



ftninformatika

Java Web Development

Modul 2

Termin 9

Sadržaj

1. JavaScript:

- I. Osobine jezika
- II. Sintaksa
- III. JSON
- IV. Objekti
- V. Klase
- VI. Ugrađene klase

Problem

U dosadašnjem pristupu u razvoju *web* aplikacija, implementacija na strani aplikacije (*web server*-a) je upravljala izgledom *web* stranice (*server-side generisan dinamički HTML sadržaj*).

Za svaku i *najmanju promenu u prikazu stranice* (npr. pretraga, poruka o greški i sl.), server je morao da generiše *potpuno novu stranicu* sa dodatkom te promene

Promene prikaza su do sada bile moguće samo na 3 načina:

- promena adrese u HTTP klijentu (*browser*-u)
- klik na link
- klik na dugme u formi

Ovakav pristup postaje nedovoljan kada je:

- promene u prikazu potrebno obavljati brzo, odnosno trenutno
- potrebno reagovati na drugačije događaje u korisničkoj interakciji
- neprihvatljivo da promene u prikazu osveže stranicu i učitaju interfejs ispočetka (*video playback, real-time pretraga* i sl.)

JavaScript

Da bi učitana *web* stranica mogla lokalno da se menja, **deo logike za prikaz je potrebno izmestiti u sam *browser*** (HTTP klijent) na korisničkom uređaju.

Ova logika mora biti implementirana u vidu programa na **standardnom programskom jeziku** koga razumeju svi *browser*-i.

Iz praktičnih razloga **programi se ne prevode** jer bi u tom slučaju morale da se prave izvršne verzije programa za sve *browser*-e na svim platformama, za svaku novu specifikaciju programskog jezika i pratećeg API-a.

Naredbe u programima se čitaju u izvornom obliku i izvršavaju *ad hoc* (**interpretiraju**), poput npr. SQL skripti.

Programi se pridružuju HTML stranicama i uz njih se na zahtev šalju *browser*-ima.

Browser je zadužen za učitavanje, prikaz HTML stranice i interakciju sa korisnikom, a poziva fragmente ovih programa po učitavanju HTML stranice i pri korisničkim događajima.

Ovakvi programi se zovu skripte.

JavaScript



- standardni jezik (sada već opšte namene) koga **podržavaju svi web browser-i**
- samo sintaksa u velikoj meri liči na sintaksu *Java* programskog jezika i tu se završavaju sličnosti
- kao i u *Java* programskom jeziku identifikatori su **case-sensitive**
- **interpretirani** jezik
- **dinamički tipiziran**
- **funkcije su objekti 1. reda**, tj. mogu da:
 - budu dodeljene promenljivoj
 - se dodaju u niz
 - budu parametri drugih funkcija
 - budu atributi objekata

JavaScript

script element

Po prijemu HTML dokumenta browser redom učitava jedan po jedan element i ako je moguće ga prikazuje.

Script element sadrži definicije i naredbe na *script* jeziku (za našu primenu na *JavaScript*-u).

Script element može biti podelement *head* ili *body* elementa i može ih biti više.

Ukoliko *script* element sadrži definicije (promenljivih ili funkcija), one tada bivaju interpretirane, učitane u memoriju stranice i dostupne naredbama u bilo kom sledećem *script* elementu na istoj stranici.

Ukoliko *script* element sadrži naredbe, one tada bivaju interpretirane i izvršene.

U slučaju *runtime* greške, ostatak naredbi u *script* elementu se preskače, a tehnički detalji o greški bivaju prikazani u konzoli browser-a.

```
<!DOCTYPE html>
<html>
  <head>
    :
    <script type="text/javascript">
      :
    </script>
  </head>
  <body>
    :
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    :
  </head>
  <body>
    :
    <script type="text/javascript">
      :
    </script>
    :
  </body>
</html>
```

JavaScript

Definicije

Učitane definicije promenljivih i funkcija važe na celoj stranici. Mogu referencirati bilo kada nakon učitanih definicija.

```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript">
      let promenljiva = "vrednost";

      function upperCase(promenljiva) {
        return promenljiva.toUpperCase();
      }
    </script>
  </head>
  <body>
    <script type="text/javascript">
      console.log(promenljiva);
      console.log(upperCase(promenljiva));
    </script>
  </body>
</html>
```

scope
promenljive *promenljiva*

JavaScript

Definicije

Učitane definicije promenljivih i funkcija važe na celoj stranici. Mogu referencirati bilo kada nakon učitanih definicija.

```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript">
      let promenljiva = "vrednost";

      function upperCase(promenljiva) {
        return promenljiva.toUpperCase();
      }
    </script>
  </head>
  <body>
    <script type="text/javascript">
      console.log(promenljiva);
      console.log(upperCase(promenljiva));
    </script>
  </body>
</html>
```

