetu naletjeli na web stranice koje od vas zahtijevaju da unesete nekakve podatke (npr. ime, prezime, adresu, e-mail, broj kreditne kartice ...itd.). Takva web stranica naziva se HTML obrazac (engl. HTML form).

Forma

- Može se reći kako je HTML forma web stranica ili njezin dio koji sadrži polja za unos podataka.
- Uneseni podaci se dalje prosljeđuju nekoj skripti (PHP, JavaScript, ASP, CGI ...) koja podatke obrađuje te na temelju toga generira nekakav izlaz (slanje na e-mail adresu, stvaranje nove web stranice ...)

Forme

- Od korisnika se zahtijeva neka povratna informacija vezana za proizvod ili pruženu uslugu, omogućena je kupovina ili registracija proizvoda.
- Forme imaju dva osnovna dijela: kontejner same forme i sadržaj, koji obuhvaća niz ulaznih naredbi i sav neophodan prateći tekst i slike.
- Kontejner forme ima dva atributa: Method i Action.
- **Method** opisuje kako će podaci prikupljeni u formi biti poslani procesoru forme. Za većinu modernih svrha, ova će vrijednost biti “post”.
- **Action** definira koji će procesor forme biti upotrebljen. Na raspolaganju su obično dvije opcije: procesor elektroničke pošte i neki skript jezik.
- Kontejner se pojavljuje u ovakvom obliku:

```
<form method="post" action="forma.cgi">
```

- URL dodijeljen kao vrijednost atributu ACTION ukazuje na CGI skriptu koja će primiti i reagirati na podatke poslane od strane preglednika kada korisnik pritisne “Submit”.

Form oznaka

- Oznaka **<form> </form>** govori web pregledniku gdje obrazac počinje i gdje završava. Unutar ovih oznaka može se smjestiti bilo kojia HTML oznaka, a to znači da se unutra može ubaciti slika, tablica ili nešto treće.

```
<FORM>
ime: <INPUT><BR>
email: <INPUT>
</FORM>
```

ime:

email:

Atributi form oznake

- **action="URL“**

Određuje put do PHP, CGI, ASP... skripte (procesa) koja će obraditi unesene podatke kad kliknemo na SUBMIT dugme

action="../mojaskripta.php" PHPskripta**,**

action="javascript: nekafunkcija();" JavaScript skripta u našem html dokumentu**,**

action="http://novadomena.com/strana.html" otvara novi html dokument

action="mailto: ime@domena.com" otvara e-mail client (slično kao kod <a href="mailto:...">)

Atributi form oznake

- **method="GET | POST“**
- Određuje na koji način će se podaci (varijable) proslijediti do skripte. Postoje samo dva načina GET ili POST. Default je GET, a to znači da ako izostavimo ovaj atribut da će se podaci tako prosljeđivati do skripte

method="GET"

- tada se uneseni podaci prosljeđuju do poslužitelja kao dio URL-a i mogu se vidjeti u location baru vašeg preglednika. Npr. *http://domena.com/skripta.cgi?boja=plava&oblik=kockast* sadrži dvije varijable (podatka): boja čija je vrijednost plava i oblik s vrijednošću kockast, sve iza upitnika jesu varijable i njihove vrijednosti.

method="GET"

- Ova metoda ima dva ograničenja.
- Prvo, pomoću ove metode **može se poslati ograničena količina informacije** npr. 256byte ili 1kB što ovisi o poslužitelju na kojem je obrazac i o poslužitelju na kojem je skripta.
- Drugo, podaci koji se šalju mogu se vidjeti u adress tj. location baru browsera. Ovo iz sigurnosnih razloga ponekad nije dopustivo, osim ako npr. želimo cjelokupni URL zajedno s varijablama ostaviti u Favorites-u.
- GET metoda je poželjna samo ukoliko se radi o malo podataka koji ne zahtijevaju neku veliku sigurnost. Također njezina upotreba je poželjna kod **action="mailto:"**.

method="POST"

- je češća jer nema ograničenja kao GET metoda.
- POST je metoda koja prosljeđuje varijable obrasca kao blok podataka http protokola.
- Trenutno većina preglednika i poslužitelja ima ograničenje od 32kB za GET metodu dok POST može poslati znatno više informacija.
- POST je neznatno sigurnija od GET no kako se radi o slanju čistog texta (nema kriptiranja) postoji mogućnost da vam podaci budu "oteti".

name="string"

- Atribut name dodjeljuje ime našem obrascu. Npr. ako smo obrazac definirali s **<form name="moj_obrazac"></form>** tada će se obrazac moći pozvati kroz *JavaScript* objekt kao **window.document.moj_obrazac.name;** umjesto **window.forms[0].name;** .

Input oznaka

- Oznaka **<input ... >** kreira većinu polja za unos podataka.
- Ova oznaka nije container te ne treba pisati njegov zatvarajući tag </input>.
- Vrsta polja ovisi o type atributu te može biti:
  type = text | password | checkbox | radio | submit | reset | button | hidden | image | file

Atributi input oznake

- TYPE: određuje tip polja
- NAME: ime polja u formi
- VALUE: početna vrijednost polja
- SIZE: određuje širinu polja
- MAXLENGTH: maximalni broj znakova
- CHECKED: checkira checkbox ili radio dugme
- BORDER: debljina okvira oko slike

Atributi input oznake 2

- SRC: URL za sliku
- ALT: alt. text za sliku
- LOWSRC: verzija slike koja nije veliki fajl
- WIDTH: širina slike
- HEIGHT: visina slike
- ALIGN: određuje poravnanje texta oko slike
- VSPACE: vertikalna udaljenost slike i texta
- HSPACE: horizontalna udaljenost slike i texta

Atributi input oznake 3

- READONLY: vrijednost polja se ne može promijeniti
- DISABLED: korisnik ne može s poljem uraditi ništa
- ACCESSKEY: definira shortcut key npr. ALT+g
- AUTOCOMPLETE: ako browser koristi automatsko popunjavanje
- TABINDEX: određuje kojim će se redom mijenjati fokus poljima kad se pritšće na TAB tipku

Atributi input oznake 4

- LANGUAGE: upotrebljeni skriptni jezik
- onClick: kada korisnik klikne
- onChange: kad polje promijeni vrijednost
- onFocus: kad polje dobije fokus
- onBlur: kad polje izgubi fokus
- onKeyPress: kad se pritisne tipka na tipkovnici
- onKeyUp: kad se tipka otpusti
- onKeyDown: kad se pritisne tipka a polje je u fokusu
- Posljednjih 7 atributa (onClick,...) pozivaju JavaScript funkciju kad se nad poljem izvrši nekakva radnja.

Kontrola unosa na formi

- Text – polje za unos teksta
- Password – posebno polje za unos teksta
- Checkbox – okvir u koji se upisuje znak za potvrdu.
- Radio – ova kontrola slična je polju za potvrdu, ali važi za cijelu grupu. Samo jedna opcija može biti izabrana u jednom trenutku.
- Hidden – ova kontrola omogućava autoru postavljanje imena i vrijednosti kontrola koje korisnici ne moraju mijenjati.
- Submit – ova kontrola kreira standardnu tipku koja se koristi za naredbu pregledniku za izvođenje neke akcije definirane u elementu <form>.
- Image – ova kontrola omogućava uporabu slike umjesto tipke Submit.
- Reset – tipka koja svim elementima forme vraća početne vrijednosti.
- Select – ovom kontrolom daje se lista opcija koje korisnik može izabrati. Lista se može prikazati kao padajući izbornik ili kao okvir.
- Textarea – ova kontrola kreira područje za unos teksta koji može biti duži od jednog reda.
- Svaka od kontrola ima istu sintaksu:

```
<input type="ControlType" name="MyCtrl">
```

Dodatak za HTML obrazac

da


Slijedi . . . JavaScript

JavaScript

1. Uvod u programiranje JavaScript-om
2. JavaScript sintaksa
3. Osnove programiranja

Uvod u programiranje JavaScript-om

1. JavaScript i Java
2. Interpreteri i kompajleri
3. O JavaScript-u

JavaScipt i Java

- JAVA je programski jezik koji je razvila tvrtka Sun Microsystems.
- Java je jezik čiji se izvorni kod postupkom kompajliranja pretvara u binarni oblik. JavaScript je jezik koji je origninalno razvila tvrtka Netscape pod nazivom LiveScript.
- JavaScript je interpreterski orijentiran jezik. Java i JavaScript su dva potpuno različita programska jezika unatoč sličnosti u nazivu jezika. Svi programski jezici, pa i ova dva imaju neke sličnosti.
- JavaScript trenutno je jedini jezik za pisanje skripti kojeg podržavaju svi popularni web preglednici. Netscape Navigator podržava samo JavaScript dok Microsoft Internet Explorer podržava i JavaScript i VBScript. JavaScript može se koristiti i za pisanje skripti na strani servera.

Interpreteri i kompajleri

- Izvorni kod je niz naredbi napisanih u tekstualnom obliku od kojih je sačinjen program. Svi programski jezici kreću od izvornog koda koji se tada u zavisnosti od jezika interpretira ili kompajlira.
- Jezici koji izvorni kod interpretiraju u pravilu su jednostavniji za programiranje, ali sporiji prilikom izvođenja. Svaki put prilikom izvođenja potrebno je intrepretirati kod i to crtu po crtu u zavisnosti od toka izvršavanja programa. Tok programa određen je grananjem i petljama koje se izvršavaju u programu.
- Jezici koji kompajliraju kod, uobičajeno imaju složeniju sintaksu i zahtijevaju striktno poštivanje pravila prilikom programiranja. Kod ovako orijentiranih jezika prvo je potrebno ispisati izvorni kod, a nakon toga predati ga kompajleru koji kao rezultat daje izvršni kod u binarnom obliku. Na Windows platformi izlaz iz kompajlera najčešće ima nastavak .exe.

Interpreteri i kompajleri

- Izvršni program koji genereira kompajler obično je moguće izvoditi na točno određenoj platformi (operativnom sustavu). Značajna prednost za programera je činjenica da izvorni kod nije čitljiv nakon kompajliranja, a druga prednost je što prestaje biti bitno u kojem je programskom jeziku pisan neki program.
- JAVA je jezik koji iz izvornog koda kompajliranjem dolazi do izvršnog koda koji ne ovisi o platformi (operativnom sustavu) na kojem će se program izvršavati.
- Nezavisnost o platformi osigurava preglednik preko JAVA Virtual Machine i interpretera za JavaScript.
- JavaScript može proširiti korisnost web stranica u odnosu na one stranice koje se oslanjaju samo na HTML. Uporabom JavaScripta moguće je provjeravati točnost unosa podataka od strane korisnika, kreirati zanimljive efekte i otvarati prozore koji će se pojaviti kada se dogodi (okine) neki od predefiniranih događaja. Ako se JavaScript iskombinira sa CSS-om tada se dobiva stranica koja je poznata pod nazivom Dinamička HTML stranica.

O JavaScript-u

- JavaScript je interpreterski orijentiran programski jezik kojeg je moguće uključiti u HTML dokument (web stranicu). Pojam interpreterski orijentiran znači da će se u preglednik učitati cijela stranica, a JavaScript kod će se izvršiti po okidanju nekog događaja. Za vrijeme izvođenja programa kod se interpretira crtu po crtu. Postoje brojni događaji, poput klika na dugme ili završetka učitavanja stranice koji će prouzročiti okidanje, a time i izvršavanje nekog dijela koda.
- Netscape je tvrtka koja je kreirala JavaScript, ali je jezik nakon toga standardiziran od strane European Computer Manufacturers Association (ECMA). Danas postoji više inačica JavaScripta (1.0, 1.1, 1.2, ...), a jezik se kontinuirano razvija s razvojem Interneta i web tehnologija.

