Osnove rada sa nizovima

Osnovni sastojak gotovo svih programskih jezika su nizovi. Nizovi omogućavaju smeštanje većeg broja vrednosti unutar jedne promenljive. I u JavaScript jeziku nizovi zauzimaju veoma značajno mesto, pa će modul pred vama biti posvećen radu sa nizovima.

Pojam nizova u JavaScriptu

U JavaScriptu nizovi su objekti. Tako se za kreiranje nizova u pozadini koristi globalni objekat `Array`. Ovaj objekat sadrži različita svojstva i metode koje se mogu koristiti za operacije nad nizovima i njihovim članovima. Tako će u nastavku lekcije prvo biti demonstrirane osnovne osobine JavaScript nizova, a zatim i različita svojstva i metode `Array` objekta.

JavaScript nizovi su dinamički, što znači da nemaju unapred utvrđenu dužinu. Njihova dužina se može menjati tokom izvršavanja skripte. Takođe, ni tipovi podataka koji se smeštaju unutar nizova nisu fiksni, te je tako unutar jednog niza moguće kombinovati vrednosti različitih tipova.

Kreiranje nizova

JavaScript nizovi kreiraju se na veoma lak način. Kako bi se na pravi način razumela svrha nizova, prvo će biti kreirana jedna promenljiva:

```
var car = "Ford";
```

Ovo je promenljiva sa nazivom `car` i vrednošću `Ford`. Kada bi promenljiva `car`, pored vrednosti `Ford`, mogla da sadrži i neke druge vrednosti koje bi predstavljale i ostale proizvođače automobila, dobio bi se jedan niz:

```
var cars = ["Ford", "Mazda"];
```

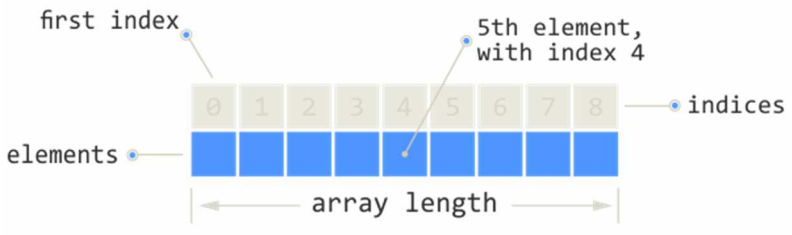
Prikazana naredba ilustruje kreiranje jednog niza. Nizovi se formiraju korišćenjem uglastih zagrada, unutar kojih se navode vrednosti, razdvojene zapetama. Struktura jednog niza prikazana je slikom 15.1.

Slika 15.1. Struktura jednog niza

Opšta sintaksa za kreiranje nizova u JavaScript jeziku je sledeća:

```
var array-name = [item1, item2, ...];
```

Slika 15.2. dočarava strukturu jednog niza.



Slika 15.2. Struktura niza

Niz je sastavljen iz elemenata, koji imaju svoje pozicije izražene kroz indekse. Indeksi počinju od nule. Dužina niza predstavlja ukupan broj elemenata u nizu.

Napomena

JavaScript niz je moguće kreirati na još jedan način, koji podrazumeva upotrebu ključne reči `new` i konstruktorske funkcije `Array`:

```
var cars = new Array("Ford", " Mazda");
```

Oba načina za kreiranje nizova proizvode identičan efekat. Zbog jednostavnosti, brzine i preglednosti, preporučuje se prvi pristup.

Pitanje

U JavaScriptu nizovi su:

- prosti tipovi
- **složeni tipovi**

Objašnjenje:

U JavaScriptu nizovi su objekti. Tako se za kreiranje nizova u pozadini koristi globalni objekat `Array`. Zbog toga nizovi nisu prosti tipovi.

