Vježba 6 - Node.js uvod

Node.js je okruženje za izvršavanje programa pisanih u JavaScript programskom jeziku. Baziran je na V8 Chrome JavaScript engine-u, otvorenog je koda i dozvoljava izvršavanje JavaScript koda na serveru. Node.js je sistem baziran na događajima (event-ima). Sve što se u Node-u desi je reakcija na neki događaj. Callback funkcije su funkcije koje se pozivaju kada se detektuje određeni događaj. Prijem jednog zahtjeva je primjer jednog događaja kao što su i otvaranje datoteke, upisivanje sadržaja u datoteku i sl. Callback je funkcija koja se prosljeđuje kao parametar nekoj drugoj funkciji koja će je pozvati kada se neki dio koda izvrši ili ako se desi neki događaj. U Node-u callback funkcije su realizovane obično tako da imaju dva parametra. Prvi govori o tome da li je došlo do greške, a drugi daje podatke ukoliko je operacija uspješno urađena. Premještanje koda u callback funkcije osigurava da se ne moraju blokirati ostale funkcionalnosti koje bi se mogle inače izvršavati. Callback funkcijom se predaje odgovornost mehanizmu koji asinhrono obavlja posao da će je pozvati kada bude vrijeme za to. U Node-u se za potrebe rada sa callback funkcijama koristi event loop mehanizam. Ovaj mehanizam ima više faza kroz koje periodično prolazi i poziva callback funkcije koje se nalaze u trenutnoj fazi. Svaka od faza je povezana sa nekim od načina kako je moglo doći do poziva callback-a: callback-i pozvani putem timera, I/O callback-i, pool novih događaja(evenata), callback funkcije pozvane sa setImmediate i na kraju je posljednja faza kojim se zatvaraju svi procesirani događaji. O tome da se poslovi obavljaju asinhrono brine sam Node. On nastoji da posao proslijedi operativnom sistemu koji će interno asinhrono obaviti zadatke kao što su prijem HTTP zahtjeva, tcp/udp socketi, dns resolve i sl. Za posao koji operativni sistem ne može odraditi asinhrono Node ima i niz threadova kojima može proslijediti posao. Sve navedeno zajedno osigurava da u vašoj Node aplikaciji ne dođe do blokiranja koda, odnosno da jedan klijent ne mora da čeka da se obradi zahtjev prethodnog klijenta.

Node možemo koristiti za sve funkcionalnosti koje jedan web server treba da ima: generisanje dinamičkog web sadržaja, rad sa datotekama, rad sa bazama podataka, pružanje funkcionalnosti web servisa, itd. Na vježbama i dalje na projektu koristit ćemo **verziju 9 Node.js-a (ili novija)**.

Da bi uspješno koristili Node.js trebate razumjeti da je on baziran na sistemu modula. Module uključujemo po potrebi, kada nam neka od funkcionalnosti zatreba. Primjeri modula koji se najčešće koriste:

- **http** - prijem, obrada i slanje http zahtjeva
- **fs** - rad sa fajlovima
- **stream** - rad sa streamovima podataka
- itd …

Modul uključujemo sa funkcijom **require()** gdje u zagradi prosljeđujemo kao parametar naziv modula ili javascript fajl koji sadrži implementaciju modula.

Primjer 1:

```
var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write('<h1>Putanja zahtjeva: '+req.url+'</h1>');
    res.end('<b>Metoda:'+req.method+'</b>')
}).listen(8080);
```

U prvoj liniji vidimo kako se uključuje modul http koji će nam omogućiti da se naša skripta ponaša kao mali web server koji osluškuje zahtjeve na portu 8080. Server se kreira sa funkcijom **createServer** koja kao parametar prima callback funkciju. Ova funkcija se poziva kada server zaprimi zahtjev. Parametri callback funkcije su request i response. Request označava dolaznu poruku (upućeni zahtjev prema web serveru), a response predstavlja odgovor servera koji se šalje u vidu streama u kojeg se mogu upisivati podaci.
U response-u specificiramo sve šta će odgovor od servera imati:

- Header-e postavljamo sa funkcijama **writeHead** - koja kao parametre prima status code odgovora i objekat sa headerima kodiranim kao naziv_header-a:vrijednost, ili sa funkcijom **setHeader** - kao parametre prosljeđujemo naziv headera i vrijednost headera zapisane kao stringovi.
- U tijelo odgovora upisujemo podakte sa funkcijama **write** i **end**. Razlika između ove dvije metode je što **end** zatvara stream nakon svog izvršavanja, odnosno označava kraj odgovora.

Snimite ovaj kod kao primjer1.js i pokrenite ga kucanjem “node primjer1.js” ili kroz visual studio code. U web pregledniku otvorite adresu “localhost:8080”.

