JavaScript Funkcije

Funkcije

- Funkcije predstavljaju skupove naredbi koje zajedno obavljaju određeni zadatak.
- Svaka funkcija je jednoznačno određena putem njenog imena, putem kojeg se vrši njeno pozivanje.
- Pozivanjem neke funkcije se izvršavaju sve naredbe definisane u telu funkcije.
- Organizovanje programa u funkcije sprečava ponavljanje delova istog koda i čini programe čitljivijim i lakšim za održavanje.

Funkcije

```
function imeFunkcije() {
    // Telo funkcije
    // ...
}
```

Funkcije

- Ime funkcije mora biti jednstveno unutar važenja funkcije (datoteka ili objekat). Mora počinjati slovom i ne sme sadržavati razmake.
- Telo funkcije počinje i završava vitičastim zagradama i sadrži proizvoljan broj JavaScript naredbi.
- Iz tela JavaScript funkcije može se pozvati neka druga funkcija, a takođe je moguće u telu JavaScript funkcije definisati neku drugu funkciju.

Funkcije

- Deo programa u kome se kreira funkcija zove se definicija funkcije
- Kada se funkcija koristi u programu, kažemo da se poziva ta funkcija
- Primer: 1.primer_funkcije

Funkcije sa parametrima

- Funkcije mogu imati proizvoljan broj parametara
- Parametri predstavljaju dodatne informacije koji su potrebni funkciji da obavi svoj zadatak
- Možemo reći da su parametri promenljivi (varijabilni) delovi funkcije i oni predstavljaju ulazne informacije
- Takođe, funkcija često treba da vrati rezultat svog izvršavanja i njega nazivamo povratnom vrednosti funkcije
- Povratna vrednost funkcije je vrednost koju funkcija vraća nakon ključne reči **return**
- Povratna vrednost može biti neki broj, tekst, niz, objekat, poziv neke druge funkcije, itd.

Funkcije

```
function izracunajPovrsinu(sirina, visina) {
    var povrsina = visina * sirina;
    return povrsina;
}

var povrsina = izracunajPovrsinu(3, 5);
```

Funkcije

- Prilikom definisanja funkcije sa parametrima, nazivi parametara su proizvoljni
- Prosleđeni parametri u funkciji se koriste kao promenljive lokalne za tu funkciju
- Prilikom pozivanja funkcije prosleđujemo joj konkretne vrednosti sa kojima će raditi. U tom slučaju nazivamo ih **argumentima** funkcija

Funkcije

- Zadatak:
- Primer **2. happy_birthday** izmeniti tako da se omogući ispisivanje rođendanske čestitke za bilo koju osobu. Ispis stihova pesme treba da se radi u funkciji koja kao parametar prima ime osobe.

Opseg važenja promenljivih (scope)

- Definiše u kom delu programa su promenljive „vidljive“
- Promenljive definisane unutar funkcije pomoću ključne reči **var** važe samo unutar te funkcije. One se zovu **lokalne** promenljve.
- Promenljive definisane izvan funkcija važe u celoj .js datoteci (i u svim JavaScript datotekama uključenim u isti HTML dokument). One se zovu **globalne** promenljive.
- Unutar istog opsega važenja ne smeju postojati dve promenljive sa istim imenima.

JavaScript Objekti

- U Objektno Orijentisanom Programiranju (OOP), objektima su predstavljeni pojave i entiteti iz sistema koji programiramo
- Objekti se defnišu kao promenljive (**var**), ali za razliku od promenljivih, objekti sadrže druge promenljive i funkcije
- Promenljive sadržane u objektu se nazivaju **atributi** objekta
- Funkcije sadržane u objektu se nazivaju **metode** objekta

JavaScript Objekt

Pretpostavimo da na sajtu želimo da imamo prezentaciju hotela. O hotelu želimo da prikažemo sledeće informacije:

- Naziv hotela
- Adresu
- Broj soba
- Broj rezervisanih soba
- Da li hotel ima teretanu

JavaScript Objekti

- JavaScript objekat koji bi predstavljao spomenuti hotel:

```
var hotel = {
    naziv: “Hotel Park”,
    adresa: “Novosadskog sajma 35”,
    brojSoba: 140,
    rezervisano: 57,
    teretana: true
};
```

JavaScript Objekti

- Evidenciju o broju slobodnih soba mogli bi da vodimo u novom atributu. Međutim, pošto broj slobodnih soba zavisi od kapaciteta hotela i broja rezervisanih soba, napravićemo novu metodu unutar objekta koja će sama da računa broj slobodnih soba.

