# Uvod u korišćenje servera realizovanog preko Node.js

## 1.Uvod

Node.js je okruženje za Javascript koje je bazirano na V8 Javascript engine-u kog koristi Google Chrome. Koristi neblokirajući, model vođen događajima za unos i ispis, što ga čini efikasnim i neopterećujućim za računar koji ga pokreće. Node-ov ekosistem za pakete, npm, je najveći ekosistem biblioteka otvorenog izvornog koda na svetu.

## 2.Instalacija

Na [node.js stranici](https://nodejs.org/en/) nađite odgovarajući instalacioni fajl za vaš operativni sistem. Izaberite verziju kod koje piše da je „Recommended for most users“, skinite je i instalirajte. Pratite instrukcije koje su vam date u toko instalacije i restartujte računar.

Nakon što vam se računar upali, otvorite Command Prompt i ukucajte „node -v“ (bez navodnika). Ukoliko dobijete neki broj kao ispis, uspešno ste instalirali Node. Pošto će nam i npm biti potreban za korišćenje ovog koda, ukucajte i npm -v. Ako opet dobijete broj, uspešno ste instalirali sve što vam treba i možete započeti sa radom.

## 3.Pokretanje

Skinite folder sa kosturom back enda i otvorite komandnu liniju u tom folderu (onom folderu gde su app.js i package.json). U terminalu ukucajte npm. Ova komanda će vam na internetu naći sve pakete koje ovaj projekat koristi i skinuti ih na vaš računar.

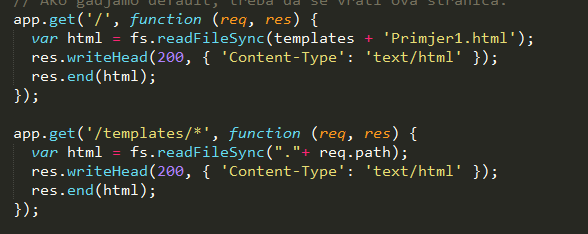
Sav kod koji server koristi se nalazi u fajlu app.js. Javascript fajlovi se u Node-u pokreću tako što kucate „node [ime fajla]“. Pokrenite app.js i trebao bi vam ispisati „Listening at http://localhost:3000” na konzolu.

## 4.Kod

Ako otvorite app.js u nekom tekst editoru, možete da vidite kod našeg servera. Prvih 11 linija predstavljaju eksterne pakete koje server koristi i neka njegova podešavanja i ne biste se trebali obazirati na njih.

Nakon deklaracije par globalnih promenljivih sledi više blokova koda koji počinju sa „app.get ...“. Ovim blokovima definišete kako će se vaš server ponašati kada mu klijent šalje zahteve. Za razliku od dosadašnjeg rada, vi više ne pristupate fajlovima direktno, nego šaljete zahteve vašem serveru (app.js), i u zavisnosti od toga kakav zahtev mu šaljete on vam daje odgovor.

U veb programiranju se većinom koriste GET, POST, DELETE i PUT zahtevi. GET zahtev se, kao što mu ime kaže, koristi kada želimo da dobijemo nešto od servera. DELETE se koristi kada želimo da izbrišemo nešto. POST se koristi kada klijent šalje podatke serveru i očekuje da se napravi neki novi entitet, a PUT se koristi kada želimo da izmenimo neki postojeći entitet. Funkcije app.get(), app.post(), app.delete() i app.put() odgovaraju ovim zahtevima.



Slika 1. Primer funkcija koje definišu odgovor na zahteve.

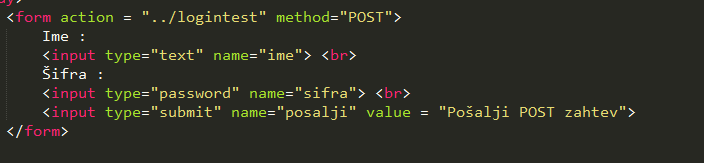
Sama funkcija app.get() (ili app.post() itd) prihvata dva argumenta. Prvi je adresa na koju je poslat zahtev i predstavlja ono što će pisati nakon localhost:3000/ u adresnom delu vašeg veb pretraživača (kada bismo rekli app.get(’/glavna.html’ , [dalji kod]), dalji kod bi se ticao zahteva na localhost:3000/glavna.html). Ako obratite pažnju, videćete da neke od adresa imaju \* u sebi. Taj simbol znači da u tom delu adrese može ići bilo šta. Ako piše da se zahtevi sa adresom ‘/templates/\*’ obrađuju na jedan način, taj način će pokrivati i zahteve na ’localhost:3000/templates/prva.html’ i ’localhost:300/templates/prva.html’.

Drugi argument je funkcija koja se poziva kada je zahtev poslat na tu adresu. Ona prihvata dva argumenta, req, tj. request ili zahtev, i res, tj. response ili odgovor. U zahtevu se može nalaziti više informacija, što je naročito bitno kada se obrađuje post zahtev. Odgovor ima zaglavlje, koje najčešće opisuje veb pretraživaču korisnika na koji način treba da obradi njegov sadržaj, ali često ima i drugih informacija, i sadržaj.

