Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/07BE13Banjanin

PARADIGMIČKA I FUNKCIONALNA ANALIZA NODE.JS PLATFORME

PARADIGMIC AND FUNCTIONAL ANALYSIS OF NODE.JS PLATFORM

Una Banjanin, Srđan Popov, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu prezentovani su osnovni koncepti i karakteristike NodeJs platforme. Takođe, prikazana je i uopšteno skript paradigma kao i sami aspekti paradigme. Implementirana aplikacija Super Chat je jednostavna web aplikacija za razmenu poruka.*

Ključne reči: *NodeJs, Objektno-orijentisana paradigma, JavaScript, AJAX*

Abstract – *This thesis is presented by the basic concepts and characteristics of the NodeJs platform. There is also presented script paradigm as well as the aspects of the paradigm. Implemented Super Chat application is a simple web application for chat.*

Keywords: *NodeJs, Object-oriented paradigm, JavaScript, AJAX*

1. UVOD

Node.Js (takođe nazvan i Node) je platforma izgradjena na Google Chrome V8 izvršnoj mašini za lako pokretanje brzih, skalabilnih i laganih aplikacija [1].

U ovom radu se bavimo kako NodeJs-om, tako i samom skript paradigmom kao i njenim aspektima. Glavna karakteristika Node-a je njegova upotreba neblokirajućeg I/O vođenog događaja sa asihronim modelom programiranja koji ostaju efikasni u upravljanju konkurentnosti. Node se razlikuje od JavaScripta koji opisujemo, naglašavajući neke nedostatke koje Node pokriva. Isto tako predstavljajući AJAX sa njegovim prednostima i manama, pokazujemo kako Node nadmašuje AJAX što se tiče upotrebljivosti razvojnih aplikacija u realnom vremenu.

2. SKRIPT PARADIGMA

2.1 Šta je skriptni jezik

Skriptni jezik je oblik programskog jezika koji se koristi za kreiranje skripti ili bita koda. Jezici za pisanje skripti često se pišu kako bi se olakšale i poboljšale karakteristike internet stranice.
Te karakteristike se obrađuju na serveru, ali se skripta određene stranice pokreće u internet pretraživaču korisnika.
Jezik za pisanje skripti kontroliše rad normalno interaktivnog programa, dajući mu redosled rada za sve „na jednoj gomili".

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Srđan Popov, vanr. prof.

2.2 Prednosti i nedostaci skriptnog jezika

Što se tiče samih prednosti ovog jezika, navešćemo nekoliko značajnih: Lako se uči i koristi, potrebno je minimalno znanje ili iskustvo u programiranju, omogućava izvođenje složenih zadataka u relativno malom broju koraka.
Nedostaci su ti što može doći do sporijeg pokretanja budući da su kodovi interpretirani i da nisu kompajlirani u mašinski kod. Takođe, ukoliko dođe do grešaka, one mogu biti teže za otklanjanje jer nije dostupno izvorno razvojno okruženje.

2.3 Zašto moderne internet stranice koriste JavaScript

Podrška za pretraživač, Korišćenjem JavaScript-a možemo dodati različite funkcije kao što su autentifikacija korisika, validacija korisnika, itd. JavaScript lako čita i piše HTML elemente i može se lako ugraditi u HTML, akcioni događaj se može kreirati korišćenjem JavaScripta u trenutku kada korisnik pritisne dugme.

2.4 Zašto koristiti skriptne jezike

Vrlo često želimo automatizovati jednostavan zadatak - pokretanje nekoliko programa u nizu, instaliranje programa, pa čak i pisanje jednostavnog skripta ili GUI-a za pokretanje programa koji zahteva mnogo parametara. Upravo skriptni jezici nam dozvoljavaju da tako nešto brzo napišemo i pokrenemo bez kompajliranja. Takođe imaju dobru podršku za pokretanje procesa i njihovo kontrolisanje.