Čitanje elemenata niza

Elementima niza pristupa se korišćenjem indeksa. Indeksi elemenata unutar niza počinju od nule, tako da prvi element niza ima indeks nula.

```
var cars = ["Ford", "Mazda"];
console.log(cars[0]);
console.log(cars[1]);
```

Primer ilustruje proces čitanja elemenata niza, korišćenjem njihovih indeksa. Kada se izvrši, prikazani kôd će unutar konzole web pregledača da ispiše:

```
Ford
Mazda
```

Moguće je obaviti i prikaz svih elemenata niza, objedinjenih u jedan `string`. Da bi se to postiglo, koristi se jedna od metoda koju poseduju svi ugrađeni JavaScript objekti - `toString()`:

```
var cars = ["Ford", "Mazda"];
console.log(cars.toString());
```

Na ovaj način se unutar konzole dobija objektna reprezentacija niza `cars`:

```
Ford,Mazda
```

Promena vrednosti elementa niza

Na veoma lak način je moguće izvršiti promenu vrednosti nekog elementa u nizu:

```
var cars = ["Ford", "Mazda"];
cars[1] = "BMW";
console.log(cars[0]);
console.log(cars[1]);
```

Nakon izvršavanja prikazanog koda, unutar konzole se dobija:

```
Ford
BMW
```

Na osnovu ispisa dobijenog unutar konzole, može se videti da je vrednost drugog elementa niza promenjena.

Dodavanje elementa u niz

JavaScript nizovi su dinamičke prirode, što znači da nemaju fiksnu, unapred definisanu dužinu (*kao kod jezika C, na primer*). To znatno olakšava izmenu strukture niza nakon inicijalizacije početnih elemenata.

```
var cars = ["Ford", "Mazda"];
```

Prikazanom naredbom obavlja se kreiranje niza sa dva elementa. Elementi unutar niza imaju indekse 0 i 1. Ako je potrebno dodati novi element unutar niza, to se može postići jednostavnim postavljanjem vrednosti elementa sa indeksom 2:

```
cars[2] = "Honda";
```

Niz cars sada poseduje tri elementa:

```
console.log(cars[0]);
console.log(cars[1]);
console.log(cars[2]);
```

Ovakav kôd će proizvesti sledeći ispis:

```
Ford
Mazda
Honda
```

Na osnovu dobijenog ispisa, jasno se vidi da je dodavanje elementa uspešno obavljeno.

JavaScript obezbeđuje veliku slobodu pri radu sa nizovima, tako da njihove elemente uopšte nije neophodno smeštati sukcesivno:

```
cars[4] = "Toyota";
```

Sada je nizu `cars` dodat još jedan element, ali na poziciju sa indeksom 4. To praktično znači da je preskočena pozicija sa indeksom 3, što je u JavaScriptu potpuno legitimno, ali retko i opravdano. Naime, ovakva praksa uglavnom može stvoriti probleme, s obzirom na to da je potrebno voditi računa o indeksima na kojima se elementi nalaze.

Čitanje dužine niza

Čitanje dužine niza postiže se korišćenjem svojstva **length:**

```
var cars = ["Ford", "Mazda"];
var length = cars.length;
console.log(length);
```

Prikazani kôd proizvodi rezultat:

```
2
```

Za realizaciju čitanja dužine niza prvi put je iskorišćen jedan element Array objekta (`length`). U nastavku će biti iskorišćeno još svojstava i metoda takvog objekta.

Sortiranje niza

Sortiranje niza može se obaviti korišćenjem metode **sort():**

```
var cars = ["Ford", "Mazda", "Honda", "Toyota", "Acura"];
cars.sort();
console.log(cars.toString());
```

Kôd proizvodi:

```
Acura,Ford,Honda,Mazda,Toyota
```

Može se primetiti da su elementi niza ispisani alfabetnim redosledom, što znači da je sortiranje niza korektno obavljeno. Ipak, ukoliko se metoda sort() primeni nad nizom sa numeričkim vrednostima, dolazi do neočekivanih rezultata:

```
var numbers = [1, 2, 55, 23, 0, 11, 123, -33];
numbers.sort();
console.log(numbers.toString());
```

Ovakav kôd sada proizvodi:

```
-33,0,1,11,123,2,23,55
```

Analizom dobijenog ispisa mogu se primetiti neke nelogičnosti. Na primer, vrednost 123 navedena je pre vrednosti 2, 23 i 55. Razlog je vrlo jednostavan. Metoda `sort()` prilikom sortiranja sve članove nizova podrazumevano tretira kao da su tekst. Tako se članovi niza iz primera, i pored toga što su `number` tipa, prvo prevode u `string`, a zatim sortiraju na osnovu indeksa njihovih karaktera u Unicode tabeli. Upravo zbog toga je vrednost 123 navedena pre vrednosti 2. Jednostavno, prvi karakter vrednosti 123 je 1, što je manje od 2, pa je otuda i kompletna vrednost 123 manja od vrednosti 2, kada se govori o tekstualnom obliku. Upravo zbog toga, `sort()` metoda u svom izvornom obliku se ne može koristiti za sortiranje numeričkih nizova. Ipak, sortiranje brojeva korišćenjem `sort()` metode je moguće postići, ali je prethodno neophodno upoznati se sa sintaksom ove metode:

```
sort(compareFunction)
```