scope
upperCase funkcije

JavaScript

Osnovni elementi sintakse

Ukoliko je naredba sama u jednoj liniji koda, može se izostaviti “;”, ali je preporuka da se ipak koristi.

```
// naredba dodele
promenljiva = "vrednost";

/*
    poziv
    funkcije
*/
funkcija();

// uzastopne naredbe
promenljiva1 = "p1"; promenljiva2 = "p2"; funkcija();

// iteracija
for (...) {

}

// selekcija
if (...) {

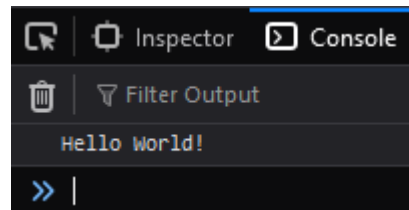
}
```

JavaScript

Funkcija *console.log(...)*

Funkcija *console.log(...)* ispisuje vrednost argumenta u konzoli *web browser*-a.

```
<script type="text/javascript">  
    console.log("Hello World!");  
</script>
```



tasterima **Control + Shift + K**
se u web browser-u otvara
Web konzola

JavaScript

Promenljive

Promenljive se definišu ključnom rečju *let*.

Ne navodi se tip promenljive.

Promenljiva poprima tip onog momenta kada joj se dodeli vrednost.

```
let a = "1"; // string
let b = 2; // number
let c = false; // boolean
let d = null;
let e;
konkatenacija
↓
console.log("vrednost promenljive a je: " + a + ", a tip: " + typeof a);
console.log("vrednost promenljive b je: " + b + ", a tip: " + typeof b);
console.log("vrednost promenljive c je: " + c + ", a tip: " + typeof c);
console.log("vrednost promenljive d je: " + d + ", a tip: " + typeof d);
console.log("vrednost promenljive e je: " + e + ", a tip: " + typeof e);
```

```
let f = "1", g = 2, h = false, i = null; // jednolinijska deklaracija
console.log("vrednost promenljive f je: " + f);
console.log("vrednost promenljive g je: " + g);
console.log("vrednost promenljive h je: " + h);
console.log("vrednost promenljive i je: " + i);
```

```
vrednost promenljive a je: 1, a tip: string
vrednost promenljive b je: 2, a tip: number
vrednost promenljive c je: false, a tip: boolean
vrednost promenljive d je: null, a tip: object
vrednost promenljive e je: undefined, a tip: undefined
```

```
vrednost promenljive f je: 1
vrednost promenljive g je: 2
vrednost promenljive h je: false
vrednost promenljive i je: null
```

JavaScript

Promenljive

Promenljiva menja tip kada joj se dodeli vrednost nekog drugog tipa.

```
let a = "1";  
console.log("vrednost promenljive a je: " + a + ", a tip: " + typeof a);  
  
a = 2;  
console.log("vrednost promenljive a je: " + a + ", a tip: " + typeof a);
```

vrednost promenljive a je: 1, a tip: string

vrednost promenljive a je: 2, a tip: number

JavaScript

Aritmetički operatori

```
console.log(3 + 2);  
console.log(3 - 2);  
console.log(3 * 2);  
console.log(3 / 2);  
console.log(3 % 2); // celobrojni ostatak pri deljenju
```

5
1
6
1.5
1

```
let a, b;  
a = 3; b = a++; console.log("a: " + a + ", b: " + b);  
a = 3; b = ++a; console.log("a: " + a + ", b: " + b);  
a = 3; b = a--; console.log("a: " + a + ", b: " + b);  
a = 3; b = --a; console.log("a: " + a + ", b: " + b);
```

a: 4, b: 3
a: 4, b: 4
a: 2, b: 3
a: 2, b: 2

JavaScript

Operatori dodele

```
let a, b;  
a = 3; b = 2; a = b; console.log(a);  
a = 3; b = 2; a += b; console.log(a);  
a = 3; b = 2; a -= b; console.log(a);  
a = 3; b = 2; a *= b; console.log(a);  
a = 3; b = 2; a /= b; console.log(a);  
a = 3; b = 2; a %= b; console.log(a);
```

2
5
1
6
1.5
1

JavaScript

Relazioni operatori

```
console.log(3 > 2);  
console.log(3 >= 2);  
console.log(3 < 2);  
console.log(3 <= 2);  
console.log((3 == 2) + ", " + (3 == 3));  
console.log((3 != 2) + ", " + (3 != 3));  
console.log(3 == "3"); // poređenje po vrednosti  
console.log(3 === "3"); // poređenje po tipu i vrednosti
```

```
true  
true  
false  
false  
false, true  
true, false  
true  
false
```