2.5 Klijentska i serverska strana

Na strani klijenta, skriptovanje vrši svu računicu na korisničkom računaru. Internet pretraživač ili određeni dodatak čita skriptu i pretvara je u vizuelnu internet stranicu. **Front end** kontekst koristi skriptovanje kroz korisnički interfejs. Većina informacija koje korisnik unosi u internet stranicu ostraje na strani klijenta i ponekad se vraća na server. JavaScript može dobiti spoljne slike i datoteke.

3. PREDSTAVLJANJE NODE.JS

Node.Js (Node) je okruženje koje je prvobitno razvijeno 2009. godine od strane Riana Dahla za razvoj aplikacija na strani servera.

Može se smatrati serverskim JavaScriptom. Napravljen je da se bavi problemima koje platforme mogu imati sa performansama u mrežnim komunikacijama – konkretno misleći na prekomerno vreme obrade internet zahteva i odgovora na isti.

Node je zasnovan na događajima, a ne na nitima. Koristi petlju događaja u okviru jedne niti umesto više niti i može da se skalira na milion istovremenih veza.

3.1 Ciljevi studije

Ubrzo nakon što je nastao WWW(World Wide Web), stvoren je i JavaScript. Kao takav, igrao je ključnu ulogu u dodavanju interakcije u korisnički interfejs internet aplikacija i internet stranica do izdanja HTML5 i modernih JavaScript okvira**.** Uprkos ovom napretku, JavaScript je smatran jezikom skriptovanja za programiranje na strani klijenta, koji kao takav, radi isključivo iz internet pretraživača. Međutim, ovaj pristup promenio se sa razvojem serverskog JavaScripta (među kojima je istaknut Node).

3.2 Pravljenje servera pomoću Node-a

Jedna od uobičajenih upotreba Node-a je za izgradnju servera. Može se koristiti za kreiranje različitih tipova servera.

3.3 Node arhitektura

Node.Js se sastoji od više komponenti. V8 je JavaScript mehanizam. Libeio obrađuje unutrašnji bazen koji se koristi da bi sinhroni POSIX pozivi postali asinhroni u petlji događaja. Asinhroni sistem datoteka se ne koristi, već se blokiranje I/O obavlja u njegovoj vlastitoj niti, takav da je ne-blokirajući u petlji događaja u Node.Js-u. Libev je petlja događaja. [2]

Node.Js koristi tehnike kao što su **kqueue, select, epoll** ili **/dev/poll** za dobijanje obaveštenja iz operativnog sistema kada dolaze nove konekcije, zatim otpremljuje nove događaje programeru, koji dalje rukuje njima.

3.4 Node.Js Moduli

Moduli predstavljaju dodatke i proširenja za Node koji pomažu u razvojnom procesu. Node modul izlaže javni API(Interfejs za programiranje aplikacija) koji se može koristiti nakon uvoza modula u trenutni skript. Mogu se kategorizovati kao jezgreni moduli, moduli trećih strana i lokalni moduli. Osnovni moduli su moduli koji dolaze sa Instalacijom Node-a i prethodno su učitani kada se pokrene Node proces.

3.5 Express Modul

Express je Node modul koji pruža minimalan i fleksibilan okvir za Node.js internet aplikacije. Obezbeđuje robusne i čiste funckije koje se dodaju Node modulima. Express se instalira koristeći NPM paket mendzer koji izdaje "npm install express" komandu. Express server se sastoji od tri bloka: *Rutera ,Rute I Middleware-a.* Ruta u Express-u je kombinacija HTTP glagola i putanje. HTTP glagol je obično jedan od četiri HTTP metode: GET, POST, PUT i DELETE, a putanja je lokacija resursa (URI).

3.6 Node Package Manager NPM

NPM je ugrađeni alat koji je podrazumevano uključen u svaku instalaciju Node-a. On pomaže u lakom upravljanju modulima u Node projektima preuzimanjem paketa, rešavanjem zavisnosti, pokretanjem testova, i instalacijom uslužnih programa komande linije.
NPM se pokreće iz prevodioca komandne linije (CLI) i upravlja svim zavisnostima Node aplikacije. NPM automatski obrađuje sve procese preuzimanja i čuva sve imenovane module u „node-moduli" folderu.

