Udžbenik

Osnove JavaScript-a

Udžbenik napisali : Suzana i Nenad Bosanac

Novi Sad, April 2017.

Osnove programiranja – udžbenik

Autor : Suzana i Nenad Bosanac

Novi Sad , april 2017

Sadržaj:

Uvod

Tema časa : Čemu služi i šta je JavaScript ?

Istorijat JavaScript-a

Šta možemo da uradimo sa njim?

Čemu služi i šta je Javascipt?

HTML(Hypert Text Markup Language) je relativno statičan jezik, bez nekih posebno atraktivnih elemenata koji bi dali zanimljivost samoj veb stranici i, s druge strane, povećali interaktivnost bez dodatnog opterećivanja komunikacionih kanala izmedju klijenta i servera.

CSS(Cascade Style Sheet) sa druge strane je jezik koji opisuje stilove HTML dokumenta.

JavaScript (JS u daljem tekstu) je programski jezik zamišljen da koristi tzv. vikend programerima ili skript programerima koji ne moraju imati nekog velikog programerskog predznanja (praktično, dovoljno je da znaju samo HTML kao polaznu osnovu). On je upravo jezik koji daje dinamičnost celoj veb stranici i čini upravo veb stranice atraktivne za korišćenje.

Istorijat i verzije JS – a

U novembru 1995. godine pod okriljem kompanije Netscape počelo je razvijanje potpuno novog programskog, skript, jezika pod nazivom LiveScript. Prvobitno rađen u kombinaciji sa programom Netscape Web server. U početku zamišljen da radi u dve namene: kao skript jezik koji bi omogućio administratorima servera da upravljaju svojim serverima i omogući povezivanje za bazama podataka; i kao skript jezik, u okviru HTML dokumenta, koji bi se izvršavao na strani klijenta.

Popularnost ovog programskog jezika je toliko narasla da su početkom 1996. Netscape i Sun (kompanija koja je osmislila programski jezik Java) objavili da će se novi skript jezik od sada zvati JavaScript.

Neophodno je na ovom mestu naglasiti da izmedu Jave i JS - a, pored osnovnih elemenata sintakse nema ničeg zajedničkog. Sličnost naziva je čist marketinški trik.

Zamišljen da se izvršava na strani klijenta uključen u HTML strane bez kompajliranja JavaScript je predstavljao sasvim novi koncept u programerskom svetu.
Netscape je počeo sa korišćenjem JavaScripta - a (verzije 1.0) u svom pretraživaču već od verzije 2.0. Sa razvijanjem JavaScript - a, u novije verzije Netscapeovog pretraživača redom bile su uključene nove verzije JavaScript - a (u Netscape Navigatoru verziji 3 - JS verzija1.2 i u Netscape Navigatoru 4 - 1.3).

Microsoft je kasno prihvatio novi skript jezik u svom MSIE 3.0 i to kao JScript (svoju verziju istog programskog jezika) koja je bila puna grešaka i nije bila pouzdana. Međutim, kasnije je JScript znatno poboljšan za MSIE 4, a u MSIE 5 počeo da ima opcije koje NN ne podržava.

JavaScript je mlad programski jezik i u svakodnevnom je razvoju. Nove mogućnosti pojavljivaće se kako se budu pojavljivale nove verzije pretraživača. Problemi kompatibilnosti među pretraživačima su jasni kada se sagleda razvoj jezika u dve suprotstavljene velike softverske kuće.

Ti problemi se u najgorem slučaju manifestuju neizvršavanjem skripta.
Danas je situacija mnogo jasnija jer je JavaScript standardizovana sa ECMAScript jezičkom specifikacijom tako da većina novih verzija browsera se pridržava navedene specifikacije ,a to u mnogome omogućava lakše korišćenje JavaScript-a u različitim browserima .

Šta JavaScript može

Mogućnosti JavaScript-a su brojne. Nabrojaćemo samo neke od njih:

- ispisivanje HTML koda
- upravljanje sadržajem formulara
- upravljanje sadržajem frejmova
- prepoznavanje verzija i tipa pretraživača (internet browser)
- zamena vrednosti SRC atributa u IMG tagu
- otvaranje i zatvaranje prozora pretraživača sa željenim dimenzijama
- očitavanje vremena i datuma
- postavljanje i očitavanje tzv. cookie – a
- matematička izračunavanja
- promena sadržaja status bar – a
- aktiviranje okvira za dijalog (alert, prompt, confi rm)
- promena URL stranice
- interakcija sa apletima
- zamena stilova zadatih CSS – om
- očitavanje događaja (klik tasterima miša, miš iznad slike, završetak učitavanja strane, pritisak na tastere na tastaturi...)

Takođe JavaScript od 2009 godine je prvi put iskorišćena da pored rada na klijent strani (tj. u okviru pretraživača) može da radi i na serverskoj strani (popularno nazvan NODE.js). Time se danas JavaScript može nazvati full stack programskim jezikom jer obuhvata mogućnost rada i na klijent i na server strani .

Čas 1

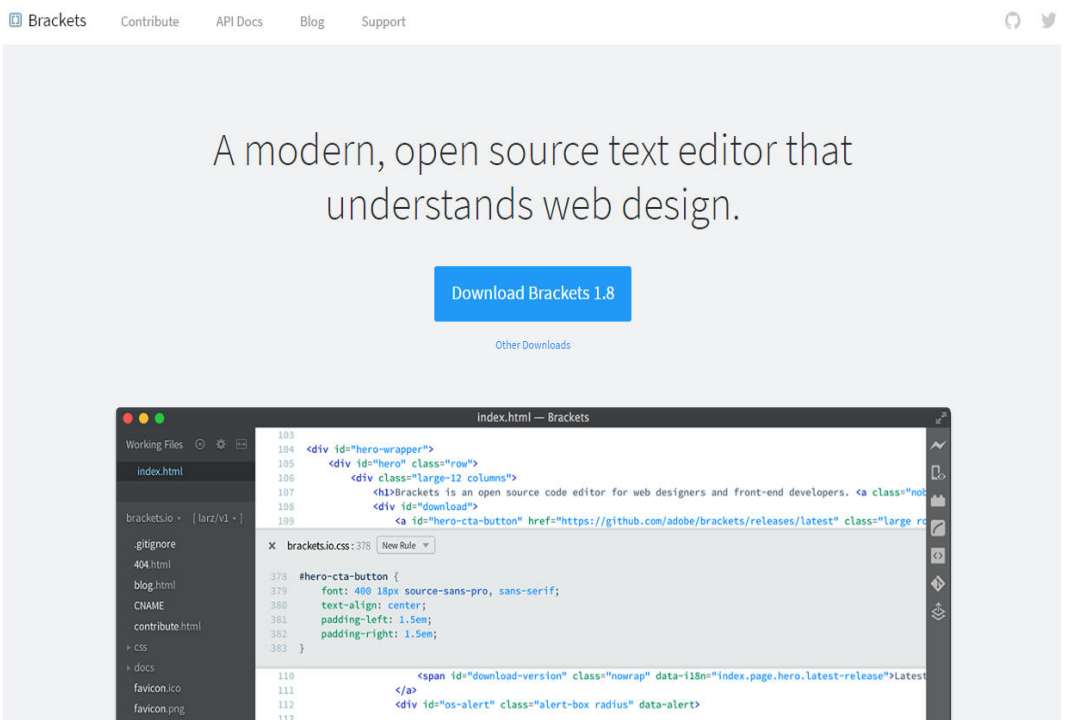
Tema časa : Instalacija okruženja za rad

Dodavanje JavaScripta u HTML stranicu

Instalacija okruženja za rad

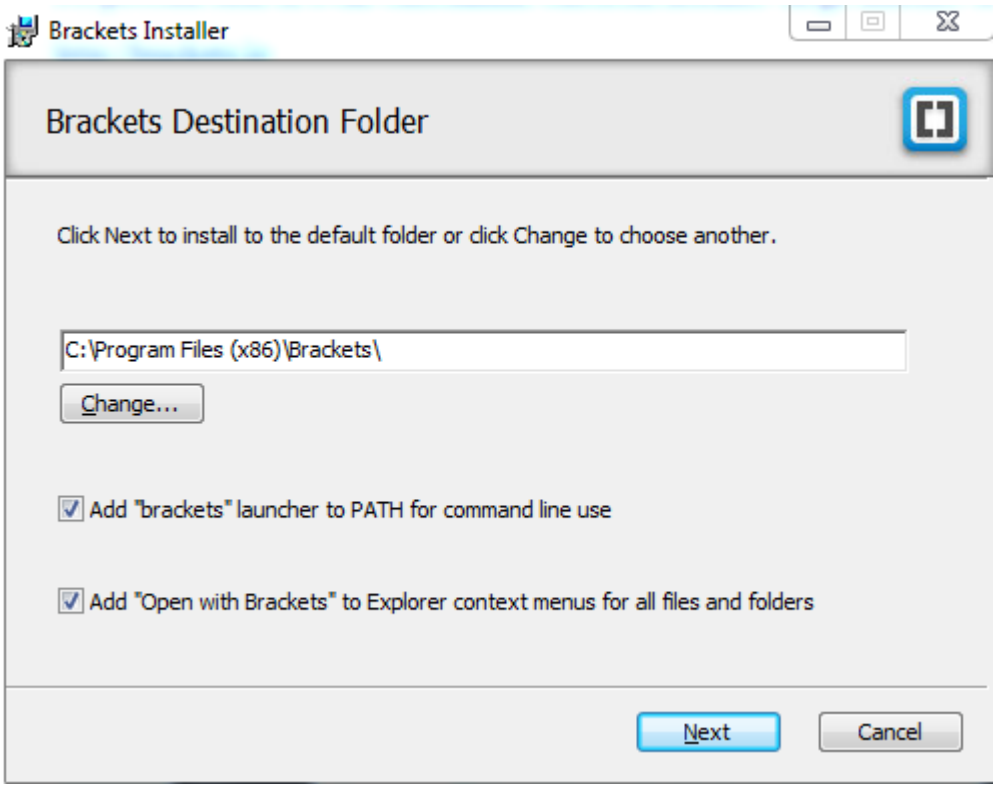
Kako bi pripremili okruženje za rad sa JavaScriptom moramo pre svega da instaliramo potreban softver. Mi ćemo koristiti moderan, open source tekstualan editor po nazivu Brackets.

Pre početka rada sa ovim tekstualnim editorom moramo da ga skinemo sa portala
http://brackets.io



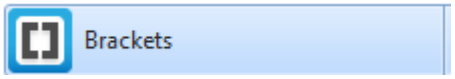
Slika 1. Program za rad – Brackets

Najnovija verzija Brackets je 1.9. Kada se downloaduje Brackets potrebno je pokrenuti Brackets.Release.1.9.msi i pojaviće se prozor kao na slici2



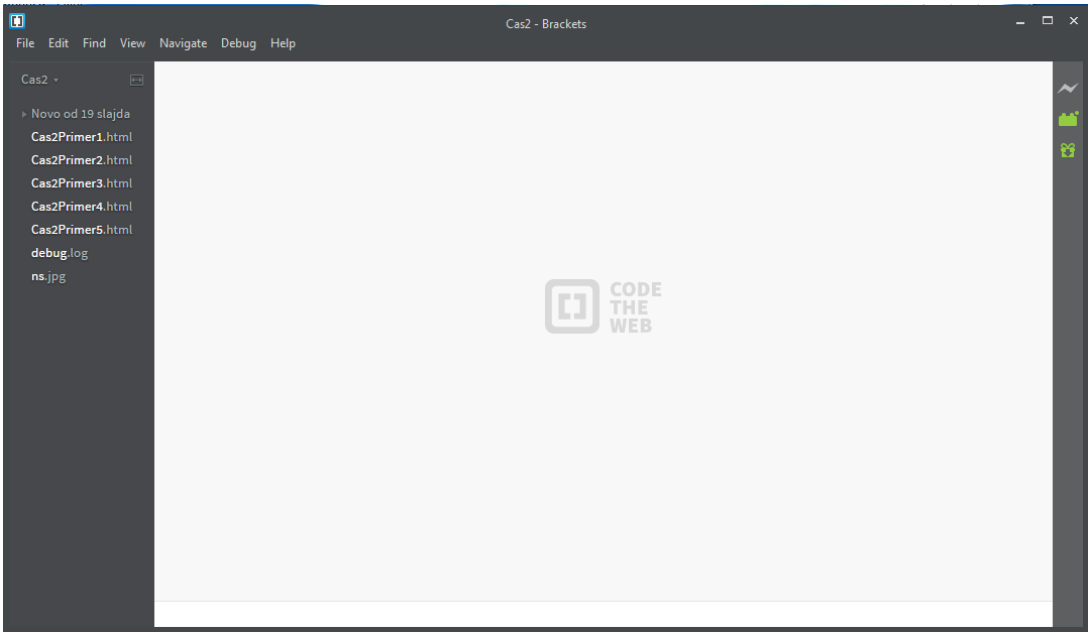
Slika 2. Instalacija Brackets-a

Nakon toga potrebno je samo da se pritiska dugme Next i u poslednjem prozoru na dugme Finish i instalacija će se završiti.
U Windows search prozoru ukucajmo Brackets i pojaviće se ikona kao na slici 3



Slika 3.Brackets ikona

Na desni klik na ikonu pin-ujmo ju na taskbar i pokrenimo ovu aplikaciju.
Otvoriće se Brackets kao na slici 4.



Slika 4.IDE Brackets

Dodavanje JavaScripta u HTML stranicu

Na početku mora se naglasiti da je JavaScript, kako mu ime kaže, skript programski jezik, i da on uglavnom služi, kao i ostali skript jezici, za izvršavanje krajnje konkretizovanih zadataka. Na internet prezentacijama on se, kao i drugi skript jezici, primenjuje u okviru samog HTML koda.

U opštem slučaju ubacivanje JavaScript vrši se uz pomoć taga **script** kao na primeru:

```
<script>
 NEKI_JAVASCRIPT_KOD_SE_UBACUJE_OVDE
</script>
```

Language *atribut se primenjivao ali se više ne koristi, ne bi li se naglasilo pretraživaču koji je skript jezik u pitanju. Sličan se postupak koristio ukoliko smo želeli da koristimo bilo koji drugi skript jezik - u* ***language*** *atributu se navodilo ime tog jezika. Kada se radilo na specifikacije* ***HTML5*** *otkriveno je da svi pretraživači imaju* ***"text/javascript"****kao defaultni atribut* ***type*** *pa je u HTML 5 ova vrednost atributa type postavljena kao default vrednost.Zato se atribut* ***type*** *ne navodi u okviru script taga.Ako koristite* ***HTML 4.01*** *atribut* ***type*** *je neophodan kako bi HTML dokument bio validan*

U opštem slučaju, **<script>** tag se ubacuje u zaglavlje HTML dokumenta (**HEAD**), mada je moguće ubaciti ga i u telo (**BODY**) stranice, ali o tome nešto više kasnije. Tokom rada sa CSS - om uočili smo da se on mora "sakriti" od ostalog koda da se pretraživač koji nije osposobljen za tumačenje CSS-a ne bi zbunjivao i ispisivao zbunjujuće delove samog CSS koda. Slična je situacija je i kod JS - a: osim unutar

SCRIPT taga kod mora biti smešten i u klasične oznake za komentar u HTML kodu **(< ! - -** i **- - >**).
Pogledajmo kroz primer ovu situaciju

Primer 1 :

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Čas 1 Primer 1</title>
<script>
<!--
   alert("Zdravo! Ovaj browser podrzava JS");
  -->
</script>
</head>
<body>
<h1>Čas 1 Primer 1 - Pozivanje JS na stranici</h1>
</body>
</html>
```

Budući da se neki stariji pretraživači zbunjuju i kod ispisa zatvarajućeg komentara postavljenog oko koda, ubacuju se dve kose crte - "slahs" - a (//), koje služe za oznaku komentara u JS - u, ispred, da bi kod JS - a u potpunosti i bez smetnji bio ignorisan kao na primeru:

```
<script
< ! - -
 NEKI_JAVASCRIPT_KOD_SE_UBACUJE_OVDE
 // - ->
</script>
```

Ovako napisan JavaScriptkod stariji pretraživač neće tumačiti, a oni koji imaju aktiviranu podršku za JavaScript će ga tumačiti, iako je ispisan kao komentar.

Za kraći skript, posebno kod provere događaja na objektima i slično, povoljan je još jedan način ubacivanja JavaScript koda, a to je u **href** atribut **<a>** taga u sintaksi:

Primer 2 :

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Čas 1 Primer 2</title>
</head>
<body>
<h1>Čas 1 Primer 2 - Pozivanje JS kroz atribut taga</h1>
<br>
<a href="javascript:alert('Zdravo!')">Pritisni me</a>
</body>
</html>
```

Slično CSS - u i kod JavaScriptu - a postoji mogućnost uvoženja eksternih kodova, snimljenih u posebnom fajlu. Postupak je sledeći: napisani JS kod snimi se kao odvojeni plain tekst dokument sa ekstenzijom .js, nakon toga na stranici na kojoj želimo da primenimo ovaj kod unesemo ga preko **src** atributa u **script** tagu u sledećoj sintaksi:

```
<script src="putanja_do_fajla.js">
</script>
```

Pogledajmo primer

Primer 3

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Čas 1 Primer 3</title>
<script src="Cas1Primer3.js"></script>
</head>

<body>
<h1>Čas 1 Primer 3 - Pozivanje eksternog JS fajla</h1>
</body>
</html>
```

Cas1Primer3.js

```
alert("Zdravo");
```

Uvoženje eksternih fajlova je moćna stvar posebno ukoliko ih koristimo na više strana naše internet prezentacije, jer smanjujemo vreme učitavanja koda koji je smešten u keš memoriju klijentovog kompjutera pri svom prvom pozivu. Pored toga, ponašanje skripta na više strana menjaće se izmenom koda samo jednog jedinog dokumenta.

Dosta starije verzije Internet Explorera npr verzija 3 nije podržavala ovu mogućnost uvođenja eksternih fajlova preko SRC atributa SCRIPT taga.

Posebno treba obratiti pažnju na one pretraživače koji ne mogu da čitaju oznake **<script>** bilo iz razloga da nemaju ugrađen interpretator JavaScripta ili koji ga imaju, ali je isključen, bez obzira koliko je procenat takvih korisnika mali.

Oznake **<noscript>** koriste se za označavanje dela koda koji će se uključiti kod takvih pretaživača. Obično se unutar para oznaka **<noscript>…</noscript**> stavlja upozorenje da posetioci koji nemaju skriptabile pretraživače neće moći u potpunosti da uživaju u vašoj stranici.

Pogledajmo primer

Primer 4

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<script src="Cas1Primer4.js"></script>
<noscript>
     Vaš browser ne podržava JavaScript!
</noscript>
<title>Čas 1 Primer 4</title>
</head>

<body>
<h1>Čas 1 Primer 4 - NOSCRIPT tag</h1>
</body>
</html>
```

Važno za napomenuti je da se JavaScript može uneti osim u <head> tag, isto tako i u <body> tag. Ovo će posebno biti važno kada budemo radili sa DOM elementima.

