
Node.js



Vuk Isić PR 39/2016
Danilo Novaković PR 136/2016

Node.js

Asinhrono, event-driven, cross-platform
JavaScript(JS) okruženje, koje omogućuje pisanje
koda za back-end u JS-u.

JS na back-end-u?

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.



Šta node.js **JESTE**

→ **Asinhron**

Operacije ne čekaju jedna drugu

→ **Event-Driven**

Događaji i akcije korisnika

→ **Open Source, Cross-Platform**

Radi na svim platformama, besplatno

→ **Single Threaded**

Sve se izvršava u jednoj niti



Šta node.js **NIJE**

→ **Okruženje**

Nije okruženje kao npr. .NET

→ **Node Wrapper-i za JS**

Pisani u C-u

→ **Multi-threaded**

Izvršava se u jednoj niti, call-back koncept

→ **Nije za početnike**

Low-level apstrakcija

Kako radi node.js?



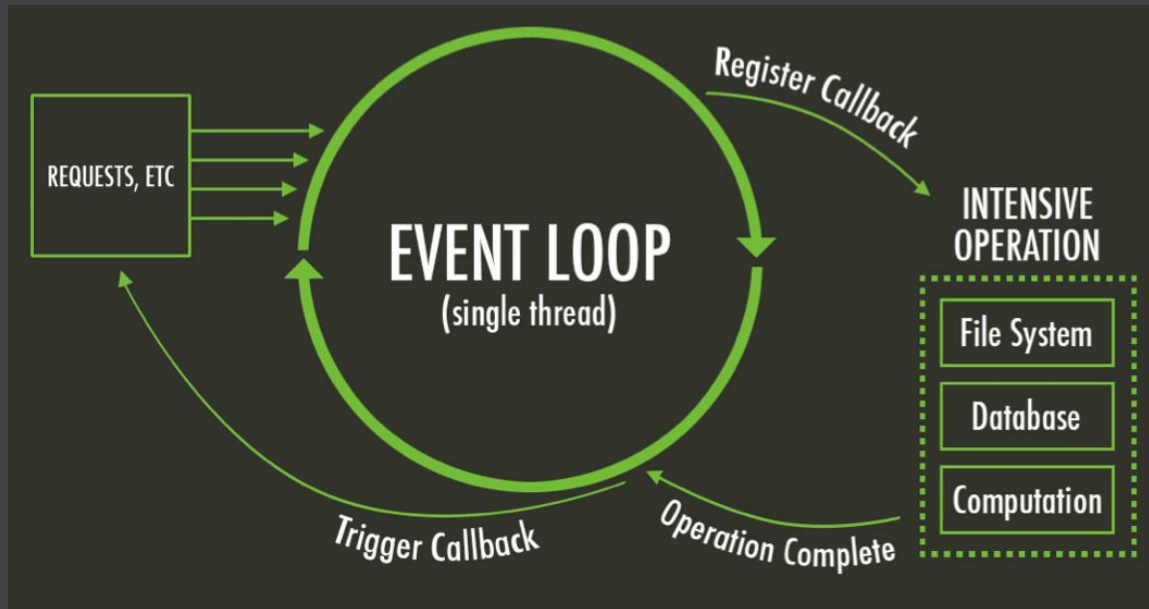
Pro Tip

*“Code JavaScript
underwater so
nobody could see
you crying”*

Kako radi node.js?

- **Komunikacija bazirana na nitima je relativno neefikasna i teška za korišćenje.**
Node.js => no locks = no deadlocks
- **Procesi su neblokirajući.**
Nema blokirajućih operacija, pa se skalabilni servisi mogu lako proizvesti.
- **Node.js ide korak dalje u pogledu event-based model-a.**
Event loop postoji u okruženju, ne koristi se neka posebna biblioteka.
- **HTTP je osnovna klasa Node.js-a**
Dizajnirana za striming i mala kašnjenja
- **Node.js dizajniran bez više niti ne znači da nemožemo iskoristiti prednosti multiple cores okruženja.**
`child_process.fork()`

Kako radi node.js?



Osnovne funkcionalnosti node.js-a?

- Rukovanje POST, GET, PUT, UPDATE, DELETE zahtevima
- Generisanje dinamičkog sadržaja.
- Open, Close i CRUD operacije sa fajlovima na serveru.
- Rad sa formama.
- CRUD operacije nad bazom podataka
- NPM

Kako radi node.js?

→ Primer: Rukovanje zahtevom za čitanje fajla

ASP . NET

- Prosledi zadatak za citanje fajla , fajl-sistemu.
- Sačeka da fajl sistem otvori i pročita fajl.
- Sadržaj fajla se šalje klijentu.
- Obrađuje se sledeći zahtev

Node . js

- Prosledi zadatak za citanje fajla , fajl-sistemu.
- Obrđuje sledeći zahtev
- Kada fajl-sistem otvori i pročita fajl, server šalje sadržaj fajla korisiku.

