**JavaScrip Full Stack Web Development**

*Mihajlo Milojević*

Sadržaj

[1. JavaScript (napredno) 6](#_Toc115707299)

[1.1 Konzola 6](#_Toc115707300)

[1.1.1 Console.log 7](#_Toc115707301)

[1.1.2 Console.log 7](#_Toc115707302)

[1.1.3 Console.warn 8](#_Toc115707303)

[1.1.4 Console.error 8](#_Toc115707304)

[1.1.5 Console.trace 9](#_Toc115707305)

[1.1.6 Console.dir 10](#_Toc115707306)

[1.2 Hoisting 12](#_Toc115707307)

[1.3 Let i const 13](#_Toc115707308)

[1.4 Arrow funkcije 16](#_Toc115707309)

[1.5 For in i for of 18](#_Toc115707310)

**Sadržaj:**

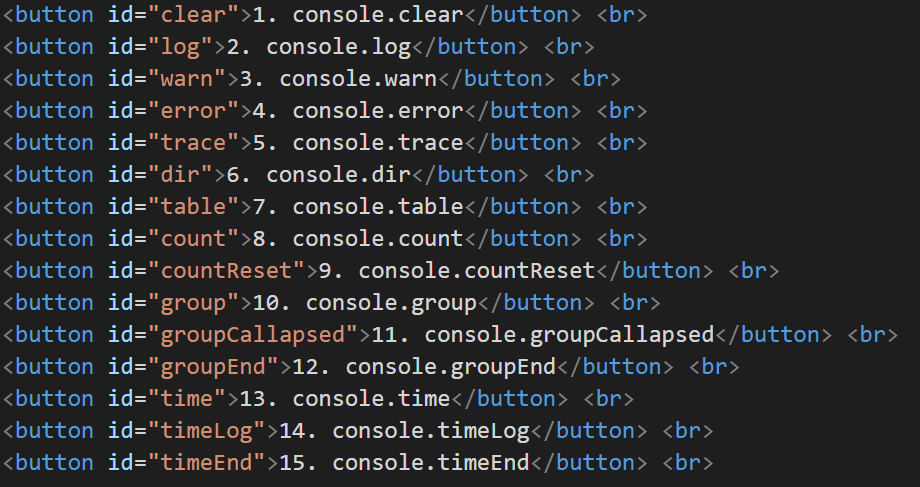
1. JavaScript (napredno)
   1. Konzola
   2. Hoisting
   3. Var, let i const
   4. Arrow funkcije
   5. *For in* i *for of*
   6. Metode nad stringovima
   7. Metode nad nizovima
   8. Metode nad objektima
   9. Desktruktuiranje objekata i nizova
   10. Klase
       1. Definicija
       2. Kreiranje
       3. Svojstva
       4. Metode
       5. Privatna svojstva
       6. Staticka svojstva i metode
       7. Nasledjivanje
   11. Asinhrono programiranje
       1. Funkcije povratnog poziva
       2. Obećanja (Promisi)
       3. Async/await
   12. Moduli
   13. JSON
   14. Fetch API
2. Nodejs
   1. Uvod
   2. Instalacija
   3. Globalne promenljive
   4. Moduli
   5. Ugrađeni moduli
      1. OS
      2. Path
      3. FS
      4. HTTP
   6. Veb serveri
   7. Fajl package.json
   8. Npm
   9. Folder node\_modules
   10. Instalacija paketa
   11. API
       1. Šta je api
       2. Zahtev i odgovor
       3. Metode
       4. Statusni kodovi
   12. Radni okvir Express
       1. Zašto?
       2. Početak
       3. Params i queries
       4. Middleware
          1. Šta?
          2. Kako?
          3. Cors
          4. Isporuka statičnih fajlva
       5. Rute
       6. Kontroleri
       7. Baza
       8. Autentifikacija
          1. JWT
          2. Kolačići
          3. Sesije
   13. Projekat
       1. Prikaz
       2. Projektovanje
       3. Baza
       4. Server
       5. API
       6. Autentifikacija i autorizacija
3. Git i GitHub
   1. Git
      1. Šta je git?
      2. Instalacija
      3. Konfiguracija
      4. Folder .git
      5. Repozitorijumi
      6. Izmene repozitorijuma
      7. Grane
   2. GitHub
      1. Šta je Github
      2. Početak
      3. Repozitorijumi
      4. Izmene repozitorijuma
      5. Preizimanje repozitorijuma
      6. Dodavanje saradnika
   3. Git u Visual Studio Code-u
   4. Veb hostovanje
   5. Projekat
      1. Prebacivanje koda na Githuba
      2. Hostovanje aplikacije
4. React
   1. Šta je react?
   2. Kreiranje i pokretanje projekta
   3. Struktura foldera i fajlova
   4. Renderovanje aplikacije na stranici
   5. JSX
   6. React komponente
      1. Klasne komponente
      2. Funkcionalne komponente
      3. Životni ciklus komponenata
   7. Props
   8. State
   9. Događaji
   10. Uslovno Renderovanje
   11. Liste
   12. Forme
   13. Rute
   14. Kuke
       1. useState
       2. useEffect
       3. useRef
       4. useContext
       5. Ostale
       6. Kreiranje sopstvenih kuka
   15. Projekat
5. Nextjs
   1. Uvod
   2. Kreiranje i pokretanje projekta
   3. Rute
   4. Dinamičke rute
   5. Kuka useRouter
   6. API rute
   7. SSR
   8. getStaticProps
   9. getStaticPaths
   10. getServerSideProps