Vežba

Potrebno je kod kuće samostalno instalirati Brackets IDE prateći korake definisane u Času 1.

Čas 2

Tema časa : Vaša prva JavaScript aplikacija

Promenljive

Operatori

Izrazi

Komentari

Očitavanje grešaka u JavaScript-u

Vaša prva JavaScript aplikacija

Sada ćemo da napravimo našu prvu JavaScript aplikaciju. U okiru HTML smo radili element forme dugme(button) . U specifikaciji piše da button podržava sledeće JavaScript metode : onclick(klikom), ondblclick(dvostruki klik), onmousedown(miš dole), onmouseup(miš gore), onmouseover(mišem preko lementa), onmouseout (ako miš napusti element) ,onkeypress (na pritisak dugmeta) , onkeydown(dugme pritisnuti), onkeyup.(dugme gore).

Za našu prvu JavaScript aplikaciju koristićemo ugrađenu funkciju **alert()** objekta window. Više o ugrađenim funkcijama i objektu window radićemo u Času 4.Ova funkcija generiše okvir za dijalog koji prikazuje svaki tekst koji se zada kao parametar. Na prozoru se prikaže i dugme OK koje omogućava korisniku da isključi ovaj okvir.

Napomena: Izgled samog prozora nemoguće je menjati! On je deo samih pretraživača koji se samo poziva ovom funkcijom!

Pogledajmo kako izgleda naša prva JavaScript aplikacija

Primer 1

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Čas 2 Primer 1</title>
</head>
<body>
<h1>Čas 2 Primer 1 - Alert metoda</h1><br>
<input type="button" name="dugme" value="Klikni me"
```

```
  onclick="javascript:alert('Druga  funkcija')" >
</body>
</html>
```

U gornjem primeru je pokazan prvi način pozivanja funkcije **alert(),** pored načina na koji želimo da je pozovemo stavimo navodne znake zatim javascript i dvotačka I onda pozivamo metodu. Jako važno je zapamtiti da se u JavaScriptu-u ne smeju ugnježdavati navodni znaci.
Sada je vreme da započnemo da se upoznajemo sa osnovama JavaScript jezika.

Variables (promenljive)

Promenljive u JavaScript - u su, slično kao i drugim programskim jezicima, parametri koji se definišu u programu i koji svoju vrednost mogu menjati u samom programu, a programeru služe za lakše obavljanje postavljenog zadatka. Promenljivama možemo zadavati vrednost ili je izračunavati raznim postupcima.

Na primer:

```
var a = 1;
```

promenljivoj pod zadatim imenom a, zadata je numerička vrednost 1.

```
var b = "smart";
```

promenljivoj pod zadatim imenom b, zadata je slovna vrednost (string ili niz) - smart.

Nazivi promenljvih u kodu treba da budu asocijativni.

NAPOMENA O KLJUČNOJ REČI "var":

Reč "var" se može i ne mora koristiti prilikom definisanja varijabli. Napomena se odnosi na to da se reč "var" koristi SAMO JEDNOM u toku koda, bez obzira da li želite da je samo inicijalizujete ili da joj u isto vreme i dodelite vrednost. Za kasnije korišćenje promenljive ključnu reč "var" ne morate koristiti.

Pogledajmo promenljive kroz primer
Primer 2

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Čas 2 Primer 2</title>
<script>
 var naziv = "Jelena "; //String
 var zanimanje = 'web developer';
 var odgovor = "On se zove 'Pera'";
 var odgovor1 = 'On se zove "Pera";
 var broj = 1; //Number
 var gradovi = ["Novi Sad", "Beograd", "Nis"];
      //Array - niz
```

```
 var tacno = true;//boolean

 var y = 123e5;//12300000
 var z = 123e-5;//0.00123

  function dijalog(){
   alert(naziv);
  }
</script>
</head>

<body>
      <h1>Čas 2 Primer 1 - Variables - promenljive</h1>
      <br>
      <input type="button" name="dugme" value="Klikni me"
           onClick="dijalog()">
</body>
</html>
```

Napomena: Pod navodnim znacima se pišu konstante I HTML znakovi, ali se ne pišu nazivi promenljivih.

Operatori

Operatori su znaci koji služe za izvršavanje određenih računskih operacija (aritmetički operatori) , operacije poređenja promenljivih (operatori poređenja) ili za logičku proveru promenljivih (logički operatori).

ARITMETIČKI OPERATORI	
Simbol	**Operacija**
+	Sabiranje
-	Oduzimanje
*	Množenje
/	Deljenje
%	Moduo
++	Inkrementiranje, uvećaj za jedan
--	Dekrementiranje, umanji za jedan
OPERATORI POREĐENJA – RELACIONI OPERATORI	
==, != ,===	Jednakost, nejednakost , jednakost po tipu i vrednosti
<,<=,>=,>	Manje od,manje ili jednako,veće ili jednako, veće
LOGIČKI OPERATORI	
!	Negacija
&&,||	Logičko I, Logičko ILI
?	provera

Tabela 1. Operatori

Na primer:

```
broj = 14 + 4 - 3;
```

promenljivoj broj zadata je vrednost dobijena sabiranja broja 14 i 4 i oduzimanjem broja 3. **Operator je znak + i -**.
Sledeći primer :

```
škola_za_obuke = "Smart" + " school";
```

promenljivoj škola_za_obuke zadata je vrednost string koji se dobija spajanjem dva stringa. **Operator je znak +.**

Izrazi (Expressions)

U primeru :

```
broj = 14 + 4 - 3;
```

operator je znak + i -, a **ceo red se naziva izraz**.
Danas ćemo nabrojimo samo aritmetiče operatore plus(+), minus (-), puta(*), podeljenjo (/), ++ uvećaj za jedan, -- umanji za jedan.

U sledećem primeru je prikazana upotreba promenljivih, operatora i izraza.

Primer 3

```
<!DOCTYPE html>
<html lang="en">
<head>
      <meta charset="UTF-8">
      <title>Čas 2 Primer 3</title>
      <script>

       var naziv = "Jelena "; //String
       var  zanimanje = "web developer";
       var broj = 1; //Number
       var gradovi = ["Novi Sad", "Beograd",
                    "Nis"];//Array - Niz

       var x = 4;
       var y = 10;

      var rezultat = x+y;
      //vezba isprobati ostale operacija

       alert(rezultat);
       //alert(naziv + zanimanje + broj);
       //probati sa x+y+naziv - 14Jelena
       //probati sa naziv+x+y Jelena410
      </script>

</head>
<body>
      <h1>Čas 2 Primer 2 –Promenljive,operatori,izrazi ,
      expressions</h1>
      <br>
</body>
</html>
```

Komentari

Veoma je korisno u JavaScript - u postavljati komentare koji mogu poslužiti kao pomoć za tumačenje koda ili stavljanje podataka o autoru skripta i napomenu o autorskim pravima. Mudra je strategija da, sekvencijalno sa razvojem koda, ubacujemo komentare ne bi li i sebi i drugima tumačili način razvoja skripta. Ovo je posebno zahvalno raditi ukoliko se kod razvija u timu.
Postoje komentari koji se pišu u jednom ili više redova.

Primer za komentar u jednom redu je:

```
a = 1; // ovo je komentar
```

Primer za višeredni komentar je:

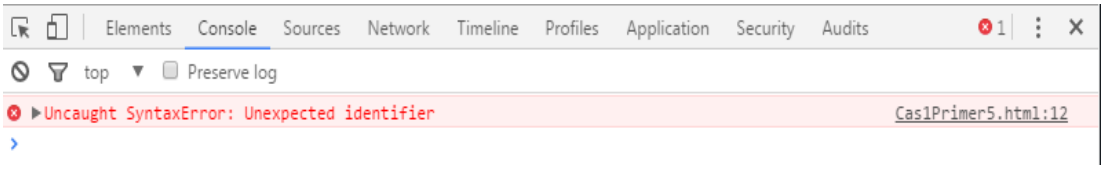
```
a = 1;
/*
      Ovo je komentar koji  ide u više redova
      i možemo da ga stavimo bilo gde u
      okviru koda
*/
a = a + 1;
```

Dakle, **jednoredne komentare** pišemo sa duplim kosim crtama ili tzv. **duplim slešom (//)**, a **višeredne** ugnježdavamo izmedu znakova **sleš zvezdica (/*) i zvezdica sleš (*/).**

Očitavanje grešaka u JavaScript-u

Kada vam se program na strani ne izvršava ili se izvršava vrlo čudno postoji način kako da proverite da li posotji greška. Svi browseri imaju ugrađene tzv. Develeoper Tools ili developerske alate za rad pomoću kojih možete da proverite da li postoji greška u vašem napisanom kodu. Ovaj alat se aktivira u svim browserima sa dugmetom F12.

Nakon aktiviranja Develeoper Tools dovoljno je potražiti karticu Console i u okviru ovog prozora sve što je prikazano sa crvenim tekstom sadrži određenu grešku u kodu koju treba ispraviti. Sa desne strane se nalazi naziv dokumenta u kojem se nalazi greška kao i linija koda u kojoj postoji greška kao što je prikazano na slici dole.



Čas 3

Tema časa : Funkcije

Pozivanje funkcija

Lokalne i globalne promenljive

Naredbe (Statements)

Identifikator metoda

Funkcije

U JavaScript - u je moguće od jednog niza naredbi i izraza, napraviti funkciju, koju možemo pozivati po potrebi iz koda. Prilikom tog pozivanja, moguće je zadati neke ulazne parametre (promenljive) koje će biti uzete u obzir u samoj funkciji. Funkcije možemo i sami praviti, ali postoje i ugrađene.

Funkcija u JavaScript - u je definicija skupa odloženih akcija. Obično se koriste za definisanje niza operacija koje treba često izvršavati. Često se funkcije projektuju tako da se mogu koristiti i u više razlicitih slučajeva u drugim dokumentima.

Sintaksa definisanja funkcije je:

```
function ime_funkcije ([parametar1][, …,  parametarn]){
  niz_operacija;
}
```

Ime funkcije treba da bude sastavljeno od brojki i slova (alfanumeričko), da ne započinje se brojem. Služi jednostavno da se funkciji zada ime na koje ćemo je kasnije pozivati.

Mali savet: zgodno je funkciju nazivati po smislenoj radnji koju će da obavlja kako bi uvek znali šta ta funkcija radi. Na primer: promenaBroja ili kvadratBroja.

Primer funkcije:

```
function kvadratBroja () {
  var a = 5 ;
  var rezultat = a * a;
  alert(rezultat);
}
```

U JavaScript kodu na jednoj stranici ne može biti više funkcija istog imena.

Kao što se vidi iz gornjeg primera pridefinisanju svake funkcije mora se zadati par običnih zagrada "()" iza imena i u njima, ukoliko je tako zahtevano zbog izvršenja funkcije ,parametri, odnosno podaci koje će funkcija koristiti tokom svog izvršenja. Ukoliko nije predviđeno da funkcija koristi bilo kakve parametre zagrade se trebaju ostaviti prazne. Ukoliko funkcija koristi više parametara oni se međusobno odvajaju zarezima unutar zagrada. Parametri se zadaju kao imena promenljivih. Bitno je shvatiti i zapamtiti da se pri pozivu funkcije kasnije u kodu, ona mora pozvati imenom i sa onoliko parametara sa koliko je definisana. Parametri se tumače redom kojim su definisani.

Primer funkcije sa parametrima:

```
function kvadratBroja (a) {
  var rezultat = a * a;
  alert(rezultat);
}
```

Značajno je uočiti i da niz naredbi koji čini funkciju mora biti zadat unutar vitičastih zagrada { }. Budući da vitičaste zagrade mogu da se koriste i van funkcija, potrebno je napomenuti da one ograđuju blok naredbi koje čine celinu.

Vrlo je bitno shvatiti da se funkcija neće nikad sama od sebe izvršiti, već da se ona mora pozvati u kodu. Samim tim njen položaj u ukupnom izgledu koda nije bitan za njeno izvršenje. Obično se dešava da se u eksternim JavaScript kodovima drže same funkcije, a da se njihovo izvršenje poziva na odgovarajućim mestima u kodu web stranice. Ukoliko su funkcije u kodu na stranici, preporučuje se da one budu pri kraju koda, a da se na početku koda nalazi kod koji nije deo funkcija i koji se, za razliku od koda u funkcijama, odmah izvršava, bez posebnog pozivanja.

Pozivanje funkcija

Kao što je posebno naglašeno u prethodnom tekstu, nijedna funkcija se neće sama po sebi izvršavati, već je neophodno pozivati ih iz odgovarajucih mesta u kodu.
Sintaksa za poziv funkcije je:

```
ime_funkcije  ([vrednost_za_parametar1] [,
                vrednost_za_ parametar2] [, …,
                vrednost_za_parametarn]);
```

Dakle, pozivanje funkcije bilo gde u JavaScript kodu vrši se ispisom njegovog imena i navođenjem vrednosti za parametre, ukoliko ih ima, u zagradi iza imena. Te vrednosti za parametre se u funkciju proslede redom po imenima promenljivih koje su zadate u defi niciji funkcije u zagradi iza njenog imena. Vrednost

parametara funkcije može biti direktno zadata vrednost ili ime neke promenljive. Ukoliko je zadata vrednost preko imena promenljive, onda će se funkciji proslediti kao trenutna vrednost promenljive.

Primer poziva funkcije :

```
kvadratBroja();
```

Primer poziva funkcije sa parametrima :

```
kvadratBroja(5);
```

Do sada su nam uvek dve zagrade pored naziva funkcije bile prazne. Te zagrade u stvari služe za definisanje ulaznih parametara. Kao u sledećem primeru gde je pozvana funkcija **param(3).**To znači da je ulaznom parametru **a** dodeljenja vrednost **3**. Vrlo je važno da kada se poziva funkcija se vodi računa sa koliko je ulaznih parametara definisana kako bi moglo da se zna sa koliko parametara da se pozove funkcija inače će biti prijavljenja greška.

Primer 1

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Čas 3 Primer 1</title>
<meta charset="utf-8">
<script>
    function param(a) {
      alert(a);
    }
</script>
</head>
<body>
<h1>Čas 3 Primer 1 - Prosleđivanje parametara funkciji</h1>
<form name="formular">
    Vrednost je već uneta samo klikni dugme:
<input type="button" name="film" value="Unesi tekst"
     onClick="param(3);"><br><br>
</form>
</body>
</html>
```

Interesantno je da je funkciju moguće pozivati ne samo iz glavnog koda (kod van funkcija), nego i iz samih funkcija. Ovo je posebno korisno ukoliko imamo jako složenu funkciju, koju, radi bolje preglednosti, možemo na ovaj način podeliti na više manjih. Moguće je, čak, da funkcija pozove samu sebe, što se naziva **rekurzija.** Iako, ponekad, znatno ubrzava i smanjuje kod, rekurzije se ne preporučuju početnicima, jer izvršavanje koda na ovaj način vrlo lako zapadne u tzv. mrtvu petlju, koja može dovesti i do ugrožavanja rada i stabilnosti browsera.

Primer 2

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Čas 3 Primer 2</title>
<script src="Cas3Primer2.js"></script>
</head>
<body>
<h1>Čas 3 Primer 2 - Primer funkcije bez i sa     parametrom,
funkcija sa povratnom vrednoscu</h1>
</body>
</html>
```

Primer: Čas3Primer2.js

```
//funkcija bez parametara
function ispisiIme(){
  alert("Moje ime je Nenad");
}

//ako imamo isti naziv funkcije poziva se ona koja je poslednja
napisana
//jako losa praksa
/*
function ispisiIme(){
  alert("Moje ime je Pera");
}
*/

//funkcija sa parametrom
function ispisiImeSaPar(ime){
  alert("Moje ime je "+ ime);
}

//funkcija koja vraca vrednost
function vratiZbir(prviBroj,drugiBroj){
  var zbir = prviBroj + drugiBroj;
  return zbir;
}

//poziv funckije bez parametra
ispisiIme();

//poziv funckije sa parametrom
//ispisiImeSaPar("Pera");

//pozovi funkciju i smesti rezltat u promenljivu
//var rezultat = vratiZbir(3,12); //15
//alert("ukupan zbir je "+rezultat);
```

Lokalne i globalne promenljive

Potrebno je napraviti jasnu razliku između tzv. globalnih i lokalnih promenljivih. U JavaScriptu - u se **globalnim promenljivama** nazivaju promenljive koje se definišu van funkcija i koje mogu biti korištene u svim funkcijama i naredbama na tekućoj stranici. One po izlasku sa date stranice prestaju da postoje. **Lokalnepromenljive**, s druge strane, su one koje se definišu unutar neke funkcije i koje se samo u toj funkciji mogu i koristiti.