JavaScript Objekti

```
var hotel = {
     naziv: “Hotel Park”,
     adresa: “Novosadskog sajma 35”,
     brojSoba: 140,
     rezervisano: 57,
     teretana: true
     brojSlobodnihSoba: function() {
          return  this.brojSoba – this.rezervisano;

     }
};
```

. Operator (member operator)

- Koristi se za pristup atributima i metodama objekta

```
var imeHotela = hotel.naziv;
var slobodnoSloba = hotel.brojSlobodnihSoba();
```

- Vrednost atributa se može menjati

```
hotel.teretana = false;
```

- Unutar objekta, za pristup atributima i metodama tog objekta umesto imena objekta koristi se ključna reč **this**.

Zadatak

- U objektu **hotel** kreirati metodu za rezervaciju sobe. Ova metoda uvećava broj rezervisanih soba hotela za jedan

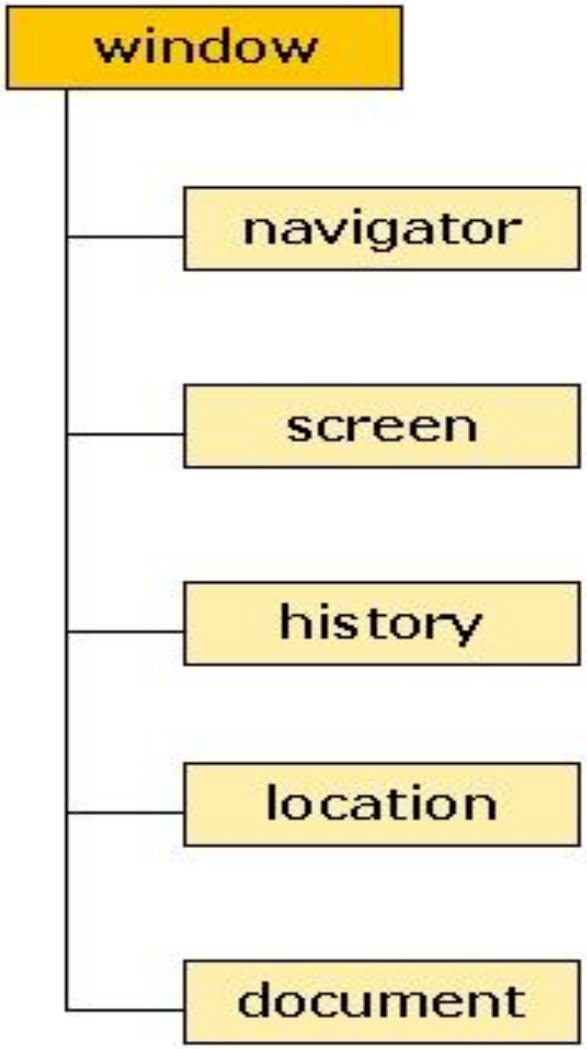
Nizovi objekata

- Za čuvanje više objekata istog tipa najlakše je koristit nizove
- Objekti u nizu nemaju ime
- Pristup se vrši preko indeksa
- PRIMER: 5. objekti_hoteli

Ugrađeni Objekti

- Web browseri sadrže skup ugrađenih objekata koji omogućavaju pristup samom browser-u ili strukturi HTML stranice iz JavaScript programa.
- Ovi objekti su dostupni pod standardnim imenima i možemo ih podeliti u 3 grupe:
  - **Browser Object Model**: Sadrži objekte koji predstavljaju prozor interner browsera (odnosno trenutno aktivan tab),
  - **Document Object Model**: Sadrži objekte koji predstavljaju trenutno prikazan HTML document
  - **Globalni JavaScript objekti**: Proširuju JavaScript jezik dodatnim funkcijama

Browser Object Model



Browser Object Model

- **window** - korenski element, predstavlja trenutno aktivan prozor ili tab web browsera. Svim ostalim objektima se pristupa preko ovog objekta.
- **navigator** - predstavlja sam web browser
- **screen** - predstavlja ekran uređaja na kojem se stranica prikazuje
- **location** - URL trenutne stranice
- **document** - predstavlja trenutno učitanu HTML stranicu. Ovaj objekat je korenski element Document Object Modela
- **console** - developer conzola browsera

window OBJEKAT – korisni atributi

- **innerWidth** , **innerHeight** - Visina i širina prozora (samo dela sa HTML stranicom).
- **pageXOffset** , **pageYOffset** - Koliko piksela je dokument skrolovan horizontalno i vertikalno.
- **screenX** , **screenY** - Pozicija browser prozora na ekranu (u odnosu na gornji levi ugao ekrana).

window Objekat – korisne metode

- **window.alert(poruka)** - Prikazuje pop-up dijalog sa prosleđenom porukom i dugmetom za potvrdu.
- **window.confirm(poruka)** - Prikazuje pop-up dijalog sa prosleđenom porukom i OK i Cancel dugmićima.
- **window.open(url)** - Otvara novi prozor (tab) sa prosleđenom adresom. Mora biti dozvoljeno od strane browser-a.
- **window.print()** - Pokreće komandu za štampanje trenutne stranice.

window Objekat - Napomena

- **window**, kao korenski objekat JavaScript objektno modela, ne mora se navoditi. Sve navedene metode i atributi će raditi i bez **window** prefiksa.

```
window.alert(„Hello world!“);
//Isto kao i
alert(„Hello world!“);
```

Primer: 6. window

console Objekat

- Služi za programski pristup developer konzoli.
- Omogućava ispis više tipova poruka u konzoli.
- Korisno za brzo debugovanje, jer konzolni ispisi ne blokiraju
- Izvršavanje JavaScript koda (za razliku od alert prozora).

