

**SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN**

Sara Sušac

**VLASTITA IMPLEMENTACIJA MAP,
REDUCE I PIPE FUNKCIJA U
JAVASCRIPT-U**

PROJEKT

DEKLARATIVNO PROGRAMIRANJE

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Sara Sušac

Matični broj: 0016143183

Studij: Baze podataka i baze znanja

**VLASTITA IMPLEMENTACIJA MAP, REDUCE I PIPE FUNKCIJA U
JAVASCRIPT-U**

PROJEKT

Mentor:

dr. sc. Bogdan Okresa Đurić

Varaždin, lipanj 2023.

Sara Sušac

Izjava o izvornosti

Izjavljujem da je ovaj projekt izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autorica potvrdila prihvaćanjem odredbi u sustavu FOI Radovi

Sažetak

U ovome projektu koristit ćemo imperativni programski jezik Javascript kako bi prikazali implementaciju map, reduce i pipe funkcija. Najprije ćemo objasniti što je poanta korištenja tih funkcija, a zatim ćemo prikazati i njihove različite primjene.

Ključne riječi: Javascript; map; reduce; pipe

Sadržaj

1. Uvod	1
2. Opis funkcija map, reduce i pipe	2
2.1. Map funkcija	2
2.2. Reduce funkcija	2
2.3. Pipe funkcija	2
3. Implementacije funkcija	3
3.1. mapFunkcija	3
3.2. reduceFunkcija	3
3.3. pipeFunkcija	4
4. Funkcionalnosti metoda	5
4.1. Primjeri korištenja mapFunkcija	5
4.2. Primjer korištenja reduceFunkcija	5
4.3. Primjer korištenja pipeFunkcija	7
4.4. Outputi funkcija	9
5. Zaključak	10
Popis literature	11
Popis slika	12
Popis isječaka koda	13

1. Uvod

Imperativno programiranje je paradigma razvoja softvera u kojoj su funkcije implicitno programirane za svaki korak potreban za rješavanje problema. U imperativnom programiranju, svaka operacija je programirana i sam kod specificira kako se problem treba riješiti, što znači da se unaprijed napisani modeli ne koriste.

Imperativno programiranje zahtijeva razumijevanje funkcija potrebnih za rješavanje problema, umjesto oslanjanja na modele koji ga mogu riješiti. Fokus imperativnog programiranja je kako problem treba biti riješen, što zahtijeva detaljan vodič korak po korak. Budući da napisani kod izvodi funkcije umjesto modela, programer mora isprogramirati svaki korak. [1]

2. Opis funkcija map, reduce i pipe

U ovome poglavlju proći ćemo ukratko kroz definicije funkcija map, reduce i pipe. Objasniti ćemo kako, kada i za što se točno one koriste.

2.1. Map funkcija

Map funkcija u programiranju koristi se za iteraciju kroz elemente liste ili kolekcije i primjenu određene funkcije na svaki element, rezultirajući novom listom ili kolekcijom s transformiranim vrijednostima.

Map funkcija prima ulaznu listu ili kolekciju i funkciju povratnog poziva. Iterira kroz svaki element ulazne liste i primjenjuje funkciju povratnog poziva. Rezultat primjene funkcije povratnog poziva na svaki element se pohranjuje u rezultirajuću listu ili kolekciju. Map funkcija vraća novu listu ili kolekciju s transformiranim vrijednostima, zadržavajući redoslijed elemenata izvorne liste. Ova funkcionalnost omogućava jednostavno transformiranje elemenata liste ili kolekcije prema željenim pravilima, bez mijenjanja izvorne strukture.[2]

2.2. Reduce funkcija

Reduce funkcija u programiranju koristi se za redukciju ili agregaciju elemenata liste ili kolekcije na jednu vrijednost. Ona iterira kroz elemente i akumulira rezultat primjenom funkcije povratnog poziva na svaki element i trenutno stanje akumulatora.