Metoda `sort()` može da prihvati jedan parametar kojim je moguće definisati funkciju za sortiranje (`compareFunction`). Osobine funkcije za sortiranje moraju biti sledeće:

- funkcija za sortiranje mora imati **dva ulazna parametra,**
- funkcija za sortiranje mora emitovati **jednu od sledeće tri vrednosti**:
    - negativnu vrednost kada je prvi parametar manji od drugog,
    - vrednost 0 kada su dva prosleđena parametra jednaka,
    - pozitivnu vrednost kada je prvi parametar veći od drugog.

Povratna vrednost metode sort()

Metoda `sort()` emituje i povratnu vrednost, koja predstavlja sortirani niz. Ipak, metoda `sort()` se **ne** izvršava nad kopijom, već nad stvarnim nizom, tako da u većini slučajeva nema ni potrebe za korišćenjem njene povratne vrednosti, zato što se sortiranje ujedno obavlja i nad nizom nad kojim je metoda pozvana.

Uzimajući u obzir sve do sada navedene osobine funkcije za poređenje, kada se koristi za poređenje brojeva, ona može da izgleda ovako:

```
function compareNumbers(a, b) {
      return a - b;
}
```

Funkcija `compareNumbers()` u potpunosti zadovoljava specifikacije funkcije za poređenje. Za početak, ona prihvata dva parametra (`a` i `b`). Unutar tela funkcije obavlja se oduzimanje drugog parametra od prvog i kao povratna se emituje dobijena vrednost. Ukoliko je prvi parametar veći od drugog, funkcija će vratiti pozitivnu vrednosti. Ukoliko je drugi parametar veći od prvog, funkcija vraća negativnu vrednost. Na kraju, ukoliko su parametri jednaki, kao rezultat se emituje nula. Na taj način se u potpunosti zadovoljava specifikacija funkcije za poređenje:

```
function compareNumbers(a, b) {
      return a - b;
}

var numbers = [1, 2, 55, 23, 0, 11, 123, -33];
numbers.sort(compareNumbers);
console.log(numbers.toString());
```

Unutar konzole se dobija sortiran numerički niz:

```
-33,0,1,2,11,23,55,123
```

Funkcija kao parametar funkcije

U upravo prikazanom primeru mogli ste videti jednu zanimljivu osobinu JavaScript funkcija. Naime, one se mogu prosleđivati kao parametri drugim funkcijama, kao da je reč o promenljivama. U nekim situacijama to može biti i više nego korisno. Upravo jedna od takvih situacija jeste i prikazani primer sortiranja numeričkih vrednosti, kod koga ova mogućnost JavaScript jezika omogućava da se pravila sortiranja nizova prilagode trenutnim potrebama.

Dodavanje elemenata korišćenjem metoda Array objekta

Kada je dodavanje elemenata u pitanju, pored bazičnog pristupa, koji podrazumeva da se poznaje indeks na koji će element biti ubačen, moguće je koristiti i ugrađene metode `push()` i `unshift()`.

Metoda **push()** dodaje element na kraj niza:

```
var cars = ["Ford", "Mazda"];
var newLength = cars.push("Honda");

console.log(newLength);
console.log(cars.toString());
```

Ispis u konzoli:

```
3
Ford,Mazda,Honda
```

Metoda **unshift()** dodaje element na početak niza:

```
var cars = ["Ford", "Mazda"];
var newLength = cars.unshift("Honda");

console.log(newLength);
console.log(cars.toString());
```

Ispis u konzoli:

```
3
Honda,Ford,Mazda
```

Obe metode kao svoju povratnu vrednost imaju novu dužinu niza. U prikazanim primerima ta nova dužina niza smešta se unutar promenljive `newLength`, a taj podatak se, zajedno sa elementima niza, ispisuje unutar konzole.

Uklanjanje elemenata iz niza

Za uklanjanje elemenata iz niza objekat `Array` poseduje dve metode: `pop()` i `shift()`.