3.7 Princip rada Node.Js-a

Glavne karakteristike Node arhitekture su korišćenje neblokirajućeg događaja i asinhroni I/O pozivi koji rade u jednoj niti. Konvencionalni internet serveri rukuju konkurentno, praveći nove niti za svaki novi zahtev koje mogu zauzeti slobodnu memoriju do maksimuma. Čak i sa ograničenom memorijom, i jednom niti, Node može postići visok stepen konkurentnosti bez potrebe za vršenjem prebacivanja konteksta između niti.

3.8 Neblokirajuća petlja događaja

Node je neblokirajući u smislu da može da servisira višestruke zahteve. Konvencionalni model blokiranja teži da blokira naknadne zahteve koji se šalju na server kada se vrši obavljanje I/O operacija, kao što je npr. čitanje sadržaja iz baze podataka. Da ne bi došlo do blokiranja, Node koristi petlju događaja (event loop), softverski šablon (patern) – šema u kojoj se aktivira povratna funckija događaja u momentu kada se neka od akcija dogodi u programu.

3.9 Jednonitni model (Single Threaded)

Node je proces koji se pokreće u petlji događaja i koristi jednu nit za servisiranje zahteva. Kada Node aplikacija treba da izvrši operacije, ona šalje asinhroni zadatak na petlju događaja, registruje funkciju povratnog poziva, a zatim nastavlja sa obradom druge operacije. Petlja događaja prati asinhronu operaciju, izvršava zadati povratni poziv i kada se završi, vraća njegov rezultat aplikaciji.

3.10 Asinhrono programiranje

Dok neblokirajući deo Node-a omogućava da prihvati gotovo sve izvršene zahteve, asinhrono programiranje omoguća da se zahtevi izvršavaju učinkovito korišćenjem ograničenih ciklusa takta i memoriji koja je na raspolaganju svojoj jednonitnoj arhitekturi. Asinhronizacija je u korenu Node-a zato što su skoro svi API-ji izloženi preko Node modula asinhroni.

3.11 Rešavanje konkurentnosti

Jedan način je „mrešćenjem" child niti, koje dele isti adresni prostor kao glavna nit izvršenja. Svaka nit dodeljuje svoj stack, registre i programski brojač. Drugi način je preko petlje događaja. Kada se dogodi neki događaj, pridruženi handler je stavljen u red za izvršenje.

3.12 Programiranje vođeno događajem (Event Driven)

Programiranje zasnovano na događajima je paradigma koja se često koristi za interkativne aplikacije. Interakcija korisnika stvara događaje, a glavna petlja izvršava odgovarajuće rukovodioce događaja (Event handler-e).

Event handler-i u Node-u su obične funkcije koje se koriste kao povratni pozivi. Node.Js API metode često koriste funkcije povratnog poziva koje se na kraju izvršavaju, kada se završi ne-blokirajuća operacija (kao što je čitanje i pisanje). Petlja događaja u Node.Js prima signale završetka i izvršava povratni poziv.

4. POREDJENJE NODE.JS SA JAVASCRIPTOM

JavaScript je baziran na prototipu, objektno orijentisanom, labavo tipiziranom skriptovanju na strani klijenta. Radi isključivo unutar internet pretraživača. Koristi se za dodavanje interaktivnosti na internet stranice. Zbog toga je potrebna pomoć nekog drugog programskog jezika ukoliko mora da izvrši bilo kakvu interakciju sa serverom. I ako je Node zasnovan na JavaScriptu, i koristi konstrukciju JavaScripta za skoro sve svoje funkcionalnosti, on nudi potpuno drugačije okruženje od JavaScripta. Node se može smatrati nadskupom JavaScripta. Ima dodatne funkcionalnosti i funkcije pored svega što sadrži JavaScript.