Reduce funkcija prima ulaznu listu ili kolekciju, funkciju povratnog poziva i početnu vrijednost akumulatora. Iterira kroz svaki element ulazne liste i kombinira ga s trenutnim stanjem akumulatora koristeći funkciju povratnog poziva. Rezultat kombinacije se ažurira kao novi vrijednost akumulatora. Nakon što se svi elementi obrađeni, reduce funkcija vraća konačni rezultat iz akumulatora. Ova funkcionalnost omogućava redukciju kompleksnih struktura podataka u pojednostavljene ili agregirane rezultate, kao što su suma, prosjek, maksimum, minimum itd.[3]

2.3. Pipe funkcija

Pipe funkcija u programiranju koristi se za slaganje i izvršavanje više funkcija jedna za drugom, pri čemu izlaz jedne funkcije postaje ulaz sljedeće funkcije. To omogućava jednostavno povezivanje i kompoziciju funkcija kako bi se izvela složenija operacija ili transformacija.

Pipe funkcija prima niz funkcija kao argumente i vraća novu funkciju koja izvršava te funkcije jednu za drugom. Izlaz prve funkcije postaje ulaz sljedeće funkcije u nizu. Svaka funkcija u nizu prima samo jedan argument i vraća rezultat koji će biti ulaz za sljedeću funkciju. Pipe funkcija omogućava kompoziciju funkcija bez potrebe za uporabom međurezultata ili privremenih varijabli. Ova funkcionalnost olakšava čitanje, održavanje i ponovnu upotrebu koda jer se funkcionalnosti mogu lako kombinirati u fleksibilne i modularne cjeline.[4]

3. Implementacije funkcija

U ovome poglavlju opisati ćemo kako su implementirane funkcije map, reduce i pipe.

3.1. mapFunkcija

Isječak kôda 1: Funkcija mapFunkcija

```
1 function mapFunkcija(listaVrijednosti, callback) {
2   const rezultat = [];
3   for (const vrijednost of listaVrijednosti) {
4     const vrijednostRezultat = callback(vrijednost);
5     rezultat.push(vrijednostRezultat);
6   }
7   return rezultat;
8 }
```

U funkciji mapFunkcija implementirali smo osnovnu logiku funkcije map koja se nalazi u jeziku Javascript.

(listaVrijednosti) je ulazni parametar koji predstavlja listu nad kojom želimo izvršiti transformaciju. (callback) je funkcija koja se primjenjuje na svaki element liste. Ova funkcija definira transformaciju koju želimo primijeniti. Unutar funkcije inicijaliziramo praznu listu (rezultat) koja će sadržavati rezultate transformacije. Zatim prolazimo kroz svaki element (vrijednost) u (listaVrijednosti). Za svaki element pozivamo funkciju (callback) s trenutnom vrijednosti i (rezultat) dodajemo u listu rezultat. Nakon što smo prošli kroz sve elemente liste, vraćamo rezultirajuću listu.

Ova funkcija omogućuje nam da primijenimo bilo koju transformaciju na svaki element liste, čime olakšava rad s podacima u Javascriptu.

3.2. reduceFunkcija

Isječak kôda 2: Funkcija reduceFunkcija

```
1 function reduceFunkcija (incijalnaListaVrijednosti, callback, početnaVrijendost){
2   let acc = početnaVrijendost
3   for (const vrijednost of incijalnaListaVrijednosti){
4     acc =callback(acc,vrijednost)
5   }
6   return acc
7 }
```

Funkcija reduce se koristi za reduciranje liste vrijednosti na jednu vrijednost primjenom određene akumulativne operacije.

(incijalnaListaVrijednosti) je ulazni parametar koji predstavlja listu vrijednosti nad kojom želimo primijeniti redukciju. (callback) je funkcija koja se koristi za obavljanje akumulativne

operacije nad elementima liste. (početnaVrijednost) je početna vrijednost akumulatora. Inicijaliziramo akumulator (acc) s početnom vrijednosti. Prolazimo kroz svaki element (vrijednost) u (incijalnaListaVrijednosti). Za svaki element pozivamo funkciju (callback) s trenutnim akumulatorom i trenutnom vrijednošću. Rezultat te operacije dodjeljujemo akumulatoru. Nakon što smo prošli kroz sve elemente liste, vraćamo konačnu vrijednost akumulatora.

Ova funkcija omogućuje nam da izvršimo različite akumulativne operacije, poput zbrajanja, množenja, filtriranja ili bilo koje druge operacije koju definiramo putem funkcije callback.