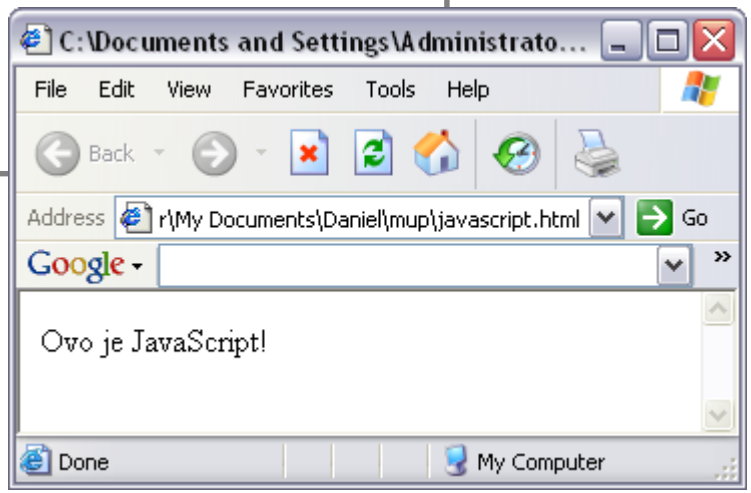
JavaScript sintaksa

1. Uključivanje JavaScript-e u HTML dokument
2. Smještanje JavaScript-a
3. Sintaksa i konvencije

Uključivanje JavaScript-e u HTML dokument

- JavaScript kod dodaje se u HTML dokument uporabom **script** oznake

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
document.write("Ovo je JavaScript!")
</script>
</body>
</html>
```



Uključivanje JavaScript-e u HTML dokument

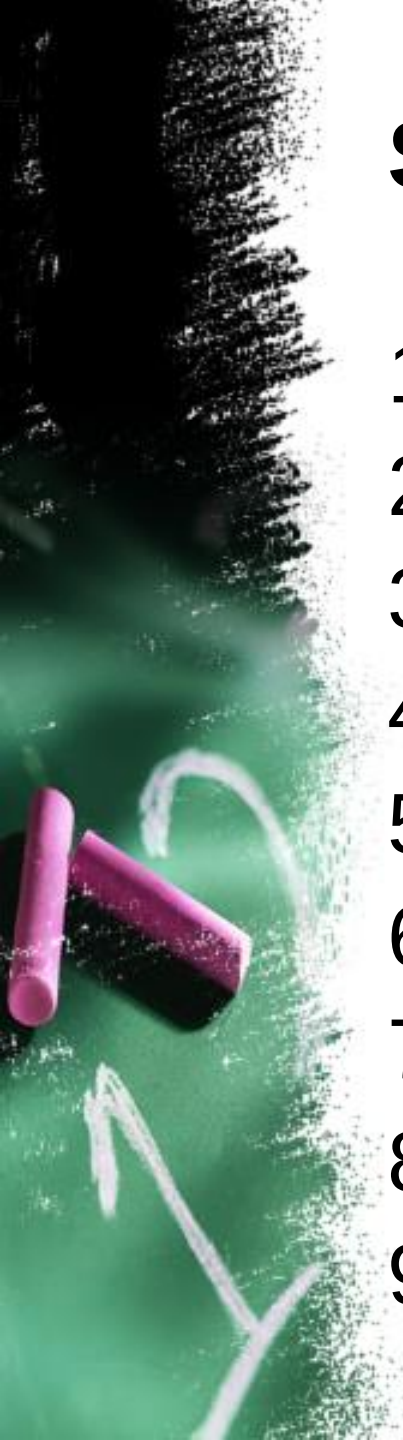
- Kako bi izbjegli prikazivanje koda skripte koju preglednik nije u mogućnosti izvršiti može se koristiti HTML oznaka komentara **<!--...-->**.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
<!-- Skrivanje JavaScript koda od starijih
   preglednika
   ... (JavaScript kod)
// kraj skrivanja JavaScript koda -->
</script>
</body>
</html>
```

- Oznake *//* u JavaScript-u imaju značenje komentara koji će spriječiti interpretiranje te linije koda.

Smještanje JavaScript-a

- Skripte se izvršavaju odmah po učitavanju u preglednik. Ovaj način obrade skripti nije uvijek poželjan. Ponekad je skriptu potrebno izvršiti za vrijeme učitavanja stranice, a nekad samo kada se aktivira neki od okidača.
- Skripte koje će se izvršavati po pozivu ili na određeni događaj stavljaju se u head (zaglavlje) HTML dokumenta. Ovim se osigurava dostupnost skripte (skripta je učitana) prije korištenja.
- Skripte koje će se izvršiti za vrijeme učitavanja stranice stavljaju se u body (tijelo) HTML dokumenta. Takve skripte uobičajeno služe za generiranje sadržaja stranice.
- Broj skripti koje se mogu uključiti u HTML dokument nije ograničen. Dozvoljeno je definirati skripte i u zaglavlju i u tijelu dokumenta.



Sintaksa i konvencije

1. Osjetljivost na velika i mala slova
2. Točka-zarez
3. Razmaci
4. String i navodnici
5. Backslash (\) i stringovi
6. Otvaranje i zatvaranje zagrada
7. Komentari
8. Varijable i imena funkcija
9. Rezervirane riječi

Osjetljivost na velika i mala slova

- JavaScript je jezik koji razlikuje velika od malih slova, što znači kako se riječi iznos, Iznos i IZNOS tretiraju kao različite riječi.

Točka-zarez

- Svi izrazi trebaju završavati sa točka-zarezom (**;**). Točka-zarez razdvaja dva izraza.
- Primjer:

```
var x = 0; var y = 10;
```

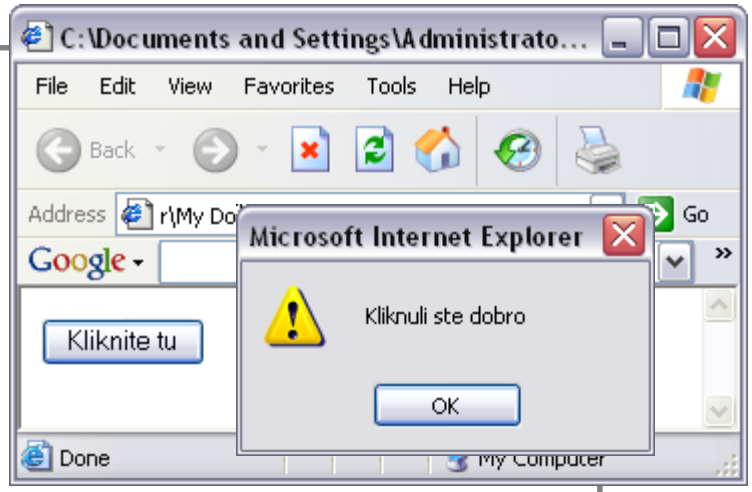
Razmaci

- JavaScript kao i HTML ignorira razmak, tab i novi red koji se pojavljuju unutar izraza. JavaScript raspoznaje razmake, tabove i novi red kao dijelove stringa.
- var x = 0; isto je kao i var x=0;
- Rezultat obaju primjera isti je, ali je dobro koristiti razmake kako bi se poboljšala čitljivost koda. Nije dobro naredbu razvući preko više redova, iako je to sintaksom JavaScripta dozvoljeno.
- Prazna mjesta obvezna su kako bi se razdvojile naredba i varijabla. Primjerice varx=0; nije ispravno napisan izraz jer je razmak između var i x neophodan za pravilno tumačenje izraza.

String i navodnici

- String je niz znakova omeđenih jednostrukim ili dvostrukim navodnicima ('jednostruki', “dvostruki”).
- Dozvoljeno je gnježđenje dvostrukih navodnika unutar jednostrukih i obratno.

```
<html>
<head>
</head>
<body>
<input type="Button"
value="Kliknite tu"
onclick=
"window.alert('Kliknuli ste dobro');">
</body>
</html>
```



- U ovom primjeru dvostruki navodnici koriste se kako bi se odredio atribut HTML oznake. Iz tog razloga poruku koja će se pojaviti u prozoru trebalo je omeđiti jednostrukim navodnicima.

Backslash (\) i stringovi

- Ovaj znak slijedit će drugi kojeg nije moguće otipkati na tipkovnici.
- Primjerice, ako želimo riječ “Kliknuli” prikazati u jednom redu, rijec “ste” u drugom redu, a riječ “dobro” u trećem redu string bi izgledao ovako: ”Kliknuli\nste\ndobro”
- Kombinacija backslash i slova obično se naziva escape sekvencom. Neke od uobičajenih sekvenci su:
- \b backspace – jedno mjesto prema natrag
- \f form feed – početak stranice
- \n new line – novi red
- \r carriage return – početak reda
- \t tab
- \' jednostruki navodnik
- \” dvostruki navodnik
- Posljednje dvije sekvence bitne su jer omogućavaju ispis navodnika bez prethodne interpretacije što je naručito bitno za značenje jednostrukih navodnika u engleskom jeziku.

Otvaranje i zatvaranje zagrada

- Sve otvorene zagrade moraju se zatvoriti, ovo uključuje (), [], {}.
- Primjer:

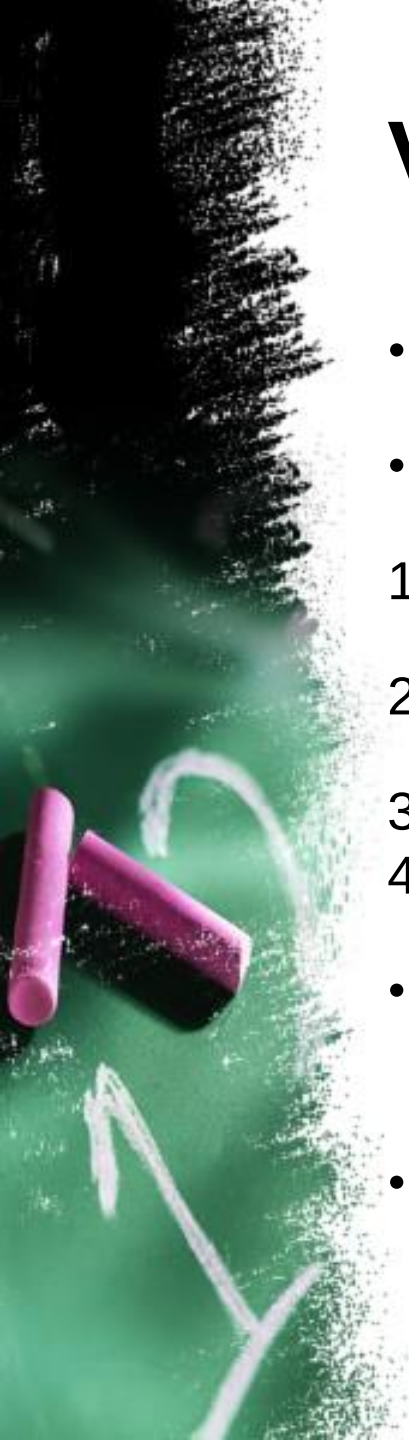
```
winpop = window.open('primjer1.html',
'popup',
'scrollbars=yes');
if ( x[0] == 10 )
{
   x[0] = 0;
   x[1] = 0;
}
```

- Vitičaste zagrade { } koriste se za omeđivanje više JavaScript izraza.
- Uglate zagrade [ ] dio su posebne podatkovne strukture koja se naziva polje.
- Okrugle zagrade ( ) omeđuju funkcije ili argumente metode.

Komentari

- Komentar se naznačuje dvostrukom kosom crtom //.
- Ovakav način označavanja komentara prihvatljiv je za komentar koji je ograničen na jednu crtu koda, ali ako se želi zakomentirati veći dio teksta u kodu poželjno je koristiti /* kao početnu oznaku komentara i */ kao završnu oznaku komentara.
- U primjeru bi to izgledalo ovako:

```
// Jednolinijski ili
/* Komentari često služe programerima
kako bi objasnili logiku koja stoji iza
nekog dijela koda ili razlog iz kojeg je kod
izmijenjen. Ovo se čini kako bi se lakše
razumjele akcije programera nakon nekog
duljeg vremena
*/
```



Varijable i imena funkcija

- Kao programer potrebno je izabrati i dodijeliti nazive varijabli i funkcija.
- Nazivi varijabli i funkcija moraju pratiti nekoliko jednostavnih pravila:

1. Prvi znak mora biti slovo abecede (malo ili veliko slovo) ili donja crta (_).
2. NIJE DOZVOLJENO koristiti brojeve kao prvi znak u nazivu
3. NIJE DOZVOLJENA upotreba razmaka u nazivu
4. NIJE DOZVOLJENA upotreba rezerviranih riječi u nazivu

- Evo nekoliko primjera ispravnih naziva

  X, zbroj_dva_broja, x13, _sve, $iznos_u_dolarima

- Preporučuje se pri izboru imena birati nazive koji će asocirati na varijablu ili funkciju koja se piše. Kako JavaScript razlikuje mala od velikih slova uobičajeno je umjesto naziva zbroj_dva_broja pisati ZbrojDvaBroja.