# 1. JavaScript (napredno)

## 1.1 Konzola

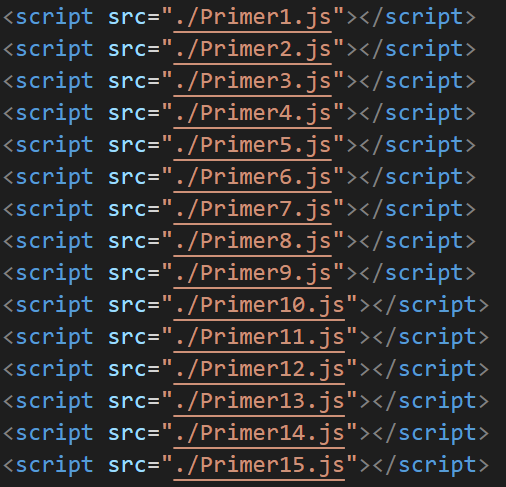
Objekat *console* je globalni objekatak koji sadrži mnoštvo metoda za debag-ovanje koga.

Da bi isprobali nekoliko najkorisnijih metoda napravićemo html stranicu sa sledećom strukturom:



*index.html*

Kod za svaku od ovih metoda smestićemo u posebne fajlove i povezati sa našim html-om pomoću sledećih scripti:

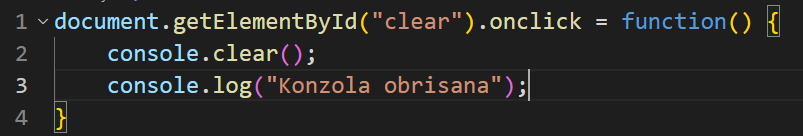


Sada možemo probati svaku od ovih metoda dodavanjem odgovarajućih *onClick* dogadjaja.

### 1.1.1 Console.log

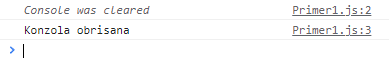
Metoda *clear* je jednostavna metoda koja briše sadržaj konzole.