Pogledajmo lokalne i globalne promenljive kroz sledeći primer

Primer 3

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Čas 3 Primer 3</title>
<script src="Cas3Primer3.js"></script>
</head>

<body>
<h1>Čas 3 Primer 3 - Lokalne i globalne promenljive
(Objekti)</h1>
</body>
</html>
```

Čas3Primer3.js

```
//globalna promenljiva
var ime = "Pera";

function ispisiIme(){
  //lokalna promenljiva
  var ime = "Ana";
  alert("Moje ime je "+ime);
}
ispisiIme();
```

U gornjem primeru pozivamo JavaScript fajl pod imenom Cas1PrimerF.js koji u sebi sadrži definisanu globalnu promenljivu ime I lokalnu promenljivu ime. Pozivom funkcije ispisiIme ispisaće se ime *Moje ime je Ana.* U ovom slučaju lokalna promenljiva je dostupna u okviru funkcije ispisIme i ona se ispisuje. Ako bi komentarisali promenljivu ime u funkciji ispisiIme rezultat ispisa bi bio *Moje ime je Pera.*

Naredbe (Statements)

U JavaScript-u postoje i posebne naredbe kojima se programiraju razna grananja, petlje i sl. Učićemo naredbe jednu po jednu. Prva koju ćemo raditi je **naredna if**. Ovo je osnovna naredba **kontrolna naredba** koja nam omogućava u JavaScript-u "donošenje odluka".Pogledajmo primer:

```
if (b > 100) {
      a = 3;
} else {
      a = 1;
}
```

Ovo znači da ukoliko je vrednost izraza b > 100 tačna, onda će promenljivoj a biti zadata vrednost 3, a inače će biti zadata vrednost 1.

Pogledajmo sledeći primer:

Primer 4

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Čas 3 Primer 4</title>
<script>

   var j = 10;

   function odluka(){
     if (j<10){
       alert("Manji je od 10");
     } else if (j>10) {
       alert("Veci je od 10");
     } else {
       alert("Jednako je  10");
     }
     //dodati else uslov
   }
</script>

</head>
<body>
 Čas 3 Primer 4 - Naredbe , statements , kontrola toka
  <br>
  <input type="button" name="dugme" value="Klikni me"
      onclick="odluka()" >
</body>
</html>
```

Događaji (metodi)

Događaji, kako je spomenuto, predstavljaju jedno od tri obeležja objekata na web stranicama, pored imena, osobina i procedura za obradu događaja. Mi ćemo na ovom kursu događaje nazvati imenom **metodi**, zbog analogije sa njihovim engleskim nazivom methods. Metod je komanda koju skript može zadati nekom objektu. Metodi su izuzetno bitne karakteristike objekta koje mi možemo očitavati i preko kojih mi možemo inicirati dalje operacije. Takođe, neki objekti nemaju defi nisane metode.

Svi metodi imaju na kraju zagrade (kao i funkcije) i uvek se pojavljuju na kraju reference objekta. Kada metod prihvata ili zahteva parametre, njihove vrednosti se stavljaju unutar zagrada.

Primer sintakse bi bio:

```
document.write("Pozdrav svima");
```

U navedenom primeru na stanici na kojoj se nalazi skript biće ispisan tekst u zagradi. Naime, ovde je pozvan **write() metod** objekta document (ovaj objekat ćemo kasnije detaljnije objasniti ,a ovde ga koristimo radi lakšeg objašnjena pojma metoda) i kao parametar mu je dostavljen string Pozdrav svima.

U daljem tekstu bavićemo se opisom pojedinih metoda na objektima i opisom koje metode na odgovarajucim objektima na stranici možemo očekivati.

Osnovni tipovi metoda na objektima na web stranici

click

Ovaj metod se izaziva klikom miša (levo dugme) na pojedine objekte na stranici (linkove, slike, elemente formulara...). Identifikator metoda je: **onClick**

change

Ovaj metod se izaziva promenom sadržaja na nekom elementu na formularu.

Identifikator metoda je: **onChange**

focus

Metod se, najčešće, primenjuje na formularima, mada može da se primeni i na prozor. Fokus je metod koji se dešava kada se, recimo, selektuje neki elemenat formulara radi njegove promene ili nekog drugog razloga. Tako na primer fokus na tekst elementu formulara prepoznaćete po tome što se kursor za unos podataka nalazi na njemu. Tada kažemo da taj elemenat ima fokus. Momenat kada taj elemenat dobija fokus predstavlja focus metod tog objekta.

Identifikator metoda je: **onFocus**

blur
Ovaj metod je suprotan metodu fokus i dešava se kada elemenat (objekat) izgubi fokus.

Identifi kator metoda je: **onBlur**

mouseover
Metod se aktivira prelaskom pokazivača miša preko objekta. Najčešće je u pitanju linkovani element (tekst, slika), mada u poslednje vreme omogućen je i za elemente tabela.

Identifi kator metoda je: **onMouseOver**

mouseout
Metod se, suprotno od prethodnog, aktivira kada pokazivač miša napusti objekat.

Identifi kator metoda je: **onMouseOut**

select
Ovaj metod se dešava kada se selektuje tekst u okviru text ili textarea elementa formulara.

Identifi kator metoda je: **onSelect**

submit
Metod se aktivira kada se pritisne dugme submit na formularu. Time se postiže mogućnost da se pre slanja formulara aktivira JavaScript funkcija za proveru unetih podataka.

Identifi kator metoda je: **onSubmit**

resize
Dešava se u trenutku kada se izvrši promena veličine prozora pretraživača.

Identifi kator metoda je: **onResize**

load
Metod označava trenutak kada je stranica prezentacije potpuno učitana u prozor pretraživača.

Identifi kator metoda je: **onLoad**

unload
Suprotno prethodnom, ovaj metod aktivira se kada posetilac napušta stranicu bilo "gašenjem" prozora bilo odlaskom na neki drugi URL. Identifi kator metoda je: **onUnload**

Identifikator metoda

Iz gornjih primera vidi se da svaki metod ima svoj identifikator. Identifikator je potreban da bi na različitim objektima u HTML dokumentu mogli da programiramo metode i njihove posledice (aktiviranje okvira za dijalog, provera formulara, otvaranje novog prozora, odlazak na novi URL i sl.). Sintaksa bi u HTML - u izgledala ovako:

```
<TAG [atribut1 ][atribut2identifikator_metoda="javascript_kod">
```

Identifikator metoda se piše kao poseban atribut HTML taga, a njegova vrednost je JavaScript kod (poziv funkcije ili cela funkcija). Preporuka je da se funkcija koja se aktivira na ovom mestu definiše na drugom mestu tj u okviru JavaScript fajla, a da se ovde samo poziva. Najčešće je to i slučaj, iz razloga što se često funkcije projektuju tako da ih možemo pozivati na različitim mestima sa drugačijim parametrima čime se ovako postiže ušteda prostora.

NAPOMENA O ZNAKU ";": Često se u JavaScript kodovima na internetu može videti da se na nekim mestima ovaj znak koristi, a na nekim ne. Razlog ovome je što u opštem slučaju znak ne mora da se stavlja na kraju reda kada postoji samo jedna naredba u nizu. Moj savet je da se dok se ne stekne sigurnost, a i kasnije, na kraju svakog reda sa naredbom znak ";" stavlja ne bi li se izbegle neprijatnosti sa neizvršavanjem skripta, jer je uočavanje ove greške jako teško u šumi linija koda.

Pogledajmo sledeći primer:

Primer 5

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Čas 3 Primer 5</title>
<script>
   function dijalog(){
     alert("Proba funckije");
   }
</script>
</head>
<body>
<h1>Čas 3 Primer 5 - Idetifikatori metoda</h1>
<br>
<!-- onClick je identifikator metoda -->
<input type="button" name="dugme" value="Klikni me"
        onclick="dijalog()" >
</body>
</html>
```

Vežba

Napravite stranicu sa jednim 5 dugmadi. Svako dugme će da radi kao jedan od gornjih primera parametre možete slobodno menjati.

Čas4

Tema časa:

Integrisani objekti

BOM (Browser Object Model)

Window objekat

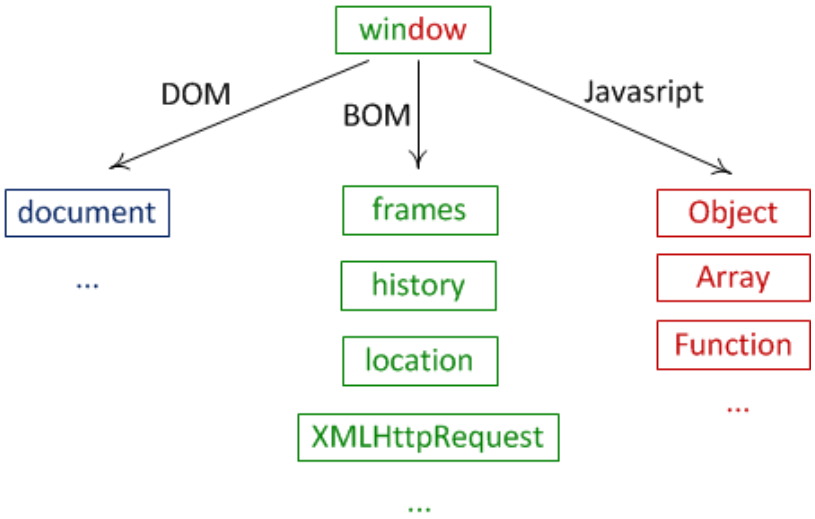
Metode objekta window kroz primere

Procedure za obradu događaja objekta window

Integrisani objekti

Web browseri sadrže skup integrisanih objekata koji omogućavaju pristup pretraživaču ili strukturi HTML stranice iz JavaScript programa. Ovi objekti su dostupni u okviru svih savremenih pretraživača i možemo ih podeliti u 3 grupe:

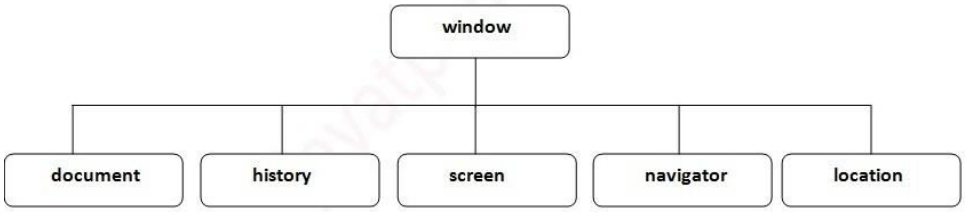
- Browser Object Model
- Document Object Model
- Globalni JavaScript objekti



Slika 5. Integrisani objekti

BOM (Browser Object Model)

BOM sadrži objekte koji predstavljaju prozor internet pretraživača tj. trenutno aktivan tab. Globalni objekat i ujedno objekat koji predstavlja prozor pretraživača je **window** objekat. Po hijerahiji kao što vidimo na slici on je roditelj svim ostalim objektima i oni ga nasleđuju.



Slika 6. BOM struktura

U nastavku navodimo detalje svakog objekta u BOM-u

Window objekat

Kao što smo rekli on je korenski objekat i svim ostalim objektima se pristupa preko ovog objekta .

Window objekat je najviši objekat u hijerarhiji objekata u JavaScriptu - u. Sve HTML i JavaScript aktivnosti odvijaju se u okviru ovog objekta. Taj prozor može biti u obliku standardne Windows ili Mackintosh aplikacije sa paletama alatki, trakama za pomeranje i sličnim detaljima ili bez nekih od njih ukoliko ga sami pravimo.

Jedinstvena pozicija ovog objekta ima zanimljive implikacije na njegovo pojavljivanje u našim skriptovima. Naime, u referenciranju objekata on se često (gotovo uvek) izbegava, odnosno, ne koristi, iz razloga što je to jedini podrazumevani objekat. Budući da svi ostali objekti predstavljaju, takoreći, njemu "potčinjene" objekte on je objekat koji se retko koristi u referencama objekata na stranicama.

Objekat window sadrži propertije ili atribute od kojih navodimo sledeće :

- ***innerWidth, innerHeight*** - prikazuje širinu i visinu prozora i to samo dela sa HTML stranicom
  Primer:
  ```
  var w = window.innerWidth;
  var h = window.innerHeight;
  ```

- ***outerWidth, outerHeight*** - prikazuje spoljnu širinu i visinu prozora uključujući i toolbarove i skrolbarove

Primer:

```
var w = window.outerWidth;
var h = window.outerHeight;
```

- ***pageXOffset, pageYOffset–*** prikazuje koliko piksela je dokument skrolovan horizontalno i vertikalno u odnosu na gornji levi ugao.
  Primer:

```
var x  = window.pageXOffset;
var y  = window.pageYOffset;
```

- ***screenX, screenY*** – prikazuje koordinate prozora pretraživača u odnosu na levi ugao ekrana
  Primer:

```
var winX = window.screenX;
var winY = window.screenY;
```

- ***closed***– za zatvoren daje vrednost True za otvoren False

- ***opener***- referenca prozora iz kojeg je otvoren postojeći

- ***document*** – predstavlja trenutno učitanu HTML stranicu.Ovaj objekat je korenski element Document Object Modela koji ćemo kasnije obrađivati

Window objekat pored propertija sadrži i korisne metode od kojih navodimo samo neke u nastavku

- ***window.alert(tekstPoruke)*** – sa ovom metodom smo se upoznali tokom naše prve JavaScript aplikacije, a namena ove metode je da prikaže pop-up dijalog sa prosleđenom porukom. Ona sadrži jedno dugme za potvrdu. Ovaj metod prima jedan parametar koji je ustvari tekst poruke koji se ispisuje u okviru pop-up poruke.

- ***window.confirm(tekstPoruke)***–ova metoda prikazuje pop-up dijalog sa unetim tekstom poruke i sadrži OK i Cancel dugmad.

- ***window.prompt(tekstPoruke,ulaznaVrednost)*** – ova metoda prikazuje pop-up ekran koji prikazuje tekst poruke i polje sa unos ulazne vrednosti. Razlikuje se od prompt metode što se funkciji vraća ulazna vrednost. Ako je ulazna vrednost definisana ona će biti ispisana u polje za unos ulazne vrednosti u suprotnom će biti prazno polje. Pop-up ekran sadrži dugme OK i Cancel. Ako se pritisne dugme OK vrednost iz polja za unos se prenosi u funkciju ,a ukoliko se pritisne dugme Cancel vrednost iz polja za unos koja će se vratiti u funkciju će biti null.

- ***window.open(url)*** – ova metoda otvara novi prozor (tab) sa definisanom adresom navedenom u url parametru

- ***window.close()*** – ova metoda zatvara prozor

- ***window.print()*** – ova metoda pokreće ekran za štampanje trenutne stranice

- ***window.setTimeOut(imeFunkcije,interval)*** – ova metoda poziva funckiju ili izvrši neki izraz nakon određenog vremenskog intervala u milisekundama. Ova metoda se izvrši samo jednom.

- ***window.setInterval(imeFunkcije,interval)*** – ova metoda poziva funckiju ili izvrši neki izraz nakon određenog vremenskog intervala u milisekundama. Ova metoda se izvršava u kontinuitetu sve dok se ne pozove metod **clearInterval()** ili dok se prozor ne zatvori.

- ***window.resizeTo(širina,visina)*** **–** ova metoda proširuje prozor na zadatu širinu i visinu

- ***window.focus()*** **–** ova metoda podešava fokus na trenutni prozor

Ukoliko želimo da na stranici prikažemo neki tekst pomoću metoda alert objekta window možemo koristiti sledeća dva potpuno ravnopravna načina:

```
window.alert("Dobar dan");
 // što je isto kao i
alert("Dobar dan");
```

Uočićete da se u drugom (inače najčešćem) načinu referenciranja ne pominje objekat window, a da je efekat potpuno isti.

Metode objekta WINDOW kroz primere

Već je nagovešteno da je moguće praviti nov prozor na svojim stranicama koji po izgledu odudaraju od izgleda standardnih, ali odgovaraju našim potrebama. To su takozvani "pop-up" prozori. Na ovom mestu upozoravamo da korišćenje ovih prozora može da naiđe na negodovanje posetilaca ukoliko nije svrsishodno ili ukoliko zasmeta kontinuitetu surfovanja. Takođe, najčešće, ovakvi prozori se koriste za izbacivanje dosadnih reklamnih poruka koje teraju posetioce da ih iznervirano zatvaraju udaljavajući ih od sadržaja prezentacije, te ne retko izaziva napuštanje takvih prezenatcija u korist onih koji takve prozore ne izbacuju. Stoga savet: koristite ih veoma ograničeno i sa najavom.

window.open()

Metod objekta window koji omogućava otvaranje novog prozora je open(). Kao što smo i naveli malo ranije ovaj metod ima parametar URL ali poseduje još nekoliko parametar koji su veoma korisni.

Svi parametri su opcioni za unos. Ako ne unesemo ni jedan parametar otvoriće se prazan tab.

Način primene je:

```
var noviProzor = window.open ("[URL]","[ime]"[,"parametri"]);
```

Objasnimo detaljnije parametre metode open() :

1. ***URL***– putanja do stranice koja treba da se prikaže u novom prozoru (opcion parametar)
2. ***ime***- naziv novog prozora koji može poslužiti kao deo reference do njega. Može da bude
    - **_blank** – URL se učitava u novom prozoru. Ovo je default vrednost
    - **_parent** – URL se učitava u frejmu roditelja
    - **_self** – URL menja trenutnu stranicu
    - **_top** – URL menja bilo koji frameset koji se učita
    - **imeProzora** – proizvoljno ime prozora (ovo ime ne definiše naslov novog prozora)
3. ***parametri*** - osobine prozora koje se mogu podešavati:
    - **toolbar** - kreira standardnu paletu alatki (toolbar) pretraživača. (koristi se samo u IE i Firefox-u)
    - **location** - kreira location bar tj. mesto gde se upisuje URL (koristi se samo u Operi)
    - **directories** - kreira What's New i What's Cool i sl. dugmiće, ne koristi se više (koristi se samo u IE)
    - **status** - kreira status bar (ne prikazuju ga browseri po defaultu)
    - **menubar** - kreira standardnu paletu menija pretraživača.
    - **scrollbars** - kreira horizontalne ili vertikalne trake za pomeranje (koristi se samo u Operi,IE I Firefox-u)
    - **resizable** - određuje da li se može menjati veličina prozora (koristi se samo u IE)
    - **width, height** - veličina prozora u pikselima (minimalna vrednost je 100)
    - **top** - udaljenost prozora od vrha .Negativna vrednost nije dozvoljena

Kako se pravi novi prozor u JavaScriptu demonstriramo u sledeće primeru.

Primer 1

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Čas 4 Primer 1</title>
<script>
  function otvori(){

     window.open("Cas4Prozor.html","_blank",
                 "width=200,height=300");
 }
</script>
</head>
<body>
<h1>Čas 4 Primer 1 - Metoda WINDOW.OPEN</h1>
<br>
<input type="button" name="pisi" value="Otvori"
  onClick="otvori()">
</body>
</html>
```

Sem width i height parametara svi ostali su logički parametri (boolean) i aktiviraju se sa yes ili sa 1, a isključuju sa no ili 0, tako da im je izgled: parametar= yes ili parametar=1 tj. parametar=no ili parametar=0

Ukoliko ih je više, parametre odvajamo zarezom, i postavljaju se kao treći parametar window.open metoda pod navodnicima. Ukoliko se ne navedu podrazumevaju se default vrednosti.

