

Curs Tehnologii Web

Node.js

Node.js este o platformă de programare JavaScript la nivelul serverului, care vă permite să creați aplicații web și dar și aplicații la nivelul serverului cu JavaScript. Este o soluție populară deoarece vă permite să utilizați același limbaj de programare (JavaScript) atât **pe partea clientului**, cât și pe **partea serverului**, fapt ce conduce la o dezvoltare mai rapidă și eficientă a aplicațiilor web.

Node.js se poate utiliza și în dezvoltarea de aplicații de rețea, cum ar fi aplicații de chat, aplicații de jocuri online, servere web etc.

Tipuri de aplicații care utilizează Node.js:

1. **Aplicații web** - Node.js este adesea folosit pentru crearea de aplicații web cum ar fi aplicații de socializare, platforme de jocuri și aplicații pentru streaming video.
2. **Aplicații de rețea** - Node.js este excelent pentru construirea de aplicații de rețea, cum ar fi aplicații de chat sau de mesagerie, servere de jocuri și aplicații de transfer de fișiere.
3. **Aplicații de timp real** - datorită naturii sale non-blocante și a sistemului de evenimente, Node.js este foarte potrivit pentru crearea de aplicații de timp real, cum ar fi aplicații de monitorizare, aplicații de streaming.
4. **Aplicații de automatizare** - cum ar fi procesarea fișierelor sau interacțiunea cu alte sisteme de automatizare.
5. **Aplicații mobile** - permite construirea de aplicații mobile hibride, adică aplicații care rulează pe **platformele iOS și Android**.

<https://nodejs.dev/en/learn/>

Node.js este construit pe baza motorului JavaScript Google Chrome V8 și este folosit în principal pentru a crea servere web - dar nu se limitează doar la asta. Node.js este un mediu de rulare JavaScript open-source și multiplatformă. Este un instrument popular pentru aproape orice tip de proiect!

O aplicație Node.js rulează într-un singur proces, fără a crea un fir nou pentru fiecare solicitare. Node.js oferă un set de primitive I/O asincrone în biblioteca sa standard care împiedică blocarea codului JavaScript și, în general, bibliotecile din Node.js sunt scrise folosind paradigme neblocante.

Node.js poate fi descărcat și instalat gratuit de pe site-ul oficial: <https://nodejs.org/en/download/>

Pentru a instala Node.js trebuie:

1. Sa descarcam versiunea corespunzatoare S.O si arhitecturii PC.
2. Executam fișierul de instalare și urmam instrucțiunile afișate pe ecran pentru a finaliza instalarea.
3. Verificam dacă Node.js a fost instalat corect, deschizând o fereastră de terminal (Command Prompt → Windows, Terminal în macOS sau Linux) și tastând comanda "**node -v**". Dacă veți vedea versiunea Node.js afișată în terminal, atunci instalarea a fost reușită.

```
C:\Users\5info>node -v
v16.13.2
```

Fisierele se ruleaza

```
C:\node_js>node app.js
Salut, Node.js!
```

node nume-fisier.js

Pentru a dezvolta/crea un nou proiect Node.js se relueaza comanda **npm init** si astfel se creaza fișierul "**package.json**" în directorul curent.

```
C:\node_js>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (node_js)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\node_js\package.json:

{
  "name": "node_js",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes) yes
npm notice
npm notice New major version of npm available! 8.1.2 -> 9.6.4
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.4
npm notice Run npm install -g npm@9.6.4 to update!
npm notice
```

Express.js (<https://expressjs.com>). Acesta este unul dintre cele mai populare web framework-uri pentru Node

npm install express --save

--save are scopul de a salva **Express** ca dependență a proiectului. După ce **Express** se instalează, el va apărea în **package.json**