Kod za prikaz ove metode je unutar fajla *Primer1.js*:



*Primer1.js*

Nakon klika na dugme „1. console.clear“ dobijamo sledeći izgled konzole:

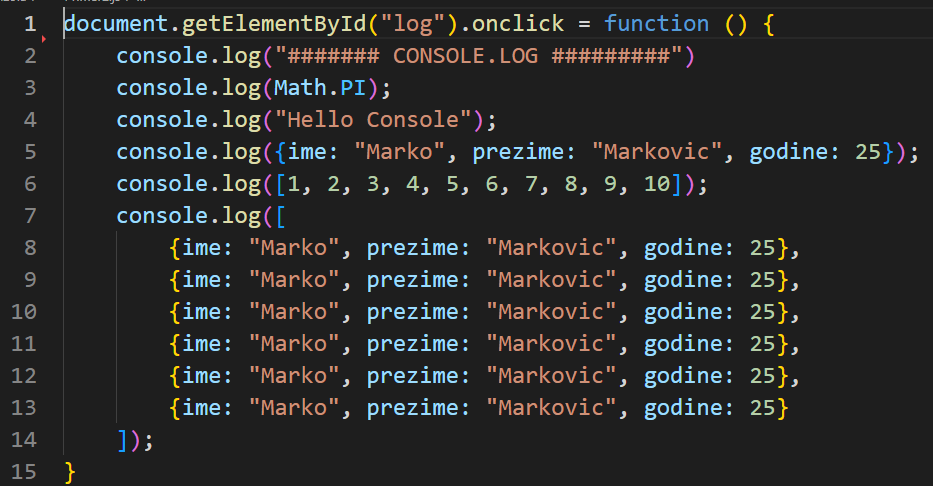


*Izgled konzole nakon pritiska na dugme „1. console.clear“*

### 1.1.2 Console.log

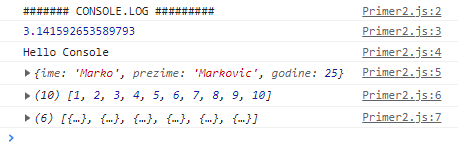
Metoda *log* je najkorišćenija i najjednostavnija. Ona prosto ispisuje ono što joj se prosledi kao parametar u konzolu.

Kod za prikaz ove metode je unutar fajla *Primer2.js*:



*Primer2.js*

A izgled konzole nakon klika na dugme „2. console.log“ je sledeći:

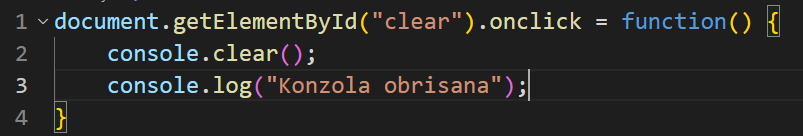


*Izgled konzole nakon pritiska na dugme „2. console.log“*

### 1.1.3 Console.warn

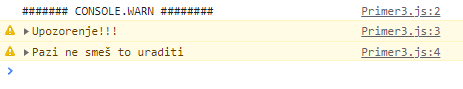
Metoda *warn* je veoma slična metodi *log*. Jedina razlika je što ona ispisuje upozorenje koje se drugačije prikazuje unutar konzole.

Kod za prikaz ove metode je unutar fajla *Primer3.js*:



*Primer3.js*

A izgled konzole nakon klika na dugme „3. console.warn“ je sledeći:

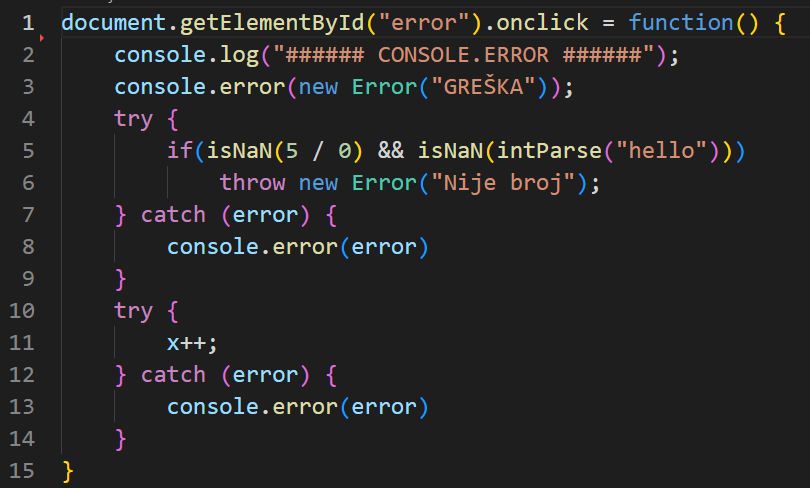


*Izgled konzole nakon pritiska na dugme „3. console.warn“*

### 1.1.4 Console.error

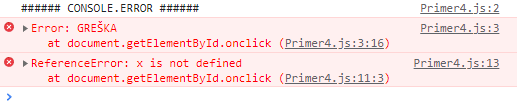
Metoda *error* je slična metodi *warn*, samo što umesto upozorenja ispisuje grešku:

Kod za prikaz ove metode je unutar fajla *Primer4.js*:



*Primer4.js*

A izgled konzole nakon klika na dugme „4. console.error“ je sledeći:

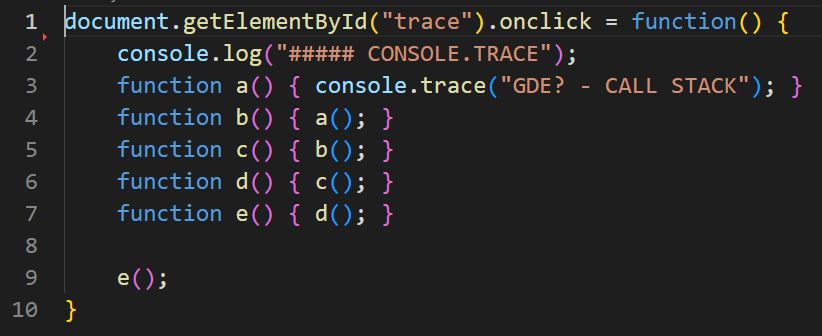


*Izgled konzole nakon pritiska na dugme „4. console.error“*

### 1.1.5 Console.trace

Metoda *trace* u konzolu ispusuje trenutni *call stack*. Ova metoda može biti veoma korisna za otkrivanje redosleda poziva ugnježdenih funkcija i otkrivana grešaka u istim.

Kod za prikaz ove metode je unutar fajla *Primer5.js*:



*Primer5.js*

A izgled konzole nakon klika na dugme „5. console.trace“ je sledeći:

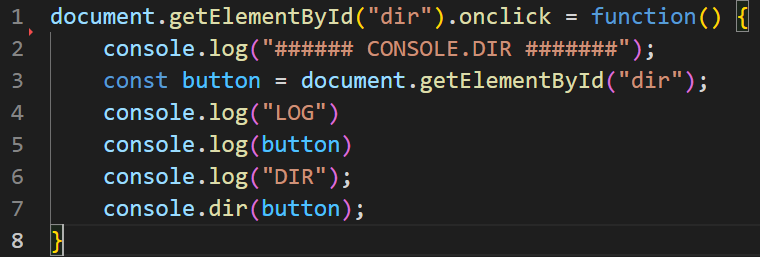


*Izgled konzole nakon pritiska na dugme „5. console.trace“*

### 1.1.6 Console.dir

Metoda *dir* u konzolu ispusuje svoje parametre kao objekte. Ovo je pogodno zato što nekad želimo da ispitamo svojstva html elemenata, a običnim logovanjem dobijamo samo html tog elementa.