Location Objekat – korisne metode

- **location.reload()** - Ponovo učitava trenutnu stranicu (refresh).
- **location.assert(url)** - Učitava novi URL i ažurira postojeću istoriju.
- **location.replace(url)** - Učitava novi URL sa novom istorijom.

history Objekat – korisne metode

- **history.back()** - Učitava prethodnu stranicu iz liste
- **history.forward()** - Učitava sledeću stranicu iz liste
- **history.go(index)** - Učitava URL na zadanom indeksu u listi.

- PRIMER: 7. ostali_objekti

Document Object Model

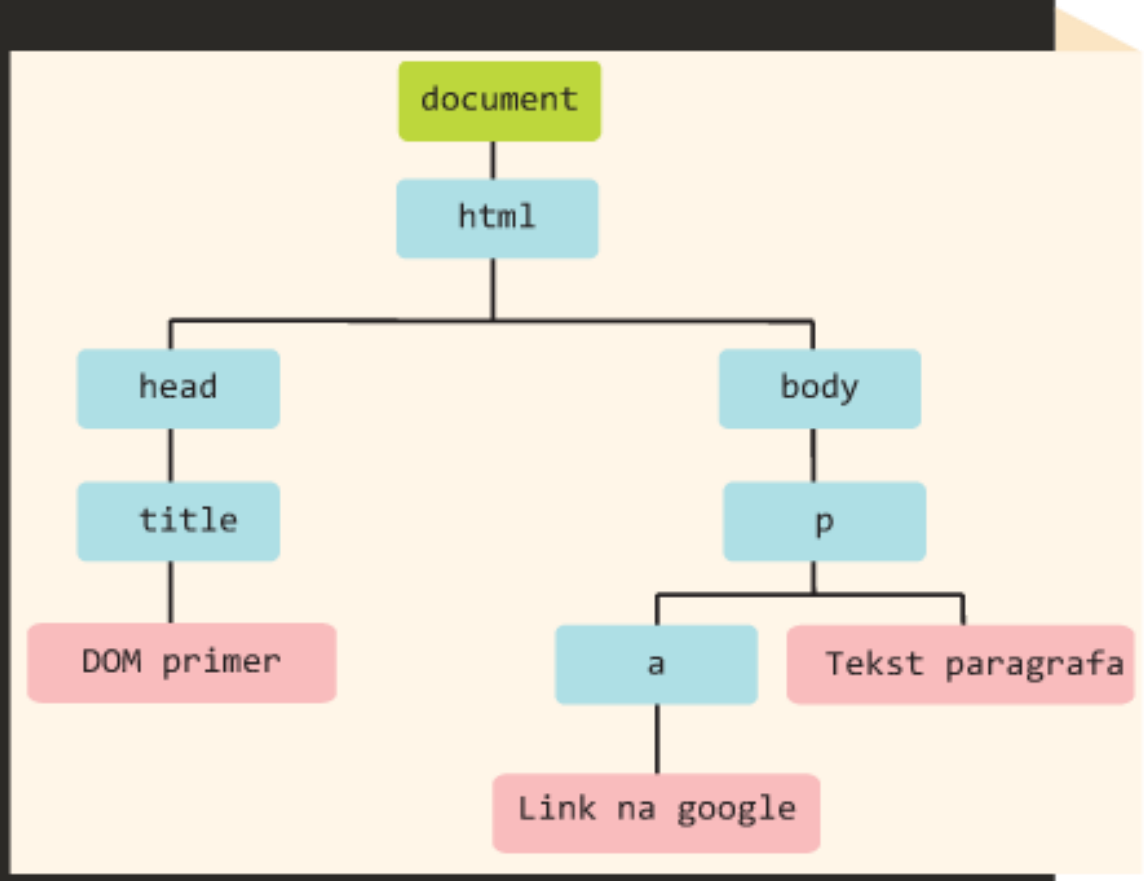
- Objektna reprezentacija HTML stranice.
- Kada se stranica učita, browser kreira **document** objekat, kao i objekte za svaki element na stranici.
- Dom manipulacije predstavljaju osnovu kreiranja dinamičkih HTML stranica