JavaScript

Logički operatori

```
console.log((true && false) + ", " + (true && true));  
console.log((true || false) + ", " + (true || true));  
console.log(!false + ", " + !true);
```

```
false, true  
true, true  
true, false
```


JavaScript

Izrazi

```
let a = 4;  
let ime = "Pera";  
  
console.log(3 + 2);  
console.log("ime: " + ime);  
console.log(3 + a);  
console.log(3 + Math.sqrt(9));  
console.log(3 + a + Math.sqrt(9));
```

```
console.log(3 + 2 * 2);  
console.log((3 + 2) * 2);
```

```
console.log((1 == 1)? "tačno": "netačno");  
console.log((1 != 1)? "tačno": "netačno");
```

```
5  
ime: Pera  
7  
6  
10  
  
7  
10  
  
tačno  
netačno
```

JavaScript

Selekcije

```
let sati;  
  
sati = 10;  
if (sati < 12) {  
    console.log("prepodne");  
}  
  
sati = 13;  
if (sati < 12) {  
    console.log("prepodne");  
} else {  
    console.log("poslepodne");  
}  
  
sati = 20;  
if (sati < 10) {  
    console.log("Dobro jutro!");  
} else if (sati < 18) {  
    console.log("Dobar dan!");  
} else {  
    console.log("Dobro veče!");  
}
```

prepodne

poslepodne

Dobro veče!

JavaScript

Switch

```
let biljka = "banana";

let rezultat;
switch (biljka) {
  case "banana":
  case "pomorandža":
    rezultat = "voće";
    break;
  case "krompir":
    rezultat = "povrće";
    break;
  default:
    rezultat = "nedefinisana";
}
console.log(biljka + " je " + rezultat);
```

banana je voće

JavaScript

Petlje

```
let granica = 3;

let it1 = 1;
while (it1 <= granica) {
    console.log("while: " + it1);
    it1++;
}

let it2 = 1;
do {
    console.log("do-while: " + it2);
    it2++;
} while (it2 <= granica)

for (let it3 = 1; it3 <= granica; it3++) {
    console.log("for A: " + it3);
}
```

while: 1
while: 2
while: 3

do-while: 1
do-while: 2
do-while: 3

for: 1
for: 2
for: 3

JavaScript

break, continue

Važi za sve vrste petlji.

```
let granica = 5;
let preskoci = 2;
let prekini = 4;
for (let it = 1; it <= granica; it++) {
  if (it == preskoci) {
    continue;
  }
  if (it == prekini) {
    break;
  }
  console.log("for B: " + it);
}
```

for: 1

for: 3

JavaScript

Nizovi

```
let brojevi;

brojevi = ["one", "dva", "tri"]; // 1. način
brojevi = Array.of("one", "dva", "tri"); // 2. način
brojevi = new Array("one", "dva", "tri"); // 3. način
console.log(brojevi);

console.log(typeof brojevi);

let kopija = Array.from(brojevi); // plitka (shallow) kopija niza
console.log(kopija);

brojevi = []; // 1. način
brojevi = new Array(); // 2. način
brojevi[0] = "one";
brojevi[1] = "dva";
brojevi[2] = "tri";
console.log(brojevi);

console.log(brojevi.includes("jedan") + ", " + brojevi.includes("one")); // da li sadrži element?

brojevi[0] = "jedan"; // zamena
brojevi.push("četiri"); // dodavanje na kraj
console.log(brojevi);

let uklonjen = brojevi.pop(); // uklanjanje sa kraja
console.log(brojevi);
console.log(uklonjen);

brojevi = brojevi.slice(1, 3); // izdvajanje elemenata u opsegu; vraća modifikovani niz
console.log(brojevi);

brojevi.splice(1, 0, "dva i po"); // umetanje elemenata; menja postojeći niz
console.log(brojevi);

let uklonjeni = brojevi.splice(1, 1); // izbacivanje elemenata; menja postojeći niz
console.log(brojevi);
console.log(uklonjeni);
```

Array(3) ["one", "dva", "tri"]

object

Array(3) ["one", "dva", "tri"]

Array(3) ["jedan", "dva", "tri"]

false, true

Array(4) ["jedan", "dva", "tri", "četiri"]

Array(3) ["jedan", "dva", "tri"]
četiri

Array ["dva", "tri"]

Array ["dva", "dva i po", "tri"]

Array ["dva", "tri"]
Array ["dva i po"]

JavaScript

for-each

```
let brojevi = ["jedan", "dva", "tri"];  
for (let it in brojevi) {  
    console.log("it: " + it + ", broj: " + brojevi[it]);  
}  
  
for (let itBroj of brojevi) {  
    console.log("broj: " + itBroj);  
}
```

it: 0, broj: jedan
it: 1, broj: dva
it: 2, broj: tri

broj: jedan
broj: dva
broj: tri


JavaScript

Funkcije

Funkcije se definišu ključnom rečju *function*.