3.3. pipeFunkcija

Isječak kôda 3: Funkcija pipeFunkcija

```
1 function pipeFunkcija(...callbacks) {  
2   return function (incijalnaListaVrijednosti) {  
3     let vrijednost = incijalnaListaVrijednosti;  
4     for (const callback of callbacks) {  
5       vrijednost = callback(vrijednost);  
6     }  
7     return vrijednost;  
8   };  
9 }
```

Funkciju pipeFunkcija smo stvorili kako bi omogućili kompoziciju više funkcija u jednu.

Definirali smo funkciju pipeFunkcija s parametrom (...callbacks). Ovaj parametar omogućuje proslijeđivanje proizvoljnog broja funkcija kao argumente. Unutar funkcije, vratili smo novu anonimnu funkciju koja ima parametar (incijalnaListaVrijednosti). Ova funkcija će biti pozvana kada se pozove rezultirajuća funkcija dobivena iz pipeFunkcija. Stvorili smo varijablu (vrijednost) koja će se koristiti za prolazak kroz svaku funkciju u callbacks. Početna vrijednost je jednaka incijalnoj listi vrijednosti. Kroz petlju prolazimo kroz svaku funkciju callback iz callbacks. Unutar petlje, svaku funkciju callback primjenjujemo na trenutnu vrijednost i rezultat pohranjujemo natrag u vrijednost. To nam omogućuje lančano izvođenje funkcija, gdje je izlaz jedne funkcije ulaz za sljedeću. Na kraju petlje, vraćamo konačnu vrijednost iz rezultirajuće funkcije.

Ovime smo omogućili kompoziciju funkcija u obliku pipe, gdje rezultat iz jedne funkcije postaje ulaz za sljedeću funkciju.

4. Funkcionalnosti metoda

Unutar ovog poglavlja pokazati ćemo po dva primjera za svaku funkciju te ćemo ih detaljno objasniti. Rezultati svih funkcionalnosti biti će prikazani unutar terminala i u potpoglavlju 4.4..

4.1. Primjeri korištenja mapFunkcija

Isječak kôda 4: Primjer1 korištenja mapFunkcija

```
1 const rezultat1 = mapFunkcija([1, 2, 3], (v) => v + 1);
2 console.log('MAP-Primjer1.:'+ ' ' +rezultat1);
```

U primjeru sa isječka koda 4 pozvali smo funkciju mapFunkcija s dvama argumentima: prva je lista [1, 2, 3] koju želimo transformirati, a druga je callback funkcija (v) => v + 1 koja definira kako želimo transformirati svaku vrijednost.

Prolazili smo kroz svaku vrijednost u ulaznoj listi. Za svaku vrijednost v u listi, primijenili smo callback funkciju (v) => v + 1 na tu vrijednost. U ovom slučaju, callback funkcija jednostavno dodaje broj 1 na svaku vrijednost. Zatim, linija console.log('MAP-Primjer1.:'+ ' ' +rezultat1); ispisuje rezultat na konzoli.

Isječak kôda 5: Primjer2 korištenja mapFunkcija

```
1 const listaObjekata = [
2   { ime: "sara", prezime: "susac" },
3   { ime: "Pero", prezime: "Perić" },
4 ];
5 const rezultat2 = mapFunkcija(listaObjekata, (obj) => `${obj.ime} ${obj.prezime}`);
6 console.log('MAP-Primjer2.:'+ ' ' +rezultat2);
```

Ovaj primjer koristi (mapFunkcija) za pretvaranje liste objekata u listu punih imena, spajajući ime i prezime svakog objekta u jedan string.

Za početak definirali smo listu objekata listaObjekata koja sadrži informacije o osobama (ime i prezime). Koristili smo funkciju mapFunkcija koja prima (listaObjekata) kao ulaznu listu i (callback) funkciju koja se primjenjuje na svaki element liste. U callback funkciji, kombinirali smo ime i prezime svakog objekta koristeći template string obj.ime i obj.prezime kako bismo dobili puno ime. Funkcija mapFunkcija prolazi kroz svaki objekt u listaObjekata, primjenjuje callback funkciju na svaki objekt i vraća rezultat kao novu listu. Dobivenu novu listu rezultata pohranjujemo u varijablu (rezultat2).

Konačno, rezultat (rezultat2) ispisujemo na konzoli pomoću console.log s odgovarajućom porukom.