Window.close()

U praksi je običaj da se posetiocu ponudi "surferski" način zatvaranja prozora pored standardnog pritiskom pokazivačem miša na X u gornjem desnom uglu ili sa Alt F4 kombinacijom, a to se radi pomoću **close()** metoda objekta window koju smo spomenuli ranije. Sada ćemo da napravimo prozor sa više parametara ,a nakon toga i primer sa zatvaranjem prozora.

Primer 2

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Čas 4 Primer 2</title>
<script>
  function otvori(){
      window.open("Cas4Prozor.html","_blank",
        "width=800,height=600,
        toolbar=1,location=1,status=1,
```

```
                menubar=1,scrollbar=1");
 }
</script>
</head>
<body>
<h1>Čas 4 Primer 2 - Metoda WINDOW.OPEN sa više
 parametara</h1>
<br>
 <input type="button" name="pisi" value="Otvori"
 onClick="otvori()">
</body>
</html>
```

Window.setTimeout()

Zatvaranje prozora nekad je poželjno automatizovati (učiniti da se izvrši posle određenog vremena) , a za to koristimo uz pomoćmetode **setTimeout**koju smo objasnili malo ranije. Pre nego objasnimo rad ove metode kroz primer naglasimo još jednom da se ova metoda izvrši **samo jednom** u odnosu na metod **setInterval** koja se izvršava ciklično dok se ne zaustavi metodom **clearInterval** ili zatvori prozor

Pogledajmo kroz primer kako radi metod setTimeout.

```
            setTimeout(window.close(),30000);
```

U gornjem primeru će metod setTimeout pozvati metod **close**, ali će se izvršiti tek za 30 sekundi ,a to znači da će prozor biti zatvoren posle 30 sekundi.

U sledećem primeru prozor se zatvara posle 5 sekundi.

Primer 3

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8">
 <title>Čas 4 Primer 3</title>
 <script>
      var prozor;

      function zatvori(){
            prozor.close();
      }

      function otvori(){
            prozor =
window.open("Cas4Prozor.html","prvi","width=800,height=60
0,toolbar=1,status=1,location=1,directories=1,menubar=1");
```

```
                setTimeout('zatvori()',5000);
          }
      </script>
      </head>
      <body>
           <h1>Čas 4 Primer 3 - Metoda WINDOW.CLOSE() </h1>
           <br>
           <input type="button" name="pisi" value="Otvori"
                onClick="otvori()">
           <input type="button" name="pisi" value="Zatvori"
                onClick="zatvori()">
      </body>
     </html>
```

Pogledajmo kroz sledeći primer kako funkcioniše metode
prompt(),print(),resizeTo() i focus()

Prompt()

Primer 4

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Čas 4 Primer 4</title>
<script>
  //prompt metod
  function unesiVrednost(){
    var unetaVrednost = window.prompt("Unesite zeljenu
     vrednost",'demo');
    alert("Uneli ste sledecu vrednost :
     "+unetaVrednost);
  }

  //print metod - stampaj stranicu
  function stampaj(){
    window.print();
  }

  //resizeTo
  var mojProzor;

  function otvori() {
    mojProzor = window.open("", "", "width=100,
     height=100");
  }

  function promeniVel() {
    mojProzor.resizeTo(250, 250);
    mojProzor.focus();
  }
</script>
</head>
<body>
```

```
<h1>Čas 4 Primer 4 - Metoda PROMPT,PRINT,RESIZETO objekta
WINDOW</h1>
<br>
<a href="#" onMouseOver="unesiVrednost();"> Unesi automatski
vrednost</a>
<br><br>
<input type="button" value="Štampaj stranicu"
     onclick="stampaj();">
<br><br>
<input type="button" value="Otvori prozor"
     onclick="otvori()">
<br><br>
<input type="button" value="Promeni veličinu"
     onclick="promeniVel()">
</body>
</html>
```

Confirm()

U sledećemo primeru ćemo da vidimo kako radi metod **confirm()**

Primer 5

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Čas 4 Primer 5</title>
<script>
  function status(){
    var x = window.confirm("Da li ste sigurni da
      zelite da napustite ovu stranu");
    //alert(x);
    if(x){
     window.close(); //u IE radi, ne radi u
Chrome, u Firefoxu radi ako se u about:config
podesi dom.allow_scripts_to_close_windows na
true
    }
  }
</script>
</head>
<body>
 <h1>Čas 4 Primer 5 - Metoda PROMPT objekta WINDOW</h1>
 <br>
 <a href="#" onClick="status();">Klikni</a>
</body>
</html>
```

Okviri za dijalog su izgleda kakav je unapred zadat u pretraživaču i ne mogu se menjati standardnim Java Script - om, mada za to postoji mogućnost sa tzv. potpisanim skriptima. Tema istih prevazilazi okvir ovog kursa.

Properti (osobine) window objekta kroz primer

U primeru koji sledu koristimo sve propertije koje smo definisali malo ranije.

Primer 6

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Čas 4 Primer 6</title>
<script>
  //Prikaz velicine ekrana
  function velicinaEkrana() {
    var s = window.innerWidth;
    var v = window.innerHeight;

    var os = window.outerWidth;
    var ov = window.outerHeight;

    alert("Širina unutrasnjeg ekrana je "+s+" a visina
     unutrasnjeg ekrana je "+v+" dok je sirina spolnjeg
     ekrana "+os+" a visina spoljneg ekrana "+ov);
      };

  //Prikazi ime prozora
  function otvoriProzor(){
    var prozor = window.open("", "MojProzor", "width = 200,
     height = 100");
    prozor.document.write("<p>Ime ovog prozora je: " +
     prozor.name + "</p>");
    prozor.opener.document.write("<p>Ovo je izvorni
     prozor</p>");
  };

  //Prikaz closed osobine objekta Window
  var mojProzor;

  function otvori() {
    mojProzor = window.open("", "MojProzor 2", "width=400,
     height=200");
  }

  function zatvori() {
    if (mojProzor) {
        mojProzor.close();
    }
  }

  function proveri() {
    if (!mojProzor) {
        alert("'mojProzor' nikada nije bio otvoren!");
    } else {
        if (mojProzor.closed) {
            alert( "'mojProzor' je zatvoren!");
        } else {
            alert("'mojProzor' nije bio zatvoren!");
        }
```

```
    }
}

</script>
</head>
<body>
<h1>Čas 4 Primer 6 - Pregled osobina objekta Window</h1>
<input type="button" value="Veličina ekrana"
     onclick="velicinaEkrana()">
<br><br>
<input type="button" value="Otvori prozor"
onclick="otvoriProzor()">
<br><br>
<button onclick="otvori()">Otvori "mojProzor"</button>
<br><br>
<button onclick="zatvori()">Zatvori "mojProzor"</button>
<br><br>
<button onclick="proveri()">Proveri status
"mojProzor"</button>
<br><br>
<div id="msg"></div>
</body>
</html>
```

Svako dugme prikazuje po jednu osobinu(properti) metode window. Na Dugme **Veličina ekrana** se prikazuje unutrašnja i spoljna širina I visina ekrana pozivajući propertije **innerHeight** i **innerWidth** kao i **outerWidth** i **outherHeight**.
Na dugme **Otvori prozor** pored otvaranja prozora ,upisujemo u prozor na kojem se nalazi ovo dugme novu vrednost koristeći **document.write** metod koji ćemo kasnije da objasnimo. Ovo smo uradili koristeći **opener** properti(osobinu) koja nam vraća fokus na roditeljski prozor tj. onaj prozor koji sadrži dugme **Otvori prozor.**Na dugme Proveri status proveravamo da li je prozor postoji I ako posotji koristimo closed properti koji ukoliko nam vrati true znači da je prozor zatvoren i ispišemo poruku `'mojProzor' je zatvoren!`u suprotnom je false i ispišemo poruku `'mojProzor' nije bio zatvoren!`

Procedure za obradu događaja objekta WINDOW

Specifičnost ovih procedura za obradu događaja jeste da se definišu u okviru body tag elementa koji direktno utiče na ponašanje window objekta. Navešćemo procedure koje koristimo za obradu događaja objekta window :

- onBlur - gubitak focusa
- "onError - aktivira se ako u JS kodu postoji sintaksna greška ili ako dode do greške u izvršavanju JS koda "onFocus =" - dobijanje fokusa
- onLoad - učitavanje sadržaja prozora
- onUnload - zatvaranje prozora

- onDragDrop - aktivira se prevlačenjem nekog sadržaja u prozor (više se ne koristi)
- onResize - promena dimenzija prozora
- onMove - promena položaja prozora na ekranu
-

Pogledajmo primer kako ove procedure rade

Primer 7

```
<!DOCTYPE html>
<html lang="en">
<head>
<script>
  function prikazi() {
     alert("Uklonjen je fokus sa prozora");
  }

</script>
<title>Čas 4 Primer 7</title>
</head>
<body onBlur="prikazi();">
<!-- Koristeci HTML onLoad onBlur onUnload IE onError
onResize -->
<h1>Čas 4 Primer 7 - Procedure za obradu događaja
 objekta WINDOW
</h1>
</body>
</html>
```

Vežba

Napravite stranicu koja ima dva dugmeta.Klikom na dugme otvori se prozor kom može da se menja veličina sa tulbarom, statusbarom, menijem, adresbarom, dimenzije prozora treba da budu 400x200px. U novom prozorutreba da se nalazi dugme na kome piši “zatvori me”. Kada kliknemo na to dugme prozor se zatvori.

Čas 5

Tema časa:

DOM (Document Object Model)

Pristup elementima DOM stabla

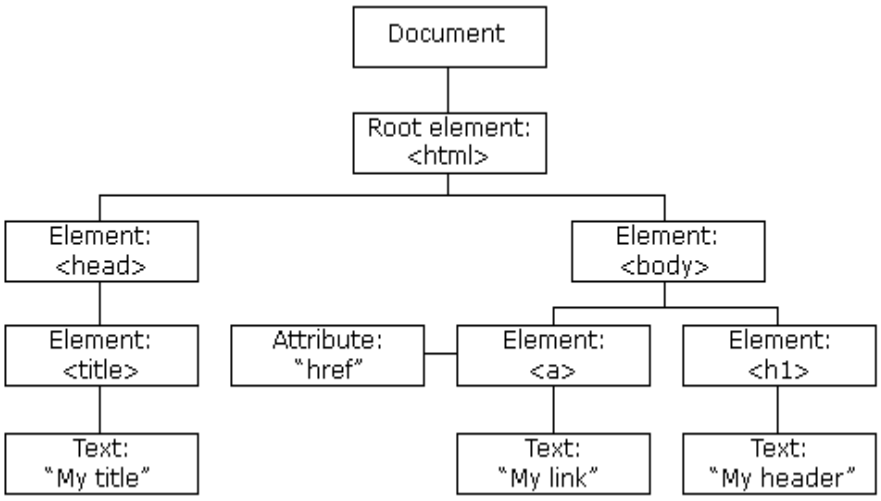
Ostali BOM objekti

DOM (Document Object Model)

DOM predstavlja model reprezentaciju u obliku strukture stabla gde svaki čvor (ili node) je ustvari objekat reprezentacija dela dokumenta.

Svi pretraživači renderuju dokument kao što je HTML stranica koristeći interni model sličan DOM-u. Čvor ili node svakog dokumenta je organizovan u stukturu stabla koju zovemo DOM stablo. Na vrhu stabla se nalazi **document** objekat.
Videli smo u času 4 da je document objekat dete window objekta u okviru BOMa.

Pogledajmo kako izgleda grafički prikaz jednog DOM stabla.



Slika 7. DOM stablo

Kada se HTML stranica izrenderuje u pretraživaču, pretraživač sačuva HTML u lokalnoj memoriji i automatski parsira I prikazuje stranicu na ekranu . To znači da kada se web stranica učita pretraživač kreira DOM stranice. Sa objekat modelom JavaScript je spreman da pravi dinamičan HTML.

Sa ovim objektima u DOM stablu možemo programski da manipulišemo koristeći JavaScript objekat document koji smo rekli da se nalazi na vrhu stabla. Uz pomoć metoda i propertija ovog objekta možemo da pristupimo i da menjamo elemente HTML dokumenta. Da pogledamo šta sve može JavaScript dinamički da radi sa HTML koristeći DOM elemente :

- JavaScript može da dodaje , menja ili uklanja sve HTML elemente i atribute na stranici
- JavaScript može da menja sve CSS stilove na stranici
- JavaScript može da reaguje na sve događaje na stranici
- JavaScript može da pravi nove događaje na stranci
-

Možemo slobodno da kažemo da DOM manipulacija predstavlja osnovu kreiranja dinamičkih HTML stranica.

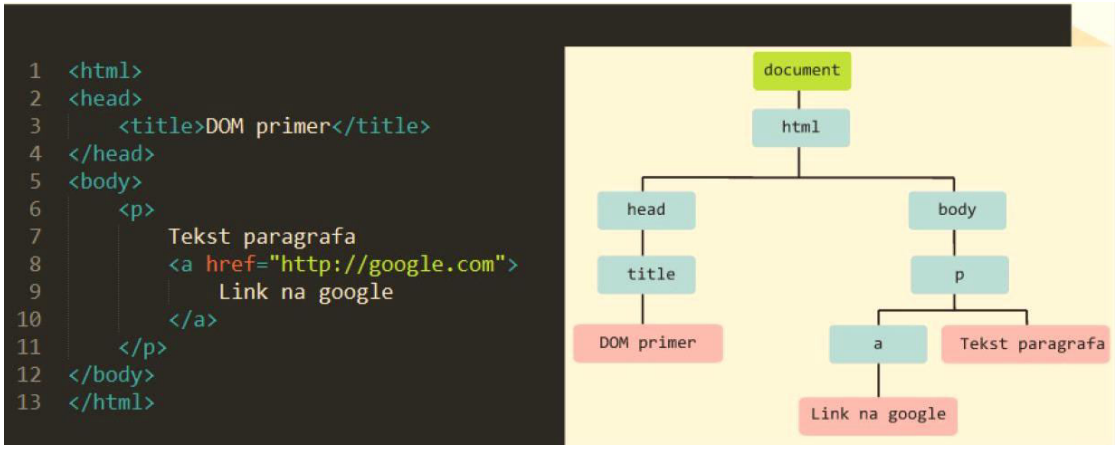
Kako bi mogli da manipulišemo sa HTML DOM objektima (u DOM svi HTML elementi su definisani kao objekti) koristimo JavaScript . Koristićemo metode i propertije za rad sa HTML DOM objektima.

Pogledajmo deo koda jedne HTML stranice

```
…
<body>

<p id="tekst"></p>

<script>
  document.getElementById("tekst").innerHTML = "Dobro
došli";
</script>

</body>
…
```

U gornjem primeru nam je **getElementById metod** ,a **innerHTML je properti**

Pogledajmo sliku dole



Slika 8. DOM stablo

Vidimo primer reprezetacije HTML koda sa leve strane renderovanu u DOM stablo koje je uradio naš pretraživač. Ukoliko bi želeli da pristupimo elementima HTML stabla koristićemo jednu od metoda JavaScripta kao i ostale propertije sa kojima možemo da dinamički promenimo izgled i strukutru postojećeg DOM stabla.

Pristup elementima DOM stabla

DOM *document* objekat je vlasnik svih objekata na vašoj web stranici ili bolje rečeno *document* objekat predstavlja ustvari web stranicu. Ako hoćemo da pristupimo bilo kom elementu HTML stranice uvek moramo da pozovemo objekat *document.*
Ako želimo da pristupimo i manipulišemo sa HTML elementima koristićemo sledeće metode:

1. **document.getElementById(id)** - pronađi element po ID
2. **document.getElementsByTagName(name)** – pronađi sve elemente po imenu taga
3. **document.getElementsByClassName(name)** – pronađi elemente po imenu klase
4. **document.querySelectorAll(selektor)** – pronađi sve elemente po CSS selektoru (ova metoda ne radi u <IE9)
5. **document.querySelector(selektor)** – pronađi prvi element po CSS selektoru (ova metoda ne radi u <IE9)
6. **pristupanje HTML elementima koristeći ime forme**

Pored metoda za pristup HTML elementima postoje i odgovarajući propertiji koji mogu da promene vrednosti HTML elemenata ili njihove CSS karakteristike. Nabrojaćemo najčešće korišćene propertije:

1. **innerHTML** - menja HTML elementa (ukoliko se ovaj properti nalazi sa leve strane jednakosti tada menja vrednost HTML elementa, a ukoliko se nalazi sa desne strane jednakosti tada samo dohvata vrednost HTML elementa)
2. **getAttribute(imeAtributa)** – dohvata atribut HTML elementa
3. **setAttribute(imeAtributa,novaVrednost)** – menja vrednost atributa HTML elementa
4. **style.*CSSproperty*** – menja CSS stil HTML elementa

Svaki od ovih propertija se može primeniti nad prethodno dostavljenom elementu. Zato ćemo prvo da naučimo kako da pristupimo HTML elementima koristeći gore navedene metode, a ujedno da isprobamo gore navedene propertije. Krenimo redom :

document.getElementByID

Ova metoda će na osnovu definisanog DOM stabla pronaći element po zadatom ID atributu. Tako pronađen element možemo da sačuvamo u nekoj promenljivoj. Bitno je za napomenuti da koriteći ovaj metod pristupamo samo jednom elementu u okviru HTML stranice jer da se podsetimo u okviru jedne HTML stranice može da postoji samo jedan jedinstveni ID.

Pogledajmo primer :

Primer 1

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Čas 5 Primer 1</title>
<!-- da li možemo i ovde da stavimo script tag ?-->
</head>
<body>
<h1>Čas 5 Primer 1 - document.getElementById</h1>

<p id="tekst">Ovaj tekst treba da se prikaže u pop-up
      ekranu</p>

<input type="button" value="Prikaži tekst iz h1 elementa"
      onclick="myFunction()">

<script>
      function myFunction() {
            var tekstZaPrikaz=
            document.getElementById("tekst").innerHTML;
            alert(tekstZaPrikaz);
      }
</script>

</body>
</html>
```

Zadržimo se ukratko na prethodnom primeru kako bi objasnili bitne pojedinosti. U prethodnom primeru imamo definisan jedan **p element** (tag) u kojem se nalazi neki tekst. Ovaj element sadrži atribut **id** sa imenom **tekst**. Mi želimo da kada se pritisne dugme "Prikaži tekst iz h1 elementa" , ceo tekst iz p elementa prenese u pop-up prozor.

