**SVEUČILIŠTE U ZAGREBU, FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

36. Chat

**NODE.JS**

Josipa Kelava

**Zagreb, prosinac 2016.**

# Općenito o Node.js-u

Node.js je okruženje koje nam omogućuje pokretanje JavaScript koda izvan web preglednika. JavaScript se većinom koristi za programiranje interaktivnih web sučelja, a Node.js omogućuje iskorištavanje tog programskog jezika u širem kontekstu - najčešće za programiranje web poslužitelja. Korištenje Node.js-a omogućuje da se cijela web aplikacija izradi korištenjem samo jednog programskog jezika – JavaScripta. JavaScript je jedna od najraširenijih tehnologija s obzirom da ga svaki moderni web preglednik podržava, a pregledavanje weba je danas postala norma prilikom korištenja računala. Uz to programiranje klijentske strane i poslužiteljske strane istom tehnologijom, programski kod može biti dijeljen, odnosno povećava se količina koda koja se može ponovno iskoristiti. Koristeći neki drugi programski jezik za programiranje poslužitelja poput PHP-a, Rubya ili Pythona tako nešto ne bi bilo moguće. Koliko je moćno okruženje Node.js pokazuje i skup njegovih korisnika od kojih možemo spomenuti neke svjetske kompanije poput: Microsoft, Netflix, PayPal.

Kao interpreter Node.js-a, koristi se *Google’s V8 JavaScript engine* zbog čega su aplikacije napisane u Node.js-u i nekoliko puta brže od aplikacija napisanih u ostalim skriptnim jezicima. V8 koristi novije tehnologije prevođenja koje često omogućuju višim programskim jezicima poput JavaScripta da se izvode sličnom brzinom kao i niži programski jezici poput C-a. Viši programski jezici često omogućuju brže razvijanje proizvoda, ali je i njihovo izvođenje često sporije. Brzina izvođenja je ključan aspekt Nodea. Node dodatno uključuje apstrakcije koje omogućuju bolje korištenje JavaScripta u kontekstu izvan web preglednika, na primjer razredi za manipuliranje binarnih podataka što je često potrebno u poslužiteljskom dijelu aplikacije.

Programska paradigma programiranja zasnovanog na događajima svojstvena je za programski jezik JavaScript, a Node iskorištava tu programsku paradigmu za ostvarivanje visoko skalabilnih poslužitelja. Arhitektura koju koristi Node zove se *event loop.* Ta arhitektura omogućuje brzo i sigurno razvijanje visoko skalabilnih poslužitelja. To je vrlo bitna značajka ove tehnologije jer je konkurentno programiranje zahtjevno i sklono pogreškama. Node zaobilazi te izazove zadržavajući dobre performanse. Da bi arhitektura programa razvijenog za okruženje Node mogla biti zasnovana na događajima, Node nudi biblioteke koje apstrahiraju operacije poput pristupanja datotekama na disku u operacije koje nazivamo događajima. To znači da program neće biti blokiran odnosno neće čekati da se ta operacija izvrši, već će biti obaviješten kada operacija završi, a u međuvremenu će se moći izvršavati neka druga radnja. Takav način izvršavanja sličan je onom u web pregledniku tijekom događaja poput klika mišem. Programiranje takvih arhitektura poznato je programerima koji nisu nužno koristili Node, ali su koristili JavaScript. Također, takve arhitekture su intuitivne i sigurne od pogrešaka koje se javljaju kod konkuretnog programiranja pa je vrijeme uloženo za učenje i razvijanje takve arhitekture znatno smanjeno.

Node radi na principu asinkronog programiranja. Glavna dretva postavlja poslove u dijeljeni red iz kojeg druge dretve mogu preuzeti i obaviti te poslove. Po završetku posla, pozvat će se odgovarajuća funkcija u glavnoj dretvi. To znači da procesor neće biti blokiran na operacijama poput čitanja i pisanja jer će te operacije biti obavljene u posebnim dretvama. Glavna dretva bit će obavještena po završetku, a do tad će moći trošiti procesorsku snagu na nešto drugo. Problem s operacijama čitanja i pisanja je taj što moraju koristiti fizičke uređaje poput tvrdog diska koji imaju pomične dijelove i često puno sporije obavljaju operacije od procesora. Tijekom obavljanja jedne operacije tvrdog diska, procesor može obaviti ciklus koji uključuje u milijunima operacija. Zato je bitno ne blokirati procesorsku snagu dok se obavljaju takve operacije.