4.2. Primjer korištenja reduceFunkcija

Isječak kôda 6: Primjer1 korištenja reduceFunkcija

```
1 const rezultat5 = reduceFunkcija([1,2,3,4], (acc,val) => {
2   if(val % 2 ===0) {
3     acc.push(val);
4   }
5   return acc;
6 }, [] );
7 console.log('REDUCE-Primjer1.:'+ ' ' +rezultat5);
```

Isječak koda 6 je primjer koji koristi reduceFunkcija za filtriranje brojeva iz liste koji su djeljivi s 2 i stvara novu listu samo s tim brojevima.

Koristili smo funkciju (reduceFunkcija) koja prihvaća tri argumenta: (incijalnaListaVrijednosti), (callback) funkciju i (početnaVrijednost). (incijalnaListaVrijednosti) je ulazna lista koju želimo reducirati. (callback) funkcija definira operaciju koja će se primijeniti na svaki element liste. (početnaVrijednost) je početna vrijednost akumulatora koji će se koristiti tijekom redukcije. U primjeru, ulazna lista je [1, 2, 3, 4], callback funkcija provjerava je li broj djeljiv s 2 i ako jest, dodaje ga u akumulator (acc). Početna vrijednost akumulatora je prazna lista []. Funkcija reduceFunkcija prolazi kroz svaki element (incijalnaListaVrijednosti) i primjenjuje (callback) funkciju na njega, a rezultat dodaje u akumulator. Na kraju, funkcija vraća akumulator kao rezultat. Dobiveni rezultat (rezultat5) je nova lista koja sadrži samo brojeve iz ulazne liste koji su djeljivi s 2.

Na kraju, rezultat (rezultat5) ispisujemo na konzoli pomoću console.log s odgovarajućom porukom.

Isječak kôda 7: Primjer2 korištenja reduceFunkcija

```
1 function spojistringove(acc, val) {
2   return acc + val;
3 }
4
5 const listastringova = ["Prikaz", " ", "rada", " ", "funkcije", " ", "reduce", "."];
6
7 const rezultat6 = reduceFunkcija(listastringova, spojistringove, "");
8 console.log('REDUCE-Primjer2.:'+ ' ' +rezultat6);
```

Primjer koristi funkciju reduceFunkcija za spajanje svih stringova iz liste u jedan spojeni string.

Definirali smo funkciju (spojistringove) koja ima dva parametra: (acc) akumulator i (val) trenutna vrijednost iz liste. Ova funkcija jednostavno spaja trenutnu vrijednost (val) s akumulatorom (acc) i vraća rezultat. Imamo ulaznu listu listastringova koja sadrži niz stringova: ["Prikaz", " ", "rada", " ", "funkcije", " ", "reduce", "."]. Koristimo funkciju (reduceFunkcija) s (listastringova) kao ulaznom listom, (spojistringove) kao callback funkciju i praznim stringom "" kao početnom vrijednošću akumulatora. Funkcija prolazi kroz svaki element (listastringova) i primjenjuje (callback) funkciju (spojistringove) na svaki element zajedno s akumulatorom. U svakoj iteraciji, trenutni string (val) se spaja s akumulatorom (acc) koristeći operator +. Na kraju svake iteracije, funkcija vraća ažurirani akumulator. Rezultat6 je spojeni string koji sadrži sve

elemente iz originalne liste, kojeg ispisujemo na konzoli pomoću `console.log` s odgovarajućom porukom.

4.3. Primjer korištenja pipeFunkcija

Isječak kôda 8: Primjer1 korištenja pipeFunkcija

```
1 function zbrajanje(listaVrijednosti) {
2   let rezultat = [];
3   for (const broj of listaVrijednosti) {
4     rezultat.push(broj + broj);
5   }
6   return rezultat;
7 }
8
9 function množenje(listaVrijednosti) {
10  let rezultat = [];
11  for (const broj of listaVrijednosti) {
12    rezultat.push(broj * broj);
13  }
14  return rezultat;
15 }
16
17 const rezultat3 = pipeFunkcija(zbrajanje, množenje)([1, 2, 3]);
18 console.log('PIPE-Primjer1: ' + ' ' + rezultat3);
```

Ovaj primjer demonstrira korištenje pipeFunkcija za slijedno izvršavanje funkcija zbrajanje i množenje na ulaznoj listi [1, 2, 3].