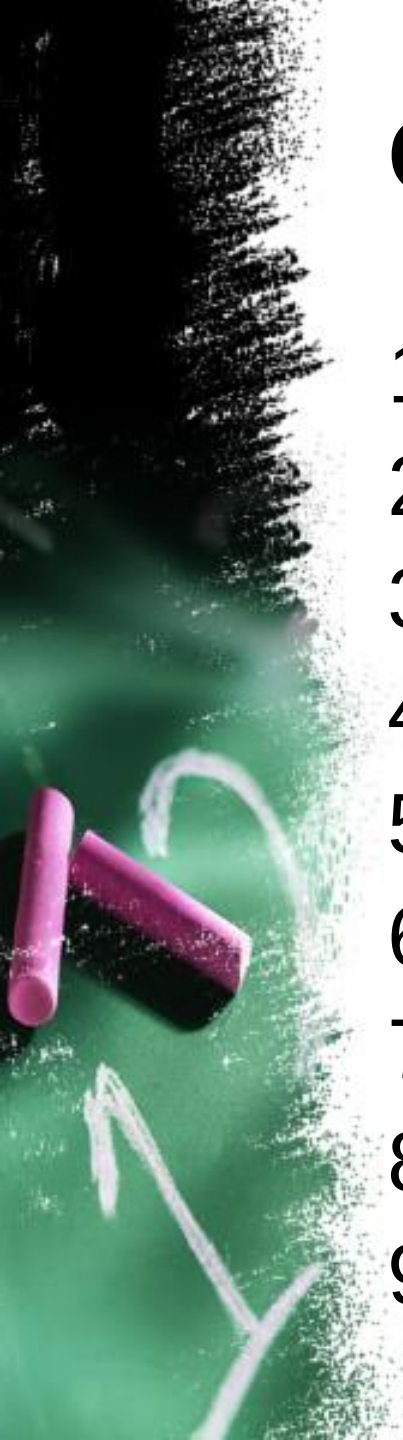
Rezervirane riječi

- Postoje brojne riječi koje su sastavni dio JavaScript jezika. Te riječi nije dozvoljeno koristiti za nazive varijabli I funkcija jer interpreter ne bi bio u mogućnosti razlučiti što je izvorna naredba jezika, a što varijabla ili funkcija.
- Lista rezerviranih riječi u JavaScriptu:

abstract
alert
arguments
Array
blur
boolean
Boolean
break
byte
callee
caller
captureEvents
case
catch
char
class
clearInterval
clearTimeout
close
closed
confirm
const
constructor
continue
Date
debugger
default
defaultStatus
delete
do
document
double
else
enum
escape
eval
export
extends
final
finally
find
float
focus
for
frames
Function
function
goto
history
home
if
implements
import
in
Infinity
innerHeight
innerWidth
instanceof
int
interface
isFinite
isNaN
java
length
location
locationbar
long
Math
menubar
moveBy
moveTo
name
NaN
native
netscape
new
null
Number
Object
open
opener
outerHeight
outerWidth
package
Packages
pageXOffset
pageYOffset
parent
parseFloat
parseInt
personalbar
print
private
prompt
protected
prototype
public
RegExp
releaseEvents
resizeBy
resizeTo
return
routeEvent
scroll
scrollbars
scrollBy
scrollTo
self
setInterval
setTimeout
short
static
status
statusbar
stop
String
super
switch
synchroniz
this
throw
throws
toolbar
top
toString
transient
try
typeof
unescape
unwatch
valueOf
var
void
watch
while
window
with
FALSE
TRUE

Osnove programiranja

“ponavljanje onoga što već znate iz drugih programskih jezika”



Osnove programiranja

1. Deklariranje varijabli
2. Tipovi varijabli
3. Uporaba operatora
4. Vidljivost varijable
5. Upravljačke strukture
6. Funkcije
7. JavaScript objekti
8. DOM (Document Object Model)
9. Događaji (events)

Deklariranje varijabli

- Deklaracija varijable (kreiranje varijable) vrši se ključnom riječi **var**:

  var x;

  var zbroj;
- Više varijabli može se deklarirati jednom **var** ključnom riječi:

  var x,y,zbroj;
- Varijablu je u JavaScript-u moguće kreirati i direktnom inicijalizacijom vrijednosti varijable:

  x=0;

  zbroj=0;
- Iako je ovakav način dozvoljen zbog preglednosti zgodnije je prije inicijalizacije izvršiti deklaraciju ili barem iskombinirati deklaraciju i inicijalizaciju kao što se može vidjeti u ovom primjeru:

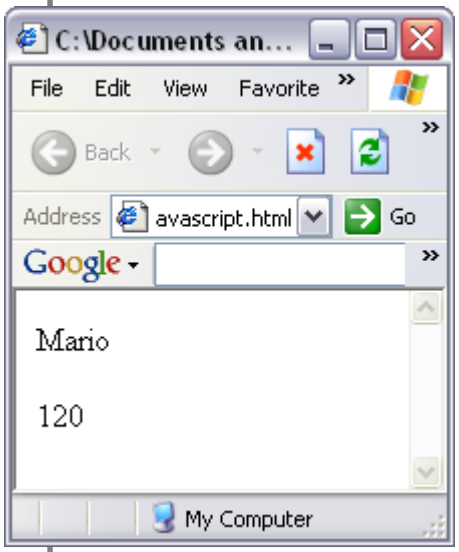
  var x=1, y=3, zbroj=0;
- Ako se posebno ne inicijalizira deklarirana varijabla poprima posebnu vrijednost JavaScripta undefined (nedefiniranu) vrijednost.

Tipovi varijabli

- Velika razlika između JavaScripta i drugih jezika poput Jave i C-a je u tome što JavaScript nije orijentirana prema tipu varijable. Ovo omogućava izmjenu tipa podatka varijable za vrijeme izvođenja skripte.

var x = 10; x = “deset”;

```
<html>
<body>
<script type="text/javascript">
// deklaracija u string i ispis
var ime = "Mario";
document.write(ime);
/* postavljanje brojčane
vrijednosti i ispis */
ime = 120;
document.write("<p>"+ime+"</p>");
</script>
</body>
</html>
```



Upotreba operatora

- Operatori djeluju na varijable. U primjeru je korišten operator dodjele vrijednosti varijabli **=**. Još jedan od operatora je **+**.

```
var x = 1, y = 3, zbroj = 0;
zbroj = x + y;
```

- Ovaj djelić koda deklarira varijable x,y i zbroj. Dodijeli varijablama x, y i zbroj vrijednosti 1, 3 i 0 respektivno. Druga crta skripta zbraja vrijednosti x i y i dodijeli tu vrijednost varijabli zbroj. Vrijednost varijable zbroj bit će 4 nakon izvršenja koda.
- Drugi operatori koriste se za uspoređivanje:

```
var x = 1, y = 3, zbroj = 0;
if (zbroj == 0 && x > -5)
{
    zbroj += x + y;
}
```

Vidljivost varijable

- Kad se varijabla deklarira unutar funkcije ona je vidljiva samo unutar te funkcije. Kada se funkcija napusti varijabla se uništi. Ove varijable nazivaju se lokalne varijable.
- Moguće je koristiti varijable s istim imenom u više funkcija jer je svaka od njih vidljiva samo unutar funkcije u kojoj je deklarirana.
- Ako se varijabla deklarira van funkcije tada je ona vidljiva od strane svih funkcija.
- Život ovako definirane varijable počinje s njenom deklaracijom, a završava zatvaranjem stranice.

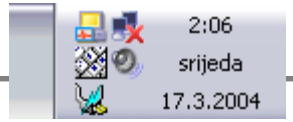
Upravljačke strukture

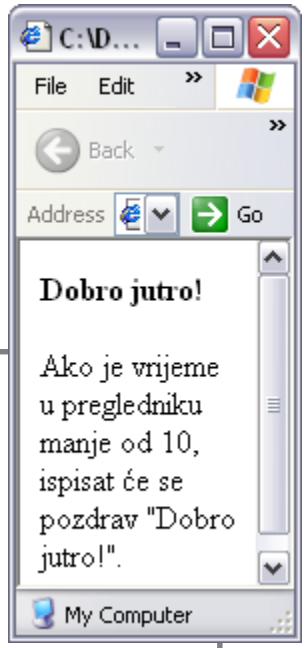
1. Uvjetne naredbe
2. Petlje

Uvjetne naredbe

- **If** je osnovna upravljačka naredba koja omogućuje programu provjeravanje nekog uvjeta te nastavak izvršavanja u zavisnosti od rezultata tog testa.
- Primjer jednostrukog odabira:

```
if((x==1)&&(y==3)
{
     sum = y-x;
}
```



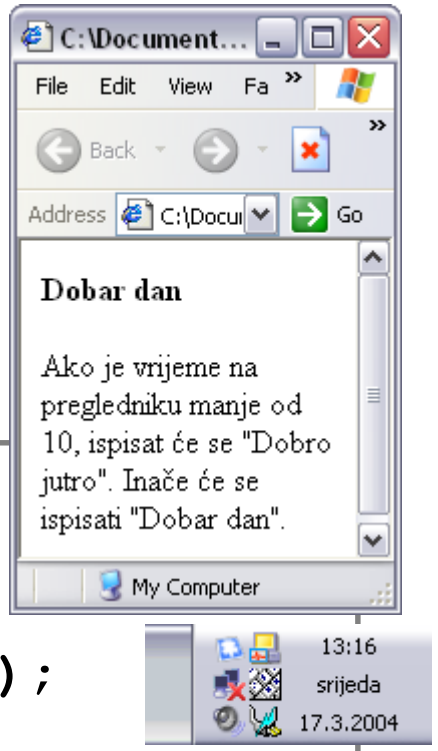



```
<html>
<body>
<script type="text/javascript">
var d=new Date(),time=d.getHours();
if(time<10)
document.write("<b>Dobro jutro!</b>");
</script>
<p>Ako je vrijeme u pregledniku manje od 10,
ispisat će se pozdrav "Dobro jutro!".</p>
</body></html>
```

Uvjetne naredbe

- Primjer dvojnog odabira:

```
if (sum==0) sum=x+y;
else subtotal=sum;
```



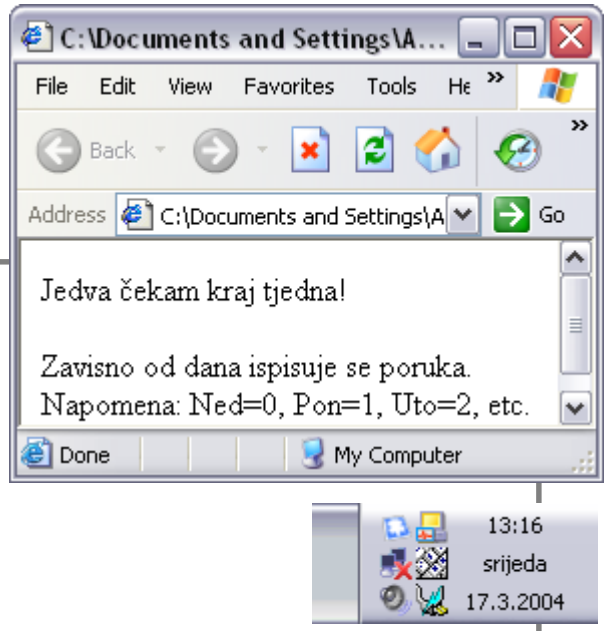

```
<html>
<body>
<script type="text/javascript">
var d=new Date(),time=d.getHours();
if (time < 10)
  document.write("<b>Dobro jutro</b>");
else
  document.write("<b>Dobar dan</b>");
</script>
<p>Ako je vrijeme na pregledniku manje od
10, ispisat će se "Dobro jutro". Inače će se
ispisati "Dobar dan".</p>
</body></html>
```