Document Object Model

- Jedini standardni objekat je document
- Svi ostali objekti zavise od strukture same HTML stranice.
- DOM predstavlja strukturu tipa stabla, sa document elementom u korenu.
- Svi elementi DOM stabla se sastoje od atributa i tekstualnog (HTML) sadržaja

Document Object Model

```
1  <html>
2  <head>
3      <title>DOM primer</title>
4  </head>
5  <body>
6      <p>
7          Tekst paragrafa
8          <a href="http://google.com">
9              Link na google
10         </a>
11     </p>
12 </body>
13 </html>
```



Document Object Model

- Svaki HTML dokument se posmatra kao DOM stablo
- Čvor na vrhu se zove korenski čvor
- Svaki čvor osim korenskog ima jednog roditelja (čvor iznad)
- Svaki čvor može da ima potomke (decu – čvorovi ispod)
- List je čvor bez dece
- Čvorovi istog nivoa (sibiling) su čvorovi sa istim roditeljem.

Navigacija Kroz DOM Stablo

- **document** objekat sadrži metode i atribute za manipulaciju DOM stablom. jQuery u velikoj meri olakšava pisanje ovih naredbi.
- PRIMER: 8. dom_hoteli

Globalni JavaScript Objekti

- Omogućavaju naprednije rukovanje numeričikim i tekstualnim tipovima podataka i proširuju JavaScript jezik dodatnim funkcijama
- Dostupni objekti:
  - String
  - Number
  - Date
  - Math

String Objekti

- Sve tekstualne vrednosti u JavaScriptu su u stvari objekti tipa String
- Ovo znači da su im dostupne sve metode i atributi definisani u String objektu
- U programskim jezicima tekst je predstavljen nizom karaktera.

String Objekti

- **length** - atribut koji predstavlja dužinu stringa
- **toUpperCase()** - pretvara sva slova u tekstu u velika (za mala: **toLowerCase()**)
- **charAt(pos)** - Vraća slovo na zadatoj poziciji
- **indexOf(str)** - Vraća prvi indeks na kojem je zadani string pronađen
- **substring(start**, end) - Vraća sve karaktere koji se nalaze između zadanih indeksa

String Objekti

- **split(char)** - Vrši "isecanje" teksta po zadanom karakteru, smešta rezultat u niz
- **trim()** - Uklanja razmake sa početka i kraja teksta
- **replace(stari, novi)** - Menja prosleđeni stari tekst sa novim.

Number Objekti

- **isNan(val)** - Proverava da li prosleđena vrednost nije broj.
- **toFixed(places)** - Zaokružuje decimalni broj na prosleđeni broj decimala (vraća string)
- **toPrecision(precision)** - Zaokružuje broj na prosleđeni broj cifara
- PRIMER: 9. numbers

Date Objekti

- Služe za rukovanje tipovima podataka koji čuvaju vreme i datume.
- Rukovanje slično kao u Java programskom jeziku.
- Kreiraju se pomoću konstruktora:

Date

```
// Bez parametara – trenutni datum i vreme
var now = new Date();

// Sa parametrima možemo kreirati objekti sa zadanim vremenom i datumom
var date1 = new Date(1996, 11, 26, 15, 45, 55);
var date2 = new Date(1996, 11, 26);
var date3 = new Date(„Dec 26, 1996 15:45:55“);
```

Date – korisne metode

- Na raspolaganju su nam metode za rukovanje pojedinačnim elementima datuma i vremena:

- getDate() - vraća dan u mesecu (1-31)
- getDay() - vraća dan u nedelji (0-6, počinje sa nedeljom)
- getFullYear() - godina kao četverocifreni broj
- getMonth() - broj meseca u godini (0-11)
- getMinutes() - minute (00-59)
- getHours() - sati (0-23)
- PRIMER: 10. date

Date – korisne metode

- getTime() funkcija
- Vraća broj milisekundi koji je protekao od 01.01.1970. Ako je datum za koji je poziva pre ovog datuma, taj broj je negativan. Ovo je interni format u kojem se čuvaju datumi u JavaScript programskom jeziku. Pogodan je za različite računske operacije nad datumom i vremenom
- PRIMER: 11. date

Math Objekat

- Predstavlja biblioteku matematičkih funkcija i konstanti za JavaScript

- **Math.round(broj)** - Zaokružuje prosleđeni broj na najbliži celi.
- **Math.ceil(broj)** - Zaokružuje prosleđeni broj na veći ceo broj.
- **Math.floor(broj)** - Zaokružuje prosleđeni broj na manji ceo broj.
- **Math.sqrt(broj)** - Računa kvadratni koren.
- **Math.random()** - Vraća slučajno generisan broj između 0 i 1.
- PRIMER: 12. math