Da bi ovo uradili potrebno je da koristeći DOM metod **getElementByID** (budući da u okviru p elementa imamo definisan ovaj atribut) dohvatimo ceo p element i smestimo ga u promenljivu **tekstZaPrikaz** koja je definisan u script tagu. Ukoliko navedemo properti **innerHTML** tada ćemo iz celog p elementa da uzmemo samo njegov HTML tekst koji se nalazi u okviru ovog p elementa. Nakon toga ovaj tekst koji nam se nalazi promenljivoj **tekstZaPrikaz** prosledimo metodi alert koja će ga ispisati.

Kao što vidimo u ovom primeru smo objedinili korišćenje metode **getElementByID** I propertija **innerHTML**.

Ukoliko bi imali sledeći kod u našem primeru

```
var tekstZaPrikaz= document.getElementById("tekst");
```

tada bi se u promenljivoj **tekstZaPrikaz** sačuvao ceo element p tj objekat i u prikazu u pop-up ekranu bi se prikazao tekst

```
[object HTMLParagraphElement]
```

document.getElementsByTagName

Ovom metodom možemo da koristeći DOM stablo pristupimo elementu po imenu taga.Bitno je za napomenuti da koristeći ovaj metod pristupamo i dohvatamo sve elemente sa navedenim tagom. To znači da ova metoda može da vrati i više od jednog elementa koji se čuvaju u nizu. Zbog toga se treba voditi računa kojem elementu po redu želimo da pristupimo.

Pogledajmo sledeći primer.

Primer 2

```
<!DOCTYPE html>

<html lang="en">
<head>
      <meta charset="UTF-8">
      <title>Čas 5 Primer 2</title>
</head>
<body>
<h1>Čas 5 Primer 2 - document.getElementByTagName</h1>

<p>Ovaj tekst ne treba da bude prikazan</p>
```

```
<p>Ovaj tekst treba da se prikaže u pop-up ekranu i da nakon
      toga promeni boju u crveno</p>

<input type="button" value="Prikaži tekst iz drugog p
      elementa" onclick="prikaziIOfarbaj()">

<script>
   function prikaziIOfarbaj() {
    var tekstZaPrikaz =
      document.getElementsByTagName("p")[1].innerHTML;
    alert(tekstZaPrikaz);
    document.getElementsByTagName("p")[1].style.color = "red";
   }
</script>
</body>
</html>
```

Objasnimo ukratko gornji primer. U odnosu na prethodni primer ovde imamo dva p elementa (taga). Primetimo da u p elementima sada nemam definisane ID atribute. Ukoliko želimo da pristupimo drugom p elementu ovaj put koristimo metod **getElementsByTagName** koji nam dohvata sve p elemente koji postoje na HTML stranici. Sada se postavlja pitanje kako da dohvatimo samo drugi po redu p element ? Pošto getElementsByTagName sam govori da dohvata skup elemenata tj niz elemenata moramo da pristupimo drugom p elementu koristeći **indeks** koji nam govori da je u pitanju drugi p element u nizu. Elementima u nizu pristupamo tako što krenemo od **0** koji je prvi element u nizu ,zatim **1** koji je drugi element u nizu itd. Dakle u skriptu sa

```
document.getElementsByTagName("p")[1].innerHTML;
```

dohvatamo HTML sadržaj drugog p elementa na našoj HTML stranici .

Dalji postupak je isti kao u prethodnom primeru . Na klik dugmeta koristeći događaj **onClick** pozivamo funkciju **prikaziIOfarbaj()** koja će ispisati tekst u alert poruci. Primetimo da ova funkcija radi još nešto . Nakon ispisa poruke ponovo pozivamo isti p element, čiji smo HTML ispisali, želimo da istaknemo ovaj tekst drugom bojom. Ovde smo iskoristili drugi properti po imenu **style** i definisali smo CSS properti koji želimo da promenimo. U našem slučaju je to boja fonta pa koristimo **color** properti i menjamo u crvenu boju. Vrednost color propertija navodimo desno od znaka jednakosti u jednostrukim ili dvostrukim navodnicima na sledeći način

```
document.getElementsByTagName("p")[1].style.color = "red";
```

Ovaj kod smo mogli i da pojednostavimo tako što smo ceo objekat tj p element mogli da sačuvamo u promenljivu tekstZaPrikaz i da selektivno ispišemo tekst u alert poruci koristeći innerHTML properti ,a zatim da selektivno promenimo boju teksta u crvenu koristeći style.color properti

Pogledajmo kako to možemo da uradimo :

Primer 3

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Čas 5 Primer 3</title>
</head>
<body>
<h1>Čas 5 Primer 3 - document.getElementByTagName - verzija
2</h1>

<p>Ovaj tekst ne treba da bude prikazan</p>
<p>Ovaj tekst treba da se prikaže u pop-up ekranu i da nakon
toga promeni boju u crveno</p>

<input type="button" value="Prikaži tekst iz drugog p elementa"
onclick="prikaziIOfarbaj()">

<script>
function prikaziIOfarbaj() {
    var tekstZaPrikaz = document.getElementsByTagName("p")[1];
    alert(tekstZaPrikaz.innerHTML);
    tekstZaPrikaz.style.color = "red";
}
</script>
</body>
</html>
```

document.getElementsByClassName

Ovom metodom možemo da pristupimo HTML elementima ukoliko imaju definisan atribut class. Ova metoda kao i prethodna pretražuje celo DOM stablo tj. ceo HTML dokument i dohvata sve elemente koji imaju zadatu vrednost class atribut. To znači da ova metoda može da vrati i više od jednog elementa koji se čuvaju u nizu. Zbog toga treba voditi računa kojem elementu po redu želimo da pristupimo.

Pogledajmo sledeći primer :

Primer 4

```
<!DOCTYPE html>
<html lang="en">
  <head>
  <meta charset="UTF-8">
  <title>Čas 5 Primer 4</title>
</head>
<body>
  <h1>Čas 5 Primer 4 - document.getElementsByClassName </h1>
```

```
  <div class="prvi">Želim da se ovaj tekst pojavi još
     jednom</div>
  <div class="drugi"></div>
  <div class="drugi">Ovde ne želim da se prikaže ništa</div>

  <input type="button" value="Prikaži tekst iz prvog div
     elementa u drugi div" onclick="prikaziUDrugiDiv()">
  <script>
     function prikaziUDrugiDiv() {
          var tekstZaPrikaz =
      document.getElementsByClassName("prvi")[0];
      if (tekstZaPrikaz.getAttribute("class")==="prvi"){
        document.getElementsByClassName("drugi")[0].innerHTML =
          tekstZaPrikaz.innerHTML;
      }
      tekstZaPrikaz.style.color = "red";
     }
  </script>
</body>
</html>
```

Objasnimo prethodni primer. Kao što vidimo iz primera imamo definisane div elemente koji sadrže atribute class i svaki od njih sadrži naziv. Kako bi mogli da pristupimo ovim elementima u okviru DOM stabla tj. HTML stranice koristimo metod getElementsByClassName sa kojim možemo da pristupimo elementima koji sadrže class atribut.

Mi želimo da pristiskom na dugme “Prikaži tekst iz prvog diva u drugi div” prebacimo tekst iz elementa div sa class atributom prvi u div element sa class atributom drugi ali u prvi po redu u nizu elemenata sa class atributom po imenu drugi. Podsetimo se da ukoliko imamo niz tada se elementima pristupa preko indeksa počevši brojanje od 0 pa nadalje.

To znači da će nakon pritiska dugmeta da se aktivira funckija prikaziUDrugiDIv koja će u lokalnu promenljivu tekstZaPrikaz smestiti ceo element sa class atributom po imenu prvi, zatim će se ispitati da li je u promenljivoj stvarno element sa atributom class koji ima vrednost prvi . Primetite da ovde koristimo properti getAttribute sa kojim dohvatamo atribute u okviru nekog elementa , u našem slučaju atribut po imenu class.

Ako je stvarno atribut class elementa div sa vrednošću prvi tada ga smeštamo u div sa atributom class sa vrednošću drugi ali u prvi element po nizu pa moramo da koristimo 0 indeks za pristup ovom elementu.

Na kraju smo samo promenuli boju div elementu sa classa atributom sa vrednošću prvi

document.querySelectorAll i document.querySelector

Sledeće metod koji ćemo da razmatramo je **document.querySelectorAll** i **document.querySelector**. Ovi metodi koriste kao parametar isključivo CSS selektore što znači da ukoliko želimo da pristupimo svim elementima u DOM stablu koristeći atribut **class** u elementima, to radimo koristeći CSS selektor za klase na sledeći način:

```
document.querySelectorAll(".imeClassAtributa");
```

Ako želimo da pristupimo svim elementima u DOM stablu koristeći atribut **id** u elementu, to radimo koristeći CSS selektor za id na sledeći način:

```
document.querySelectorAll("#imeIDAtributa");
```

Takođe možemo kao parametre ove metode da koristimo razne kombinacije koristeći CSS2 I CSS3 selektore. Napominjem da ovaj metod nije podržan kod pretraživača Internet Explorer verzije manje od 9.

Metod **document.querySelector()** radi na potpuno isti način sa jednom razlikom , a to je što vraća samo prvi element niza pronađen po zadatom CSS selektoru u okviru DOM stabla.

Pogledajmo sledeći primer:

Primer 5

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Čas 5 Primer 5</title>
</head>
<body>
<h1>Čas 5 Primer 5 - document.querySelectorAll() </h1>

<div class="prvi">Želim da se ovaj tekst pojavi još
jednom</div>
<div class="drugi"></div>
<div class="drugi">Ovde ne želim da se prikaže ništa</div>

<input type="button" value="Prikaži tekst iz prvog div elementa
u drugi div" onclick="prikaziUDrugiDiv()">

 <script>
  function prikaziUDrugiDiv() {
    var tekstZaPrikaz = document.querySelector(".prvi");
    if (tekstZaPrikaz.getAttribute("class")==="prvi"){
      document.querySelectorAll(".drugi")[0].innerHTML =
     tekstZaPrikaz.innerHTML;
    }
    tekstZaPrikaz.setAttribute("style","color:red");
  }
```

```
</script>
</body>
</html>
```

Primetimo da je gornji primer isti kao prethodni primer 4 ,s tim što smo koristili metode document.querySelectorAll i document.querySelector.

Sa metodom **document.querySelector** pristupimo elementu div sa klasom prvi jer postoji samo jedan element te klase pa nema potrebe da koristimo metod document.querySelectorAll.

Takođe primetimo da smo u poslednjem redu funkcije prikaziUDrugiDiv koristili properti **setAttribute** sa kojim smo podesili atribut **style** da nam prikaže crveni tekst elementa div klase prvi.

Pristupanje HTML elementima koristeći ime forme

Za interakciju sa korisnikom izuzetno je važno preuzimanje vrednosti elemenata forme u okviru HTML stranice. Ovaj način pristupanja elementima HTML forme je bio aktuelan ranije ali se i danas dosta često koristi. Vrednost parametara forme preuzimamo sledećom sintaksom

```
document.naziv_forme.naziv_elementa.value
```

Pogledajmo primenu kroz sledeći primer

Primer 6

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Čas 5 Primer 6</title>
  <script>
    function prikaziVrednost(){
     var a = document.forma.ime.value;
     alert(a);
    }
  </script>

</head>
<body>
 <h1>Čas 5 Primer 6 - Preuzimanje vrednosti parametra
  forme i ispis na onMouseOut događaj</h1>
 <form id="idForma" name="forma">
 <input id="input1" type="text" name="ime" value="Pređi
  mišem preko input polja" onMouseOut=" prikaziVrednost
()">

 </form>
</body>
</html>
```

U gornjem primeru smo videli kako možemo da upotrebimo atribut name HTML elemenata za pristup određenom elementu u okviru DOM stabla.

Ostali BOM objekti

Pored već objašnjenih objekata window i document u BOM postoje još nekoliko objekata , a to su navigator, screen, location i console, history .

Ukratko navodimo šta koji radi u okviru BOM-a kroz kratke primere

Navigator

Navigator objekat predstavlja sam web pretraživač i kroz svoje metode nudi uglavnom informacije o web pretraživaču, operativnom sistemu na kojem se pokreće browser, da li kolačići omogućeni u okviru pretražavača, lokaciju trenutno korišćenog pretraživača i sl. Pošto detaljan prikaz rada svih propertija prevazilazi opseg ovog kursa prikazaćemo samo jedan od njih , a to je prikaz verzije pretraživača sa pratećim podacima.

Primer 7

<!DOCTYPE html>

```
<html lang="en">
<head>
<meta charset="utf-8">
<title>Čas 5 Primer 7</title>
</head>
<body>
<h1>Čas 5 Primer 7 - Navigator objekat</h1>
<p>Kliknite na dugme da se prikaže verzija vašeg browsera.</p>

<button onclick="prikaziPodatke()">Try it</button>

<p id="podaci"></p>

<script>
function prikaziPodatke() {
    var x = "Version info: " + navigator.appVersion;
    document.getElementById("podaci").innerHTML = x;
}
</script>
</body>
</html>
```

Primetimo u gornjem primeru da smo primenili specifičnost kod korišćenja propertija **innerHTML** ,a to je da ukoliko se nalazi na levoj strani znaka jednakosti vrši se izmena HTML nad definisanim elementom. U našem slučaju vrednost HTML p elementa će dobiti vrednost promenljive **x** tj. u p elementu će se ispisti

sadržaj promenljive x koja čuva sve podatke koje je dobila od propertija **navigator.appVersion**.

Screen

Screen objekat predstavlja ekran uređaja na kojem se prikazuje se prikazuje HTML stranica. Sadrži nekoliko propertija koji vraćaju širini ,visinu ekrana, broj bitova kojom se prikazuje boja itd.

Location

Ovaj objekat može da se koristi za dobijanje trenutne adrese ili URL I za redirekciju pretraživača na novu stranicu.

Location metod ovog objekta koristi se za zadavanje URL - a stranice koja se prikazuje na stranici čime se omogućava promena stranice što je efekat identičan onom koji se dobija klikom na klasični HTML link.

Location objekat sadrži metode i propetije . Navešćemo korisne metode

- ***location.reload()*** – ponovo učitava trenutnu stranicu ili popularno nazvano refresh – osvežavanje stranice
- ***location.replace()*** – učitava novi URL sa novom istorijom

Ovaj objekat sadrži nekoliko propertija(osobina) :

- ***location.href*** – vraća href (URL) trenutne stranice
- ***location.hostname*** – vraća ime domena web stranice
- ***location.pathname*** – vraća putanju I ime fajla trenutne stranice
- ***location.protocol*** – vraća web protokol koji se koristi (HTTP ili HTTPS)
- ***location.assign*** – učitava novi dokument

Pogledajmo kroz primer kako se koriste ovi propertiji (osobine) location objekta

Primer 8

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Čas 5 Primer 8</title>
<meta charset="utf-8">
<script>
   function vratiURL (){
    document.getElementById("demo1").innerHTML =
        "Lokacija stranice ili URL je: " + window.location.href;
   }
function vratiHostName (){
    document.getElementById("demo2").innerHTML =
        "Ime web host domena  je: " + window.location.hostname;
```

```
  }
  function uradiAssign() {
    window.location.assign("https://www.google.com");
  }
</script>
</head>
<body>
  <h1>Čas 5 Primer 8 - LOCATION objekat - DODATAK</h1>
  <p><a href="#" onclick="vratiURL()">window.location.href</a> vraca href ili
       URL trenutne stranice npr.
       https://www.google.com/index.html?q=name</p>
  <p id="demo1"></p>
  <p><a href="#" onclick="vratiHostName()">window.location.hostname</a>
       vraća ime domena web hosta npr. www.google.com</p>
  <p id="demo2"></p>
  <p>window.location.pathname vraća path i ime fajla trenutne stranice npr
       /index.html</p>
  <p id="demo3"></p>
  <p>window.location.protocol vraća web protokol (http:// or https://)</p>
  <p id="demo4"></p>
  <p><a href="#" onclick="uradiAssign()">window.location.assign</a> učitava
       novi dokument npr (www.google.com)</p>
  <p id="demo5"></p>
</body>
</html>
```

Postojanje ovog metoda objasnićemo i kroz praktičan primer kada se prave link padajući meniji koji mogu sadržavati veliki broj opcija, a smeštanjem u SELECT elemenat formulara, tj. u njegove OPTION elemente štedi prostor.