Kod za prikaz ove metode je unutar fajla *Primer6.js*:



*Primer6.js*

A izgled konzole nakon klika na dugme „6. console.dir“ je sledeći:

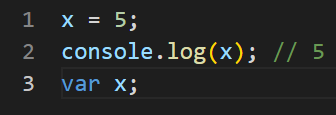


*Izgled konzole nakon pritiska na dugme „6. console.dir“*

## 1.2 Hoisting

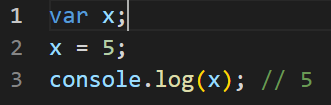
Hoisting u JavaScript-u je podizanje deklaracije na vrh.

Primer:



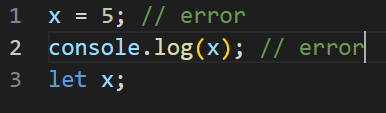
*Primer 1 - 1*

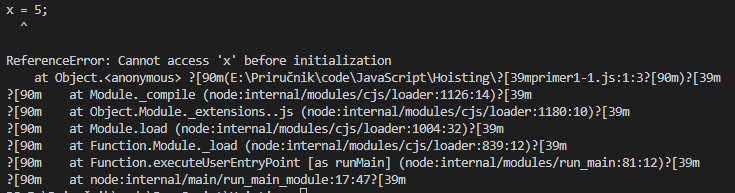
Ovde smo uspeli da dodelimo vrednost promenljive *x* i ispišemo je u konzolu pre deklarisanja. To je zato što je JavaScript podigao deklaraciju na vrh oblasti važenja promenljive *x*. Tako je da je prethodni kod identičan sledećem:



*Primer 1 - 2*

Ovo važi samo za promenljive deklarisane pomoću ključne reči *var*.

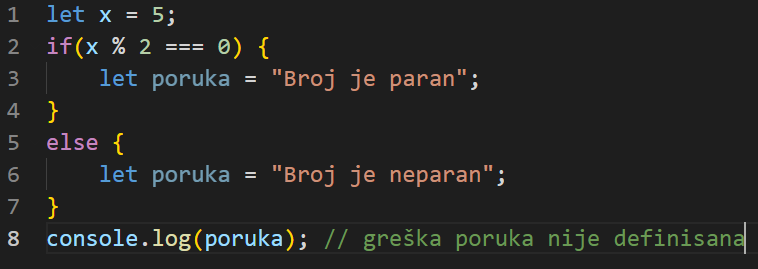


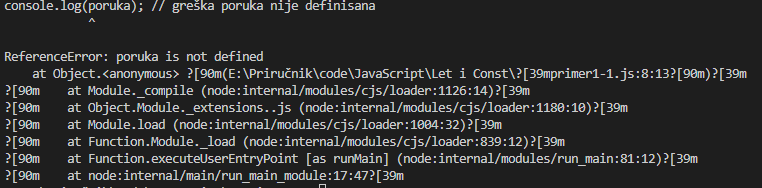


*Primer 1 – 3*

## 1.3 Let i const

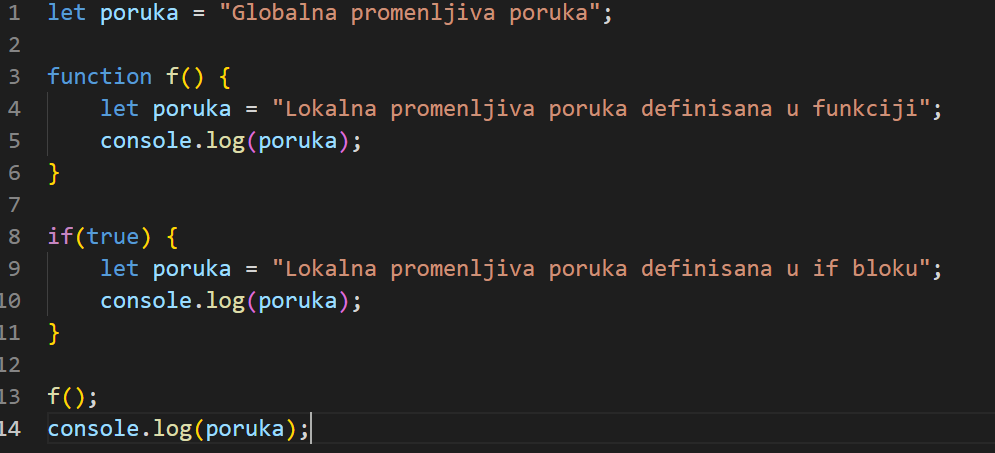
Ključna reč *let* nam omogućava da definišemo promenljive sa blokovskom oblašću važenja (za razliku od ključne reči *var* koja ima ili globalnu ili funkcijsku oblast važenja).