Zadatak 1. Ručno kreirajte datoteku imenik.txt. U ovoj datoteci podaci su zapisani u CSV formatu na način da se u jednom redu nalazi sljedeće:

Ime, prezime, adresa, broj telefona

Napravite u Node-u skriptu koja će čekati na GET zahtjev i u slučaju kada se on desi skripta odgovara u obliku JSON objekta koji sadrži niz JSON objekata redova, npr za sljedeći red:

Mujo,Mujić,Ferhadija 1,123/123-123

Odgovor bi bio: *[{‘ime’:’Mujo’,’prezime’:’Mujić’,’adresa’:’Ferhadija 1’,’broj_telefona’:’123/123-123’}]*

*Da bi iščitali sadržaj neke datoteke koristite fs paket. U ovom primjeru možete koristiti funkciju **readFile**. Ona kao parametre prima putanju do datoteke i callback funkciju koja se poziva kada je čitava datoteka pročitana. Callback kao parametre ima podatke o grešci koja se desila i pročitane podatke iz datoteke u vidu buffera. Buffer možete konvertovati u string tako što pozovete funkciju **toString(kodiranje)**, gdje kodiranje može biti: ‘utf-8’,’utf-16le’, ‘ucs-2’, ‘latin1’ i ‘binary’.*

*Kako bi zadatak bio ispravan trebate podesiti i odgovarajući tip odgovora (Content-Type), a odgovor mora biti validan u JSON formatu. Da li je odgovor ispravno zapisan možete provjeriti na https://jsonlint.com/. HTTP zahtjev možete uputiti otvaranjem linka **localhost:8080** u web pregledniku ili slanjem GET zahtjeva kroz Postman na isti link.*

Zadatak 2. Proširite zadatak 1 tako da ukoliko je proslijeđen parametar "q" ispiše sve redove čije ime sadrži vrijednost iz parametra q. Uzmite tako da je pretraga neosjetljiva na velika slova.

Da bi izdvojili parametre iz url-a možete koristiti url paket. Parametar se nalazi u query dijelu url-a i možete ga dobiti koristeći klasu URLSearchParams iz url paketa. Url zahtjeva dobijate kao u primjeru 1.

Primjer 2. Napišite dio koda koji će prepoznati da li je upućen POST zahtjev i u slučaju da jeste dodat će novi red u datoteku imenik.txt. Podaci u zahtjevu neka su zapisani u x-www-form-urlencoded formatu. **Primjer isprobajte slanjem zahtjeva kroz Postman!**
Rješenje:

```
const http = require('http');
const fs = require('fs');
const url = require('url');
http.createServer(function(req,res){
  if(req.method=='POST'){
     let tijeloZahtjeva = '';
     req.on('data',function(data){
        tijeloZahtjeva+=data;
     });
     req.on('end',function(){
        //primljen čitav zahtjev
        let parametri = new url.URLSearchParams(tijeloZahtjeva);
        let novaLinija = parametri.get('ime')+","+parametri.get('prezime')+
           ","+parametri.get('adresa')+","+parametri.get('broj_telefona');
        fs.appendFile('imenik.txt',novaLinija,function(err){
           if(err) throw err;
           console.log("Novi red uspješno dodan!");
           res.writeHead(200,{});
           res.end(parametri.toString());
        });
     });
  }
}).listen(8080);
```

U ovom primjeru koristimo tri modula, http i fs smo ranije koristili i url modul za rad sa url-ovima.

- Kreiramo server kao i u prvom primjeru

- Tijelo iz POST zahtjeva dobivamo koristeći dva eventa **‘data’** i **‘end’**. Data event se poziva kada se dobavi dio tijela zahtjeva, a end kada je dobavljen čitav zahtjev.
- Kada se dobavi čitav zahtjev tada možemo procesirati tijelo zahtjeva. U ovom primjeru predpostavlja se da su podaci u tijelu zahtjeva formatirani u obliku **‘x-www-form-urlencoded’** tako da možemo koristiti URLSearchParams (on omogućava parsiranje i rad sa query dijelom url-a, što x-www-form-urlencoded podaci ustvari i jesu).
- Parametre iz URLSearchParams dobijamo sa get metodom, gdje kao parametar prosljeđujemo naziv parametra.
- **fs.appendFile** dodaje na kraj datoteke novi sadržaj. Putanja do datoteke se navodi u prvom parametru funkcije, a dodani sadržaj u drugom parametru. Treći parametar je callback funkcija.
- Na kraju u odgovor upisujemo status kod 200 i zatvaramo odgovor sa nizom parametara koji su upravo dodani u datoteku.

Zadatak 3. Napravite HTML formu koja će pozivati prethodni primjer i ispravite ga da, ukoliko je zahtjev upućen sa forme, nakon upisa novog reda u datoteku na ekranu prikaže tabela (HTML dokument) svih stavki iz imenika.
Zadatak 4. Dodajte u Primjer 2 funkcionalnost koja će propoznati da li je zahtjev upućen na url **localhost:8080/json** i ukoliko jeste preuzet će podatke iz tijela zahtjeva pretpostavljajući da su oni formatirani u JSON formatu. Poslije preuzimanja podataka dodajte ih u novi red u imenik.txt.