Osnovna razlika između ove dve metode jeste lokacija sa koje se obavlja uklanjanje elementa.

Za uklanjanje elementa sa kraja niza koristi se metoda **pop()**:

```
var cars = ["Ford", "Mazda"];
var last = cars.pop();

console.log(last);
console.log(cars.toString());
```

Metoda `pop()` kao svoju povratnu vrednost ima element koji je uklonila iz niza. Tako će prikazani kôd proizvesti efekat:

```
Mazda
Ford
```

Da bi se element uklonio sa početka niza, koristi se metoda **shift()**:

```
var cars = ["Ford", "Mazda"];
var last = cars.shift();

console.log(last);
console.log(cars.toString());
```

Efekat je:

```
Ford
Mazda
```

Čitanje indeksa elementa (pretraga niza)

Array objekat poseduje metodu koju je moguće koristiti za pretragu nizova. Naime, reč je o metodi koja za prosleđenu vrednost vraća indeks na kojem se unutar niza takva vrednost nalazi. Reč je o metodi **indexOf()**:

```
var cars = ["Ford", "Mazda"];
var pos = cars.indexOf("Mazda");
console.log(pos);
```

Metoda `indexOf()` prihvata jedan parametar, koji predstavlja vrednost koja će biti potražena unutar niza. Kao svoju povratnu vrednost, metoda `indexOf()` emituje indeks na kojem se nalazi tražena vrednost. U prikazanom primeru to će biti:

```
1
```

Ukoliko tražena vrednost ne postoji unutar niza, metoda indexOf() vraća vrednost -1:

```
var cars = ["Ford", "Mazda"];
var pos = cars.indexOf("Honda");
console.log(pos);
```

Tražena vrednost (`Honda`) ne postoji unutar niza `cars`. Zbog toga je povratna vrednost metode `indexOf()`, a ujedno i vrednost koja se dobija unutar konzole:

```
-1
```

Prolazak kroz niz

Kada se govori o nizovima u programiranju, veoma često se može čuti fraza *prolazak kroz niz*. Reč je zapravo o operaciji koja podrazumeva sukcesivno čitanje svakog elementa pojedinačno i to uglavnom redom.

Prolazak kroz niz se najčešće obavlja kada je potrebno:

- sprovesti određene intervencije nad svakim od elemenata niza ili
- pojedinačno ispisati svaki od elemenata jednog niza.

U ovakvim situacijama ručno navođenje indeksa ne dolazi u obzir, već se mora pribeći nešto drugačijim rešenjima.

Osnovni pristup za postizanje prolaska kroz niz jeste upotreba petlji. Petlje su već obrađene u jednom od prethodnih modula i tada je rečeno da će petlje ponovo biti razmatrane kada bude bilo reči o nizovima. Sem korišćenjem petlji, prolazak kroz niz se može postići i upotrebom jedne metode Array objekta. Tako će u nastavku ove lekcije biti prikazani sledeći pristupi koji omogućavaju prolazak kroz nizove:

- `for` petlja,
- `for...in` petlja,
- `for...of` petlja,
- `forEach()` metoda `Array` objekta.

Prolazak kroz niz korišćenjem for petlje

For petlja se veoma često koristi za prolazak kroz nizove:

```
var arr = ["Honda", "Toyota", "Acura"];

for (let i = 0; i < arr.length; i++) {
      console.log(arr[i]);
}
```

U primeru je deklarisan i inicijalizovan jedan niz sa tri člana. `For` petlja se koristi za prolazak kroz niz i ispis vrednosti elemenata. Za formiranje uslova je iskorišćeno svojstvo `length` objekta `Array`, čime je dobijena dužina niza.

Uslov je formiran tako da se `for` petlja izvršava sve dok je vrednost brojača manja od dužine niza. Tako prikazani primer proizvodi sledeći rezultat:

```
Honda
Toyota
Acura
```

Prolazak kroz niz korišćenjem for...in petlje

Za prolazak kroz nizove JavaScript poseduje još jednu vrste petlje. Reč je o petlji `for...in`.

Petlja `for...in` primarno je dizajnirana za prolazak kroz svojstva objekata. S obzirom na to da su i nizovi jedna vrsta objekata, ova petlja se može koristiti i za prolazak kroz nizove.