Primer 9

```
<!DOCTYPE html>
<html lang="en">
<head>
<html>
<title>Čas 5 Primer 9</title>
  <meta charset="utf-8">
  <script language="JScript">
    function odvedi() {
      //pristupamo padajućoj listi
      var element = document.formular.veze;
      //izdvajamo selektovanu opciju tj njegov indeks
       var ind = element.selectedIndex;
      //izdvajamo vrednost selekovane opcije iz liste
      var lokacija = element.options[ind].value;
     document.location = lokacija;
      //skracena verzija
      //var element = document.getElementById("veze");
```

```
      //var lokacija =
      element.options[element.selectedIndex].value;
      //document.location = lokacija;
    }
</script>
<body>
 <h1>Čas 5 Primer 9 - LOCATION objekat</h1>
 <form name="formular">
 <select name="veze" id="veze" size="1"
     onChange="odvedi()">
  <option value="#" selected>------
  <option value="http://www.yahoo.com">Yahoo
  <option value="http://www.msn.com">MSN
  <option value="http://www.hotmail.com">HotMail
 </select>
</form>
</body>

</html>
```

U formi smo definisali select listu u kojoj smo naveli linkove poznatih prezentacija na internetu i jedno polje koje ne vodi nigde. Biranjem odgovarajuće vrednosti aktivira se događaj **onChange** koji poziva funkciju **odvedi()** koja, opet, vodi na odgovarajući link naveden u **value** atributu odgovarajućeg polja uz pomoć **location** metoda objekta document.

Console

Ovaj objekat služi za programski pristup developer konzoli. Ovaj objekat omogućava ispis više tipova poruka u konzoli. Koristi se za brzo debugovanje ili otkrivanje grešaka ili testiranje rada programskog koda napisanog u JavaScript jeziku.

Da bi videli rezultat ispisa u konzoli koristi se **log** metoda ovog objekta. Da bi u pretraživaču videli rezultat ispisa koristi se dugme F12 sa kojim otvaramo developer konzolu u svim pretraživačima. Zatim odaberemo tab prozora koji se zove console i tu se ispisuju poruke koje smo prosledili kroz **console.log** metodu.

Pogledajmo primer

Primer 10

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Čas 5 Primer 10</title>
  <script>
    console.log("Ovaj tekst se ispisuje u konzolnom prozoru");
  </script>
</head>
<body>
      <h1>Čas 5 Primer 10 - Console objekat</h1>
```

```
      <h2>Pokrenite developer konzolu sa F12 dugmetom na
      tastaturi</h2>

      <p>Selektujte "Console" tab u developer konzoli i
      pritisnite osvežite stranicu.</p>
</body>
</html>
```

History

Kao što mu i samo ime govori ovaj objekat se koristi za manipulaciju sa istorijom pretraživanja.Najčešće metode ovog objekta su

- ***history.back()*** – učitava prethodnu stranicu iz liste
- ***history.forward()*** – učitava sledeću stranicu iz liste
- ***history.length*** – vraća količinu stranica u istoriji
- ***history.go(index)*** – učitava URL na zadanom indeksu u listi

Pogledajmo primer kako history objekat radi.

Primer 11

```
<!DOCTYPE html>
<html lang="en">
 <head>
 <meta charset="utf-8">
<title>Čas 5 Primer 11</title>
 <script>
    var u_istoriji = history.length;
    console.log("Istorija sadrži "+u_istoriji+" stranice");
    function idiNazad() {
      window.history.back();
    }
    function uradiAssign() {
        window.location.assign("https://www.google.com");
    }
 </script>
</head>
<body>
 <h1>Čas 5 Primer 11 - History objekat</h1>
 <p><a href="#" onclick="uradiAssign()">Pritisni me da se
     napuni history</a> učitava  novi dokument npr
     (www.google.com)
 </p>
 <input type="button" value="Vrati se na prethodnu stranicu"
     onclick="idiNazad()">
</body>
</html>
```

U primeru smo prikazali kako metode objekta history rade. Ako prvo sa linkom **“Pritisni me da se napuni history”** pozivamo funkciju **uradiAssign()** koja otvori novu stranicu I prikaže google.com . To znači da smo u memoriju upisali history objekat i definisali novu stranicu .

Taada u konzoli vidimo da se nakon osvežavanja stranice doda novi element u istoriju. Sada ako pritisnemo dugme **“Vrati se na prethodnu stranicu”** pozivamo funkciju **idiNazad()** koja pozive history metodu **back()** i iz istorije se vrati stranica koja je prethodno bilo učitana.

Čas 6

Tema časa:

Globalni JavaScript objekti

parseFloat i parseInt

Logičke operacije u izrazima

For petlja

While i do while strukture

Očitavanje pretraživača (browser)

Globalni JavaScript objekti

Ovi objekti pripadaju takođe BOM-u i zovu se još I built-in objekti ili ugrašeni objekti. Oni omogućavaju naprednija rukovanja numeričkim, tekstualnim, datumskim i matematičkim tipovima podataka i proširuju JavaScript jezik dodatnim funkcijama i njihovim funkcionalnostima. To su sledeći objekti :

- Date
- Number
- String
- Math

Date

U JavaSciptu moguće je očitati datum i vreme sa klijentovog računara i te vrednosti koristiti u JavaScript kodu. Jedna od mogućnosti je da se u zavisnosti od dana, na primer, strana prikazuje na različite načine čime može dodatno da animira posetioce. Očitavanje datuma i vremena radi se pomoću objekta **Date** u JavaScript - u. Objekat zamislite kao jednu crnu kutiju koja je puna informacija a do njih možete doći samo ako ih pozovete na pravi način. Objekat Date zamislite kao nekog ko zna sve podatke o vremenu. Da bi ste radili sa datumima prvo morate da pozovete objekat ,a to se radi saključnom reči **new**. Metode objekta pozivate tako što pored naziva promenljive koja instancira tj. poziva objekat dodate tačku i naziv metode koja vam je potrebna.

NAPOMENA: Ukoliko se izostavi ključna reč new u referenci kasniji skriptovi se neće izvršiti iako ovaj hoće.

Objekat **Date** uzima snimak internog sata sa računara i vraća datumski objekat za taj period. Prava vrednost ovog objekta u nekom momentu je vreme u milisekundama počev od nula časova 1. januara 1970. godine po Griniču (GMT) - svetskoj referentnoj tački za sve vremenske konverzije. Tako ovaj objekat sadrži informacije i o datumu i o vremenu.

Metodi objekta **Date** koriste se za izdvajanje pojedinih podataka iz ove grupe. Pogledajmo koje metode postoje u objektu Date()

- getTime(), opseg 0 - ... počev od pomenute tačke u vremenu 1970.
- getYear(), opseg 70 - ... godina minus 1900, četiri cifre
- getFullYear() – vraća trenutnu godinu npr. 2017
- getMonth(), ospeg 0 - 11, meseci u toku godine (januar = 0)
- getDate(), opseg 1 - 31, datum u toku meseca
- getDay(), opseg 0 - 6, dani u nedelji (nedelja = 0)
- getHours(), opseg 0 - 23, sati u danu
- getMinutes(), opseg 0 - 59, minuti u satu
- getSeconds(), opseg 0 - 59, sekunde u minuti

Pogledajmo primer kako koristiti objekat Date I njegove metode:

Primer 1

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Čas 6 Primer 1</title>
  <script>
    function danas() {
      var danas = new Date;
      var time = danas.getTime();
      var dan = danas.getDay();
      var danUMesecu = danas.getDate()
      var mesec = danas.getMonth()+1;

      //ne koristi se vise ni u IE
      var godina = danas.getYear();

      //ovo resava problem sa godinom
      var godina1 = danas.getFullYear();

      var sati = danas.getHours();
      var minuta = danas.getMinutes();
      var sekundi = danas.getSeconds();

      var datum = danUMesecu+"/"+mesec+"/"+godina1;
      document.formular.datum.value = datum;
      alert(" Dan u nedelji je "+dan+", a dan u mesecu je
     "+danUMesecu+" mesec je "+ mesec+" godina je "+ godina+".
     Trenutno je "+sati+":"+minuta+":"+sekundi+" sati "+time);
    }
  </script>
  </head>
  <body>
```

```
    <h1>Čas 6 Primer 1 - DATE objekat</h1>
    <form name="formular">
       Dan/mesec/godina <br>
    <input type="button" name="film" value="Klik"
     onClick="danas()">
    <input type="text" name="datum">
 </form>
</body>
</html>
```

Preporuka je da se koristi metod getFullYear() umesto getYear, koji u oba pretraživača daje isti četvorocifreni rezultat. Novi metod čitanja godine uveden je zbog čitanja godina posle 2000 godine.

Number

Number objekat je još jedan built-in ili ugrađeni objekat koji je globalno dostupan. JavaScript kao što smo i rekli poznaje samo jedan tip numerika i to je number. Ovaj objekat pored nekoliko propertija (osobina) sadrži i metode koje vrše provere da li je cifra numerik , metode za razne konverzije , metode za zaokruživanje na decimalu I slično.

Pogledajmo funkcije objekta Number:

- **isNan(vrednost)** – proverava da li prosleđena vrednost nije broj
- **toFixed(broj_decimala)** – zaokružuje decimalni broj na prosleđeni broj decimala (vraća string)
- **toPrecision(broj_cifara)** – zaokružuje broj na prosleđeni broj cifara

String

String objekat sadrži veoma korisne metode i propertije za manipulaciju stringovnim vrednostima. Najpoznatiji string properti je **length** koji vraća broj karaktera u stringu.

Pogledajmo primer:

```
var str = "Hello World!";
var n = str.length;
//Rezultat  n je 12
```

String objekat ima veliki broj metoda za rad sa tringovima. Navodimo neke najčešće korišćene:

- **replace(staraVrednost,novaVrednost)** – pretražuje se string koji odgovara vrednosti zadatu u prvom parametru i zamenjuje je sa vrednošću navedenom u drugom parametru

```
var str = "Visit Microsoft!";
var res = str.replace("Microsoft", "Smart");
 //rezultat je  Visit Smart
```

- **substr(pozicija,dužina)** – izvlači karaktere od zadate pozicije pa dužinu zadatih karaktera. Ako se ne navede dužina izvlače se svi karakteri od zadate pozicije pa do kraja stringa

```
var str = "Hello world!";
var res = str.substr(1, 4);
//rezultat je ello
```

- **toLowerCase(), toUpperCase**() – pretvara ceo zadati string u mala odnosno velika slova

```
var str = "Hello World!";
var res = str.toLowerCase();
//rezultat je hello world
```

- **split()** – deli string u niz podstringova

```
var str = "Kako si danas?";
var res = str.split(" ");
//rezultat je Kako,si,danas?
```

- **charAt()** – vraća karakter u stringu na zadatoj poziciju

```
var str = "SMART";
var res = str.charAt(0);
//rezultat je S
```

- **concat()** – spaja dva stringa

```
var str1 = "Hello ";
var str2 = "world!";
var res = str1.concat(str2);
//rezultat je Hello world!
```

- **indexOf()** – vraća poziciju zadatog karaktera ili stringa

```
var str = "Hello world, welcome to the universe.";
var n = str.indexOf("welcome");
//rezultat je 13
```

parseFloat i parseInt

Ove dve funkcije su tzv. top level funkcije ili globalne funkcije koje ne pripadaju ni jednom objektu.

Ove dve funkcije imaju za zadatak da iz nekog stringa koji je zadat kao njihov parametar, izvuku broj sa prvih mesta u tom stringu i prikažu ga u odgovarajućem formatu. Pa tako:

- **parseFloat(string)** - uzima string argument i vraća broj sa decimalnim zarezom

  Evo nekoliko primera :

```
var a = parseFloat("10")// 10
var b = parseFloat("10.00")// 10
var c = parseFloat("10.33") // 10.33
var d = parseFloat("34 45 66")// 34
var e = parseFloat(" 60 ") // 60
var f = parseFloat("40 kvadrata") // 40
var g = parseFloat("Stan ima 40 kvadrata") // NaN
```

- **parseInt(string,radix)** – uzima string argument i vraća broj (ne zaokružuje već vraća samo broj ispred decimalnog zareza). Ako

koristimo drugi parametar radix on nam predstavlja osnovu rezultata (npr 10 rezultat će biti pretvoren u decimalu, ako koristimo 8 broj će biti pretvoren u oktalan rezultat, ako je 16 broj će biti pretvoren u heksadecimalan ispis , po defaultu je decimalan ispis)

Evo nekoliko primera:

```
var a = parseInt("10")//
var b = parseInt("10.00")//
var c = parseInt("10.33")//
var d = parseInt("34 45 66")//
var e = parseInt(" 60 ")//
var f = parseInt("40 kvadrata")//
var g = parseInt("Stan ima 40 kvadrata")//

var h = parseInt("10", 10)//
var i = parseInt("010")//
var j = parseInt("10", 8)//
var k = parseInt("0x10")//
var l = parseInt("10", 16)//
```

Pogledajmo ove funkcije kroz primer

Primer 2

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Čas 6 Primer 2</title>
  <script>
    //parseFloat parsira string u vraca broj sa decimalnim
     zarezom
    //odredjuje da li je prvi karakter broj, i parsira ga do
     kraja broja
    //parseFloat vraca samo prvi broj,ako je prvi karakter
     string
    //i ne moze se prevesti u number tada vraca NaN
    var a = parseFloat("10") + " ";
    var b = parseFloat("10.00") + " ";
    var c = parseFloat("10.33") + " ";
    var d = parseFloat("34 45 66") + " ";
    var e = parseFloat("   60   ") + " ";
    var f = parseFloat("40 kvadrata") + " ";
    var g = parseFloat("Stan ima 40 kvadrata") + " ";
    var n = a + b + c + d + e + f + g;
    alert(n);
    //parseInt
    var a = parseInt("10") + " ";
    var b = parseInt("10.00") + " ";
    var c = parseInt("10.33") + " ";
    var d = parseInt("34 45 66") + " ";
    var e = parseInt(" 60 ") + " ";
    var f = parseInt("40 kvadrata") + " ";
    var g = parseInt("Stan ima 40 kvadrata") + " ";
```

```
    var h = parseInt("10", 10) + " ";
    var i = parseInt("010") + " ";
    var j = parseInt("10", 8) + " ";
    var k = parseInt("0x10") + " ";
    var l = parseInt("10", 16) + " ";
    var n1 = a + b + c + d + e + f + g;
    var n2 = h + i + j + k + l;

    alert(n1);
    alert(n2);
  </script>
</head>

<body>
  <h1>Čas 6 Primer 2 - PARSEFLOAT i PARSEINTEGER</h1> </body>

</html>
```

U sledećem primeru ćemo da saberemo vrednost dva elementa forme. Napravićemo jednostavan digitron.

Primer 3

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
<script>
  function zbir(){
    var a=document.forma.prvi.value;
    var b=document.forma.drugi.value;
    alert(a+b);
  }
</script>
<title>Čas 6 Primer 3</title>
</head>
<body>
<h2>Čas 6 Primer 3 - Preuzimanje vrednosti parametra forme
konkatenacija i ispis </h2>
<form name="forma">
  <input type="text" name="prvi">
  <input type="text" name="drugi">
  <input type=button value="Saberi" onClick="zbir()">
</form>
</body>
</html>
```

Ako u prvo polje unesemo vrednost 1, a u drugo 5 dobićemo rezultat 15 što nijejednako 1+5. Zašto? Zato što vrednosti koje preuzme iz polja forme JS posmatra kao Stringove (nizove znakova ili dve reči). Da bi ih zaista sabrao

moramo mu reći da on prebaci u celobrojne vrednosti. To se radi sa metodom **parseFloat** kao u sledećem primeru.

Primer 4

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Čas 6 Primer 4</title>
  <script>
   function zbir(){
   var a=document.getElementsByTagName("input")[0].value;
   var b=document.getElementsByTagName("input")[1].value;
     alert(parseFloat(a)+parseFloat(b));
   }
  </script>
</head>
<body>
 <h3>Čas 6 Primer 4 - Preuzimanje vrednosti parametra
     forme,zbir brojeva i ispis</h3>
 <form name="forma">
  <input type="text" name="prvi">
  <input type="text" name="drugi">
  <input type=button value="Saberi" onClick="zbir()">
 </form>
</body>
</html>
```

Mi možemo rezulatat da ne prikazujemo u alertu već da ga u običnom digitronu ispišemo u nekom elementu forme. U sledećem primeru se u polje rezultat ispisuje vrednost zbira.

Primer 5

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Čas 6 Primer 5</title>
<script>
  function zbir(){
    var a=document.getElementsByTagName("input")[0].value;
    var b=document.getElementsByTagName("input")[1].value;
    var c=parseFloat(a)+parseFloat(b);
    document.getElementsByTagName("input")[2].value=c;
}
</script>
</head>
<body>
  <h1>Čas 6 Primer 5 - Preuzimanje vrednosti parametra
forme, zbir brojeva i dodeljivanje vrednosti parametru
forme i ispis na onMouseOut događaj</h1>
```

```
    <form name=forma>
      <label for="prvi">Prvi broj</label>
      <input type="text" name="prvi"><br>
      <label for="drugi">Drugi broj</label>
      <input type="text" name="drugi"><br>
      <label for="rezultat">Rezultat</label>
      <input type="text" name="rezultat">
      <input type=button value="Saberi" onClick="zbir()">
    </form>
    </body>
    </html>
```

Sada ćemo da napravimo još dva polja jedno će se zvati manje a drugo veće.
U prva dva polja ćemo upisivati brojeve ako zbir manji od 10 rezultat ćemo upisivati u polje koje se zove mane a ako je veći od 10 u drugo koje se zove veće.

