Nizovi

Osnovni koncept programiranja je kako organizovati i čuvati podatke.

Jedan način da organizujemo podatke je da napravimo listu.
Kao ovu:

```
Spisak za kupovinu:
1. Jogurt
2. Banane
3. Sir
```

U JavaScript-u takođe možemo da napravimo listu, u vidu niza:

```
var spisak = ['Jogurt', 'Banane', 'Sir'];
```

Ove liste mogu sadržati bilo koji tip podataka (stringove, brojeve i booleane) i ove liste su numerisane, svaka stavka ima numerisanu poziciju.

Kreiranje niza

Niz kreiramo tako što definišemo promenjivu, a zatim dodamo vrednosti unutar uglastih zagrada.

```
var spisak = ['Jogurt', 'Banane', 'Sir'];

document.write(spisak);
```

Pristup sadržaju niza

Svaka stavka u nizu ima numerisanu poziciju. Pojedinačnim stavkama možemo pristupiti koristeći brojeve, baš kao i u običnoj listi.

Potrebno je obratiti pažnju na to da JavaScript počinje da broji od `0`, a ne od `1`, tako da će prva stavka u nizu biti na poziciji 0.
Prvu stavku u nizu možemo izabrati ovako:

```
var spisak = ['Jogurt', 'Banane', 'Sir'];
document.write(spisak[0]);  // Ispis: 'Jogurt'
```

Pojedinačnim karaktrerima u stingu možemo pristupiti na ovaj način:

```
var js = 'JavaScript';
document.write(js[6]);  // ispis: r
```

`r` će biti ispisano jer se ono nalazi na šestom mestu u stringu. Ovo funkcioniše zbog toga što JavaScript čuva stringove na sličan način kao i nizove.

Izmena elemenata

Do sada smo naučili kako pristupiti elementima nizova ili stringova koristeći njihov indeksni broj. Sada ćemo videti kako možemo promeniti elemente niza pomoću njihovih indeksa.

```
var spisak = ['Jogurt', 'Banane', 'Sir'];
spisak[2] = 'Cokolada';
document.write(spisak)
```

U primeru iznad `spisak` niz je sadržao četiti stavke.

Nakon toga smo jednu od stavki promenili sa `"Sir"` na `"Cokolada"`.

`spisak[2] = "Cokolada";` govori programu da izmeni stavku na poziciji 2 niza `spisak` na "Cokolada".

length osobina

Ponekad nam je potrebno da znamo koliko stavki se nalazi u nizu.

To možemo saznati pomoću ugrađene osobine `length`. Kada je dodamo nekoj promenjivoj koja sadrži niz, kao rezultat ćemo dobiti broj stavki u tom nizu.

```
var spisak = ['Jogurt', 'Banane', 'Sir'];
document.write(spisak.length);  // ispis: 3
```

U primeru iznad ispisali smo `spisak.length` kao rezultat dobićemo `length` osobinu `spisak` niza.

push i pop metode

JavaScript poseduje ugrađene metode za nizove koje mogu pomoći u nekim uobičajenim zadacima. Ovo su neke od njih.

`.push()` metoda nam omogućava da dodajemo stavke na kraj niza, kao u primeru ispod:

```
var spisak = ['stavka 0', 'stavka 1', 'stavka 2'];
spisak.push('stavka 3', 'stavka 4');
```

Nakon izvršavanja koda niz `spisak` će izgledati ovako:

```
['stavka 0', 'stavka 1', 'stavka 2', 'stavka 3', 'stavka 4'];
```

Metoda `.pop()` nam služi da uklonimo poslednju stavku iz niza.

```
var spisak = ['stavka 0', 'stavka 1', 'stavka 2'];
spisak.pop();
document.write(spisak);
```

U primeru iznad, pomoću `.pop()` metode će iz niza `spisak` biti uklonjena `stavka 2` sa kraja.

Ostale Array metode

Postoji mnogo metoda za nizove osim `.push()` and `.pop()`. Možete ih naći u Mozilla Developer Network (MDN) documentaciji.