Za početak smo definirali funkciju zbrajanje koja prima listu vrijednosti i vraća novu listu u kojoj su vrijednosti svakog elementa udvostručene. Koristili smo petlju za prolazak kroz svaki element broj u listi (listaVrijednosti). U svakoj iteraciji petlje, vrijednost (broj) je zbrojena sa samim sobom (broj + broj) i rezultat je dodan u listu (rezultat) pomoću metode push. Na kraju, funkcija zbrajanje vraća listu (rezultat) koja sadrži udvostručene vrijednosti svakog elementa iz ulazne liste.

Također smo definirali funkciju množenje koja prima listu vrijednosti i vraća novu listu u kojoj su vrijednosti svakog elementa kvadrirane. Koristili smo sličnu logiku kao i u funkciji zbrajanje, ali smo umjesto zbrajanja koristili množenje (broj * broj) za svaki element liste. Konačno, koristili smo funkciju (pipeFunkcija) koja prima dva argumenta - funkcije (zbrajanje) i (množenje).

Pomoću pipeFunkcija smo stvorili novu funkciju koja će se izvršiti s argumentom [1, 2, 3]. Ta nova funkcija će prvo primijeniti funkciju zbrajanja na ulaznu listu, a zatim rezultirajuću listu proslijediti funkciji množenja.

Konačni rezultat rezultat3 će biti rezultat izvršavanja cijelog niza funkcija na ulaznoj listi [1, 2, 3]. Na kraju, rezultat (rezultat3) ispisujemo na konzoli pomoću `console.log` s odgovarajućom porukom.

Isječak kôda 9: Primjer2 korištenja pipeFunkcija

```
1 // Funkcija za konverziju svih slova u velika slova
2 const pretvorbaSlovaUVelika = (text) => text.toUpperCase();
3
4 // Funkcija za dodavanje razmaka između slova, uključujući razmak
5 const dodajRazmakIzmeđuSlova = (text) => text.split('').map((char) => char + '-').
  join('');
6
7 //funkcija koja na mjesto razmaka stavlja .
8 const zamjenaRazmakaSaTočkom = (text) => text.replace(/s/g, '.');
9
10 // Primjer korištenja pipeFunkcije za pretvorbu slova u Velika, dodavanja razmaka i
  zamjenu razmaka točkom
11 const rezultat4 = pipeFunkcija(pretvorbaSlovaUVelika, dodajRazmakIzmeđuSlova,
  zamjenaRazmakaSaTočkom)('Test pipe funkcije');
12 console.log('PIPE-Primjer2.:'+ ' ' +rezultat4);
```

U kodu sa isječka 9 vidimo primjenu funkcija (pretvorbaSlovaUVelika), (dodajRazmakIzmeđuSlova) i (zamjenaRazmakaSaTočkom) putem pipeFunkcija na zadani tekst "Test pipe funkcije". Ovaj primjer pokazuje korištenje (pipeFunkcija) za slijedno izvršavanje različitih transformacija teksta na ulaznom tekstu "Test pipe funkcije".

(pretvorbaSlovaUVelika) je funkcija koja prima tekst kao ulaz i vraća taj tekst pretvoren u velika slova pomoću metode toUpperCase(). Na primjer, ako je ulazni tekst "Test", rezultat će biti "TEST".

(dodajRazmakIzmeđuSlova) je funkcija koja prima tekst kao ulaz i dodaje razmak između svakog slova (uključujući i razmake) pomoću metoda split(""), map() i join(""). Na primjer, ako je ulazni tekst "Test", rezultat će biti "T-e-s-t-".

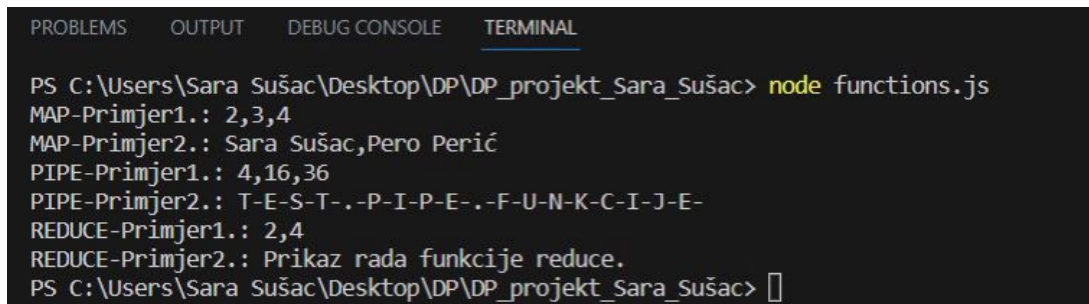
(zamjenaRazmakaSaTočkom) je funkcija koja prima tekst kao ulaz i zamjenjuje sve razmake točkom (.) pomoću metode replace. Na primjer, ako je ulazni tekst "Test pipe funkcije", rezultat će biti "Test.pipe.funkcije".