Uvjetne naredbe

- Switch – case naredba je pogodna pri višestrukom grananju programa. Ona omogućava izbor jednog između više mogućih puteva daljnjeg izvođenja programa, što ovisi o vrijednosti koju poprima varijabla odnosno izraz.
- Primjer:

```
switch(n){
        case 1: // ako je n jednak 1
                // kod
        break;
        case 2: // ako je n jednak 2
                // kod
        break;
        ...
// ako ni jedan uvjet nije zadovoljen
        default:
                // kod
}
```

Uvjetne naredbe

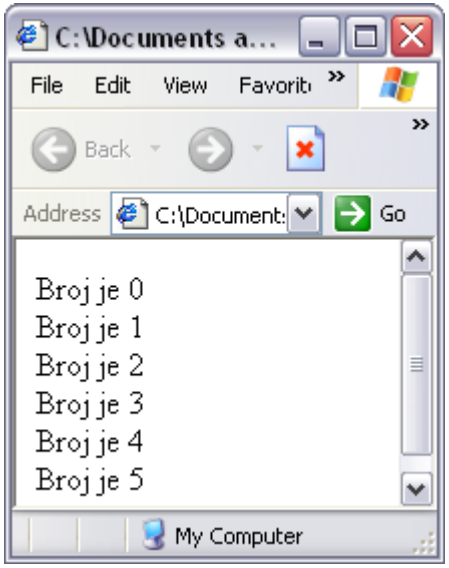



```
<html>
<body>
<script>
var d = new Date(),t;
var Dan=d.getDay();
switch (Dan){
   case 5: t="Konačno petak";          break;
   case 6: t="Savršena subota";        break;
   case 0: t="Pospana nedjelja";       break;
   default:t="Jedva čekam kraj tjedna!";
}
document.write(t);
</script>
<p>Zavisno od dana ispisuje se poruka.<br />
Napomena: Ned=0, Pon=1, Uto=2, etc.</p>
</body>
</html>
```

Petlje

- **While** – Dok je zadovoljen uvjet, izvršava jednu ili više naredbi. While petlje posebno su korisne u slučajevima kada se ne zna koliko puta treba ponoviti petlju, ali je poznato u kojem slučaju petlju treba napustiti.

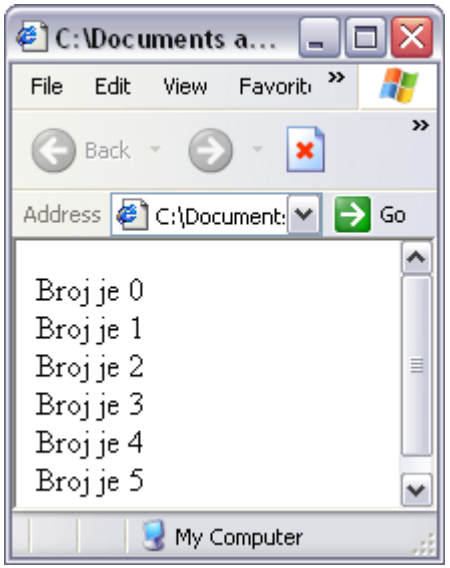
```
<html>
<body>
<script type="text/javascript">
var i=0;
while(i<=5)
{
   document.write("Broj je "+i);
   document.write("<br />");
   i++;
}
</script>
</body>
</html>
```



Petlje

- **Do ... While –** Ova petlja identična je prethodnoj, ali uz razliku što će se svakako izvršiti barem jednom.

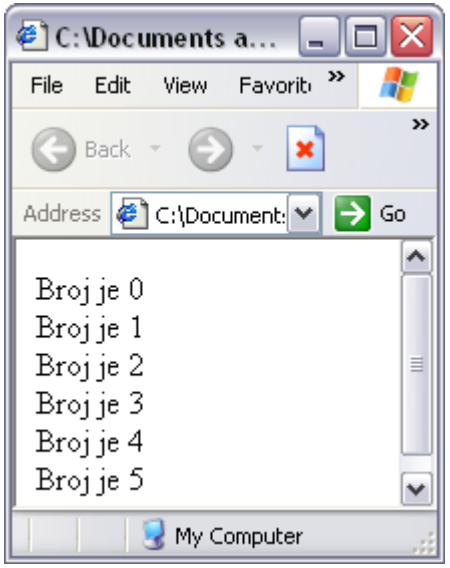
```
<html>
<body>
<script type="text/javascript">
var i=0;
do
{
   document.write("Broj je "+i);
   document.write("<br />");
   i++;
}while(i<=5);
</script>
</body>
</html>
```



Petlje

- **For –** For petlje korisne su kada se točno zna broj ponavljanja petlje.

```
<html>
<body>
<script>
for(i=0;i<=5;i++)
{
   document.write("Broj je "+i);
   document.write("<br>");
}
</script>
</body>
</html>
```



Funkcije

1. Definicija funkcije
2. Poziv funkcije
3. Return naredba

Funkcije

- Rješavanje složenih programskih problema moguće je zahvaljujući dekompoziciji problema. Funkcije su strukturni blokovi koji dozvoljavaju dekompoziciju.
- Uporaba takvih strukturnih blokova ima tri prednosti:
- Za vrijeme rada na jednom strukturnom bloku, sva pažnja usmjerava se na jedan, manji dio problema.
- Moguća je podjela zadaća na više programera.
- Moguće je iskoristiti gotov blok na nekom drugom mjestu u programu.
- Kako bi se efektno moglo pokazati poziv funkcije koristit će se alert tip prozora. Način na koji se poziva ovaj prozor iz JavaScripta je:

```
alert(“Poruka korisniku.”);
```

Definicija funkcije

- Da bi se kreirala funkcija potrebno je definirati joj ime, argumente i neke naredbe:

```
function myFunction(arg1,arg2,...){
   blok naredbi
}
```

- Fja bez argumenata mora uključivati okrugle zagrade:

```
function myFunction(){
   blok naredbi
}
```

- Argumenti su varijable koje će se koristiti u funkciji. Vrijednosti varijabli bit će vrijednosti koje su se prenijele prilikom poziva funkcije.
- Stavljanjem funkcije u zaglavlje dokumenta osiguravate dostupnost funkcije prije njenog poziva.
- Neke funkcije vraćaju vrijednost izrazu koji ih je pozvao:

```
function myFunction(a,b){
   return a+b;
}
```

Poziv funkcije

- Funkcija se ne izvršava prije poziva funkcije. Može se pozvati funkcija koja sadržava argumente:

```
myFunction (arg1,arg2,...);
```

- ili ona koja ne sadržava argumente:

```
myFunction ();
```

Return naredba

- Funkcija koja vraća rezultat mora koristiti **return** naredbu. Ova naredba specificira vrijednost koja će biti vraćena pozivatelju funkcije.

```
function myFunction(a,b){
  return a+b;
}
```

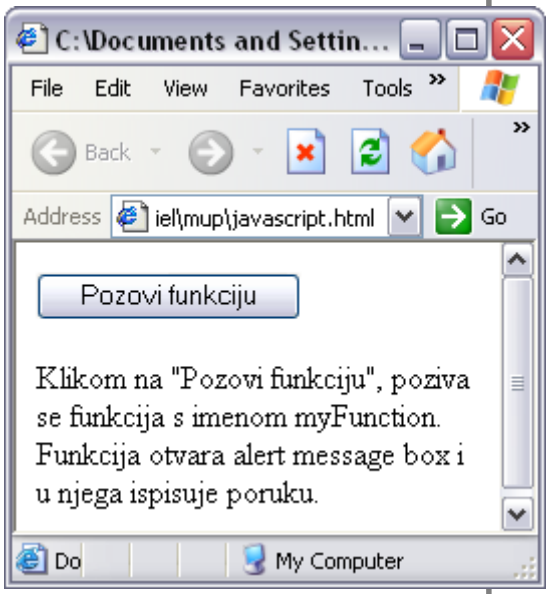
- Prilikom poziva ove funkcije potrebno je navesti oba argumenta:

```
var sum = myFunction(2,3);
```

- Varijabla suma poprima vrijednost 5.

Return naredba

```
<html>
<head>
<script type="text/javascript">
function myFunction()
{
   alert("Pozdrav!");
}
</script>
</head>
<body>
<form><input type="button"
onclick="myFunction()"
value="Pozovi funkciju"></form>
<p>Klikom na "Pozovi funkciju", poziva se
funkcija s imenom MojaFunkcija. Funkcija
otvara alert message box i u njega ispisuje
poruku.</p></body></html>
```



Malo ponavljanja

Vidljivost varijable

- Kad se **varijabla** deklarira **unutar funkcije** ona je **vidljiva** samo **unutar** te **funkcije**. Kada se funkcija napusti varijabla se uništi. Ove varijable nazivaju se **lokalne varijable**.
- Moguće je koristiti varijable s istim imenom u više funkcija jer je svaka od njih vidljiva samo unutar funkcije u kojoj je deklarirana.
- Ako se **varijabla** deklarira **van funkcije** tada je ona vidljiva od strane **svih funkcija**. **Život** ovako definirane varijable **počinje** s njenom **deklaracijom**, a **završava zatvaranjem stranice**.

Procjena varijabli

- Varijabla ili niz kojoj se nije pridružila neka vrijednost dobiva vrijednost **undefined** (nedefinirano) .
- Rezultat procjene nepridružene varijable ovisi o načinu deklariranja:
  - Ako je nepridružena varijabla definirana **bez var**, procjena rezultira sa **runtime error**.
  - Ako je nepridružena varijabla definirana **sa var**, procjena rezultira sa **undefined value**, ili sa **NaN (Not a Number) kod brojeva**.