*Primer 1 - 1*

Ovde je promenljiva poruka definisana samo izmedju „{„ i „}“ u *if* (*else*) bloku tako da bilo koja upotreba van tog bloga nije važeća.



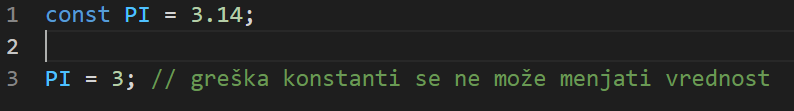


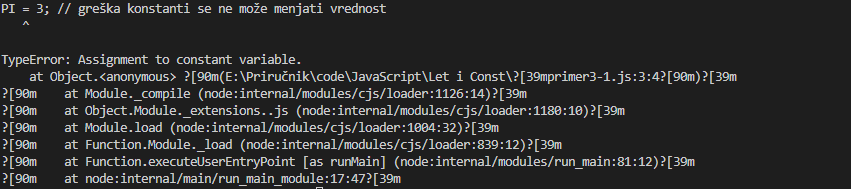
*Primer 2 - 1*

Promenljive definisane pomoću ključne reči *let*:

* ne mogu biti redefinisane unutar istog bloka važenja
* moraju biti definisane pre upotrebe
* imaju blokovsku oblast važenja

Ključna reč *const* nam omogućava da definišem promenljive istih osobina kao i one definisane pomoću *let* samo što njihova vrednost ne može biti promenjena nakon inicijalizacije.

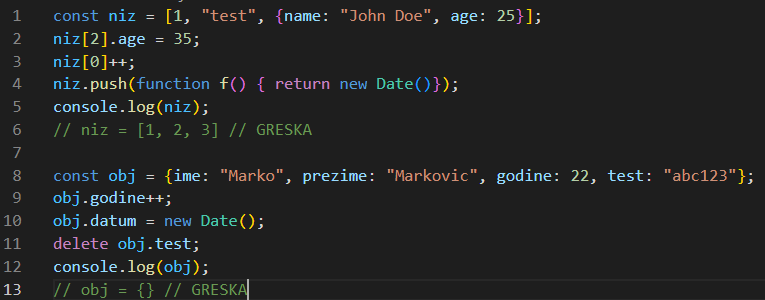


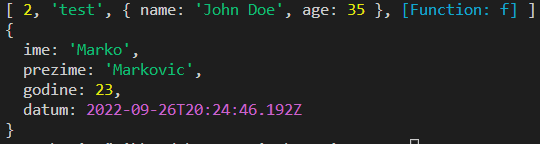


*Primer 3 – 1*

Nažalost const ne može da definiše prave konstante.

Objekti i nizovi su u JavaScriptu predstavljeni kao reference, tako da kada definisemo niz ili objekat pomoću ključne reči *const* mi kreiramo konstantu referencu, dok se u niz i dalje mogu dodavati, brisati i menjati elementi, a objektima se mogu dodavati, brisati i menjati polja.