U odgovor na get zahtev se najčešće smeštaju fajlovi sa fajl sistema servera. Biblioteka za obradu fajl sistema se u ovoj aplikaciji zove fs i može da čita fajlove preko readFileSync(lokacija fajla) i piše fajlove preko writeFileSync(lokacija fajla, šta će se pisati u fajl). U odgovorima na get zahteve se uglavnom koristi da uzme html/css/javascript fajlove koje će klijent tražiti.

Pomoću funkcije res.writeHead(kod, sadržaj) mi u definišemo zaglavlje odgovora. U slučaju našeg servera to je definisanje toga koji tip dokumenta taj odgovor sadrži. Pomoću res.end() mi smeštamo sam sadržaj u odgovor i šaljemo ga klijentu.

App.post() možete da koristite da obrađujete podatke iz formi na vašem sajtu. Korišćenjem action atributa forme možete da podesite gde forma šalje svoj zahtev i onda u samom serveru podesite kako će on reagovati na to. Svi podaci iz vaše forme će biti smešteni u telo POST zahteva pod onim imenom koje ste definisali name atributom inputa i možete im veoma lako pristupiti.



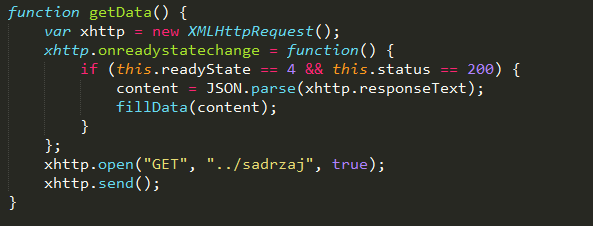
Slika 2. Forma u HTML-u



Slika 3. Način obrade podataka iz te forme

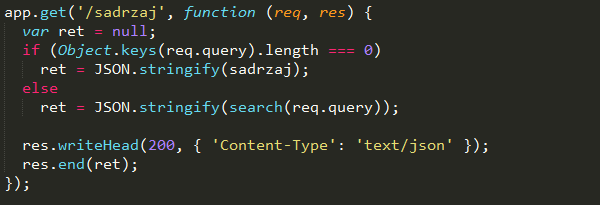
Nakon što obradite POST zahtev možete ili da pošaljete neki HTML koji će se prikazati klijentu ili preusmerite korisnika na drugu adresu. Preusmeravanje se vrši funkcijom res.redirect(adresa). Pretraživač će nakon kraja obrade zahteva biti preusmeren na tu adresu.

Ukoliko je potrebno da neka stranica dobije podatke kojima će se popuniti, potrebno je da ona napravi zahtev ka našem serveru. To se radi pomoću XMLHttpRequest objekata. Njegov onreadystatechange atribut je funkcija koja će se pozvati kad god se menja njegov readyState atribut. Kada je readyState jednak broju 4, to znači da je zahtev gotov i da je stigao odgovor servera. Pošto će on slati podatke u obliku JSON-a, parsiramo ih JSON-om.



Slika 4. Primer GET zahteva izvršenog u Javascriptu

Open metoda XMLHttpRequest-a prihvata tri argumenta, vrstu zahteva, URL gde se šalje zahtev i to da li zahtev treba da je asinhron ili ne (stavite ovo uvek na true).



Slika 5. Primer funkcije za odgovor na GET iz slike 4.

Odgovor u ovom slučaju dosta podseća na odgovore koji vraćaju html, CSS ili javascript, samo što šaljete podatke koje pretvorite u JSON string.

U slučaju da želite da gledate pojedinačni element iz liste podataka, potrebno je da napravite jednu generičku stranu za njegovo prikazivanje (u našem serveru to je *element.html*). Ako upalite server i odete samo na tu stranu, primetićete da vam ništa neće prikazati. Ali, ako pristupite toj strani kroz linkove koji su u glavnoj strani, prikazivaće detalje izabranog elementa. To je zbog toga što ti liknovi URL-u strane dodaju upitni string.



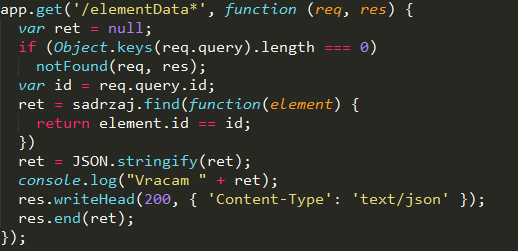
Slika 4. Primer upitnog stringa posle URL-a

Upitni string je sve ono što se u URL-u nalazi posle upitnika. Njemu Javascript lako može da pristupi preko window.location.search. Ako definišemo elementdata\* kao adresu na koju se klijent obraća serveru za pojedinačne elemente, XMLHttpRequest objekat će izgledati isto kao u prošlom primeru, samo što će ići na sledeću adresu:



Slika 6. Primer adresiranja zahteva koji očekuje samo jedan element kao odgovor

Server može pristupiti svakom delu upitnog stringa preko req.query.[ime dela]. U našem slučaju imamo samo id, ali moguće je staviti više delova upitnog stringa u URL tako što ih razdvojima znakom &. Sa serverske strane će onda obrada zahteva izgledati ovako:



Slika 7. Serverski odgovor na zahtev objašnjen iznad.