Primjer procjene

Bez var

```
function f1() {
    return y - 2;
}
f1() //stvara runtime error
```

Sa var

```
function f2() {
    return var y - 2;
}
f2() //daje NaN
```

Primjer procjene

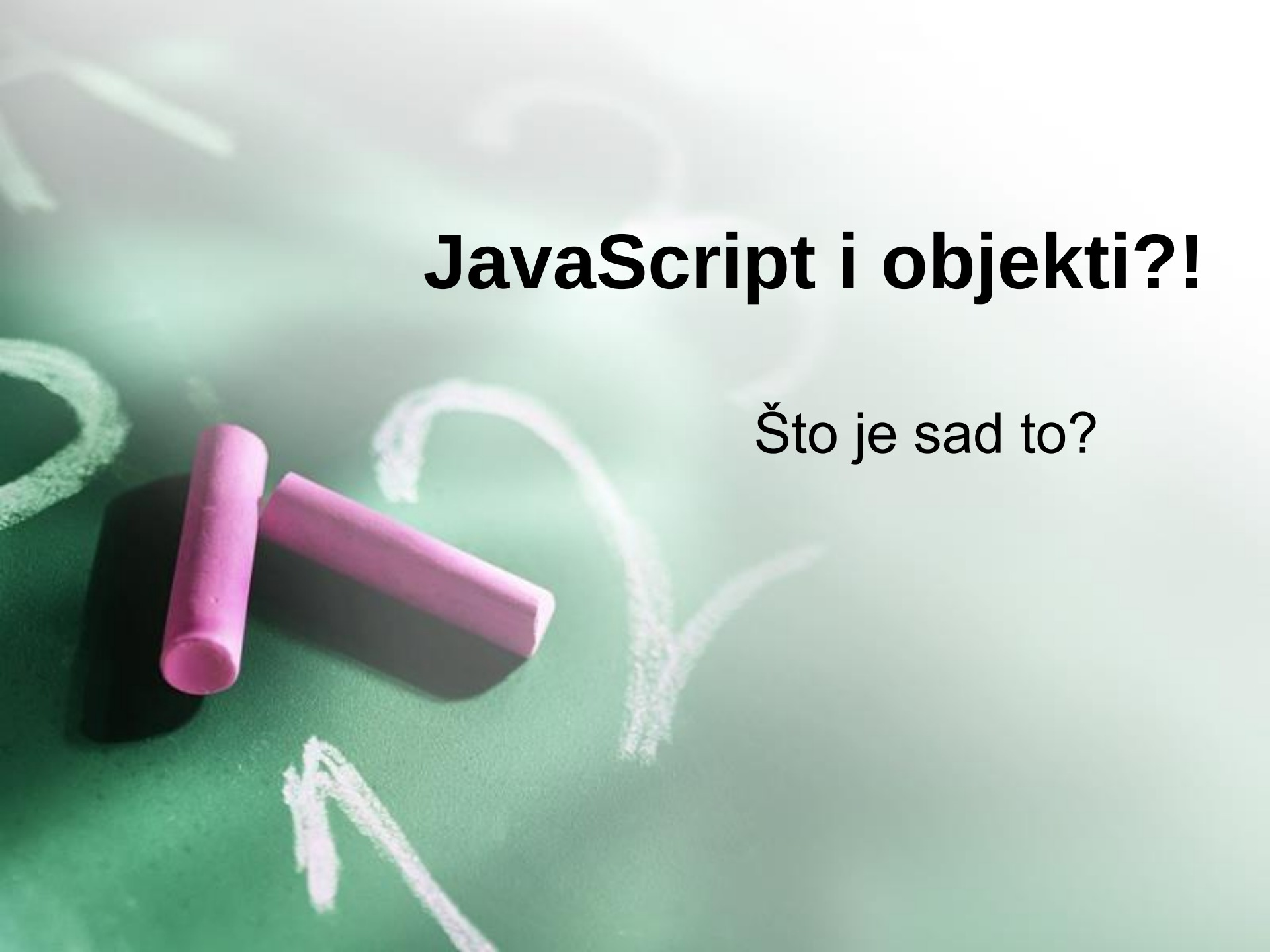
Undefined je moguće koristiti za provjeru ima li varijabla vrijednost. Ako varijabli ulaz nije pridružena nikakva vrijednost, if naredba daje true.

```
var ulaz;
   if(ulaz === undefined){
      RadiOvo();
    }
  else {
     RadiOno();
  }
```

Strogo jednako (===)

Daje true ako su operandi jednaki i istog tipa.

3 === var1



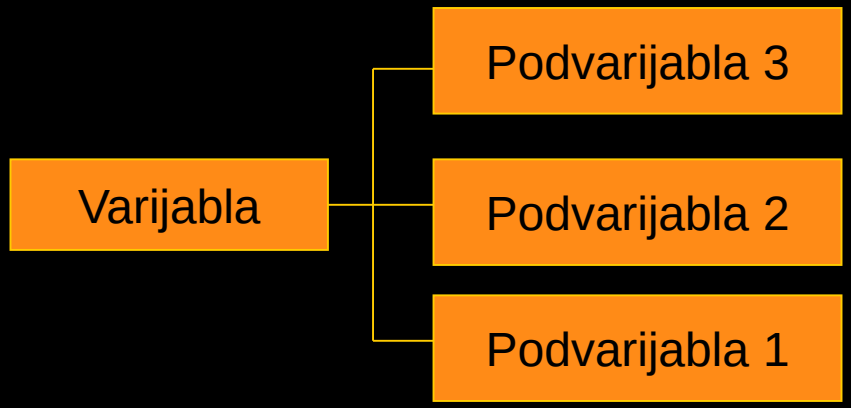
JavaScript i objekti?!

Što je sad to?

JavaScript Objekti drukčije

Objekt? #1

- Što su objekti? Zamislite da je objekt varijabla sa nizom pod-varijabli.
- Evo pokušaja grafičkog objašnjenja:



Podvarijablama objekta pristupa se na vrlo sličan način kao i varijablama, a osnovna je razlika postojanje točke.
primjer:

varijabla.podvarijabla

Objekt? #2

- Pretpostavimo kako imamo varijablu koja je skupina manjih varijabli i kojom moramo opisati psa.
- Npr. neka je varijabla "Pas" i neka su podvarijable "Broj_repova", "Broj_ruku", "Broj_nogu" & "Broj_ociju".

- Da bi koristili ove podvarijable moramo im pristupiti na sljedeći način:
  - Pas.Broj_repova ,
  - Pas.Broj_ruku ,
  - Pas.Broj_nogu i
  - Pas.Broj_ociju.

Primjer objekta Pas

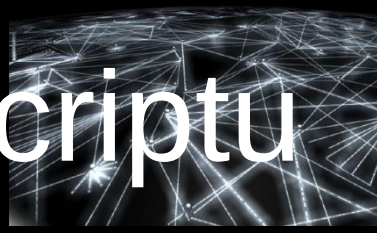
Primjer:

- Pas.Broj_repova = 1;
- Pas.Broj_ruku = 0;
- Pas.Broj_nogu = 4;
- Pas.Broj_ociju = 2;

Opišimo psa kojeg smo definirali pod-varijablama.

Opisani pas ima jedan rep, nema ruku, ima četiri noge i dva oka.

Kreiranje objekta u JavaScriptu



- Kako bi kreirali objekt u JavaScriptu?
- Postoji niz načina za kreiranje objekata. Jedan od njih je jednostavno ih početi korisiti.
- U prethodnom primjeru kada smo definirali značajke ili svojstva (eng. properties) psa (Pas) (rep, ruke, itd.) kreirali smo objekt.
- To je bilo vrlo jednostavno, ali postoji i bolji način za kreiranje objekata.

Kreiranje objekta u JavaScriptu

- Definirajmo objekt funkcijom, koristeći new sintaksu.

```
function zivotinja (br_repova,br_ruku,br_nogu,br_ociju){
    this.Broj_repova = br_repova;
    this.Broj_ruku = br_ruku;
    this.Broj_nogu = br_nogu;
    this.Broj_ociju = br_ociju;
}

var Pas = new zivotinja(1,0,4,2);
```

- Sada se postavlja pitanje što je to novo (new)?

Kreiranje objekta u JavaScriptu

- **new je posebna riječ u** JavaScript-u koja se odnosi na objekt koji se kreira.
- Kada kreirate objekt

var Pas = new zivotinja(1,0,4,2); dodjeljujete riječ Pas **objektu** this (ovaj).

- Što bi se dogodilo kada bi morali kreirati cijeli niz životinja, a pretpostavka je kako je za opis svake životinje dovoljno definirati broj repova, ruku, nogu i očiju... ?

Kreiranje objekta u JavaScriptu

- primjer koji nam omogućava ovakav način kreiranja objekata:

```
var Ptica = new zivotinja(1,0,2,2);
var Majmun = new zivotinja(1,2,2,2);
var Riba = new zivotinja(1,0,0,2);
var Zohar = new zivotinja(0,0,6,2);
```

umjesto utipkavanja 16 linija!!

Metode

- Nije sve gotovo!
- Objekti pored značajki, svojstava (variabli), mogu imati i metode (eng. methods).
- *Method,* metoda je jednostavno funkcija unutar objekta.
- Možemo učiniti ovo:
  Pas.Broj_repova=1; ili pristupiti metodi, (funkciji) definiranoj za objekt.
  Pas.oglasi_se();

Metode

- Ovakav poziv tjera da objekt Pas (pas) započne lajati (oglasi_se).
- Metoda je akcija. Ona tjera objekt na rad, akciju.
- Kako definirati metodu?
  - Evo primjera:

naziv_objekta.naziv_metode = naziv_funkcije

- Tako, nakon definicije funkcija dodijelite ih objektu.
  Najbolje je to vidjeti na primjeru.

Primjer

```
<html>
<head>
<script type="text/javascript">
<!-- sakrivanje JavaScript koda od starijih preglednika →
function oglasi_se (string) {
//Izlaz je neka vrsta govora
document.write(string);
}
function rasti (naziv_svojstva) {
//Tjera dio objekta na rast
this[naziv_svojstva]++;
}
function ispisi_svojstva() {
//ispisuje značajke, eng. properties objekta
for(var naziv_svojstva in this) {
document.write(naziv_svojstva + "=" + this[naziv_svojstva] + "<br>");
    }
}
```

Objekti #1

- Nadam se kako ste primijetili nekoliko novosti vezanih za objekte.
- Prije smo referencirali nazive značajki koristeći tzv. točka notaciju, kao što je: Pas.Broj_repova.
- Postoji još jedan način referenciranja značajke i to pomoću uglatih zagrada: Pas["Broj_repova"].
- Potrebno je samo koristiti string (niz znakova) kao naziv značajke.

Objekti #2

- Iduća novost nalazi se u metodi ispisi_svojstva, koristio se sljedeći kod:
- for(var naziv_svojstva in this) { document.write(naziv_svojstva + "=" + this[naziv_svojstva] + "<br>"); } Prvo, document.write() metoda samo ispisuje tekst. Naziv ovakvog izraza je **for..in** statement. Izraz for..in prolazi preko naziva značajki, svojstava objekta.
- Format takvog izraza je:
- **for** (*neka_varijabla* **in** *naziv_objekta*) { *izrazi* }

  Uzima varijablu (u ovom slučaju naziv_svojstva), i izjednačava je sa imenom prve značajke u objektu (u ovom slučaju Broj_repova).

  Nakon toga prelazi na iduću značajku.

  Nastavlja se sa istim postupkom dok se ne prođe preko svih značajki. Ovo uključuje i varijable i metode.

Objekti #3

- Postoji još jedna novost i to u **with** izrazu.
  Ponekad, kada radite sa jednim objektom (npr. Pas), dosadit će stalno pisati Pas.
  Pomoći će izraz **with**! Moguće je objekt Pas postaviti kao tzv. predefinirani (eng. default) objekt.

Sintaksa je:

- **with** (*naziv_objekta*) { *izrazi* }
  Ovako napisan izraz podrazumijeva kako se svi izrazi napisani između vitičastih zagrada {} odnose na dani objekt.
  Imali smo: Pas.Broj_repova=3; Pas.Broj_ruku=2; Pas.Broj_nogu=Pas.Broj_ruku*3; Pas.Broj_ociju=Pas.Broj_repova*5;

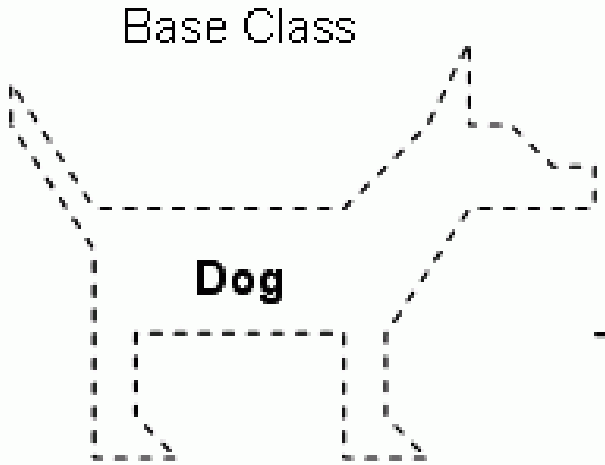
  Uporabom with izraza, pojednostavljeno: with (Pas) { Broj_repova=3; Broj_ruku=2; Broj_nogu=Broj_ruku*3; Broj_ociju=Broj_repova*5; }

Objekti #4

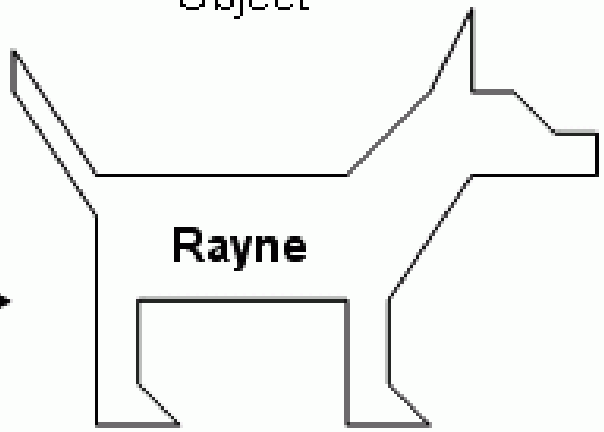
Nadam se da je ovaj brzi uvod ipak koristio.