4.1 Modularni sistem

Jedan od nedostataka u JavaScriptu je nedostatak modularnosti. Jedini način za povezivanje različitih skripti je korišćenjem drugih jezika, poput HTML-a. Umesto definisanja određenog broja globala, Node je uveo modularni sistem. Node obuhvata velik broj osnovnih modula kao što su *http, net, fs.*

4.2 Globalni objekat

Node implementira globale sa jasnim razdvajanjem. Koriste se sledeća dva globalna objekta: *Global* i *Process.*

4.3 Privremena memorija (Buffer)

Još jedan nedostatak u JavaScriptu je njegova podrška za rukovanje binarnim podacima. Node Buffer klasa je rešila ovaj nedostatak obezbeđujući API za jednostavnu manipulaciju podacima. Buffer je dodatak Node-u za četiri primitivna tipa podataka – *boolean, number, string.*

4.4 Poređenje Node-a i AJAX-a

Jedina sličnost između Node-a i AJAX-a je u tome što oba rade na JavaScriptu. Dok se Node uglavnom koristi za operacije na strani servera za razvoj kompletne serverske aplikacije, AJAX se koristi za operacije na strani klijenta za dinamički rad ažuriranja sadržaja stranice bez osvežavanja.

4.5 AJAX (Asinhroni JavaScript i XML)

AJAX je skup tehnika za kreiranje visoko interaktivnih internet stranica i internet aplikacija. Koristi se za upućivanje na sve metode komuniciranja sa serverom od klijenta koji koristi JavaScript. Koristeći AJAX, aplikacija može da pozove specifičnu procedure na serveru, i izvrši osvežavanje samo specifičnog dela internet stranice.

4.6 Razvoj aplikacija u realnom vremenu pomoću Socket.io modula

Web Sockets dozvoljava simultanu komunikaciju u oba smera, između klijenta i servera, ali u potpuno novom protokolu. Socket.io je biblioteka za internet aplikacije u realnom vremenu. Socket.io je vođen događajima i izlaže i komponente na strani klijenta i na strani servera. I klijentska i serverska strana u suštini čine istu stvar: dozvoljavaju slanje (ili emitovanje) događaja i pružaju način za definisanje rukovodioca događaja.

5. NODE.JS BEZBEDNOST

Porast potražnje za JavaScriptom u polju programiranja proširio se u opsegu od programiranja na strani klijeta do programiranja na strani servera. Kao rezultat toga, SSJS (Sever Side JavaScript) funckije su dostupne skoro svuda.

5.1 Skriptiranje na unakrsnim lokacijama (XSS)

XSS je napad koji omogućava napadaču da ubaci zlonamerni skript u internet aplikaciju. XSS ranjivosti su prouzrokovane neuspehom internet aplikacije da pravilno uradi validaciju unosa korisnika.

5.2 Odbijanje usluge (Denial of Servise - DoS)

DoS je napad koji informacije ili podatke čini nedostupnim za njihove hostove. To je jedan od najjednostavnih oblika mrežnog napada. Umesto pokušavanja krađe ili izmene podataka, cilj ovog napada je sprečavanje pristupa servisu ili resursu. To se obično postiže opterećivanjem servera velikom količinom zahteva, povezujući resurse servera i sprečavanjem ispunjava legitimnih zahteva.

6. IMPLEMENTACIJA WEB APLIKACIJE ZA RAZMENJIVANJE PORUKA POMOĆU EXPRESS-A I SOCKET.IO BIBLIOTEKE

U WebSocket-u, server može da šalje podatke klijentu, ali to takođe može i klijent serveru. Stoga, WebSocket je vrsta komunikacione cevi koja je otvorena u dva smera.
U aplikaciji ćemo pored Socket.io biblioteke koristiti i Express.js web okvir (framework)

6.1 Razvojno okruženje

Prvo što treba uraditi je pokrenuti npm, naš menadžer paketa. Da bismo to učinili, prvo treba da otvorimo Node.Js terminal, kreiramo novi repozitorijum koji će sadržati naš projekat, lociramo se unutar njega i zatim inicijalizujemo npm.