Ne navodi se tip za parametre i povratne vrednosti.

Ako funkcija nema return naredbu, tada vraća *undefined*.



```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Funkcije</title>
  <script type="text/javascript">
    function naslov() {
      console.log("Primer sa funkcijama");
      console.log("-----");
    }
    function saberi(operandA, operandB) {
      return operandA + operandB;
    }
    console.log(typeof naslov);
  </script>
</head>
<body>
  <h1>Pritisnite F12 da biste otvorili konzolu.</h1>
  <script type="text/javascript">
    naslov();

    let a = 3;
    let b = 2;
    let rezultat = saberi(a, b);
    console.log(a + " + " + b + " = " + rezultat);
  </script>
</body>
</html>
```

function

Primer sa funkcijama

3 + 2 = 5

JavaScript

Funkcije

Funkcije su objekti 1. reda, tj. mogu da:

- budu dodeljene promenljivoj
- se dodaju u niz
- budu parametri drugih funkcija

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Funkcije</title>
  <script type="text/javascript">
    function saberi(operandA, operandB) { // 1. način
      return operandA + operandB;
    }
    let oduzmi = function(operandA, operandB) { // 2. način
      return operandA - operandB;
    }
    let operacije = [saber, oduzmi]; // funkcije su objekti 1. reda
  </script>
</head>
<body>
  <h1>Pritisnite F12 da biste otvorili konzolu.</h1>
  <script type="text/javascript">
    let a = 3;
    let b = 2;
    let sabiranje = operacije[0](a, b);
    console.log(a + " + " + b + " = " + sabiranje);
    let oduzimanje = operacije[1](a, b);
    console.log(a + " - " + b + " = " + oduzimanje);
  </script>
</body>
</html>
```

reference na funkcije

niz funkcija

3 + 2 = 5

3 - 2 = 1

JavaScript

Funkcije

Funkcije su objekti 1. reda, tj. mogu da:

- budu dodeljene promenljivoj
- se dodaju u niz
- budu parametri drugih funkcija

```
let brojevi = [3, 4, 2, 1, 5];
// funkcije su objekti 1. reda; callback je funkcija koja je parametar neke druge funkcije
let comp = function(broj1, broj2) { // 1. način (imenovana funkcija)
    return broj1 - broj2;
}
brojevi.sort(comp);

brojevi.sort(function(broj1, broj2) { // 2. način (bezimena funkcija)
    return broj1 - broj2;
});

brojevi.sort((broj1, broj2) => { // 3. način (arrow funkcija, tj. lambda izraz)
    return broj1 - broj2;
});
console.log(brojevi);
```

Array(5) [1, 2, 3, 4, 5]

JavaScript

Ugrađene funkcije

```
// provera da li se string ne može parsirati u broj  
console.log(isNaN("nije broj") + ", " + isNaN("1.5"));
```

true, false

```
// izvršava string kao JavaScript kod  
eval("for (let it = 0; it < 3; it++) {console.log(it)}");
```

0

1

2

```
console.log(parseInt("1")); // parsira string celobrojni number  
console.log(parseFloat("1.5")); // parsira string u razlomljeni number
```

1

1.5

```
let kodiraniURL = escape("localhost:8080/URL sa razmakom"); // kodira url  
console.log(kodiraniURL);
```

localhost%3A8080/URL%20sa%20razmakom

```
let dekodiraniURL = unescape(kodiraniURL); // dekodira url  
console.log(dekodiraniURL);
```

localhost:8080/URL sa razmakom

JavaScript

Number klasa

```
console.log(Number.MAX_VALUE);  
console.log(Number.MIN_VALUE);  
console.log(Number.POSITIVE_INFINITY);  
console.log(Number.NEGATIVE_INFINITY);  
console.log(Number.NaN);  
console.log(Number.parseInt("1.5"));  
console.log(Number.parseFloat("1.5"));  
console.log(Number.isInteger(1.5) + ", " + Number.isInteger(1));
```

```
let broj;
```

```
broj = 1.532465; // 1. način  
console.log(typeof broj);
```

```
broj = Number("1.532465"); // 2. način  
console.log(typeof broj);
```

```
broj = new Number("1.532465"); // 3. način  
console.log(typeof broj);
```

```
console.log(broj.toFixed(2)); // broj cifara iza decimalne tačke
```

```
broj = new Number("nije broj");  
console.log(broj);
```

```
1.7976931348623157e+308  
5e-324  
Infinity  
-Infinity  
NaN  
1  
1.5  
false, true
```