Prednost asinkronog programiranja je što će procesorska snaga biti bolje iskorištena, ali postoji i nedostatak jer će u slučaju CPU zahtjevnih operacija drugi poslovi biti blokirani dok čekaju u redu da ih glavna dretva izvrši. Pokazalo se da aplikacije programirane ovim pristupom mogu podržavati znatno veći broj istodobnih konekcija od onih koji otvaraju dretvu za svaku konekciju ako takva aplikacija ne obavlja operacije koje su procesorski jako zahtjevne. Asinkrono programiranje također olakšava proces razvoja jer je jednostavnije u odnosu na konkuretno programiranje.

# Instalacija Node.js-a

Node je dostupan na gotovo svim operacijskim sustavima poput Windowsa, Linuxa i OSX-a. Node je tehnologija otvorenog koda i moguće ga je preuzeti direktno sa GitHub repozitorija. Taj kod je najnovija verzija Nodea pa vjerojatno sadrži i neke mogućnosti koje nisu dostupne u stabilnoj verziji, ali vjerojatno nisu još dovoljno testirane i sigurne. Sa službene stranice moguće je preuzeti stabilnu verziju. Za razvojno okruženje moguće je koristiti širok spektar alata s obzirom da danas postoji mnogo ekstenzija koje omogućuju razvoj Node aplikacija u alatima inicijalno namijenjenih za druge tehnologije.

Instalacijom Noda instalirava se i sustav za upravljanje i paktima koji se zove npm. Sam Node je veoma koristan, ali često postoje funkcionalnosti koje nisu dio Nodea, a želimo ih koristiti iznova u razvoju različitih aplikacija. Te funkcionalnosti je dobro razviti zasebno i nastaviti ih nadograđivati. Takav kod se distribuira u paketima preko sustava npm. To je način za korištenje programskih kodova drugih programera, a isto tako i način za dijeljenje svojih kodova. Međusobno dijeljenje i razvijanje koda podiže općenitu kvalitetu proizvoda. Npm koristi package.json dokument i u njega upisuje module koje želimo dodati. U polje dependencies upisuju se ime modula i željena verzija tog istog modula.

# Moduli Node.js-a

## 3.1 Express

Jedan od najpopularnijih modula razvijenih za Node.js je Express. Instalacija Expressa uz korištenja npm-a je vrlo jednostavna. Sve što je potrebno napraviti je izvesti naredbu ‘npm install express’. Express služi kao MVC okruženje za Node. Karakteristike Expressa su jednostavnost, minimalizam, fleksibilnost i skalabilnost.  Osnovno korištenje Expressa sastoji se od definiranja putanja (*eng. routes*) i funkcija za obradu zahtjeva poslanih na te putanje. Funkcije za obradu zahtjeva na nekoj putanji definiramo za HTTP metode. To znači da za neku putanju možemo definirati više funkcija za obradu zahtjeva, naprimjer jednu za obradu HTTP GET zahtjeva a drugu za obradu HTTP POST zahtjeva. Te funkcije su zapravo promatrači HTTP zahtjeva. Koristeći Express ne moramo pisati vlastite mehanizme za usmjeravanje zahtjeva iznova za svaku aplikaciju, već iskorištavamo mehanizme koji su već razvijeni i koji se redovito poboljšavaju i nadograđuju. Primjer jedne jednostavne aplikacije uz korištenje Expressa dan je u nastavku:

var express = require('express');  
var app = express();  
app.get('/api', function (req, res) {  
  res.send('Hello World');  
})

app.listen(10001);

U primjeru možemo vidjeti da aplikacija ima jednu putanju /api. Exress metodom get() osluškuje posjećivanje te putanje. Funkcija koja obrađuje zahjtev upućen prema toj putanji ima parametre req i res ondnosno zahtjev (request) i odgovor (response). Funkcija odgovara sa send() metodom koja sama zaključuje koje podatke treba poslati, naprimjer što treba pisati u zaglavlju http odgovora. Metoda listen(port) nam govori na kojim vratima će polužitelj slušati zahtjeve. Željeni modul nije dovoljno samo instalirati pomoću npm-a. Potrebno ga je i uključiti u programski kod. Upravo to nam omogućava funkcija require(‘express’).

