Javascript osnove III

**Funkcije nad nizovima II**

* sort (function) - se koristi za sortiranje nizova. Npr:

const fruits = ["Banana", "Kiwi", "Apple", "Strawberry"];

fruits.sort(); // ["Apple", "Banana", "Kiwi", "Strawberry"];

Napomena! Kada se sortiraju brojevi, sortiranje se vrsi po njihovim ASCII vrijednostima (tj. sortirace se kao da su stringovi. U ovom slucaju, funksiji sort mozemo proslediti “nasu logiku”, koja ce definisati kako sortiranje da se odradi. Na primjer:  
  
const arr = [50, 100, 1, 2, 35, 10];

console.log(arr.sort());

arr.sort(function(a, b) {

return a - b;

});

Na ovaj nacin se koristi razlika (broj) za komparaciju i ako je ona veca od 0, onda se radi zamjena, a ako je manja, onda ne (broj s lijeve strane je manji).  
  
- reverse (function) - takodje se koristi za sortiranje, ali od vece ka manjoj vrijednosti.

Vaze ista pravila kao i za sort.

* splice (function) - “utiskuje” vrijednosti u niz. Na primjer:

const arr = [1, 2, 3, 6, 7];

arr.splice(3, 0, 4, 5); // arr ce nakon ovoga biti [1, 2, 3, 4, 5, 6, 7];

splice dakle ima nekoliko argumenata:

* prvi je pozicija od koje se pocinje (index),
* drugi koliko elemenata se brise,
* ostalo su elemeni koji se “utiskuju”).

splice takodje mozemo koristiti za brisanje elemenata iz niza bez undefined “rupa”. Na primjer

arr.splice(1,1); // arr je nakon ovoga [1, 3, 4, 5, 6, 7];

na ovaj nacin smo rekli:

* pocni od prvo elementa
* izbrisi jedan
* ne dodaj nista novo,
* slice (function) - “isjeca” odredjeni dio niza i vraca ga kao novi niz (ne mutria originalni za razliku od splice)

Na primjer:

const arr = [1, 2, 3];

const newArr = arr.slice(1, 2); // newArr = [2];

slice moze imati par argumenata:

* prvi je index od kojeg se pocinje (ako nije prosledjen, onda je 0-ti index),
* drugi index na kojem se zavrsava isjecanje (ukoliko nije prosledjen, ide se do kraja niza);

**Funkcije “viseg reda” nad nizovima**

Funkcije vise rega (higher order functions) su jednostavno funkcije koje primaju drugu funkciju kao argument. Vec smo vidjeli primjer jedne - forEach.

const arr = [3, 6, 1];

arr.forEach((element, index) => {

console.log(`On position ${index} is value: ${element}`);

});

Druge bitne funkcije viseg reda koje ce se cesto koristiti nad nizovima:

* map - prolazi kroz sve elemente niza i eventualno radi nesto sa svakim pojedinacnim elementom, te azurira origiralan element u nizu. Map vraca novi niz sa tim azuriranim vrijednostima. Na primjer:

const arr = [1, 2, 3];

const newArr = arr.map(function (element) {

return element \* 2;

});

console.log(newArr);

newArr ce nakon ovoga biti [2, 4, 6];

* filter - isto kao map, prolazi kroz sve elemente niza i kreira novi niz na osnovu ispunjenosti uslova. Npr.

const temperatures = [17, 25, 33, 19, 21, 30, 29];

const newArr = temperatures.filter(function (element, index) {

return element >= 20;

});

console.log(newArr);

newArr ce sadrzati sve vrijednosti vece od 20: [25, 33, 21, 30, 29].

* reduce - funkcija koja se koristi za “svodjenje na jednu vrijednost”. Na primjer sabiranje elemenata niza:

const arr = [1, 2, 3];

const total = arr.reduce(function (acc, element) {

return acc += element;

}, 10);

total ce nakon ovoga biti 6.

Reduce moze vrsiti i drudge operacije (sto god da se odredi u funkciji) i takodje moze imati startnu vrijednost kao drugi parametar:  
  
const arr = [1, 2, 3];

const total = arr.reduce(function (acc, element) {

return acc += element;

}, 100);

total je nakon ovoga 106.

Reduce krece od nultog lijevog (prvog) elementa. Ako zelimo poceti od kraja niza, onda koristimo reduceRight.

* every provjerava da li je za svaki element niza uslov zadovoljen.

const items = [10, 11, 16, 12, 8];

const allAbove5 = items.every(function (element) {

return element > 5;

});

allAbove5 ce biti true, posto su svi elementi veci od 5.

* some provjerava da li neki od elemenata zadovoljavaju uslov.

const items = [10, 11, 16, 12, 8];

const someAbove10 = items.some(function (element) {

return element > 10;

});

someAbove10 ce biti true, posto imamo element vece od 10.

* find vraca element iz niza ako je uslov zadovoljen (ili undefined)

const items = [10, 11, 16, 12, 8];

const find10 = items.find(function (element) {

return element === 10;

});

find10 ce imati vrijednost 10.

* indexOf vraca index (ili -1 ako ne postoji) od nekog elementa

const items = [10, 11, 16, 12, 8];

const index16 = items.indexOf(16); // 2

* lastIndexOf vraca index poslednjeg pojavljivanja elementa u nizu. Npr:

const items = [10, 11, 16, 12, 8, 10];

const lastIndex10 = items.lastIndexOf(10); // 5

Kompletnu listu svih metoda nad nizovima mozete naci na linku:   
  
<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array>

Zadatak 1:   
  
Isfiltrirati sledeci niz:

const persons = [

{

firstName: “Homer”,

lastName: “Simpson”,

gender: “male”,

role: “parent”,

age: 43,

hobbies: [“sleeping”, “eating”, “watching tv”]

}, {

firstName: “Bart”,

lastName: “Simpson”,

gender: “male”,

role: “child”,

age: 12,

hobbies: [“skating”, “football”, “watching tv”]

}, {

firstName: “Lisa”,

lastName: “Simpson”,

gender: “female”,

role: “child”,

age: 9,

hobbies: [“reading”, “playing music”, “studying”, “swimming”]

}, {

firstName: “Marge”,

lastName: “Simpson”,

gender: “female”,

role: “parent”,

age: 37,

hobbies: [“cooking”, “watching tv”]

},

];

tako da dobijemo nove nizove samo roditelja, odnosno samo djece. Uraditi primjer koristeci filter funkciju + rucnu implementaciju upotrebom for petlje.

Zadatak 2:

Izracunati koliko ukupno imaju godina svi clanovi prethodnog niza zajedno.

Zadatak 3:

Napraviti novi niz koji ce sadrzati sve (unikatne) hobije koje upraznjava porodica Simpson.

Zadatak 4:

U svaki objekat dodati countHobbies polje koje se sadrzati broj hobija za ordedjenog clana.

Zadatak 5:

Napisati funkciju compareCurlyBraces(input) koja broji ponavljanje “{“ i “}” u nekom stringu. Ukoliko je broj isti, funkcija treba da vrati true, u suprotnom false.

Primjer stringa: “My name is {Branimir}”

Zadatak 6:

Radimo na malom konzolnom digitronu. Digitron moze da radi samo 4 operacije: +, -, \*, /. Jezgro naseg digitrona je funkcija koja prima niz brojeva kao prvi parametar i operand kao drugi. Funkcija treba da vrati rezultat nakon izvrsavanja trazene operacije nad svim elementima