```
number
```

```
number
```

```
object
```

```
1.53
```

```
Number { NaN }
```

JavaScript

String klasa

```
let string;

string = 'Hello World!'; // string literali mogu se navode apostrofima ili navodnicima
string = "Hello World!"; // 1. način
console.log(typeof string);

string = String("Hello World!"); // 2. način
console.log(typeof string);

string = new String("Hello World!"); // 3. način
console.log(typeof string);

console.log(string.length);
console.log(string.substring(1, 10));
console.log(string.split(" "));
console.log(string.indexOf("l"));
console.log(string.lastIndexOf("l"));
console.log(string.charAt(6));

console.log("a".localeCompare("b")); // kao compare(...) u jeziku Java
console.log("a".localeCompare("a"));
console.log("b".localeCompare("a"));

string = "$%^ ^&*()";
// kao matches(...) u jeziku Java, ali matches(...) vraća boolean
console.log(string.match("[a-zA-Z0-9]+$") != null);
string = "1a";
console.log(string.match("[a-zA-Z0-9]+$") != null);

let id = 1;
let naziv = "Film 1";
// template literal se navode obrnutim apostrofima
console.log(`<a href="/filmovi/prikazid=${id}">${naziv}</a>`);
```

string

string

object

12
ello Worl
Array ["Hello", "World!"]
2
9
W

-1
0
1

false

true

Film 1

JavaScript

Date klasa

```
let datum;  
  
datum = Date("2020-01-01T12:00"); // ISO datum i vreme  
console.log(typeof datum);  
  
datum = new Date("2020-01-01T12:00");  
console.log(typeof datum);  
  
console.log(datum);  
console.log(datum.valueOf()); // number reprezentacija datuma  
  
datum = new Date("nije datum");  
console.log(datum);  
console.log(datum.valueOf());  
  
console.log(Date.now())  
datum = new Date(Date.now());  
console.log(datum);  
  
console.log(`${datum.getFullYear()}-${datum.getMonth() + 1}-${datum.getDate()}`);  
console.log(`${datum.getHours()}:${datum.getMinutes()}`);  
  
console.log(datum.toISOString()); // ISO datum i vreme  
console.log(datum.toISOString().split("T")[0]); // ISO datum  
console.log(datum.toISOString().split("T")[1]); // ISO vreme
```

string

object

Date Wed Jan 01 2020 12:00:00 GMT+0100 (Central European Standard Time)
1577876400000

Invalid Date
NaN

1716892087368

Date Tue May 28 2024 12:28:07 GMT+0200 (Central European Summer Time)

2024-5-28
12:28

2024-05-28T10:28:07.368Z
2024-05-28
10:28:07.368Z

JavaScript

Map klasa

```
let proizvodi = new Map();
proizvodi.set("0001", "proizvod1");
proizvodi.set("0002", "proizvod2");
console.log(proizvodi);

console.log(typeof proizvodi);

console.log(proizvodi.has("0001") + ", " + proizvodi.has("0003")); // da li sadrži ključ?
console.log(proizvodi.get("0001"));

// iteracija po vrednostima
for (let itProizvod of proizvodi.values()) {
  console.log(itProizvod);
}
// iteracija po parovima (ključ, vrednost)
for (let [itSifra, itProizvod] of proizvodi.entries()) {
  console.log(itSifra + ": " + itProizvod);
}

proizvodi.delete("0001");
console.log(proizvodi);
```

Map { 0001 → "proizvod1", 0002 → "proizvod2" }

object

true, false
proizvod1

proizvod1
proizvod2

0001: proizvod1
0002: proizvod2

Map { 0002 → "proizvod2" }

JavaScript

JSON

JSON (*Java Script Object Notation*) je tekstualna sintaksa kojom se može zapisati proizvoljna hijerarhijska struktura podataka.



```
{
  "filmovi": [
    {
      "id": 1,
      "naziv": "Avengers: Endgame",
      "trajanje": 182,
      "zanrovi": [
        {
          "id": 1,
          "naziv": "naučna fantastika"
        },
        {
          "id": 2,
          "akcija": "akcija"
        }
      ]
    },
    {
      "id": 2,
      "naziv": null,
      "trajanje": null,
      "zanrovi": []
    }
  ]
}
```


JavaScript

Objekti

JavaScript podržava kreiranje **objekata bez klasa**.

Objekti se definišu JSON sintaksom:

- Definicija *JavaScript* objekta se navodi između znakova `{ i }`
- Navode se parovi **"nazivAtributa": vrednostAtributa** odvojeni znakom `,`
- Vrednost atributa može biti primitivnog tipa ili referenca na drugi objekat, pa i niz

```
let film = {  
  "id": 1,  
  "naziv": "Avengers: Endgame",  
  "trajanje": 182  
};  
console.log(film);  
console.log(typeof film);
```

Object { id: 1, naziv: "Avengers: Endgame", trajanje: 182 }
object

JavaScript

Objekti

Atributima (*properties*) se pristupa po nazivu.

Izmena se vrši operatorima dodele.