## 3. 2 MongoDB

Još jedan koristan modul je ‘monogodb’. Za njegovu instalaciju potrebno je napraviti sve kao i za Express, odnosno npm install mongodb. Taj modul nam omogućuje korištenje Node.js-a sa nerelacijskom bazom podataka MongoDB. MongoDB koristi skupine kolekcija. Kolekcija sadrži dokumente koji imaju dinamicku strukturu (BSON - Binary JSON).  Osnovno korištenje MongaDB u aplikaciji prikazno je u sljedećim linijama:

var mongodb = require('mongodb');  
var MongoClient = mongodb.MongoClient;  
MongoClient.connect(url, function (err, db) {  
 if (err) {  
   console.log('error’);  
 } else {  
   console.log('Connection to', url);

//operacije prema bazi   
   var users = db.collection('users');  
        db.close();  
  }  
});

Uključivanje *MongoDB native driver*a radi se pomoću funkcije require kao i kod Expressa. Potrebno je napraviti instancu MongoClienta koja je zapravo sučelje preko kojeg ćemo se spajati na bazu podataka.  MonogoClient se spaja na bazu podataka pomoću urla koji mu se zada kao parametar i sadrži *callback* funkciju u kojoj se odvija komuniciranje s bazom (čitanje ili pisanje). Url predstavlja lokaciju na kojoj se nalazi  baza podataka. *Callback* funkcija sadrži varijablu db u kojoj su svi podaci sa željene baze. Ta funkcija u gore navedebom primjeru sarži jedan primjer dohvaćanja podataka s baze.

## 3.3 Dotenv

## Dotenv je modul Node.js-a koji nam omogućuje učitavanje varijabli okruženja iz .env datoteka u process.env. Dotenv modul instaliramo i uključujemo pomoću npm install dotenv te require(‘dotenv’). Potrebno je još samo učitati sve varijable okrženja. to se radi pomoću naredbe require(‘dotenv’).load(). Sada su sve varijable okruženja učitane u process.env i možemo im prisupati preko njega. Ako je naprimjer env datoteka imala liniju DB\_HOST=localhost onda joj u programskom kodu možemo pristupiti kao process.env.DB\_HOST.

## 3.4 ES7, Babel i ES5

ECMAScript(ES) je specifikacija je skriptnih jezika bazirana na JavaScirptu.  Node.js podržava verziju specifikacije ES5. Najnovija verzija specifikacije za JavaScript je ES7. Kako bi Node.js mogao raditi sa verzijom ES7 potrebno je koristiti modul babel koji omogućuje prevođenje verzije ES7 u verziju ES5. Instalacija babela izvodi se pomoću naredbe npm install --save-dev babel-core. Sljedećih par linija programskog koda prikazuje primjenu babela u izradi aplikacija.

var babel = require("babel-core");  
import { transform } from 'babel-core';  
import \* as babel from 'babel-core';

Babel ima različite varijante metode transforme koja omogućuje zadanog programskog koda ili datoteke u odgovarajuću verziju prikladno za Node.js.

## 3.5 Morgan

Jedan koristan modul je i Morgan. On na omogućuje vođenje dnevnika HTTP zahtjeva. Pojednostavljuje cijeli proces zapisivanja zahtjeva. U sljedećem primjeru možemo vidjeti metodu koja se izvršava nakon svakog zahtjeva:

app.use(function(request, response, next) {  
 console.log(request.method + " to " + request.url);  
 next();  
});

Morgan radi uz Express te ovakvu funkciju zamjenjuje sljedećim programskim linijama:

var morgan = require("morgan");

app.use(morgan());

Unutar funkcije morgan() mogu se dodati razni parametri poput ‘dev’ i ‘common’ koji samo defniraju način na koji će se zahtijevi ispisivati.

# Zaključak

Node. js je okruženje koje nam omogućujue da se cijela web aplikacija izradi korištenjem samo JavaScripta. On radi uz *Google’s V8 JavaScript engine* zbog čega su aplikacije napisane u Node.js dosta brze. Još jedan od glavnih razloga brzine koju postižu aplikacije pisane ovim jezikom je činjenica da Node.js radi asinkrono te se tako izbjegava stalno čekanje sustava na neke rezultate. Node. js ima još puno modula koji ubrzavaju rad, olakšavaju izradu i pomažu pri izradi aplikacija. Jedan modul koji je najkorišteniji i gotovo uvijek dolazi u kompletu s izradom aplikacija u Node.js-u je Express.