(pipeFunkcija) je funkcija koja prima niz funkcija kao argumente i stvara novu funkciju koja će izvršiti te funkcije slijeva nadesno. U ovom primjeru, koristimo (pipeFunkcija) za stvaranje nove funkcije koja će prvo primijeniti funkciju (pretvorbaSlovaUVelika) na ulazni tekst, zatim rezultirajući tekst proslijediti funkciji (dodajRazmakIzmeđuSlova), a na kraju rezultirajući tekst proslijediti funkciji (zamjenaRazmakaSaTočkom).

Konačni rezultat (rezultat4) je rezultat izvršavanja niza funkcija na ulaznom tekstu "Test pipe funkcije". U ovom slučaju, rezultat će biti "T-E-S-T-.P-I-P-E-.F-U-N-K-C-I-J-E". Rezultat se uspisuje na konzoli.

4.4. Outputi funkcija

Na slici 1. možemo vidjeti sve outpute koje smo dobili koristeći funkcije iz prijašnjih podpoglavlja. Ustvari se radi o screenshotu konzole, jer na njoj sve testiramo. Možemo vidjeti da sve funkcije pravilno rade.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Sara Sušac\Desktop\DP\DP_projekt_Sara_Sušac> node functions.js
MAP-Primjer1.: 2,3,4
MAP-Primjer2.: Sara Sušac,Pero Perić
PIPE-Primjer1.: 4,16,36
PIPE-Primjer2.: T-E-S-T-.-P-I-P-E-.-F-U-N-K-C-I-J-E-
REDUCE-Primjer1.: 2,4
REDUCE-Primjer2.: Prikaz rada funkcije reduce.
PS C:\Users\Sara Sušac\Desktop\DP\DP_projekt_Sara_Sušac> 
```

Slika 1: Outputi funkcija

5. Zaključak

Na temelju svega što je napravljeno, možemo zaključiti da funkcija `mapFunkcija` omogućuje primjenu zadane funkcije na svaki element liste, vraćajući novu listu rezultata.

Funkcija `reduceFunkcija` omogućuje smanjivanje liste vrijednosti na jednu pojedinačnu vrijednost primjenom zadane funkcije na svaki element liste.

Funkcija `pipeFunkcija` omogućuje slijedno izvršavanje više funkcija, pri čemu je izlaz jedne funkcije ulaz za sljedeću.

U primjerima korištenja funkcija, koristili smo različite transformacije nad listama i tekstom. Kroz primjere smo demonstrirali primjenu mapiranja, reduciranja i slijednog izvršavanja funkcija. Ove tehnike mogu biti korisne u različitim situacijama, kao što su obrada podataka, transformacija teksta ili manipulacija listama.

Popis literature

- [1] „What is imperative programming ? | Definition from TechTarget.” (), adresa: <https://www.techtarget.com/whatis/definition/imperative-programming>.
- [2] „Array.prototype.map() - JavaScript | MDN.” (), adresa: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map.
- [3] „Array.prototype.reduce() - JavaScript | MDN.” (), adresa: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce.
- [4] „Creating an ES6ish Compose in Javascript | by Drew Tipson | Medium.” (), adresa: <https://medium.com/@dtipson/creating-an-es6ish-compose-in-javascript-ac580b95104a>.

Popis slika

1.	Outputi funkcija	9
----	----------------------------	---

Popis isječka koda

1.	Funkcija mapFunkcija	3
2.	Funkcija reduceFunkcija	3
3.	Funkcija pipeFunkcija	4
4.	Primjer1 korištenja mapFunkcija	5
5.	Primjer2 korištenja mapFunkcija	5
6.	Primjer1 korištenja reduceFunkcija	6
7.	Primjer2 korištenja reduceFunkcija	6
8.	Primjer1 korištenja pipeFunkcija	7
9.	Primjer2 korištenja pipeFunkcija	8