```
console.log("id: " + film.id); // 1. način  
console.log("id: " + film["id"]); // 2. način  
console.log("naziv: " + film.naziv);  
console.log("trajanje: " + film.trajanje);
```

```
film.id = "2"; // 1. način  
film["id"] = 2; // 2. način  
film.naziv = "Life";  
film.trajanje = 110;  
console.log(film);
```

```
id: 1  
id: 1  
naziv: Avengers: Endgame  
trajanje: 182
```

```
Object { id: "2", naziv: "Life", trajanje: 110 }
```

JavaScript

Objekti

Atributima (*properties*) se pristupa po nazivu.

Izmena se vrši operatorima dodele.

```
let atribut = "id";  
console.log("id: " + film.id); // 1. način  
console.log("id: " + film[atribut]); // 2. način  
console.log("naziv: " + film.naziv);  
console.log("trajanje: " + film.trajanje);
```

```
film.id = "2"; // 1. način  
film[atribut] = 2; // 2. način  
film.naziv = "Life";  
film.trajanje = 110;  
console.log(film);
```

id: 1

id: 1

naziv: Avengers: Endgame

trajanje: 182

Object { id: "2", naziv: "Life", trajanje: 110 }

JavaScript

Objekti

Atributi se mogu dodati u već kreirani objekat.

Atributi se nakon dodavanja mogu ukloniti operatorom *delete*.

```
film.zanr = "horor";  
console.log(film);  
  
delete film.zanr;  
console.log(film);
```

```
Object { id: "2", naziv: "Life", trajanje: 110, zanr: "horor" }
```

```
Object { id: "2", naziv: "Life", trajanje: 110 }
```

JavaScript

Objekti

Atributi mogu biti i reference na objekte i čak i nizovi referenci.

```
film.zanr = {"id": 4, "naziv": "horor"};  
console.log(film);
```

```
{...}  
  id: "2"  
  naziv: "Life"  
  trajanje: 110  
  zanr: Object { id: 4, naziv: "horor" }
```

```
film.zanrovi = [  
  {"id": 1, "naziv": "naučna fantastika"},  
  {"id": 4, "naziv": "horor"}  
];  
console.log(film);
```

```
{...}  
  id: "2"  
  naziv: "Life"  
  trajanje: 110  
  zanrovi: (2) [...]  
    0: Object { id: 1, naziv: "naučna fantastika" }  
    1: Object { id: 4, naziv: "horor" }
```

JavaScript

Objekti

JSON.stringify(...) funkcija serijalizuje JSON objekat u *String* koji sadrži upravo onu sintaksu kojom je kreiran taj objekat.

JSON.parse(...) funkcija deserijalizuje *String* koji je napisan JSON sintaksom u odgovarajući objekat.

```
let jsonFilm = JSON.stringify(film);
console.log(jsonFilm);
console.log(jsonFilm.naziv)

film = JSON.parse(jsonFilm);
console.log(film);
console.log(film.naziv);
```

```
{ id: "2", naziv: "Life", trajanje: 110 }
undefined
```

```
Object { id: "2", naziv: "Life", trajanje: 110 }
Life
```

JavaScript

Objekti

Funkcije su objekti 1. reda pa mogu biti atributi drugih objekata (njihove metode).

Metode se takođe mogu dinamički dodavati i uklanjati.

Atributima unutar metoda se uvek mora pristupiti kroz referencu *this*.

```
let film = {  
  // atributi  
  "id": 1,  
  "naziv": "Avengers: Endgame",  
  "trajanje": 182,  
  "zanrovi": [], // many-to-many  
  // metode  
  "addZanr": function(zanr) { // 1. način  
    if (this.zanrovi.includes(zanr)) {  
      return;  
    }  
    this.zanrovi.push(zanr);  
  },  
  removeZanr(zanr) { // 2. način  
    let index = this.zanrovi.indexOf(zanr);  
    if (index > -1) {  
      this.zanrovi.splice(index, 1);  
    }  
  }  
};  
  
film.addZanr({"id": 1, "naziv": "naučna fantastika"});  
console.log(film);
```

paziti na zarez

```
Object {  
  id: 1  
  naziv: "Avengers: Endgame"  
  trajanje: 182  
  zanrovi: Array [  
    0: Object { id: 1, naziv: "naučna fantastika" }  
    length: 1  
    <prototype>: Array []  
  ]  
  addZanr: function addZanr(zanr)  
  removeZanr: function removeZanr(zanr)  
  <prototype>: Object { ... }  
}
```

JavaScript

Klase

Javni atributi ne moraju da se deklariraju.