Još neke JavaScript metode koje se često koriste su `.join()`, `.slice()`, `.splice()`, i `.concat()`

1. Metoda `.join()` povezuje sve elemente niza u string.

```
var elements = ['Fire', 'Wind', 'Rain'];

document.write(elements.join());
// ispis: Fire,Wind,Rain

document.write(elements.join(''));
// ispis: FireWindRain

document.write(elements.join('-'));
// ispis: Fire-Wind-Rain
```

2. Metoda `.slice()` vraća kopiju dela niza kao novi niz od izabranog početka do kraja. Originalni niz nije izmenjen.

```
var animals = ['ant', 'bison', 'camel', 'duck', 'elephant'];

document.write(animals.slice(2));
// ispis: Array ["camel", "duck", "elephant"]

document.write(animals.slice(2, 4));
// ispis: Array ["camel", "duck"]

document.write(animals.slice(1, 5));
// ispis: Array ["bison", "camel", "duck", "elephant"]
```

3. Metoda `.splice()` menja sadržaj niza uklanjanjem postojećih elemenata i/ili dodavanjem novih elemenata.

```
var months = ['Jan', 'March', 'April', 'June'];
months.splice(1, 0, 'Feb');
// ubacule element na poziciju 1
document.write(months);
// ispis: Array ['Jan', 'Feb', 'March', 'April', 'June']

months.splice(4, 1, 'May');
// zamenjuje 1 element na poziciji 4
document.write(months);
// ispis: Array ['Jan', 'Feb', 'March', 'April', 'May']
```

4. `.concat()` metoda se koristi za spajanje dva ili više nizova. Ova metoda ne menja postojeće nizove, već vraća rezultat kao novi niz.

```
var array1 = ['a', 'b', 'c'];
var array2 = ['d', 'e', 'f'];

document.write(array1.concat(array2));
// ispis: Array ["a", "b", "c", "d", "e", "f"]
```

Petlje

Jedna od najznačajnijih mogućnosti računara je što mogu ponoviti neki zadatak više puta. Petlje nam omogućavaju da kažemo računaru da krene da izvršava neki blok koda više puta tako da mi ne moramo pisati isti proces iznova i iznova.

Petlje su posebno korisne kada imamo niz u kom želimo da uradimo nešto za svaku od njegovih stavki, kao što je ispisivanje svake stavke na ekranu.

Petlje možemo podeliti na dve osnovne vrste:

1. `for` petlje, koje služe da izvršimo neki blok koda određen broj puta.
2. `while` petlje, koje služe da izvršimo neki blok koda nepoznat broj puta.

for petlje

Umesto da unosimo isti kod nanovo i nanovo, možemo da napravimo da računar prođe kroz niz za nas. To možemo uraditi pomoću `for` petlje.

Primer:

```
var zivotinje = ["medved", "slon", "lav"];
for (var i = 0; i < zivotinje.length; i++)
{
document.write(zivotinje[i] + "<br>");
}
```

Ispis bi izgledao ovako:

```
medved
slon
lav
```

Sintaksa na prvi pogled može delovati komplikovano, pa ćemo je podeliti na četiri dela:

1. Unutar obicnih zagrada `for` petlje, kao početno stanje postavljamo `var i = 0`, što znači da će petlja početi da broji od `0`.
2. Kao uslov za zaustavljanje postavljamo `i < zivotinje.length`, što znači pa će se petlja izvršavati sve dok je `i` manji od dužine niza. Kada `i` postane jednak dužini niza, petlja će prestati da se izvršava.
3. `i++` služi da se nakon svakog izvršavanja `i` poveća za 1.
4. Unutar `{ }` vitičastih zagrada se stavlja kod koji program treba da ponavlja.

Ispis pomoću petlje funkcinoše zbog toga što je `i` promenljiva koju smo kreirali unutar zagrada `for` petlje, jednaka broju. Pri prvom izvršavanju `i` će biti jednak `0`, pri drugom `1` i tako dalje.

To nam omogućava ispisivanje `zivotinje[0]`, `zivotinje[1]`, `zivotinje[2]` programski umesto ručno, ubacivanjem `i` unutar uglastih zagrada ovako: `zivotinje[i]`.

for petlja unazad

Kao što možemo napraviti da `for` petlja prolazi unapred kroz niz, tako možemo napraviti i da prolazi unazad.

Da bi to učinili moramo izmeniti kod unutar običnih zagrada `for` petlje:

1. Kao početno stanje potrebno je da stavimo dužinu niza minus 1
2. Kao uslov za zaustavljanje treba da stavimo `i >= 1`.
3. Nakon svakog izvršavanja `animalIndex` treba da se smanjuje za 1 `i--`.

```
var zivotinje = ["medved", "slon", "lav"];
for (var i = zivotinje.length - 1; i >= 0; i--)
{
document.write(zivotinje[i] + "<br>");
}
```

while petlja

`for` petlje su odlične, ali imaju ograničenje, a to je da moramo da znamo koliko puta želimo da se petlja izvrši.