Rad sa modulima

```
//_greet.js  
  
module.exports = function(name) {  
  console.log(`Hello ${name}`);  
};  
  
/*-----*/  
  
//modules_single_export.js  
const greet = require("./_greet");  
  
greet("John");
```

```
//_greet_and_wave.js  
const greet = function(name) {  
  console.log(`Hello ${name}`);  
};  
  
const wave = function(name) {  
  console.log(`*Waving towards ${name}*`);  
};  
  
module.exports = { greet, wave };  
  
/*-----*/  
  
// modules_multiple_exports.js  
const gaw = require("./_greet_and_wave");  
  
gaw.wave("John");  
gaw.greet("John");
```

Rad sa fajlovima

```
// FILES
Learn more or give us feedback
const fs = require("fs");

const readFilePath = "text_file.txt";
const writeFilePath = "_text_file_copy.txt";
const encoding = "utf8";

// Asynchronous
fs.readFile(readFilePath, encoding, (err, data) => {
  console.log("<Async>", data);
  fs.writeFile(writeFilePath, data, () => {
    console.log("<Async> Finished writing to file");
  });
});

// Synchronous
const textFileContent = fs.readFileSync(readFilePath, encoding);
console.log("<Sync>", textFileContent);

fs.writeFileSync(writeFilePath, textFileContent);
console.log("<Sync> Finished writing to file");

// FOLDERS
const fs = require("fs");

// fs.unlinkSync("_text_file_copy.txt"); // Removes file

fs.mkdirSync("_new_folder");
fs.rmdirSync("_new_folder");
```

— Live Demo Simple Server

```
const http = require("http");

const server = http.createServer((req, res) => {
  console.log("request was made: " + req.url);

  statusCode = 200;
  headers = { "Content-Type": "text/plain" };
  content = "Hello from server";

  res.writeHead(statusCode, headers);
  res.end(content);
});

port = 3000;
server.listen(port, "127.0.0.1");
console.log(`Listening to port ${port}...`);
```

Serving HTML

```
const http = require("http");
const fs = require("fs");

const server = http.createServer((req, res) => {

  statusCode = 200;
  headers = { "Content-Type": "text/html" };
  res.writeHead(statusCode, headers);

  const readStream = fs.createReadStream(__dirname + "/index.html", "utf8");
  readStream.pipe(res);

});

port = 3000;
server.listen(port, "127.0.0.1");
console.log(`Listening to port ${port}...`);
```

Live Demo Routing

```
const http = require("http");
const fs = require("fs");

const server = http.createServer((req, res) => {
  console.log("request: " + req.url);

  if (req.url === "/index.html" || req.url === "/") {
    res.writeHead(200, { "Content-Type": "text/html" });
    fs.createReadStream(__dirname + "/index.html", "utf8").pipe(res);
  } else if (req.url === "/api/students") {
    students = [
      { name: "Danilo", age: 23 },
      { name: "Vuk", age: 23 }
    ];

    res.writeHead(200, { "Content-Type": "application/json" });
    res.end(JSON.stringify(students));
  } else {
    res.writeHead(404, { "Content-Type": "text/html" });
    fs.createReadStream(__dirname + "/404.html", "utf8").pipe(res);
  }
});

port = 3000;
server.listen(port, "127.0.0.1");
console.log(`Listening to port ${port}...`);
```

Express.js

```
const express = require('express')  
const app = express()
```

- Fleksibilno i minimalno okruženje za Node.js
- Namenjen za rad sa Web aplikacijama
- Osnova je HTTP protokol
- Brzo i “lako” kreiranje API-a

The logo for Express.js, featuring the word "Express" in a thin, black, sans-serif font. The letter 'E' is stylized with a horizontal bar that extends to the left and then curves back to the right, creating a unique graphic element.

Express.js Server

```
const express = require("express");
const bodyParser = require("body-parser");

const app = express();

// setting up template engine (like Razor in ASP.NET)
app.set("view engine", "ejs");

// GET requests

app.get("/", (req, res) => {
  res.send(`this is a homepage. query: ${JSON.stringify(req.query)}`); // text as response
});

app.get("/contact", (req, res) => {
  res.sendFile(__dirname + "/contact.html"); // serving static html
});

app.get("/student/:id", (req, res) => {
  let data = {
    age: 23,
    name: "Djura Djuric",
    id: req.params.id,
    subjects: ["Web2", "BP2", "Grafika"]
  };
  res.render("student", { student: data });
});

// POST requests

// create application/x-www-form-urlencoded parser
const urlencodedParser = bodyParser.urlencoded({ extended: false });

app.post("/contact", urlencodedParser, (req, res) => {
  res.send(`You have sent: ${JSON.stringify(req.body)}`);
});

// Listen

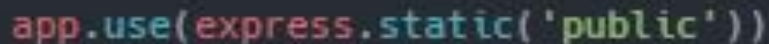
const port = 3000;
app.listen(port, () => {
  console.log(`Listening to port ${port}...`);
});
```


Serving Static Files



A file explorer window showing the project structure. The 'public/' directory is expanded, showing 'style.css'. Other files in the root include 'views/', '.env', '.gitconfig', '.gitignore', '.hyperdev-assets', 'README.md', 'package.json', and 'server.js'.