```
class Film {
  // može da postoji najviše 1 konstruktor
  constructor(id = 0, naziv = "", trajanje = 0) { // opcioni parametri
    this.id = id;
    this.naziv = naziv;
    this.trajanje = trajanje;
    this.zanrovi = []; // many-to-many
  }

  addZanr(zanr) {
    if (this.zanrovi.includes(zanr)) {
      return;
    }
    this.zanrovi.push(zanr);
  }

  removeZanr(zanr) {
    let index = this.zanrovi.indexOf(zanr);
    if (index > -1) {
      this.zanrovi.splice(index, 1);
    }
  }
}

let film = new Film(1, "Avengers: Endgame", 182);
console.log(film);
console.log(typeof film);
```

```
Object {
  id: 1
  naziv: "Avengers: Endgame"
  trajanje: 182
  zanrovi: Array []
  <prototype>: Object {
    constructor: class Film { constructor(id, naziv, trajanje) }
    addZanr: function addZanr(zanr)
    removeZanr: function removeZanr(zanr)
    <prototype>: Object { ... }
  }
}
object
```


JavaScript

Klase

Privatni atributi moraju da se deklarišu i navode se sa prefiksom #.

```
class Film {  
  #id;  
  #naziv;  
  #trajanje;  
  #zanrovi;  
  
  constructor(id = 0, naziv = "", trajanje = 0) {  
    this.#id = id;  
    this.#naziv = naziv;  
    this.#trajanje = trajanje;  
    this.#zanrovi = [];  
  }  
  :  
}  
  
let film = new Film();  
film.#id = 1; // nije dozvoljeno  
console.log(film.#id); // nije dozvoljeno  
film.id = 1;  
console.log(film.id);  
console.log(film);
```

1

```
Object {  
  id: 1  
  #id: 0  
  #naziv: ""  
  #trajanje: 0  
  #zanrovi: Array []  
  <prototype>: Object { ... }  
}
```

JavaScript

Getter-i i setter-i

Mogu da ih imaju i objekti bez klasa.

```
class Film {  
  #id;  
  #naziv;  
  #trajanje;  
  #zanrovi;  
  
  constructor(id = 0, naziv = "", trajanje = 0) {  
    this.#id = id;  
    this.#naziv = naziv;  
    this.#trajanje = trajanje;  
    this.#zanrovi = [];  
  }  
  
  get id() {  
    return this.#id;  
  }  
  
  set id(id) {  
    this.#id = id;  
  }  
  :  
}  
  
let film = new Film();  
film.#id = 1; // nije dozvoljeno  
console.log(film.#id); // nije dozvoljeno  
film.id = 1;  
console.log(film.id);  
console.log(film);
```



1

```
Object {  
  #id: 1  
  #naziv: ""  
  #trajanje: 0  
  #zanrovi: Array []  
  <prototype>: Object { ... }  
}
```

JavaScript

Klase

Unutar *static* funkcija, *static* atributima se takođe mora pristupati kroz referencu *this*.

```
class Bioskop {
    static zanrovi = new Map();

    static inicijalizuj() {
        // kreiranje žanrova
        this.zanrovi.set(1, new Zanr(1, "naučna fantastika"));
        this.zanrovi.set(2, new Zanr(2, "akcija"));
        this.zanrovi.set(3, new Zanr(3, "komedija"));
        this.zanrovi.set(4, new Zanr(4, "horor"));
        this.zanrovi.set(5, new Zanr(5, "avantura"));

        :
    }
    :
}

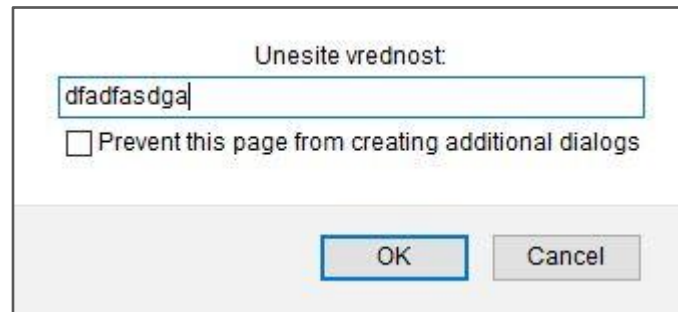
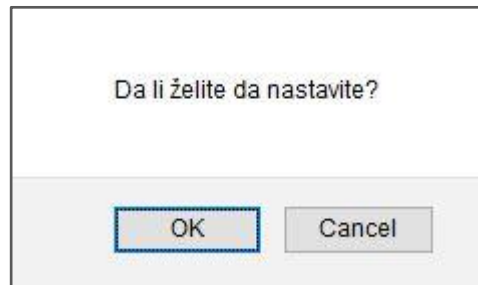
Bioskop.inicijalizuj();
```

JavaScript

Popup Boxes

primer26.html