Vježbom i korištenjem objekata u programiranju priviknut ćete se na objekte, a ako nešto ne zapamtite pogledajte na ovoj stranici primjer.

Kako OO u JS?



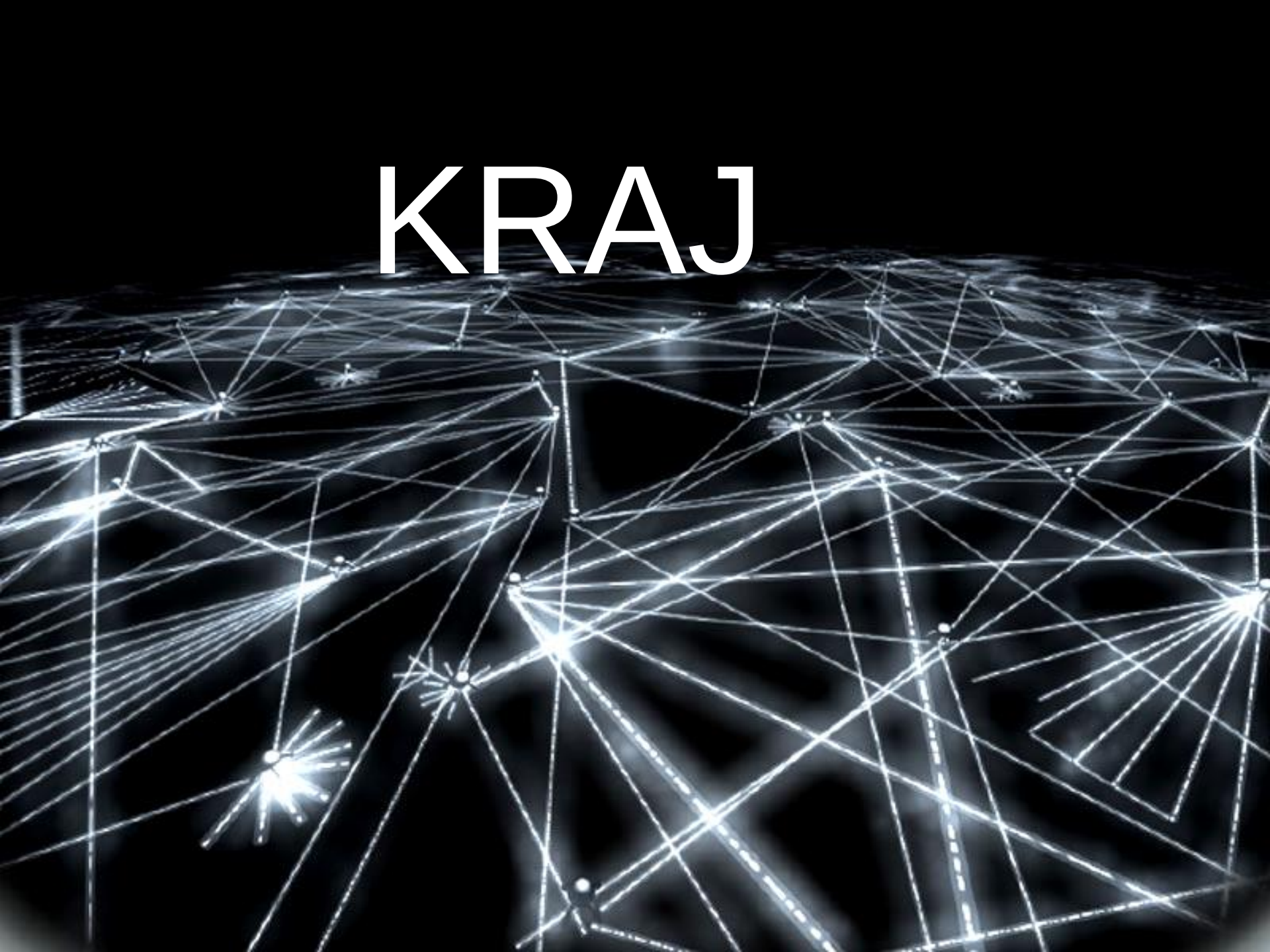
Create Instance



Properties	**Methods**
Color	Sit
Eye Color	Lay Down
Height	Shake
Length	Come
Weight	

Property values	**Methods**
Color: Gray, White, and Black	Sit
Eye Color: Blue and Brown	Lay Down
Height: 18 Inches	Shake
Length: 36 Inches	Come
Weight: 30 Pounds	

http://dean.edwards.name/weblog/2006/03/base/


KRAJ

JavaScript Objekti klasično

JavaScript objekti

- **Booleov objekt**
- **String objekt**
- **Objekt polja**
- **Objekt datuma**
- **Matematički objekt**

Booleov objekt

- Brojčani podaci i stringovi imaju neograničen broj mogućih vrijednosti, dok Booleovi podaci imaju samo dvije moguće vrijednosti, true ili false.

Metode	Opis	NN	IE	ECMA
toString()	Vraća string Booleovu vrijednost. (true or false)	3.0	3.0	1.0
valueOf()	Vraća vrijednost specificiranog objekta	4.0	4.0	1.0

String objekt #1

- String je niz znakova koji kod JavaScripta predstavlja tekst. Jedna od mogućnosti JavaScripta je **spajanje** stringova sa **'+'** operatorom i to tako da se drugi string doda prvome, na primjer:

```
ime = "Pero " + "Perić";                  // rezultat je "Pero Perić"
recenica = "Moje ime je" + " " + ime;// rezultat je "Moje ime je Pero Perić"
```

- Za duljinu stringa odnosno, **broj znakova u stringu** koristi se **lenght** svojstvo (**property**) stringa. Ako varijabla sadrži string, duljini stringa se može pristupiti na sljedeći način:

```
zadnje_slovo=ime.charAt(ime.lenght-1)
```

String objekt #2

- Za **izdvajanje drugog, trećeg i četvrtog znaka** stringa, može se koristiti sljedeći izraz:

```
podstring=ime.substring(1,4);
```

- Za **pronalazak pozicije** prvog slova 'r' u stringu ime, može se koristiti sljedeći izraz:

```
r=ime.indexOf('r');
```

- Postoji još niz drugih metoda kojima se može upravljati stringovima. Metode i opisi metoda upravljanja stringovima se nalaze u tablici

String objekt #3

Metode	Opis	NN	IE	ECMA
length	Vraća duljinu stringa	2.0	3.0	1.0
anchor()	Vraća string kao anchor: <a>string	2.0	3.0	
big()	Vraća string, formatiran sa big: <big>string</big>	2.0	3.0	
blink()	Vraća string formatiran s treptanjem: <blink>string</blink>	2.0		
bold()	Vraća podebljani string: string	2.0	3.0	
charAt()	Vraća znak na određenom mjestu	2.0	3.0	1.0
charCodeAt()	Vraća unicode od znaka na određenom mjestu	4.0	4.0	1.0
concat()	Vraća spojena dva stringa	4.0	4.0	
fixed()	Vraća string formatiran kao teletype text: <tt>string</tt>	2.0	3.0	
fontColor()	Vraća string u određenoj boji: string	2.0	3.0	
fontSize()	Vraća string u određenoj veličini: string	2.0	3.0	
fromCharCode()	Inverzna funkcija funkcije charCodeAt.	4.0	4.0	
indexOf()	Vraća mjesto prve pojave znaka ili –1 ako tog znaka nema.	2.0	3.0	
italics()	Vraća string u kurzivu, italic: string	2.0	3.0	
lastIndexOf()	Isto kao indexOf, ali s desna na lijevo.	2.0	3.0	
link()	Vraća string kao hyperlink: string	2.0	3.0	

String objekt #4

Metode	Opis	NN	IE	ECMA
match()	Ponaša se slično kao indexOf i lastIndexOf, ali vraća znakove koji se podudaraju ili "null", umjesto brojčane vrijednosti.	4.0	4.0	
replace()	Zamijeni znak s odabranim znakom.	4.0	4.0	
search()	Vraća cjelobrojnu vrijednost, tako da string sadrži određeni znak, inače vraća -1.	4.0	4.0	
slice()	Vraća string koji sadrži određeni znak definiran indexom.	4.0	4.0	
small()	Vraća string formatiran na small: string	2.0	3.0	
split()	Mijenja određeni znak zarezom.	4.0	4.0	1.0
strike()	Vraća string formatiran prekrižen: <strike>string</strike>	2.0	3.0	
sub()	Vraća string formatiran subscript: _{string}	2.0	3.0	
substr()	Vraća specificirane znakove. (14,7) vraća 7 znakova, počevši od 14-og znaka.	4.0	4.0	
substring()	Vraća znakove koji su specificirani. (14,7) vraća znakove između 7-og i 14-og.	2.0	3.0	1.0
sup()	Vraća string formatiran superscript: ^{string}	2.0	3.0	
toLowerCase()	Vraća string konvertiran u mala slova	2.0	3.0	1.0
toUpperCase()	Vraća string konvertiran u velika slova	2.0	3.0	1.0

Objekt polja #1

Polje je niz podataka, kao i objekt. Kao što podatak unutar objekta ima ime, svaki podatak u polju ima broj ili index. Kod JavaScripta polja su indeksirana upisivanjem indeksa unutar uglatih zagrada nakon imena polja. **Polje može sadržavati bilo koji tip JavaScript podataka kao i reference na druga polja, objekte ili funkcije**. JavaScript **ne podržava izravno višedimenzionalne nizove**. Polje se definira ključnom riječi new Array.

```
<html>
<body>
<script type="text/javascript">
var imena = new Array(4)
imena[0] = "Pero"
imena[1] = "Mate"
imena[2] = "Ante"
imena[3] = "Ivo"
for (i=0; i<4; i++)
 {
   document.write(imena[i] + "<br>")
 }
</script>
</body>
</html>
```

Višedimenzionalno polje u JavaScript-u

```
var myArray = new Array(3);
  for (var i = 0; i < 3; i++)
  {
      myArray[i] = new Array(3);
      for (var j = 0; j < 3; j++)
      {          myArray[i][j] = ''; }
   }
```

Jednostavnije

```
myArray[0][0] = '';
myArray[0][1] = '';
myArray[0][2] = '';
myArray[1][0] = '';
myArray[1][1] = '';
myArray[1][2] = '';
myArray[2][0] = '';
myArray[2][1] = '';
myArray[2][2] = '';
```

Objekt polja #2

Metode	Opis	NN	IE	ECMA
length	Vraća broj elemenata polja	3.0	4.0	1.0
concat()	Vraća polje nastalo lijepljenjem dvaju polja	4.0	4.0	1.0
join()	Vraća string slijepljenih vrijednosti svih elemenata polja	3.0	4.0	1.0
reverse()	Vraća polje obrnutim redoslijedom	3.0	4.0	1.0
slice()	Vraća specificirani dio polja	4.0	4.0	
sort()	Vraća sortirano polje	3.0	4.0	1.0

Objekt datuma #3

- JavaScript omogućuje tip ili klasu objekta koji predstavlja **datum i vrijeme**, i može se iskoristiti za manipuliranje tim tipom objekta. Date objekt kod JavaScripta se radi operatorom **new i Date() konstruktorom**

```
<html>
<body>
<script type="text/javascript">
var d = new Date()
document.write("Danas je: ")
document.write(d.getDate())
document.write(".")
document.write(d.getMonth() + 1)
document.write(".")
document.write(d.getFullYear())
var s = new Date()
document.write(", sada je ")
document.write(s.getHours())
document.write(":")
document.write(s.getMinutes())
document.write("  sati.")
</script>
</body>
</html>
```