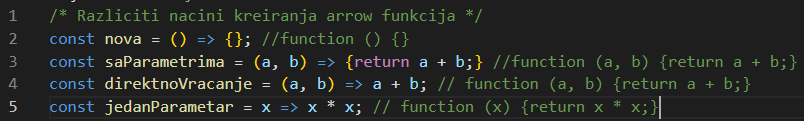




*Primer 4 - 1*

## 1.4 Arrow funkcije

Arrow funkcije su skraćeni oblik definisanja funkcija. Postoji više načina za njihovo definisanje.

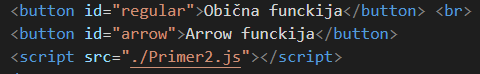


*Primer 1*

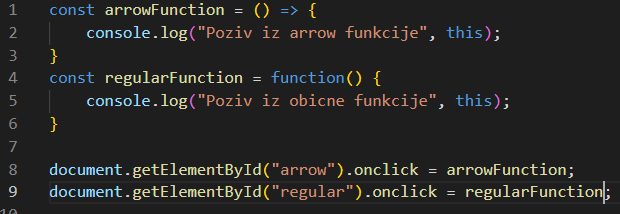
Arrow funkcije se definisu sintaksom: *(parametri) => {/\*telo\*/}*, ali postoje i skraćene varijante.

* Ako imamo samo jedan parametar možemo izostaviti zagrade: *event => {console.log(event);}*.
* Ako funkcija odmah vraća vrednost možemo izostaviti vitičaste zagrade i naredbu return: *(value) => parseInt(value)*

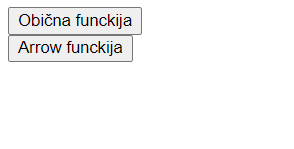
Kod arrow funkcija je veoma bitno da se kljucna reč this ponaša drugačije u odnosu na običnu funkciju:



*Primer 2 - HTML*



*Primer 2 - JavaScript*



*Primer 2 - Izgled u pregledaču*



*Primer 2 - Ispis u konzoli nakon klika na oba dugmeta*

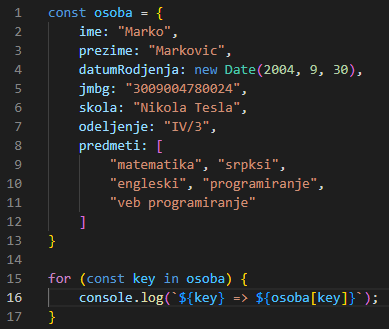
Kao što možete da primetite *this* u arrow funkciji i običnoj funkciji nije isto. U običnoj funkciji *this* se odnosi na objekat koji je pozvao funkciju (u ovom slučaju button), dok se u arrow funkcijama *this* odnosi na objekat u kom je funkcija definisana (u ovom slučaju *window* jer je *arrowFunction* globalna promenljiva).

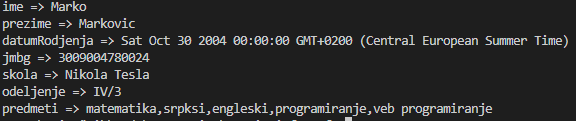
Ovo ponašanje može biti problematično ako se radi metodama objekata. Kada pozovemo metodu nad objektom očekujemo da ona sadrži referencu na sam objekat, a samim tim i na sva njegova svojstva i metode.

## 1.5 For in i for of

*For in* i *for of* se koriste za lakse prolaženje kroz objekte i nizove.

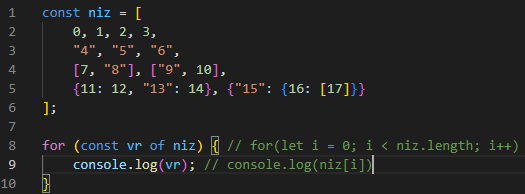
For in se koristi za prolazak kroz sve ključeve (svojstva obejka).

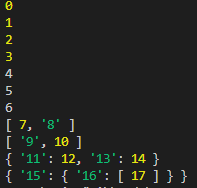




*Primer 1*

For of se koristi za prolazak kroz sve vrednosti niza:





*Primer 2*