Sintaksa `for...in` petlje je sledeća:

```
for (property in object) {
      statements
}
```

`Property` označava promenljivu koja će pri svakoj iteraciji preuzimati vrednost koja se odnosi na naziv svojstva objekta. Drugim rečima, `for...in` petlja dozvoljava prolazak kroz nazive svojstava nekog objekta. To će potvrditi sledeći primer:

```
var arr = ["Honda", "Toyota", "Acura"];

for (let property in arr) {
      console.log(property);
}
```

U primeru je iskorišćena `for...in` petlja za prolazak kroz nazive svojstava jednog niza. Unutar konzole se dobija:

```
0
1
2
```

Promenljiva sa nazivom `property` će u primeru prilikom svake iteracije dobijati vrednost sledećeg naziva svojstva objekta `arr`. S obzirom na to da su nizovi objekti, a nazivi svojstava zapravo indeksi, promenljiva `property` će prilikom svake iteracije dobijati vrednost indeksa elementa (0, 1, 2...).

`For...in` petlja se može iskoristiti i za dobijanje vrednosti članova nizova:

```
var arr = ["Honda", "Toyota", "Acura"];

for (let property in arr) {
      console.log(arr[property]);
}
```

Sada je promenljiva property iskorišćena za definisanje indeksa arr niza, te se na taj način dobijaju vrednosti članova nizova. Ispis unutar konzole je identičan kao i prilikom korišćenja for petlje.

Prolazak kroz niz korišćenjem for...of petlje

Pored for in petlje, prikazane u prethodnim redovima, JavaScript poseduje još jednu petlju koju je moguće koristiti za prolazak kroz nizove. Reč je o petlji for of.

Petlja for of, za razliku od for in petlje, omogućava prolazak kroz vrednosti jednog niza. Sintaksa ove petlje je sledeća:

```
for (value of object) {
      statement
}
```

Sintaksa je praktično identična sintaksi for in petlje. Jedina razlika je u upotrebi ključne reči of umesto in.

Sada se kôd za prolazak kroz niz iz prethodnih primera može napisati ovako:

```
var arr = ["Honda", "Toyota", "Acura"];

for (let car of arr) {
      console.log(car);
}
```

Ovoga puta nema potrebe za čitanjem elemenata niza na osnovu njihovih indeksa, s obzirom na to da u svakoj iteraciji promenljiva car dobija vrednost tekućeg elementa niza.

Napomena

Petlju for of, za razliku od nešto ranije prikazane for in petlje, nije moguće koristiti nad objektima tipa Object, već isključivo nad objektima tipa Array, String i Map.

forEach() metoda Array objekta

Sem korišćenjem petlji, prolazak kroz niz se može obaviti i korišćenjem specijalne ugrađene metode Array objekta - forEach**:**

```
arr.forEach(callback(value, index, array));
```

Metoda forEach() kao svoj parametar prihvata jednu funkciju. Reč je o funkciji koja će biti izvršena po jednom, za svaki od elemenata niza.

Funkcija koja se prosleđuje metodi `forEach()` prihvata sledeće parametre:

- `value` – vrednost trenutnog člana niza,
- `index` – indeks trenutnog člana niza,
- `array` – niz kroz koji se prolazi.

Bitno je reći da je, od tri prikazana parametra, samo prvi obavezan.

Preostala dva su opciona. Tako se prolazak kroz niz iz prethodnih primera korišćenjem metode `forEach()` može obaviti na sledeći način:

```
var cars = ["Honda", "Toyota", "Acura"];
cars.forEach(writeCar);

function writeCar(item) {
      console.log(item);
}
```

U prikazanom primeru kreirana je funkcija `writeCar()`, koja prihvata jedan parametar i obavlja ispis njegove vrednosti unutar konzole. Funkcija `writeCar()` se prosleđuje metodi `forEach()`, a sama metoda `forEach()` se poziva nad nizom `cars`. Sve ovo će za efekat imati pozivanje funkcije `writeCar()` tri puta, po jednom za svaki od elemenata niza `cars`.