6.2 Arhitektura aplikacije

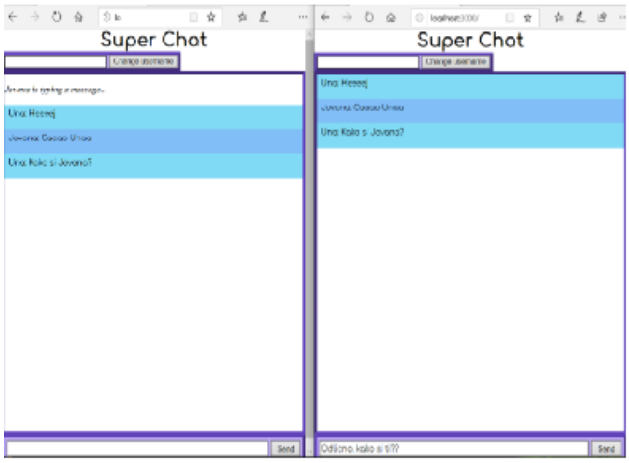
Razlikujemo dva dela u razvoju aplikacije – klijentski i serverski deo. Serverskom stranom će upravljati Node.js, koji će pokretati sve pakete i internet stranice. Klijentski deo će biti prikazan na računaru klijenta. On će imati direktan pristup datotekama (html/css i js).

6.3 Slanje i primanje podataka

Kada se korisnik poveže na aplikaciju, postavićemo mu podrazumevano korisničko ime,npr. „anonimus“. Da bismo to uradili, moramo otići na stranu servera (app.js) i dodati ključ Socketu. U stvari, socket predstavlja svakog klijenta konektovanog na naš server.

Što se tiče razmene poruka, princip je potpuno isti kao i za promenu korisničkog imena.

Za događaj new_message pozivamo io objekat socketa, pozivajući tako sve povezane sockete. Dakle, ova linija koda će zapravo poslati poruku svim socketima, što je nama i cilj jer želimo da poruka koju je korisnik poslao bude vidljiva svima, pa i njemu samom.
Da bismo aplikaciju učinili realnijom, dodali smo feedback poruku koja obaveštava korisnika kada drugi korisnik kuca poruku - porukom “<Korisnik> is typing a message..” To smo postigli dodavanjem JQuery event listener-a na događaj kucanja i šaljemo socket događaj nazvan „typing“. Sa druge strane, slušamo „kucanje“ i emitujemo poruku. Emitovanje znači slanje poruke svima drugima osim socket-u koji ga je pokrenuo.



Slika 1. Konačan izgled aplikacije

7. ZAKLJUČAK

U ovom radu pokazano je da je Node transformisao upotrebljivost JavaScripta, čineći Node kompletnim programskim jezikom. Od internet pretraživača, do serverskih skripti izvan pretraživača, on je omogućio dostupnost runtime okruženja, biblioteke koja sadrži mnoštvo besplatnih korisnih modula koji se mogu uvesti pomoću ugrađenog alata NPM.

Pokazalo se da je postavljanje Node okruženja jednostavno, a dostupan je na svim glavnim operativnim sistemima. Zasniva se na poznatoj sintaksi JavaScripta, ali razlike postoje. Node može biti pomešan sa AJAX-om, ali su dva potpuno različita alata, čija je jedina sličnost to što koriste JavaScript kao bazu.

Pored svih njegovih prednosti, Node ima i neke sigurnosne rupe.

Dakle, ako slučaj upotrebe ne sadrži intenzivne operacije koje opterećuju procesor, niti pristupa bilo kojoj blokadi resursa, mogu se iskoristiti prednosti Node-a i iskusiti brz i skalabilan razvoj aplikacije.

8. LITERATURA

[1] Node.js v5.1.0 Documentation
https://nodejs.org/en/blog/release/v5.1.0/

[2] What is Node?
https://www.amazon.com/dp/B005ISQ7JC/ref=pe_385040_117923520_TE_M1DP

Kratka biografija:

Una Banjanin, rođena je 01.03.1994. u Novom Sadu. Nakon završene Osnovne škole upisuje srednju školu, Gimnaziju „Svetozar Marković” u Novom Sadu. 2013. godine upisuje „Fakultet tehničkih nauka” u Novom Sadu, smer „Računarstvo i automatika”. 2018. godine dobija zvanje Diplomirani inženjer elektrotehnike i računarstva. 2018/19. godine upisuje master akademske studije, smer „Primenjene računarske nauke i informatika – inženjering informacionih sistema”. kontakt: ubanjanin@gmail.com