Primer 6

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Čas 6 Primer 6</title>
<script>
  function zbir(){
    var a=document.getElementsByTagName("input")[0].value;
    var b=document.getElementsByTagName("input")[1].value;
    var c=parseFloat(a)+parseFloat(b);
    if(c<10){
      document.getElementsByTagName("input")[2].value=c;
      //ocisti drugo polje
      // document.getElementsByTagName("input")[2].value="";
    } else {
      document.getElementsByTagName("input")[3].value=c;
      //ocisti drugo polje
      // document.getElementsByTagName("input")[3].value="";
    }
  }
  //BONUS
  function izmeniHTML() {
    alert(document.getElementById("demo").innerHTML);
    document.getElementById("demo").innerHTML = "Izmenio sam
tekst u paragrafu 'demo'!";
  }
</script>
</head>
<body>
  <h1>Čas 6 Primer 6 - Preuzimanje vrednosti parametra forme
     sabiranje provera uslova i upis u odgovarajuće polje</h1>
  <form name="forma">
   <label for="prvi">Prvi broj</label>
   <input type="text" name="prvi"><br>
   <label for="drugi">Drugi broj</label>
   <input type="text" name="drugi"><br>
   <label for="manji">Manji</label>
   <input type="text" name="manji"><br>
```

```
   <label for="veci">Veci</label>
   <input type="text" name="veci"><br>
   <input type=button value="Saberi" onClick="zbir()">
 </form><br>
  BONUS
  <p  id="demo"  onclick="izmeniHTML()">Klikni  me  i  promeni  mi
     HTML sadržaj (innerHTML).</p>
</body>
</html>
```

Logičke operacije u izrazima

Do sada smo koristili naredbu if za grananje sada ćemo da proširimo naše poznavanje ove metode upotrebom još nekih operatora koje do sada nismo koristili.

Uslovi u if naredbi mogu biti zadati pomoću jednostavnih operacija poređenja ili uz pomoć kompleksnih logičkih izraza.
Kao što je poznato iz matematičke logike svi logički izrazi bez obzira na kompleksnost mogu kao rezultat imati samo dve vrednosti: true ili false.
Operatori koji se koriste u ovim izrazima su:

- == - označava poređenje "jednako je sa"
- != - označava poređenje "nije jednako sa"
- > - označava poređenje "veće je od"
- >= - označava poređenje "veće je ili jednako sa"
- < - označava poređenje "manje je od"
- <= - označava poređenje "manje je ili jednako sa"

Logički izrazi koji se mogu koristiti u zadavanju uslova if ... else konstrukcija konstruišu se od više jednostavnih logičkih operacija poređenja međusobno povezanih tako da daju izraz čiji rezultat može biti vrednost true ili false.

Logičke operacije koje nam stoje na raspolaganju su:

- && - veza "i", vraća vrednost true ukoliko su oba podizraza koja čine kompleksan logički izraz tačni;
- || - veza "ili", vraća vrednost true ukoliko je barem jedan podizraz tačan;
- ! - "ne", stavlja se ispred izraza ukoliko želimo da tačan izraz proglasimo netačnim, a netačan tačnim;

Primer 7

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Čas 6 Primer 7</title>
  <meta charset="utf-8">
  <script>
    function poredenje() {
      var danas = new Date;
      var dan = danas.getDay();
      var mesec = danas.getMonth() + 1;
      var godina = danas.getYear();
      //probati par primera samostalno
      if (dan == 1 && mesec == 2) {
        alert("Danas je ponedeljak februar");
      }else {
        alert("Danas je neki drugi dan!")
      }
    }
  </script>
</head>
  <body>
    <h1>Čas 6 Primer 7 - Logičke operacije i operatori -
     dva uslova</h1>
    <form name="formular">
      <input type="button" name="film" value="Klik"
     onClick="poredenje()"> </form>
  </body>
</html>
```

Primeri kompleksnih logičkih izraza:

`(b > 2 && b < 6)` – tačan je ukoliko je b veći od dva, a manji od šest

`(b == 1 || b == 3)` – tačan je ako je b 1 ili 3

`(b == 1 || (b > 2 && b < 6))` – izraz je tačan ukoliko je b jednako 1 ili ukoliko je veći od 2 i manji od 6

`(!(b > 2 && b < 6) || b == 3)` – izraz je tačan ako je b 3 ili nije veće od 2 i manje od 6

U prvom primeru su data dva uslova a može ih zadati više i mogu biti povezani različitim operatorima. U sledećem primeru imamo tri uslova.

Primer 8

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Čas 6 Primer 8</title>
  <meta charset="utf-8">
  <script>
    function poredenje() {
      var danas = new Date;
      var dan = danas.getDay();
      var mesec = danas.getMonth() + 1;
      var godina = danas.getFullYear();
      if (dan == 1 && mesec == 2 && godina == 2016) {
        alert("Danas je ponedelja februar 2016 godine");
      }else{
        alert("Neki drugi datum je danas");
      }
    }
  </script>
</head>
<body>
  <h1>Čas 6 Primer 8 - Logičke operacije i operatori - tri
uslova</h1>
  <form name="formular">
    <input type="button" name="film" value="Klik"
onClick="poredenje()"> </form>
</body>

</html>
```

FOR petlja

Sledeća kontrolna struktura koju ćemo upoznati je **for** petlja.

Petlje u programiranju služe za slučajeve kada želimo da se određena radnja izvrši tačno određeni broj puta. Na primer, možda će vaš skript morati da pregleda unos klijenta u polje za tekst formulara da bi se uverio nije li svaki uneseni znak broj ili slovo ili da nije prazno mesto i sl. Na početku petlje postavlja se uslov i po njegovom ispunjenu petlja se završava i nastavlja sa čitanjem koda.

U for petlji u uslovu zadajemo koliko će se tačno puta petlja ponoviti pre nego što se iz nje izađe. Često je potrebno znati koliko se puta petlja ponovila u određenom momentu, što se rešava postavljanjem tzv. indeksa petlje, odnosno, promenljive koja se posebno defi niše i čija se vrednost menja sa svakim izvršavanjem petlje.
Sintaksa ove petlje je:

```
for ([početni izraz]; [uslov]; [iteracija ]){
     naredbe
}
```

Postavljanje indeksa petlje vrši se na sledeći način:

Primer 9

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Čas 6 Primer 9</title>
  <script>
        function petlja() {
      //pocetni izraz; uslov; iteracija
      for (i = 0; i < 5; i++) {
        console.log(i);
      }
    }
  </script>
</head>
<body>
  <h1>Čas 6 Primer 9 - FOR petlja</h1>
  <form name="formular">
    <input type="button" name="film" value="Klik"
onClick="petlja()">
  </form>
  <p id="rez"></p>
</body>
</html>
```

U prikazanom slučaju petlja će se izvršiti pet puta, jer postavljeni uslov ne dozvoljava da promenljiva i dobije vrednost 5, a obnavljajući izraz progresivno za jedan uvećava vrednost promenljive i za svaki prolazak kroz nju. Svaki put kada se petlja ponovi vrednost indeksa je drugačija. Rezultat svakog prolaza kroz petlju se ispisuje u konzolu .

Ako bi hteli da prekinemo izvršenje petlje u nekom od prolaza kroz nju i da iskočimo iz for petlje, koristimo ključnu reč **break**

Primer 10

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Čas 6 Primer 10</title>
  <script>
    function petlja() {

      for (i = 0; i < 5; i++) {
        if(i==3){
          break;
        }
        console.log(i);
      }
    }
```

```
  </script>
</head>
<body>
  <h1>Čas 6 Primer 10 - FOR petlja - break</h1>
  <form name="formular">
    <input type="button" name="film" value="Klik"
     onClick="petlja()">
  </form>
  <p id="rez"></p>
</body>
</html>
```

Ako bi hteli da prekinemo izvršenje neke od iteracija u petlji u nekom od prolaza kroz nju , koristimo ključnu reč **continue.** Tada se ne prekida izvršenje for petlje već se preskače ona iteracija koja je definisana uslovom i u kojoj se nalazi ključna reč continue.

Pogledajmo primer

Primer 11

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Čas 6 Primer 11</title>
  <script>
    function petlja() {

      for (i = 0; i < 5; i++) {
        if(i==3){
          continue;
        }
        console.log(i);
      }
    }
  </script>
</head>
<body>
  <h1>Čas 6 Primer 11 - FOR petlja - continue</h1>
  <form name="formular">
    <input type="button" name="film" value="Klik"
     onClick="petlja()">
  </form>
  <p id="rez"></p>
</body>
</html>
```

VAŽNA NAPOMENA: prilikom korišćenja for petlje u vašem kodu morate obratiti pažnju na beskonačno ponavljanje petlje iz kojeg nema povratka. Ovakve petlje se zovu I mrtve petlje I mogu da izazovu prekid rada vašeg pretraživača

Na primer bez dela koda i++ petlja će se beskonačno mnogo puta ponavljati, jer nema promene vrednosti promenljive i te nema mogućnosti da bude zadovoljen uslov za izlazak iz petlje. Na slučajeve u kojima uslov za izlazak iz petlje nikad ne može biti zadovoljen nailazi se često ukoliko se koriste određene matematičke

operacije koje nisu dovoljno osmišljene (tj. ukoliko promenljiva menja vrednost po nekoj nedovoljno osmišljenoj formuli) stoga vam savetujem da ove kontrolne strukture koristite na način na koji je prikazan u primerima tj. preko tzv. indeksa petlje.
Evo primera jedne beskonačne petlje:

Primer 12

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Čas 6 Primer 12</title>
  <script>
        function petlja() {
         for (i = 0; i < 9; i+2) {
                console.log(i);
          }
       }
  </script>
</head>
<body>
  <h1>Čas 6 Primer 12 – MRTVA FOR petlja</h1>
  <form name="formular">
    <input type="button" name="film" value="Klik"
     onClick="petlja()">
  </form>
  <p id="rez"></p>
</body>
</html>
```

WHILE i DO WHILE strukture

Struktura **while** je kontrolna struktura koja je u suštini jako slična strukturi **for**. Naredbe unutar petlje neće se izvršiti ukoliko pri prvom prolasku uslov nije ispunjen. Kroz petlju će se ponovo prolaziti sve dok je zadati uslov ispunjen.
Sintaksa ove petlje je:

```
while (uslov) {
   niz naredbi
}
```

Pogledajmo jednostavan primer ove petlje

Primer 13

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Čas 6 Primer 13</title>
  <meta charset="utf-8">
  <script>
```

```
    function petlja_dva() {
      i = 0;
      while (i < 5) {
        alert(i);
        i++;
      }
    }
  </script>
</head>
<body>
  <h1>Čas 6 Primer 13 - WHILE petlja</h1>
  <form name="formular">
    <input type="button" name="film" value="Klik"
onClick=" petlja_dva()"> </form>
</body>
</html>
```

Slična struktura, sa jednom značajnom razlikom, je kontrolna struktura do while – naredbe unutar ove strukture izvršiće se UVEK, bez obzira na uslov, NAJMANJE JEDNOM. To je sasvim jasno ukoliko se ima u vidu njena sintaksa:

```
do{
   naredbe
} while(uslov)
```

Primer 14

```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>Čas 6 Primer 14</title>
  <meta charset="utf-8">
  <script>
    function petlja_dva() {
      i = 1;
      do {
        alert('da li vam je dosadilo');
        i++;
      }
      while (i < 5) {
        alert(i);
      }
    }
  </script>
</head>
<body>
  <h1>Čas 6 Primer 14 - DO - WHILE petlja</h1>
  <form name="formular">
    <input type="button" name="film" value="Klik"
onClick="petlja_dva()"> </form>
</body>

</html>
```

Kod će MORATI da prođe kroz naredbe barem jednom da bi proverio uslov. Ukoliko ovaj nije ispunjen izaćiće se iz petlje. Nekad je izuzetno važno postaviti ovakvu kontrolnu strukturu.
VAŽNA NAPOMENA: prilikom korišćenja while i do while struktura u vašem kodu morate obratiti pažnju na tu da se tokom njihovog izvršavanja nešto dešava sa promenljivom ili izrazom koji koristite u uslovu, jer će u protivnom petlja zapasti u beskonačno ponavljanje iz kojeg nema povratka.

Na primer:

```
var i=1;
do{
  document.write("Red "+i+"<BR>");
  i++;
} while(i<11)
```

U primeru se vrednost promenljive i prilikom svakog prolaska kroz petlju povećava za jedan sve dok ta vrednost ne postane 10 čime je zadovoljen uslov za izlazak iz petlje. Bez dela koda i++ petlja će se beskonačno mnogo puta ponavljati, jer nema promene vrednosti promenljive i te nema mogućnosti da bude zadovoljen uslov za izlazak iz petlje. Na slučajeve u kojima uslov za izlazak iz petlje nikad ne može biti zadovoljen nailazi se često ukoliko se koriste određene matematičke operacije koje nisu dovoljno osmišljene (tj. ukoliko promenljiva menja vrednost po nekoj nedovoljno osmišljenoj formuli) stoga vam savetujem da ove kontrolne strukture koristite na način na koji je prikazan u primerima tj. preko tzv. indeksa petlje.

Očitavanje pretraživača (browsera)

Iz istorije razvoja JavaScript progamskog jezika vidi se da je jezik razvijan u dve različite "škole" - Netscape i Microsoft . Neposredna posledica je razvijanje jezika i njegovog objektnog modela u dva različita pravca. Najveću štetu od ovog sukoba ima sam jezik i, na žalost, sami programeri.
Iako je najgore što može da se dogodi da zbog postojećih razlika u objektnim modelima u različitim pretraživačima skript neće da se izvrši u nekom od njih, zahtevi klijenta mogu da budu takvi da se takve razlike premoste te da se na krajnje jednostavnom kodu napravi grananje i dva načina izvršavanja za različite tipove pretraživač što samo produžava posao.
Razlike u objektnim modelima JavaScript - a u pretraživačima su brojne i nemoguće ih je na jednom mestu spomenuti. Najbolji savet koji može da se da je da se prati dokumentacija koja se stalno menja I dopunjuje ili da se drži preporuka W3C konzorcijuma koji za cilj ima prevazilaženje ovih razlika.Preporuka sajta na kojem se mogu pratiti izmene dokumentacije vezano za JavaScript je

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

kao i

https://www.w3.org/standards/webdesign/script

Na ovom mestu, međutim, obratićemo pažnju na očitavanje koji pretraživač se nalazi kod klijenta kod kog naš kod treba da se razvije kao početak bilo kakvog angažovanja povodom prevazilaženja pomenutih razlika.
Ukucajte sledeći kod u vaš editor u HTML dokument i posmatrajte šta ćete dobiti kao rezultat u različitim verzijama pretraživača.

Primer 15

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Čas 6 Primer 15</title>
  <meta charset="utf-8">
  <script>
    function koji() {
      //IE do v11
      if (document.all) {
        alert("U pitanju je internet explorer 8");
      }
      //Netscape < v6
      if (document.layers) {
        alert("U pitanju je Netscape ");
      }
      //Chrome,Firefox,Opera,IE 11
      if (document.getElementById && !document.all) {
        alert("U pitanju je Netscape
            6,Opera,Chrome,Firefox");
      }
    }
  </script>
</head>
  <body>
     <h1>Čas 6 Primer 15 - Koja je verzija browsera u
       pitanju ?</h1>
     <form name="formular">
     <input type="button" name="film" value="Klik"
       onClick="koji()"> </form>
  </body>
</html>
```

U objektnom modelu **Internet Explorera do v11** , na primer, javlja se osobina **all** objekta document koja u sebi sadrži sve objekte strane. Zbog toga se u ispitivanju može koristiti sledeći način:

```
if (document.all){
  alert("U pitanju je Internet explorer");
}
```

Slično ovome u starom pretraživaču **Netscape** – ovom objektnom modelu **do v6** (Netscape je pretraživač koji se više ne koristi, poslednja verzija je 9 čija podrška

je prestala 2008 godine) postojala je osobina **layers** objekta document te bi sintaksa u ovom slučaju bila

```
if (document.layers){
  alert("U pitanju je Netscape");
}
```

Za najnovije verzije **Google Chrome,Firefox,Opera i stare Netscape** – ove pretraživače **od verzije 6** (koji imao mnogo izmenjen objektni model u odnosu na verzije 4) ovo ispitivanje bi bilo istinito i dalo rezultate:

```
if (document.getElementByld &&   !document.all){
     alert("U pitanju je   Netscape 6");
}
```

Vežba 1

1. Napravi dve funkcije jedna će u tekst polje upisati današnji datum.
2. A druga će u tekst polje upisati vremem kada se strana otvori.
3. Na istoj strani napraviti dugme klikom na koje se odlazi na drugu unapred napravljenu stranu. Kada se strana otvori da iskoči alert sa natpisom na koji brauzer koristimo.

Vežba 2

1. Napraviti formu sa dva polja i jednim dugmetom. U prvo polje se unosi jedna vrednost u drugo polje druga vrednost i klikom na dugme u alertu treba da se ispiše rezultat množenja.
2. Napisati funkciju da svake tri sekunde iskoči poruka “ja iskočim posle tri sekunde.”
3. Napraviti dva polja u koje se unose celobrojne vrednosti i zatim klikom na dugme da se pojavi alert “veći je od 20” akoje zbir veći od 20 ili “manji je od 20” ako je zbir manji od dvadeset.

Primeri i zadaci

Tema časa:

Primeri za vežbanje

Zadaci za proveru znanja

Primeri za vežbanje

Primer 1

U ovom primeru ćemo definisati otvaranja u novom prozoru. Potrebno je napraviti tri stranice kao u tekstu pod nazivima VezbaPrimer1.html, VežbaPrimerA.html i VežbaPrimerB.html i posmatrati kako reaguju prozori u različitim pretraživačima na promenu različitih parametara.

VežbaPrimer1.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Primeri za vežbanje - Primer 1</title>
  <meta charset="utf-8">
<script>
   function novProzor(url,ime,parametri){
      var winX = window.open(url,ime,parametri);
      winX.setTimeout('window.close()','30000');
    }
</script>
</head>
<body>
  <h1>Primeri za vežbanje - Primer 1</h1>
  <br>
    <a href="#"
     onClick="novProzor('VezbaPrimerA.html','prozor1','status
     =0,toolbar=0,scrollbars=0,location=1')">prozor1</a><br>
     <a href="#"
     onClick="novProzor('VezbaPrimerB.html','prozor2','width=
     500,height=500,top=10,toolbar=0,location=0,scrollbars=0,
     resizable=0,status=0,menubar=0')">prozor2</a><br><br>
     <a href="#"
     onClick="novProzor('VezbaPrimerA.html','prozor3','menuba
     r=no,width=680,height=350,top=10,left=10')">prozor3</a>
     <br>
```

```
    <a href="#"
      onClick="novProzor('VezbaPrimerA.html','prozor4','width=
      500,height=500')">prozor4</a><br><br>
</body>
</html>
```

VežbaPrimerA.htm

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Primeri za vežbanje - Primer A</title>
  <meta charset="utf-8">
</head>
<body>
  <h1>Primeri za vežbanje - Primer A</h1>
  <br>
  <form>
    <input type="button" name="dugme" value="Zatvori prozor"
      onClick="window.close();">
    <input type="button" name="dugme" value="Zatvori oba
      prozora" onClick="opener.close(); window.close();">
  </form>
</body>
</html>
```

VežbaPrimerB.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Primeri za vežbanje - Primer B</title>
  <meta charset="utf-8">
</head>
<body onBlur="window.close();">
  <form>
    <input type="button" name="dugme" value="Zatvori prozor"
      onClick="window.close();">
  </form>
</body>
</html>
```

Primer 2

U ovom primeru vežbamo primena onLoad, onUnload i onMove metoda window objekta.