Objekt datuma #4

Metode	Opis	NN	IE	ECMA
Date()	Vraća novi objekt datuma	2.0	3.0	1.0
getDate()	Vraća datum od Date objekta. (1-31)	2.0	3.0	1.0
getDay()	Vraća dan u tjednu. (0-6)	2.0	3.0	1.0
getMonth()	Vraća mjesec u godini. (0-11)	2.0	3.0	1.0
getFullYear()	Vraća godinu. (2000)	4.0	4.0	1.0
getYear()	Vraća godinu kao vrijednost od 4 znaka (ili godinu sa 2 znaka ako je datum manji od 1.1. 2000). Koristiti getFullYear za zamjenu !!	2.0	3.0	1.0
getHours()	Vraća sat između 0 i 23	2.0	3.0	1.0
getMinutes()	Vraća minute. (0-59)	2.0	3.0	1.0
getSeconds()	Vraća sekunde. (0-59)	2.0	3.0	1.0
getMilliseconds()	Vraća milisekunde. (0-999)	4.0	4.0	1.0
getTime()	Vraća milisekunde protekle od 1/1-1970	2.0	3.0	1.0
getTimezoneOffset()	Vraća vremensku razliku između vremena na računalu i GMT	2.0	3.0	1.0
getUTCDate()	Vraća datum podešen prema World Time Standard, UTC = Universal Coordinated Time. Za povrat na lokalno vrijeme koristiti getDate metodu	4.0	4.0	1.0
getUTCDay()	Vraća UTC dan	4.0	4.0	1.0
getUTCMonth()	Vraća UTC mjesec	4.0	4.0	1.0
getUTCFullYear()	Vraća UTC 4 znamenke za godinu	4.0	4.0	1.0

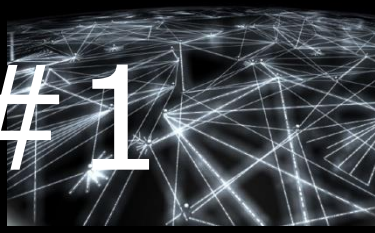
Objekt datuma #5

Metode	Opis	NN	IE	ECMA
getUTCHourc()	Vraća UTC sat	4.0	4.0	1.0
getUTCMinutes()	Vraća UTC minute	4.0	4.0	1.0
getUTCSeconds()	Vraća UTC sekunde	4.0	4.0	1.0
getUTCMilliseconds()	Vraća UTC millisekunde	4.0	4.0	1.0
parse()	Vraća string vrijednost datuma koliko je millisekundi proteklo od 1/1-1970.	2.0	3.0	1.0
setDate()	Postavlja novi datum u Date objekt	2.0	3.0	1.0
setFullYear()	Postavlja novu godinu u Date objekt	4.0	4.0	1.0
setHours()	Postavlja novi sat u Date objekt. (0-23)	2.0	3.0	1.0
setMilliseconds()	Postavlja nove millisekunde u Date objekt. (0-999)	4.0	4.0	1.0
setMinutes()	Postavlja nove minute u Date objekt. (0-59)	2.0	3.0	1.0
setMonth()	Postavlja novi mjesec u Date objekt. (0-11)	2.0	3.0	1.0
setSeconds()	Postavlja nove sekunde u Date objekt. (0-59)	2.0	3.0	1.0
setTime()	Postavlja milisekunde nakon 1/1-1970	2.0	3.0	1.0
setYear()	Postavlja novu godinu u Date objekt	2.0	3.0	1.0

Objekt datuma #6

Metode	Opis	NN	IE	ECMA
setUTCDate()	Postavlja UTC Date objekt	4.0	4.0	1.0
setUTCDay()	Postavlja UTC Date objekt	4.0	4.0	1.0
setUTCMonth()	Postavlja UTC Date objekt	4.0	4.0	1.0
setUTCFullYear()	Postavlja UTC Date objekt	4.0	4.0	1.0
setUTCHourc()	Postavlja UTC Date objekt	4.0	4.0	1.0
setUTCMinutes()	Postavlja UTC Date objekt	4.0	4.0	1.0
setUTCSeconds()	Postavlja UTC Date objekt	4.0	4.0	1.0
setUTCMilliseconds()	Postavlja UTC Date objekt	4.0	4.0	1.0
toGMTString()	Vraća string vrijednost datuma.	2.0	3.0	1.0
toLocaleString()	Vraća string vrijednost datuma.	2.0	3.0	1.0
toString()	Vraća string vrijednost datuma.	2.0	4.0	1.0

Matematički objekt #1



- Brojevi su osnovni tip podataka koje nije potrebno dodatno objašnjavati.
- **Kod JavaScripta**, za razliku od drugih programskih jezika poput C-a, **svi brojevi su decimalne vrijednosti** (*floating-point*).
- JavaScript programi rade s brojevima koristeći osnovne matematičke operatore kao što su:
  - + za zbrajanje,
  - - za oduzimanje,
  - * za množenje, i
  - / za dijeljenje.
- Kao dodatak ovim osnovnim aritmetičkim operacijama, JavaScript podržava kompleksne matematičke operacije pomoću velikog broja matematičkih funkcija koje su dio JavaScript jezika.
- **Sve matematičke funkcije su spremljene kao svojstva (properties) Math objekta.**
- Na primjer, za izračun sinusa numeričke vrijednosti x može se koristiti sljedeći izraz:

```
sinus_x=Math.sin(x);
```

Matematički objekt #2

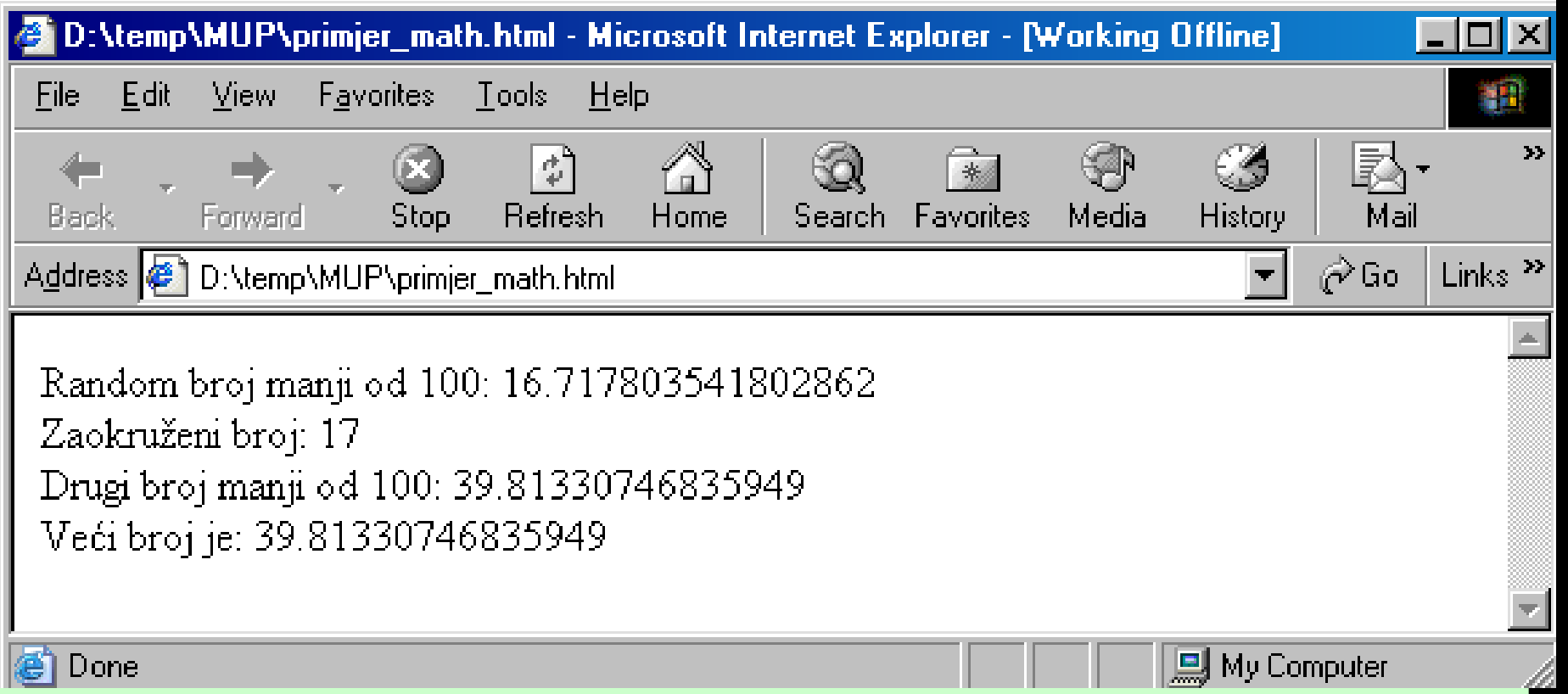
- Za izračun korijena brojčane vrijednosti može se koristiti sljedeći izraz

```
korijen=Math.sqrt(x*x+y*y)
```

- Postoji **nekoliko specijalnih matematičkih vrijednosti** korištenih kod JavaScripta.
- Kada neka numerička vrijednost postane veća od najvećeg mogućeg broja, rezultat je beskonačna vrijednost koju JavaScript prikazuje kao **Infinity**.
- Isto tako kada vrijednost nekog broja postane manja od najmanje moguće, rezultat je **-Infinity.**
- Drugi specijani slučaj je kada je vraćena vrijednost matematičke operacije (kao kod dijeljenja s nulom) **nedefinirani rezultat ili greška**. U ovom slučaju rezultat JavaScripta je vrijednost Not-a-Number koja se prikazuje kao **Nan.**
- **Not-a-Number** vrijednost se ne može usporediti s drugim brojevima pa čak ni sa samim sobom!
- Iz tog razloga specijana funkcija **isNan()**potrebna je za **testiranje** ovih vrijednosti