```
// prikazuje pitanje i vraća korisnikov odgovor u vidu boolean rezultata
let nastavak = confirm("Da li želite da nastavite?");
if (nastavak) {
    // traži unos od korisnika i vraća unesenu vrednost
    let vrednost = prompt("Unesite vrednost:");
    // prikazuje poruku
    alert("Uneli ste: " + vrednost);
}
```



Primeri

- *primer1.html, primer2.html, ...*

JavaScript

script element

Sadržaj *script* elementa je moguće izvesti u eksternu datoteku sa ekstenzijom *.js* i referencirati je *src* atributom.

I tada definicije i naredbe ostaju u izvornom obliku i ne prevode se.

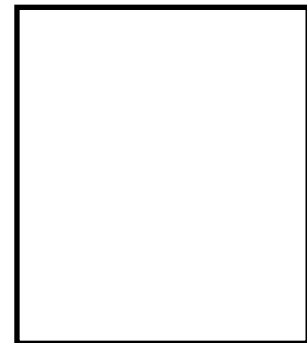
Ovo omogućuje da se iste definicije i naredbe referenciraju u više *script* elemenata i čak i u različitim HTML dokumentima.

JavaScript biblioteke se sastoje od jedne ili više *.js* datoteka.

```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript" src="eksterna_datoteka.js"></script>
  </head>
  <body>

  </body>
</html>
```

eksterna_datoteka.js



JavaScript

script element

I spoljašnje skripte se izvršavaju onog momenta kada se učitava *script* element koji ih referencira.

U slučaju da naredbe u skriptama zavise od HTML elemenata koji stoje iza *script* elementa (jer bi se tada učitali nakon njega), one tada ne bi radile.

Dodatno, izvršenje naredbi iz *script* elementa ne čeka na izvršenje naredbi iz prethodnog *script* elementa. Ukoliko naredbe u njemu zavise od definicija iz nekog od prethodnih *script* elemenata, one tada ne bi radile.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Bioskop</title>

  <script src="js/model/zanr.js"></script>
  <script src="js/model/film.js"></script>
  <script src="js/model/projekcija.js"></script>
  <script src="js/model/korisnik.js"></script>
  <script src="js/model/bioskop.js"></script>

  <script src="js/util/konzola.js"></script>

  <script src="js/ui/zanrUI.js"></script>
  <script src="js/ui/filmUI.js"></script>
  <script src="js/ui/projekcijaUI.js"></script>
  <script src="js/ui/korisnikUI.js"></script>
  <script src="js/ui/applicationUI.js"></script>
</head>
<body>
  <h1>Pritisnite F12 da biste otvorili konzolu.</h1>
</body>
</html>
```

JavaScript

script element

Atribut *defer*:

1. Odlaže izvršavanje skripte do momenta nakon učitavanja kompletnog HTML dokumenta.
2. Kao posledicu prethodnog obezbeđuje da se skripte izvrše **u redosledu navođenja, nakon učitavanja kompletnog HTML dokumenta.**

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Bioskop</title>

  <script defer src="js/model/zanr.js"></script>
  <script defer src="js/model/film.js"></script>
  <script defer src="js/model/projekcija.js"></script>
  <script defer src="js/model/korisnik.js"></script>
  <script defer src="js/model/bioskop.js"></script>

  <script defer src="js/util/konzola.js"></script>

  <script defer src="js/ui/zanrUI.js"></script>
  <script defer src="js/ui/filmUI.js"></script>
  <script defer src="js/ui/projekcijaUI.js"></script>
  <script defer src="js/ui/korisnikUI.js"></script>
  <script defer src="js/ui/applicationUI.js"></script>
</head>
<body>
  <h1>Pritisnite F12 da biste otvorili konzolu.</h1>
</body>
</html>
```


Primer

- *Modul2Termin9Bioskop*

Zadatak 1

Po ugledu na primer *Modul2Termin9Bioskop*, implementirati konzolnu *JavaScript* aplikaciju za evidenciju dostave hrane:

1. Napraviti prazan HTML dokument *dostava.html*
2. Uz HTML dokument napraviti direktorijum *js*, a u njemu direktorijume *model*, *util* i *ui*
3. U direktorijumu *model* napraviti datoteke *kategorija.js*, *restoran.js*, *artikal.js* i *dostava.js*
4. U direktorijumu *ui* napraviti datoteke *kategorijaUI.js*, *restoranUI.js*, *artikalUI.js* i *applicationUI.js*
5. U direktorijum *util* kopirati datoteku *konzolaUI.js*
6. Uvezati skripte u datoteku *dostava.html* u odgovarajućem redosledu
7. Implementirati i testirati aplikaciju

Dodatni materijali

- <https://www.w3schools.com/js/>
- https://www.w3schools.com/jsref/jsref_obj_regexp.asp
- <https://www.w3schools.com/jsref/>
- <https://www.youtube.com/watch?v=et8xNAc2ic8> 😊