Na kraju će unutar konzole biti ispisano:

```
Honda
Toyota
Acura
```

Nešto ranije je prikazano da funkcija koja se prosleđuje metodi `forEach()` može da prihvati i dva dodatna parametra, koji predstavljaju indeks i niz kroz koji se prolazi, respektivno:

```
var cars = ["Honda", "Toyota", "Acura"];

cars.forEach(writeCar);

function writeCar(item, index, array) {
      console.log(index + ': ' + item);
}
```

Sada je funkcija `writeCar()` proširena sa dva dodatna ulazna parametra. Drugi parametar `(index)` koristi se za ispis indeksa elemenata:

```
0: Honda
1: Toyota
2: Acura
```

Primer – Program za spajanje dva niza

U nastavku će biti prikazan primer JavaScript programa za spajanje dva niza. Program treba da omogući korisniku unos dva niza i da zatim takva dva niza spoji u jedan. Evo kako može izgledati kôd takvog programa:

```
let arrayA = prompt("Please, enter the elements of the first array:");
let arrayB = prompt("Please, enter the elements of the second array:");

arrayA = arrayA.split(' ');
arrayB = arrayB.split(' ');

let finalArray = [];

for (let i = 0; i < arrayA.length; i++) {
      finalArray[i] = arrayA[i];
}

for (let i = 0; i < arrayB.length; i++) {
      finalArray[arrayA.length +  i] = arrayB[i];
}
alert(finalArray.toString());
```

Na početku se, već dobro poznatom metodom `prompt()`, preuzimaju vrednosti nizova od korisnika. Zbog jednostavnosti, usvojili smo pravilo da se elementi jednog niza odvajaju praznim mestom (razmakom). Stoga da bi korisnik uneo jedan niz sa tri elementa dovoljno je da napiše: 1 2 3. Elementi se međusobno razdvajaju razmakom.

Nakon što korisnik unese nizove, oni se smeštaju unutar promenljivih `arrayA` i `arrayB`.

Ipak, bitno je razumeti da su takvi *nizovi* i dalje u `string` formatu. Stoga je na neki način `stringove` potrebno obraditi i od njih stvoriti nizove. To se postiže korišćenjem metode `split()`. Reč je o metodi `String` objekta, koja omogućava podelu `stringova` na osnovu definisanog karaktera ili grupe karaktera. Ovoj metodi se u primeru prosleđuje karakter razmak, pa će ona stringove da podeli na svim onim mestima na kojima postoji takav karakter. Metoda `split()` kao svoju povratnu vrednost emituje niz, pa je u primeru upotrebom ove metode korisnički unos pretvoren u nizove.

Preostalo je još samo da se dva dobijena niza spoje u jedan niz. To se obavlja korišćenjem dve `for` petlje. Unutar prve petlje elementi niza `arrayA` kopiraju se u niz `finalArray`.

Korišćenjem druge `for` petlje obavlja se kopiranje elemenata niza `arrayB` u niz `finalArray`.

Obratite pažnju na način na koji se dolazi do indeksa na koje je potrebno smestiti elemente `arrayB` niza (dužina `arrayA` niza se stalno uvećava za vrednost brojača).

Poslednjom naredbom obavlja se prikaz finalnog niza korisniku.

Rezime

- Nizovi omogućavaju smeštanje većeg broja vrednosti unutar jedne promenljive.
- U JavaScriptu nizovi su objekti tipa `Array`.
- JavaScript nizovi su dinamički: njihova dužina nije unapred utvrđena i mogu da prihvataju elemente različitih tipova.
- Nizovi se formiraju korišćenjem uglastih zagrada, unutar kojih se navode vrednosti, razdvojene zapetama.
- Niz je sastavljen iz elemenata, koji imaju svoje pozicije izražene kroz indekse; indeksi počinju od nule.
- Elementima niza pristupa se korišćenjem indeksa.
- Čitanje dužine niza postiže se korišćenjem svojstva `length`.
- Sortiranje niza može se obaviti korišćenjem metode `sort()`.
- Metoda `push()` dodaje element na kraj niza.
- Metoda `unshift()` dodaje element na početak niza.
- Za uklanjanje elementa sa kraja niza koristi se metoda `pop()`.
- Da bi se element uklonio sa početka niza, koristi se metoda `shift()`.
- Čitanje indeksa elementa, odnosno pretraga niza, može se obaviti korišćenjem metode `indexOf()`.
- Fraza *prolazak kroz niz* podrazumeva sukcesivno čitanje svakog elementa pojedinačno.
- Sem korišćenjem petlji, prolazak kroz niz se može obaviti i korišćenjem specijalne ugrađene metode `Array` objekta – `forEach()`.