Matematički objekt #3

Primjeri za matematički objekt

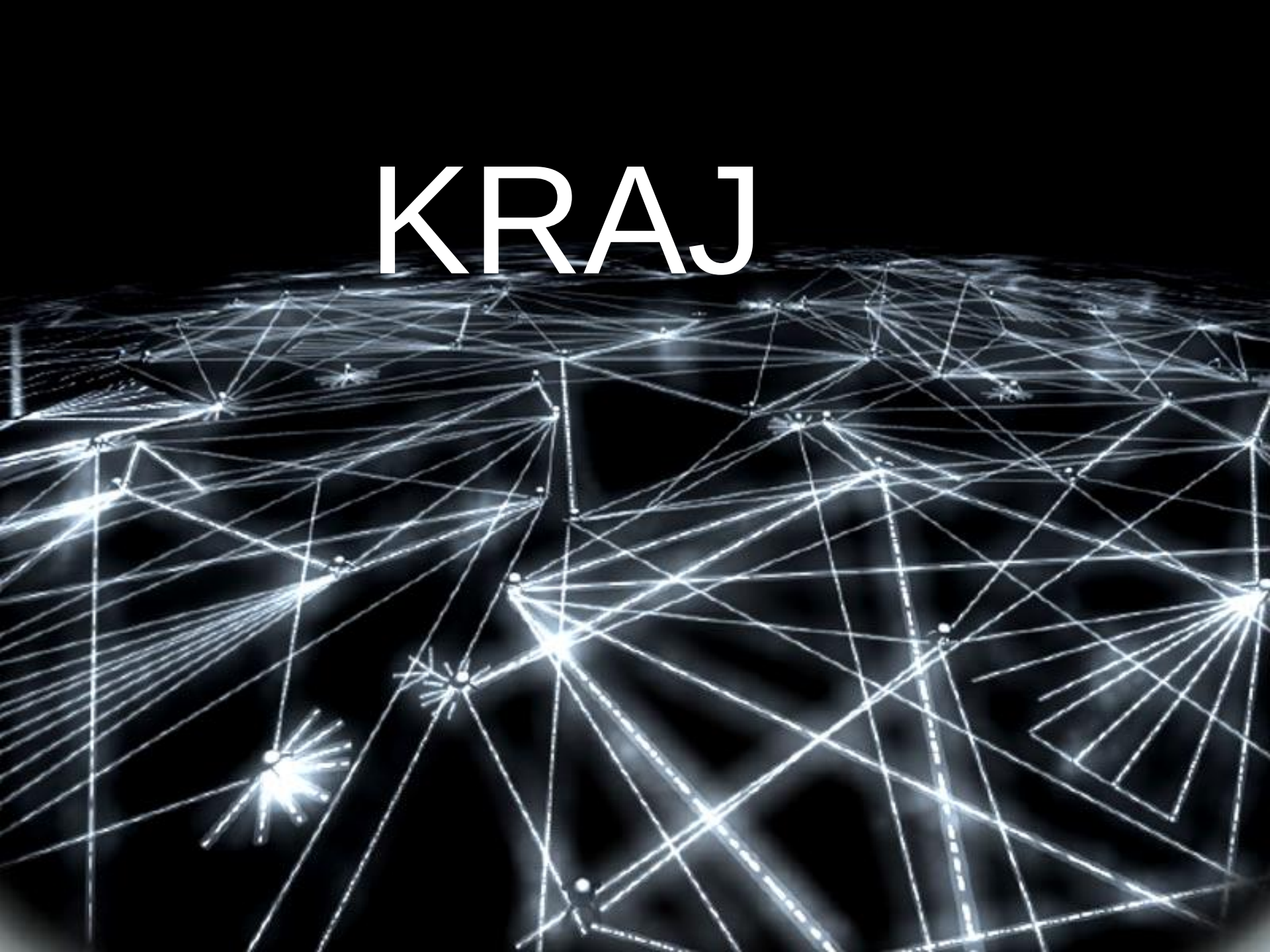


Matematički objekt #4

Svojstvo	Opis	NN	IE	ECMA
E	Vraća bazu prirodnog logaritma	2.0	3.0	1.0
LN2	Vraća prirodni logaritam od 2	2.0	3.0	1.0
LN10	Vraća prirodni logaritam od 10	2.0	3.0	1.0
LOG2E	Vraća logaritam po bazi 2 od E	2.0	3.0	1.0
LOG10E	Vraća logaritam po bazi 10 od E	2.0	3.0	1.0
PI	Vraća PI	2.0	3.0	1.0
SQRT1_2	Vraća kvadratni korijen od 0.5	2.0	3.0	1.0
SQRT2	Vraća kvadratni korijen od 2	2.0	3.0	1.0

Matematički objekt #5

Metode	Opis	NN	IE	ECMA
abs()	Vraća apsolutnu vrijednost broja	2.0	3.0	1.0
acos()	Vraća arccos broja	2.0	3.0	1.0
asin()	Vraća arcsin broja	2.0	3.0	1.0
atan()	Vraća arctan broja	2.0	3.0	1.0
atan2()	Vraća kut od x osi do točke	2.0	3.0	1.0
ceil()	Vraća najbliži cijeli broj veći ili jednak broju	2.0	3.0	1.0
cos()	Vraća kosinus broja	2.0	3.0	1.0
exp()	Vraća bazu logaritma na neki eksponent	2.0	3.0	1.0
floor()	Vraća najbliži cijeli broj manji ili jednak broju.	2.0	3.0	1.0
log()	Vraća logaritam broja	2.0	3.0	1.0
max()	Vraća veći od dva broja	2.0	3.0	1.0
min()	Vraća manji od dva broja	2.0	3.0	1.0
pow()	Vrijednost na neku potenciju	2.0	3.0	1.0
random()	Vraća slučajni broj	2.0	3.0	1.0
round()	Zaokruži broj na najbliži cijeli broj	2.0	3.0	1.0
sin()	Sinus broja	2.0	3.0	1.0
sqrt()	Kvadratni korijen broja	2.0	3.0	1.0
tan()	Tangens broja	2.0	3.0	1.0


KRAJ

Ponovimo i nastavimo dalje

JavaScript objekti

Klasa	Vrijednost	Metode
Boolean	true/false	toString(), valueOf()
String	string	length, charAt(), indexOf(), substring()
Array	polje	length, join(), sort()
Date	datum	Date(), getDate(), getMonth(), getFullYear()
Math		abs(), cos(), sin(), random(), round(), sqrt(), E, PI

DOM (Document Object Model)

- Objekt je skup varijabli (parametara) i funkcija (metoda).
- Preglednik pruža mogućnost pristupa nizu predefiniranih objekata.
- Prozor preglednika u kojem se prikazuje stranica naziva se window object (objekt prozora).
- HTML stranica koju prikazuje preglednik naziva se document object (objekt dokumenta).
- Objekt dokumenta vjerojatno je najčešće korišteni objekt JavaScripta na klijent strani.
- HTML elementi koji se dodaju stranici proširuju hijerarhiju objekata. Primjer je form element i elementi koji se nalaze unutar form elementa.
- Ovo znači kako je moguće referencirati svaki od dodanih objekata.
- DOM je skup objekata koje JavaScript jeziku dodaje preglednik.

DOM (Document Object Model)

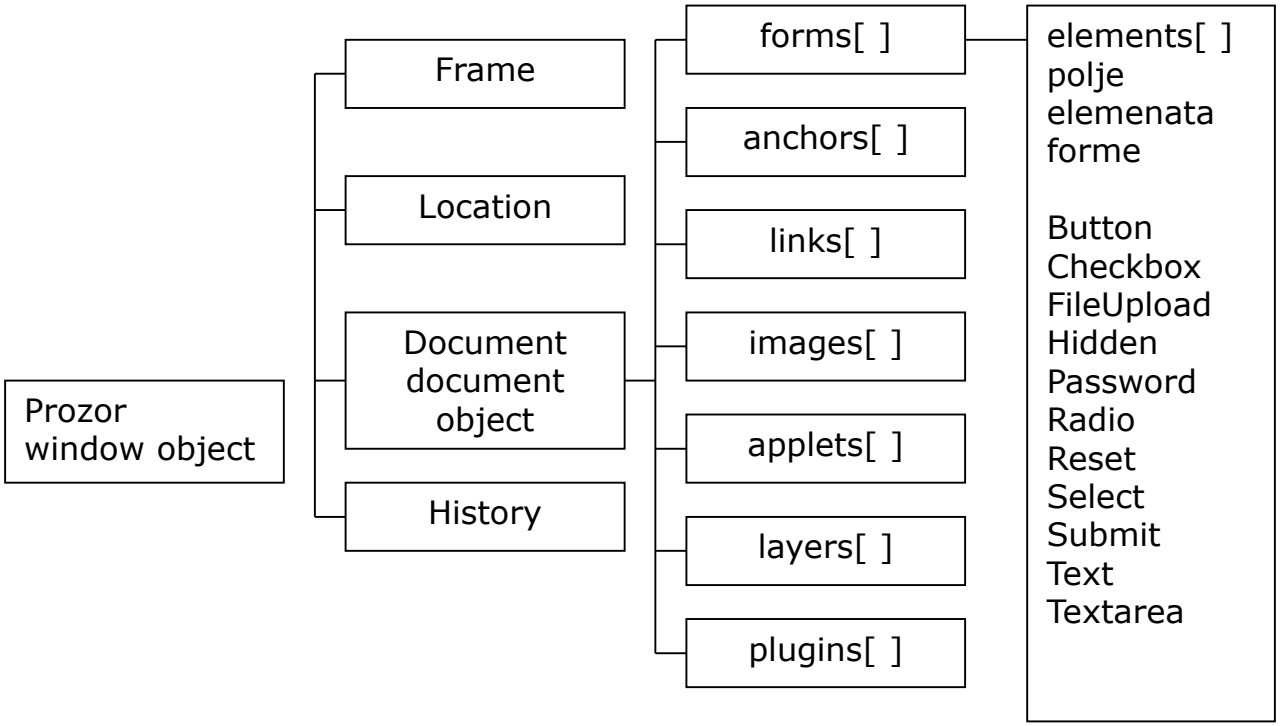
- **window.document.forms[0]** – se odnosi na prvu formu unutar html dokumenta. Forme su u DOM-u implementirane kao polja. Ako postoji više od jedne forme u dokumentu brojevi forme počinju od 0 i rastu za 1.
- **window.document.Form1** – se odnosi na formu unutar html dokumenta naziva Form1.
- **window.document.Form1.Ime.value** – se odnosi na vrijednost upisanu u textbox naziva Ime u formu unutar html dokumenta naziva Form1 od strane klijenta.

DOM (Document Object Model)

```
<html>
<head>
      <title>DOM jednostavna forma</title>
</head>
<body>
<form name="Form1">
Ime: <input type="text" name="Ime">
<br />
<input type="button"
value="Pošalji informacije">
</form>
<form name="Form2">
Prezime: <input type="text" name="Prezime">
<br />
<input type="button"
value="Pošalji informacije">
</form></body></html>
```

DOM (Document Object Model)

- Iako je moguće referencirati objekt na formi bilo preko imena forme bilo preko njenog položaja u polju formi preporučuje se koristiti referenciranje preko imena zbog jednostavnijeg praćenja naziva objekata.
- Dijagram koji ilustrira Document Object Model (DOM)



Događaji (events)

- Događaji su okidači koji pozivaju (pokreću) jednu od definiranih funkcija. Klijent strana aplikacije (JavaScript program) neće se izvršiti (interpretirati) dok je ne pokrene neki događaj.
- Događaj može biti akcija poput klika na neki objekt ili prijelaz miša preko nekog objekta. Provjera unosa na formi u pravilu se pokreće kada korisnik klikne na «Pošalji», odnosno kada se na objektu okine onClick događaj.
- U JavaScriptu možemo registrirati okidanje ovih događaja:

onClick	onSubmit	onMouseOver
onMouseOut	onFocus	onChange
onBlur	onLoad	onUnLoad

onClick

- Događaj koji okida kada se klikne na elemente:
  - button
  - checkbox
  - radio button
  - reset button
  - submit button
- Primjer uporabe ovog događaja

```
<input type="button"
value="Klikni me"
onClick="window.alert('Kliknuli ste');">
```

onSubmit

- Ovaj događaj okida kada korisnik šalje rezultate popunjene na formi.
- Uobičajeno je ovaj događaj vezati na provjeru podataka forme.
- Primjer uporabe ovog događaja

```
<form
action="http://www.pmfst.hr/formtest.asp"
onSubmit="return provjera();">
```

- U ovom primjeru kada korisnik klikne na submit izvršit će se funkcija provjera(). Ako svi podaci na formi uspješno prođu test funkcija će vratiti true, a podaci će se tada proslijediti serveru. U protivnom korisnik će morati ispraviti ili dopuniti podatke na formi.

onMouseOver, onMouseOut

- Ovaj događaj okida kada se korisnik pozicionira mišem na hyperlink.

```
<a href="http://www.pmfst.hr/"
onMouseOver="window.status='Prirodoslovno-matematički fakultet'; return true;">
```

- Ovaj događaj okida kada se korisnik pozicionira mišem van hyperlinka.

```
<a href="http://www.pmfst.hr /"
onMouseOver="window.status=''; return true;">
```

onFocus, onChange, onBlur

- onFocus – Objekt forme postaje aktivan. Ovaj događaj okida kada se korisnik tabulatorom ili mišem postavi na neki element u HTML formi.

```
<input type="text" name="Mjesec"
onFocus="window.status=('Upišite mjesec od
01 do 12');return true;">
```

- onChange – Ovaj događaj okida kada korisnik napušta objekt, a vrijednost objekta se promijenila. Primjer:

```
<input type="text" name="Mjesec"
onChange="window.status=('Vrijednost se
promijenila!!!'); return true;">
```

- onBlur – Ovaj događaj okida kada korisnik napusti objekt u HTML formi, a bez obzira na to je li se vrijednost objekta promijenila ili ne.

onLoad, onUnLoad

- onLoad – Ovaj događaj okida nakon što preglednik učita dokument.
- onUnLoad – Ovaj događaj okida kada se počne korisnik pomiče na novi dokument.

JavaScript Events and HTML Form Elements

Events / HTML Elements	Blur	Click	Change	Focus	Load	Mouseover	Select	Submit	Unload
Button		X							
Checkbox		X			X				X
Document								X	
Form						X			
Link		X							
Radio		X							
Reset		X							
Selection	X		X	X					
Submit		X							
Text	X		X	X			X		
Textarea	X		X	X			X		