VezbaPrimer2.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Primeri za vežbanje - Primer 2</title>
</head>
 <h1>Primeri za vežbanje - Primer 2</h1>
 <br>
 <body onLoad="alert('Dobro dosli!')"
     onUnload="alert('Dovidjenja!')" onMove="alert('Pa, vi me
     pomerate!')">
</body>
</html>
```

Primer 3

U ovom primeru vežbamo preuzimanje vrednosti elementa kroz prompt metod objekta window.

VezbaPrimer3.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Primeri za vežbanje - Primer 3</title>
  <meta charset="utf-8">
  <script>
    function ime() {
      var odgovor = window.prompt('Da li zelite da imate jos
     jedan cas', 'odgovor');
      alert(odgovor);
    }
  </script>
</head>
<h1>Primeri za vežbanje - Primer 3</h1>
<br>
<body>
  <form name="Form1">
    <input type="Button" value="Klikni" onClick="ime();">
</form>
</body>
</html>
```

Primer 4

U ovom primeru ćemo napraviti realni sat. Prikazuje tačno vreme na korisnikovom kompjuteru u tekst polju formulara.

VezbaPrimer4.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Primeri za vežbanje - Primer 4</title>
  <meta charset="utf-8">
  <script>
    function pokaziVreme() {
      var sad = new Date();
      var sati = sad.getHours();
      var minuti = sad.getMinutes();
      var sekunde = sad.getSeconds();
      if (sekunde <= 9) {
        sekunde = "0" + sekunde;
      }
      if (minuti <= 9) {
        minuti = "0" + minuti;
      }
      if (sati <= 9) {
        sati = "0" + sati;
      }
      document.vreme.sat.value = (sati + ":" + minuti + ":" +
     sekunde);
      //var tiktak = window.setTimeout("pokaziVreme()", 1000);
    }

    function startujSat() {
      pokaziVreme();
    }
  </script>
</head>
<body onLoad="startujSat()">
  <h1>Primeri za vežbanje - Primer 4</h1>
  <form name="vreme">
    <input type="text" name="sat" size="8" value>
  </form>
</body>
</html>
```

Primer 5

U ovom primeru ćemo da prikažemo kako se vrši provera sadržaja tekst polja formulara na e-mail strukturu (znak @ i tačku).

VezbaPrimer5.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Primeri za vežbanje - Primer 5</title>
  <meta charset="utf-8">
  <script>
      function proveriMejl(polje){
          if (polje.value == ""){
            alert("Molim Vas upisite svoju mejl adresu!!!");
            polje.focus();
            return false;
          }
          else if (!isEmail(polje.value)){
            alert("Molim Vas upisite ISPRAVNU mejl adresu!!!");
             polje.focus();
            return false;
          }
          else{
            return true;
          }
      }
      var whitespace = " \t\n\r";

      function isEmail (s) {
        if (isEmpty(s)) {
          if (isEmail.arguments.length == 1) {
            return defaultEmptyOK;
          } else {
            return (isEmail.arguments[1] == true);
          }
        }
        if (isWhitespace(s)) {
          return false;
        }
        var i = 1;
        var sLength = s.length;
        while ((i < sLength) && (s.charAt(i) != "@")){
          i++
        }
        if ((i >= sLength) || (s.charAt(i) != "@")) {
          return false;
        } else {
          i += 2;
        }
        while ((i < sLength) && (s.charAt(i) != ".")) {
          i++
        }
        if ((i >= sLength - 1) || (s.charAt(i) != ".")) {
          return false;
```

```
        } else {
          return true;
        }
      }
    function isEmpty(s) {
        return ((s == null) || (s.length == 0));
    }
    function isWhitespace (s) {
      var i;
      if (isEmpty(s)) {
        return true;
      }
      for (i = 0; i < s.length; i++) {
        var c = s.charAt(i);
        if (whitespace.indexOf(c) == -1) {
          return false;
        }
      }
      return true;
    }
</script>
</head>
<body>
  <h1>Primeri za vežbanje - Primer 5</h1>
  <form name="formular" method="POST" onSubmit="return
proveriMejl(document.formular.email);">
    <input type="text" name="email" size="15"
maxlength="50"><br><br>
    <input type="submit" value="Posalji"><br><br>
  </form>
</body>
</html>
```

Primer 6

U ovom primeru ćemo pokazati kako da napravite vaš prvi slide show ili slučajni prikaz slika na stranici. Uz stranicu ide folder images sa slikama sa nazivima: 1.jpg pa sve do 10.jpg. Putanje do slika zadate su u nizu slika i dalje se metodom slučajnog odabira bira indeks po kome će biti tražena putanja slike koja se prikazuje. Funkcija se poziva iz taga BODY na idenifikator događaja onLoad.

VezbaPrimer6.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Primeri za vežbanje - Primer 6</title>
  <script>
    var slika = new
Array("images/1.jpg","images/2.jpg","images/3.jpg",

"images/4.jpg","images/5.jpg","images/6.jpg",

"images/7.jpg","images/8.jpg","images/9.jpg",
                          "images/10.jpg");

    function randomize() {
        var a = Math.round(Math.random() * 10);
        if (a != 10){
          document.images[0].src = slika[a];
          alert("Prvi uslov "+(a+1)+".jpg");
        } else {
          document.images[0].src = slika[10 - a];
          alert("Drugi uslov "+(10 - a +1)+".jpg");
        }
    }
  </script>
</head>

<body onLoad="randomize();">
  <h1>Primeri za vežbanje - Primer 6</h1>
  <img src="images/1.jpg" border="0" width="250" height="200"
     alt="" name="s1">
  <br>
  <br>
</body>

</html>
```

Primer 7

U sledeća dva primera vežbaćete proveru da li su svi uneseni karakteri brojevi. Prvi skript očitava koji tasteri su pritisnuti i upozorava ukoliko nisu pritisnuti tasteri za brojeve. Drugi očitava svaki karakter unetog stringa i poredi sa celim brojevima.

VezbaPrimer7a.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Primeri za vežbanje - Primer 7a</title>
  <script>
    //which i keyCode vracaju Unicode broj unete karaktere
    // npr 1 je 49 a 9 je 57
    function proveriBrojeve(a) {
      var charCode;
      if (navigator.appName == "Netscape") {
        alert("Netscape");
        charCode = a.which;
        alert(charCode);
      }else {
        alert("Nije Netscape");
        charCode = a.keyCode;
      }
      status = charCode;
      if (charCode > 31 && (charCode < 48 || charCode > 57)) {
        alert("Morate uneti SAMO BROJEVE");
        return false;
      }
      return true;
    }
  </script>
</head>
<body>
  <h1>Primeri za vežbanje - Primer 7a</h1>
  <form name="formular" method="POST">
    <input type="text" name="telefon" size="15" maxlength="50"
onKeyPress="return proveriBrojeve(event)">
    <br>
    <br>
    <input type="submit" value="Posalji"> </form>
</body>

</html>
```

VezbaPrimer7b.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>Primeri za vežbanje - Primer 7b</title>
  <script>
    function proveriBrojeve() {
      var element = document.formular.telefon;
      broj = element.value;
      if (broj == "") {
        alert("Unesite nesto!");
        element.focus();
        return false;
      }
      else {
        for (i = 0; i < broj.length; i++) {
          var znak = broj.substring(i, i + 1);
          if (znak < "0" || znak > "9") {
            alert("Morate uneti SAMO BROJEVE");
            element.focus();
            return false;
          } else {
            return true;
          }
        }
      }
      return true;
    }
  </script>
</head>

<body>
  <h1>Primeri za vežbanje - Primer 7b</h1>
  <form name="formular" method="POST" onSubmit="return
     proveriBrojeve();">
    <label for="telefon">Unesite telefon :</label>
    <input type="text" name="telefon" size="15" maxlength="50">
    <br>
    <br>
    <input type="submit" value="Posalji"> </form>
</body>
```

Primer 8

U ovom primeru ćemo vežbati funkciju timeout() tako što ćemo definisati funckiju koja koja će sačekati 3 sekunde zatim započeti izračunavanje unesenih brojeva u input polja I na kraju ispisati rezultat nakon 5 sekundi.

VezbaPrimer8.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Primeri za vežbanje - Primer 8</title>
<script>
  function cekaj(){
    setTimeout("alert('Proslo je 3 sekunde! i dalje racunam
     ...')",3000);
    //napraviti funkciju koja sabira brojeve
    // i prikaže rezultat nakon 5 sekundi
    setTimeout(izracunaj,5000);
  }
  function izracunaj(){
      var a = document.forma.prvi.value;
      var b = document.forma.drugi.value;
      var c = parseFloat(a)+parseFloat(b);
      document.forma.veci.value = c;
      // ako stavim alert sta se desi?
       alert("Proslo je 5 sekundi . Rezultat je "+c);
  }
</script>
</head>
<body>
  <h1>Primeri za vežbanje - Primer 8</h1>
<form name="forma">
  <label for="prvi">Prvi broj</label>
  <input type="text" name="prvi"><br>
  <label for="drugi">Drugi broj</label>
  <input type="text" name="drugi"><br>
  <label for="manji">Manji</label>
  <input type="text" name="manji"><br>
  <label for="veci">Veci</label>
  <input type="text" name="veci"><br>
<input type=button value="Saberi" onClick="cekaj()">
</form>
</body>
</html>
```

Primer 9

U ovom primeru ćemo očitati i prikazati veličinu prozora u pikselima nakon svake 3 sekunde.

VezbaPrimer9.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Primeri za vežbanje - Primer 9</title>
  <script>
    var x;
    var y;
    var h;
    var w;
    function ocitaj(){
      if(navigator.appName=="Netscape"){
        //alert("usao");
        x = window.pageXOffset;
        y = window.pageYOffset;
        //ovo dole se ne koristi vise , a i ne radi
        //h = document.height;
        h = document.body.clientHeight;
        //ovo dole se ne koristi vise , a i ne radi
        //w = document.width;
        w = document.body.clientWidth;
      } else {
        x = document.body.scrollLeft;
        y = document.body.scrollTop;
        h = window.document.body.scrollHeight;
        w = window.document.body.scrollWidth;
      }
      var status =
     "XScroll:"+x+"/YScroll:"+y+"/Visina:"+h+"/Sirina:"+w;
      alert(status);
      setTimeout("ocitaj()",3000);
    }
  </script>
</head>

<body>
  <h1>Primeri za vežbanje - Primer 9</h1>
  <a href="#" onclick="ocitaj()">Prikazi pozicije</a>
</body>
</html>
```

Primer 10

U ovom primeru ćemo promeniti boje pozadine .Klikom na dugme menjaćemo boje pozadine

VezbaPrimer10.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Primeri za vežbanje - Primer 10</title>
  <meta charset="utf-8">
  <script language="JavaScript">
    function newbg(thecolor){
    //prvi nacin radi ali je izbacen iz standarda
    //document.bgColor=thecolor;
    //drugi nacin
    document.body.style.backgroundColor=thecolor;
  }
  </script>
</head>
<body>
  <h1>Primeri za vežbanje - Primer 10</h1>
  <div align="center">
  <form>
   <input type="button" value="White" onclick="newbg('white');
      ">
   <input type="button" value="Blue" onclick="newbg('blue');">
   <input type="button" value="Beige" onclick="newbg('Beige');
      ">
   <input type="button" value="Yellow" onclick="newbg('yellow'
     );">
   </form>
   </div>
</body>
</html>
```

Primer 11

U ovom primeru ćemo promeniti boju pozadine elemenata tabele. Prikazani kod menjaće boju pozadine ćelije. Isti način koristimo za izmenu boje reda i izmenu boje tabele (tada tako pozicioniramo identifi kator događaja). Radi samo u Internet Explorer – u.

VezbaPrimer11.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Primeri za vežbanje - Primer 11</title>
  <meta charset="utf-8"> </head>
<body>
  <h1>Primeri za vežbanje - Primer 11</h1>
  <table align="Center" border="1" cellspacing="0"
      cellpadding="0" width="100">
    <tr>
      <td onMouseover="this.style.backgroundColor='lightgrey'"
      onMouseout="this.style.backgroundColor='#FFFFFF'"> <
      /td>
      <td onMouseover="this.style.backgroundColor='lightgrey'"
      onMouseout="this.style.backgroundColor='#FFFFFF'"> <
      /td>
    </tr>
  </table>
</body>
</html>
```

Primer 12

U ovom primeru vežbamo prosleđivanje parametara funkciji sa document.getElementByID metodom

VezbaPrimer12.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Primeri za vežbanje - Primer 12</title>
  <meta charset="utf-8">
  <script>
    function param(a) {
      alert(a);
    }
  </script>
</head>
  <body>
    <h1>Primeri za vežbanje - Primer 12</h1>
  <form name="formular">
    Vrednost je već uneta samo klikni dugme:
```

```
    <input type="button" name="film" value="Unesi tekst"
      onClick="param(3);"><br><br>

    Unesite neki tekst:
    <input type="text" id="nekiTekst" name="ime">
    <input type="button" name="film" value="Unesi tekst"
      onClick="param(document.getElementById('nekiTekst').value
      );">
  </form>
  </body>
</html>
```

Zadaci za ponavljanje

Zadatak 1

Napravite na vašoj HTML strani formular koji se sastoji od sledećih polja:

1. button sa natpisom "Nov prozor",
2. button sa natpisom "Autor",
3. button sa natpisom "30 sekundi".

Napravite program koji radi sledeće:

Po svom učitavanju prikazuje alert okvir za dijalog u kome vas pozdravlja po imenu.

Pritiskom na dugme "Nov prozor" pravi potprozor u kome se nalazi button sa natpisom "Zatvori" i koji posle klika na njega zatvara potprozor. Uključiti opciju koja isključuje potprozor ukoliko ovaj izgubi fokus.

Pritiskom na dugme "Autor" aktivira se alert okvir za dijalog u kome je napisana informacija o autoru skripta.

Pritiskom na dugme "30 sekundi" posle 30 sekundi pojavljuje se alert okvir za dijalog u kome piše upozorenje o tome da je prošlo 30 sekundi.

Napraviti ispis poruke na status baru koja se ispisuje posle klika na neko od dugmadi.

Zadatak 2:

Napravite JavaScript na vašoj stranici po sledećim uputstvima:

1. Stranica index.htm treba da ima belu pozadinu.
2. Na stranici treba da se nalaze dugmadi koja vrše sledeće funkcije: jedno dugme posle pritiska poziva confi rm okvir za dijalog u kome postavlja pitanje "Da li ste sigurni?" i u slučaju potvrdnog odgovora šalje na stranicu druga.htm.
3. Drugo dugme na pritisak oboji pozadinu strane u crvenu boju.
4. Treće dugme na pritisak otvara novi prozor koji ima prikazan samo meni bar, a koji nije skrolabilan, niti mu je moguće promeniti dimenzije, a u kome je prikazana stranica treca.htm koja u sebi ima dugme koje na pritisak u otvarajućoj stranici otvara hvala.htm i koja taj prozor zatvara.
5. Stranica hvala.htm treba samo da ima poruku "Hvala sto ste slušali kurs!".
6. Stranica druga.htm otvara alert okvir za dijalog i u njemu ispisuje poruku "Svaka čast!" i ima sadržaj select element koji je navigacioni i u kom se sastoje linkovi na koje se ode već posle njihove selekcije. Linkovi su proizvoljni i jedino je bitno da su pravilno zadati.

Zadatak 3:

Napraviti JavaScript na vašoj stranici koji radi sledeću stvar:

1. U zavisnosti od toga koji je dan u pitanju ispisuje se adekvatna poruka u alert prozoru po tipu "Dobro došli! Danas je taj_dan!"
2. Nakon toga otvara se alert prozor u kome se ispisuje tekst "Vi koristite tip_pretraživača!"
3. Po učitavanju stranice u status bar - u ispisuju se pozicije klizača.