- public/
 - style.css
- views/
- .env
- .gitconfig
- .gitignore
- .hyperdev-assets
- README.md
- package.json
- server.js



A code editor snippet showing the Express.js static middleware configuration. The code is color-coded: 'app' is blue, 'use' is green, 'express' is red, 'static' is blue, and 'public' is green.

```
app.use(express.static('public'))
```

/*

http://localhost:3000/images/kitten.jpg
http://localhost:3000/css/style.css
http://localhost:3000/js/app.js
http://localhost:3000/images/bg.png
http://localhost:3000/hello.html

*/

— Live Demo

Web2_Node_Js_Opt/Demos_Node/

FAQ

Aplikacije za koje je node.js najbolje rešenje?

Da li node.js može da zameni tradicionalna back-end rešenja ?

Skalabilnost?



Aplikacije za koje je node.js najbolje rešenje

- WebSocket Server
- Fast file upload client
- Data streaming
- Ad server
- Stock exchange softver.



Node.js vs ASP.NET

Glavne razlike ::

- Jezik
JavaScript vs. C#
- Razvoj aplikacije
Nivo apstrakcije
- Okruženje



Skalabilnost

Glavne razlike ::

- Event-loop
Single-thread
- Multithreading
- "Male" vs. "Velike" aplikacije

Node.js vs. ASP.NET - timestamp demo

```
var express = require('express');
var app = express();

// enable CORS
var cors = require('cors');
app.use(cors({ optionSuccessStatus: 200 }));

app.use(express.static('public'));

app.get('/', function(req, res) {
  res.sendFile(__dirname + '/views/index.html');
});

app.get('/api/timestamp/:date_string?', function(req, res) {
  const param = isNaN(Number(req.params.date_string))
    ? req.params.date_string
    : Number(req.params.date_string);

  const date = typeof param !== 'undefined' ? new Date(param) : new Date();

  if (date !== 'Invalid Date') {
    res.json({ unix: date.getTime(), utc: date.toUTCString() });
  } else {
    res.json({ unix: null, utc: 'Invalid Date' });
  }
});

// listen for requests :)
var listener = app.listen(process.env.PORT, function() {
  console.log('Your app is listening on port ' + listener.address().port);
});
```

```
// Additional settings in Startup.cs
// TimestampController.cs
[HttpGet]
[Route("convert/{param}")]
public IActionResult Convert(string param)
{
  long unix = -1;
  string utc = "Invalid Date!";
  DateTime temp = DateTime.Now;
  if (String.IsNullOrEmpty(param))
  {
    // Use current DateTime
    utc = DateTimeToString(temp);
    unix = DateTimeToMilliseconds(temp);
    return Ok(new { unix, utc });
  }
  else if (TryConvertToDateTime(param, out temp))
  {
    // Convert given DateTime
    utc = DateTimeToString(temp);
    unix = DateTimeToMilliseconds(temp);
    return Ok(new { unix, utc });
  }
  else
  {
    // Invalid Request parameter
    return BadRequest(new { unix, utc });
  }
}
```



Prednosti

- **Asinhron, konkurentno okruženje**
Event-driven
- **JavaScript je "jednostavan" za učenje**
- **NPM**
- **JavaScript na serveru i klijentu**
- **Upotreba velikih fajlova**
- **Zajednica**



Mane

- **Skalabilnost?**

Količina i kompleksnost koda
- **Rad za bazama podataka zna da bude problematičan**
- **Ugnježdene call-back funkcije**
- **Nije za početnike**

Node.js is easy to learn and difficult to master.
- **Nije namenjen za CPU-intenzivne zadatke.**

—

Zaključak

Node.js se pojavio kao ideja da je preko JS-a napravi single-threaded event driven server (back-end kod).

Popularnosti JS-a raste.

Enterprise aplikacije najčešće koriste .NET, Java, PHP za back-end

Da li će u budućnosti back-end kod biti pisan u node.js-u umesto u PHP, .NET, Java?

Pitanja ?





Hvala na pažnji !

Repo: https://github.com/vukisic/Web2_Node_Js_Opt/