Kada nam je potrebno da se petlja izvršava sve dok je neki uslov ispunjen koristimo `while` petlju.

```
while (uslov)
{
// kod koji treba da se izvrsi
}
```

1. Petlja počinje sa `while`.
2. Unutar običnih zagrada se unosi uslov. Sve dok uslov kao rezultat vraca `true`, petlja će se izvršavati.
3. Unutar vitičastih zagrada se stavlja kod koji petlja treba da izvršava.

do while petlja

`do while` petlja je je vrsta `while` petlje. Ova petlja će po prvi put izvršiti blok koda bez proveravanja da li je uslov `true`, nakon toga će ponavaljati petlju sve dok je uslov `true`.

```
do {
// kod koji treba da se izvrsi
}
while (uslov);
```

1. Petlja počinje sa `do`.
2. Unutar vitičastih zagrada se stavlja kod koji petlja treba da izvršava
3. Nakon toga se piše `while` , pa unutar običnih zagrada se unosi uslov. Sve dok uslov kao rezultat vraca `true`, petlja će se izvršavati.

Beskonačna petlja

Sve petlje bi trebalo da se izvrše konačan broj puta, u suprotnom su to onda beskonačne petlje. Beskonačna petlja znači da je došlo do greške u logici petlje jer uslov za zaustavljanje nikada nije postignut.

Ovu vrstu greške je teže otkriti jer je sintaksa ispravna i kod će se izvršiti. Kada dođe do beskonačne petlje, program će se zamrzniti i neće odgovarati na dalje naredbe.

Ispod je primer koda koji će se izvršavati zauvek..

```
// Initiate an infinite loop
while (true)
{
// execute code forever
}
```

```
for (let i = 0; i < array.length; i--)
{
//some code
}
```

Zadatak 1.

Potrebno je napisati JavaScript program koji pomoću for petlje na ekranu ispisuje brojeve od nula do unetog broja.

- Broj do kog želimo da se broji se unosi pomoću prompta
- Nakon toga se kreira for petlja pomoću koje se na ekranu ispisuju brojevi od nula do tog broja, jedan ispod drugog.

Zadatak 2.

Izmeniti prethodni zadatak tako da program ispisuje samo parne brojeve.

- Broj do kog želimo da se broji se unosi pomoću prompta
- Nakon toga se kreira for petlja pomoću koje se na ekranu ispisuju parni brojevi od nula do tog broja, jedan ispod drugog.

Zadatak 3.

Izmeniti Zadatak 1 tako da for petlje ide unazad, odnosno da program ispisuje brojeve od unetog broja do nula.

- Broj od kog želimo da se broji se unosi pomoću prompta
- Nakon toga se kreira for petlja pomoću koje se na ekranu ispisuju brojevi od unetog broja do nula, jedan ispod drugog.

Zadatak 4.

Potrebno je napisati JavaScript program za pogađanje nasumičnog broja od 0 do 10.

- Nasumičan broj od 0 do 10 se generiše pomoću matematičke metode
- Nakon toga korisnik pomoću prompta unosi željeni broj
- Vrednosti unetog i nasumičnog broja se upoređuju i pomoću do while petlje se vraća prompt sve dok korisnik ne pogodi broj
- Kada korisnik pogodi broj na ekranu se pojavljuje alert sa tekstom Cestitamo.

Zadatak 5.

Potrebno je napisati JavaScript program koji pomoću petlje ispisuje tablicu množenja.

- Željeni broj se unosi pomoću prompta
- Nakon toga se na ekranu ispisuje tablica množenja za taj broj

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50

Zadatak 6.

Potrebno je napisati JavaScript program koji pomoću petlje generiše HTML tabelu sa zeljenim brojem redova.

- Željeni broj redova se unosi promoću prompta
- Nakon toga se pomoću for petlje generiše tabela sa unetim brojem redova
- Prva kolona tabele je redni broj, druga ostaje prazna.

#	**Ime i prezime**
1.	
2.	
3.	
4.	
5.	

Zadatak 7.

Potrebno je izmeniti prethodni zadatak tako da korisnik osim kreiranja tabele ima mogućnost popunjavanja polja u drugoj koloni.

- Željeni broj redova se unosi promoću prompta
- Nakon toga se za svaki red otvara prompt u koji se unosi vrednost tog polja u tabeli
- Kada su popunjena sva polja, na ekranu se pojavljuje tabela.

#	Ime i prezime
1.	Petar Petrovic
2.	Milan Maric
3.	Jovana Pavlovic
4.	Slobodan Bozic
5.	Marija Pavic