

Vježba2 (closure-iterators-generators-promises)

1. Napisati funkciju koja vraća funkciju after i prima dva argumenta. Prvi argument je broj koji predstavlja koliko puta funkcija after treba biti pozvana prije nego se izvrši callback funkcija, a drugi argument je callback funkcija koja će se izvršiti. (hint closure)
2. Napisati funkciju movieSelector koja prima niz objekata koji sadrže informacije o filmovima (id, title i score). Iskoristiti funkcije filter i map kako bi povrat funkcije movieSelector bio niz koji sadrži samo imena filmova sa score-om većim od pet i to uppercase-ane. Primjer za objekt

```
movies = [ { id: 1, title: "Pan's Labyrinth", score: 9 },  
  { id: 37, title: "Gentelman", score: 6 },  
  { id: 11, title: "Batman", score: 5 },  
  { id: 44, title: "Birds of Prey", score: 1 },,];
```


očekivani rezultat je: ["PAN'S LABYRINTH", "GENTELMAN"]
3. Napisati funkciju numSelectString koja prima niz brojeva i vraća string. U ovoj funkciji treba iskoristiti funkcije filter, sort i reduce kako bi se kao povrat dobio string koji sadrži samo neparne brojeve iz niza, odvojene zarezom koji su poredani uzlazno. Npr za niz [17, 34, 3, 12, 23] povrat je string „3, 17, 23”.
4. Napisati funkciju koja prima niz i u njoj kreirati for petlju koja iterira kroz elemente niza i vraća sumu elemenata.
5. Napisati „iterator funkciju“ koja kao argument prima niz elemenata, te vraća funkciju koja pri svakom pozivu vraća sljedeći član niza sa konkateniranim stringom „je n-ti element niza”. (hint. closure).
6. Kreirati funkciju russianRoulette koja prima jedan argument (broj, nazovimo ga n), i vraća funkciju. Vraćena funkcija ne prima argumente, a vraća string „click!” prvih n – 1 poziva. Prilikom n-tog poziva, funkcija će vratiti string „bang”. Svakim sljedećim pozivom funkcije, povratna vrijednost će biti string „reload to play again”. (hint „closure”)
7. Kreirati funkciju **blackJack** koja prima niz (brojevi od 1 do 11) i vraća funkciju (nazovimo je **dealer**). Funkcija dealer prima dva argumenta (dva broja) i vraća funkciju (nazovimo je **player**). Ako je funkcija **player** pozvana PRVI PUT, vraća sumu brojeva koji su argumenti funkcije **dealer**. Ako je funkcija **player** pozvana DRUGI PUT, vratiti će sumu ranija dva argumenta i prvog člana niza, ukoliko je suma manja ili jednaka 21. Ako je veća od 21, funkcija **player** vraća string „bust!”. Ukoliko prethodna suma nije veća od 21, tada će se svakim sljedećim pozivom funkcije **player** vratiti nova suma - zbroj posljednje sume i sljedećeg broja u nizu koji je proslijeđen funkciji **blackJack**. Ukoliko je nova suma veća od 21, povrat funkcije player je „bust!”. Ako je funkcija **player** vratila string „bust!”, tada će svaki sljedeći poziv funkcije **player** vratiti string „You are done!”. Ideja ovog zadatka je korištenje i shvaćanje koncepta closure, te ga je obavezno koristiti.
8. Kreirati „iterator funkciju“ koja prima niz i vraća objekt koji sadrži next metodu. Svaki poziv next metode vraća sljedeći element niza i njegov index u nizu (metoda next vraća **niz** koji se sastoji od dva člana).
9. Nadopuniti zadatak 3 na način da next() funkcija vraća **objekt** sa vrijednostima value i done {value: vrijednost, done: false}. Property value sadrži vrijednost trenutnog elementa niza, a property done ima vrijednost false sve dok se ne dodje do posljednjeg člana niza (tada mu se vrijednost mijenja u true). Ukoliko se next() funkcija poziva dalje, treba vratiti objekt koji sadrži samo property done postavljen na true {done: true}.

10. Napisati „iterator funkciju“ koja prima string riječi odvojenih razmakom, te vraća objekt koji na sebi ima metodu next. Svakim pozivom metode next, vraća se jedna po jedna riječ.
11. Riješiti prethodni zadatak koristeći generator funkciju.
12. Napisati funkciju simulacijaRazgovora koja će vratiti „hej“ ili „besmislica“ svako tri sekunde, u ovisnosti o tome je li riječ koja je prosljeđena u funkciju glasi „generator“. Nije dozvoljeno koristiti petlje, a funkcija simulacijaRazgovora će se pozvati samo jednom. (Pojašnjenje. koristiti generator funkciju. U funkciju simulacijaRazgovora se šalje niz stringova. Funkcija simulacijaRazgovora završava kada se za svaku riječ iz niza ispiše „besmislica“ ili „hej“. Hint- clearInterval() se poziva kada je povrat iz next-a {value:undefined, done:true}).
13. Koristeći async await pattern, ispisati rečenicu koja se sastoji od imenice i glagola. Pozvati async funkciju i prosljediti joj imenicu. Unutar async funkcije koristiti ključnu riječ await kako bi se dobio povrat iz ne-async-funkcije koja prima imenicu i konkatenira hardkodirani glagol i to vraća. Povrat ispisati nakon tri sekunde.
14. Kreirati promise koji će se nakon 1000 ms resolve-ati u string „Resolved“, koristeći metodu setTimeout(). Funkcija koja je zadužena za printanje resolved promisa, se prosljeđuje u promise preko then() metode. (https://developer.mozilla.org/enUS/docs/Web/JavaScript/Reference/Global_Objects/Promise)
15. Kreirati promise koji će se odmah reject-ati. Iz funkcije koja se pozvala kada je promise reject-an printati „Error“(hint. nad promise objektom pozvati metodu catch).
16. Promisi su asinhroni i sada ćemo to i dokazati. Kreirati promise koji će se resolve-ati sa vrijednošću „Promise has been resolved!“. Nakon toga ubaciti sljedeći kod:

```
promise.then(() => console.log('Promise has been resolved!'));  
  
console.log("I'm not the promise!");
```

Koji redosljed ispisa očekujete? Zašto?
17. Niže je naveden objekt fakePeople koji imitira podatke izvučene iz baze podataka. Napisati funkciju fakeAPICall(i) koja vraća promise koji će se resolve-ati u podatke osobe na i-tom mjestu. Promise resolve-ati nakon random broja milisekundi između 1000 i 3000. Napisati funkciju getAllData, koja koristi Promise.all kako bi se 3 API poziva fakeAPICall(i) izvršila istovremeno. Argument i odabrati proizvoljno. U slučaju da je i veći od 3, reject-ati promise.

Promise.all

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

```
const fakePeople = [  
  
  { name: 'Ivo', hasPets: false},  
  
  { name: 'Eva', hasPets: true},  
  
  { name: 'Marko', hasPets: true}  
]
```