Napomene:

* Za svaku vezbu je potrebno napisati po jednu funkciju koja izvrsava to sto se trazi u vezbi. Parametre te funkcije potrebno je zakljuciti iz toga sta je funkciji potrebno, npr. u prve dve vezbe je potreban samo niz, u trecoj niz i broj N.
* “Proizvoljna” vrednost znaci da funkcija treba da radi pravilno sa bilo kojom vrednoscu. Npr. ako se pominje proizvoljan broj, funkcija treba da radi bez obzira da li je broj pozitivan, negativan ili nula. Ako se pominje proizvoljni niz, funkcija treba da radi i ako je niz prazan, koje god podatke da ima. Sa druge strane, “proizvoljni niz brojeva” moze da ima bilo koju duzinu, ali da sadrzi samo brojeve. Na isto se misli kada se pominje “prosledjena” vrednost, vrednost koju prima funkcija i sl.

1. Pomocu **forEach** funkcije proci kroz proizvoljni niz i ispisati u konzoli sve njegove elemente, jedan po jedan
2. Proci kroz proizvoljni niz brojeva i ispisati u konzoli sve njegove elemente, uvecane za 2
3. Proci kroz proizvoljni niz brojeva i ispisati u konzoli sve njegove elemente, uvecane za proizvoljan broj N
4. Proci kroz proizvoljni niz i na svakom elementu pozvati prosledjenu funkciju (callback)
5. Ispisati u konzoli proizvoljan string 2 sekunde nakon pozivanja funkcije. Koristiti **setTimeout** (<https://developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/setTimeout>)
6. Ispisati u konzoli jedno po jedno slovo proizvoljnog stringa 2 sekunde nakon pozivanja funkcije (svako slovo treba pojedinacno da bude ispisano sa console.log, ali mogu sva slova biti ispisana u isto vreme). Koristiti **split** metod za konvertovanje stringa u niz slova (<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/split>). String se moze podeliti na slova tako sto na njemu pozove string.split('') sa praznim stringom.
7. Napisati funkciju koja prima drugu funkciju (callback) i poziva je nakon 2 sekunde.
8. Napisati funkciju koja prima callback i poziva je nakon N sekundi.
9. Pomocu **sort** metode na nizovima, poredjati proizvoljni niz brojeva u opadajucem redosledu, od najveceg ka najmanjem (<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort>). **sort** prima callback, i rasporedjuje elemente u zavisnosti od toga da li callback vraca pozitivan ili negativan broj. Sam callback prima 2 elementa, koje poredi.
10. Pomocu **sort** metode poredjati proizvoljni niz objekata u rastucem redosledu, prema vrednosti atributa (property-a) **value**.Dakle, ako imamo niz objekata:

var array = [

{ value: 5 },

{ value: 2 },

{ value: 1 },

{ value: 8 },

]

nakon poziva sortByValue(array) treba da se dobije

[

{ value: 1 },

{ value: 2 },

{ value: 5 },

{ value: 8 },

]

1. Napisati funkciju **callFnNTimes** koja prima callback i broj N, i poziva callback N puta.
2. Napisati funkciju **any**, koja prima niz i callback, i vraca **true** ako callback vrati **true** (ili “truthy” vrednost) za bilo koji element niza, i **false** u suprotnom. Ova funkcija treba da radi slicno kao **find** metod na nizovima (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Array/find), s tim sto **find** vraca prvi element niza za koji callback vrati truthy vrednost, ili undefined u suprotnom, a **any** treba da vraca boolean.
3. Napisati funkciju **every**, koja prima niz i callback, i vraca **true** ako callback vrati **true** (ili “truthy” vrednost) za sve elemente niza, i **false** u suprotnom.
4. Napisati funkciju **attachMethod** koja prima objekat i funkciju, i dodaje tu funkciju na objekat kao metod (moze da se zove bilo kako, npr. **temporaryMethod**).

Primer koriscenja:

var object = { a: 1 };

function logA() {

console.log(this.a);

}

attachMethod(object, logA);

object.temporaryMethod(); // loguje 1

1. Napisati funkciju **callOn**, koja isto kao **attachMethod** prima objekat i funkciju i dodaje metod, ali odmah poziva taj metod (bez argumenata) i zatim ga uklanja sa objekta pomocu **delete** operatora.

Primer koriscenja:

var object = { a: 1 };

function logA() {

console.log(this.a);

}

callOn(object, logA); // loguje 1

object.temporaryMethod(); // ERROR! temporaryMethod vise ne postoji na tom objektu

1. Napisati funkciju **multiplyBy5** koja uvecava broj 5 puta, odnosno prima broj i vraca 5 puta veci broj
2. Napisati funkciju **getMultiplyBy5** koja vraca funkciju istu kao **multiplyBy5** iz prethodne vezbe. Moguce je ovo izvesti tako sto funkcija jednostavno ima

return multiplyBy5;

Medjutim, preporucljivo je da vrati novu funkciju, posto se sledeca vezba dosta lakse resava na taj nacin.

1. Napisati funkciju **getMultiplyByN** koja prima broj N i vraca funkciju koja uvecava prosledjeni broj N puta