<https://expressjs.com/en/starter/hello-world.html>

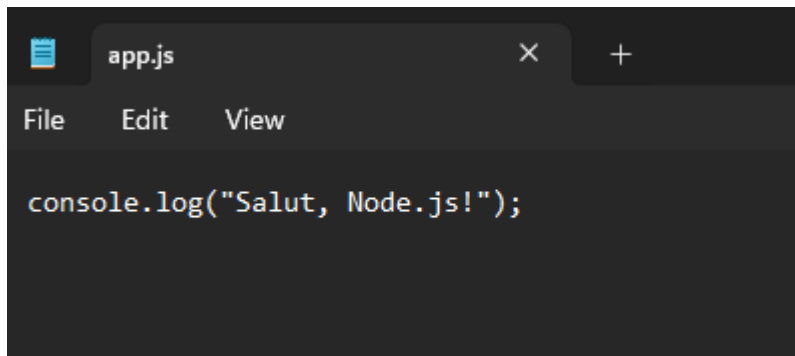
```
package.json
File Edit View

{
  "dependencies": {
    "express": "^4.18.2"
  },
  "name": "proiect_nd",
  "version": "1.0.0",
  "main": "index.js",
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": ""
}
```

```
C:\node_js>node
Welcome to Node.js v16.13.2.
Type ".help" for more information.
> 1+1+1
3
> new Date()
2023-04-11T06:24:12.383Z
> a=9;b=10;
10
> numere={a:10,b:20,c:30}
{ a: 10, b: 20, c: 30 }
> numere.a
10
> numre.b=30
Uncaught ReferenceError: numre is not defined
> numere.b=30
30
> .exit
C:\node_js>
```

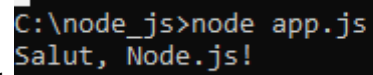
Aplicatia 1

Se creeaza fisierul js.



```
app.js
File Edit View
console.log("Salut, Node.js!");
```

Se ruleaza fisierul



```
C:\node_js>node app.js
Salut, Node.js!
```

resurse online de studiu:

- <https://nodejs.org/en/download/package-manager>
- <https://nodejs.org/en/docs>
- <https://www.codecademy.com/learn/javascript>
- <https://replit.com/languages/nodejs>

Aplicatia 2

`npm install express`

<https://expressjs.com/en/starter/hello-world.html>

```
const express = require("express");
const app = express();

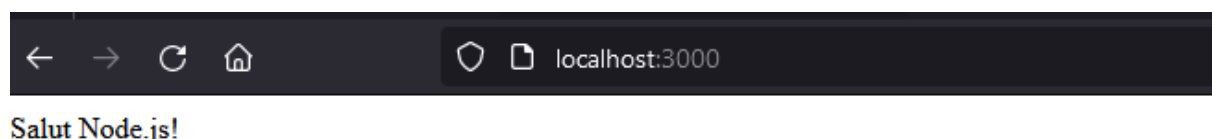
app.get("/", (req, res) => {
  res.send("Salut Node.js!");
});

app.listen(3000, () => {
  console.log("Serverul a pornit pe portul 3000.");
  console.log("Serverul a pornit pe portul 3000.");
});
```

his app starts a server and listens on port 3000 for connections. The app responds with “Salut Node.JS!” for requests to the root URL (/) or *route*. For every other path, it will respond with a **404 Not Found**.

The example above is actually a working server: Go ahead and click on the URL shown. You’ll get a response, with real-time logs on the page, and any changes you make will be reflected in real time. This is powered by [RunKit](#), which provides an interactive JavaScript playground connected to a complete Node environment that runs in your web browser. Below are instructions for running the same app on your local machine.

```
C:\node_js>node ap_2.js
Serverul a pornit pe portul 3000.
Serverul a pornit pe portul 3000.
```



← → ↻ 🏠 localhost:3000

Salut Node.js!

Aplicatia 3 → folosind Node.js se crează o pagină web:

1. se creeaza un fișier package.json pentru a instala dependențele necesare și a gestiona proiectul daca rulam comanda `npm init` în linia de comandă ...

2. se instaleaza dependentele necesare pentru proiect, vom avea nevoie de Express.js care se instaleaza cu comandă în linia de comandă: **npm install express**
3. se creează un fișier **ap_4.js** pentru a configura și porni serverul. În acest fișier, vom importa Express.js si vom crea o instanță a aplicației. Configuram aplicația să folosească fișierele statice din folder public și adaugam o rută pentru afișarea unei pagini simple.

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

// Configurare folder public
app.use(express.static('public'));

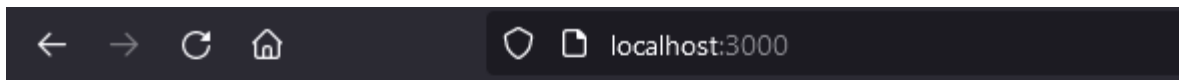
// Ruta de start
app.get('/', (req, res) => {
  res.sendFile(__dirname + '/public/index.html');
});

// Pornire server
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Serverul rulează pe portul ${PORT}`);
});
```

4. Creează un fișier **index.html** într-un folderul **public**

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Pagina mea</title>
  </head>
  <body>
    <h1>Bine ați venit pe pagina mea!</h1>
    <p>Salut NODE.JS</p>
  </body>
</html>
```

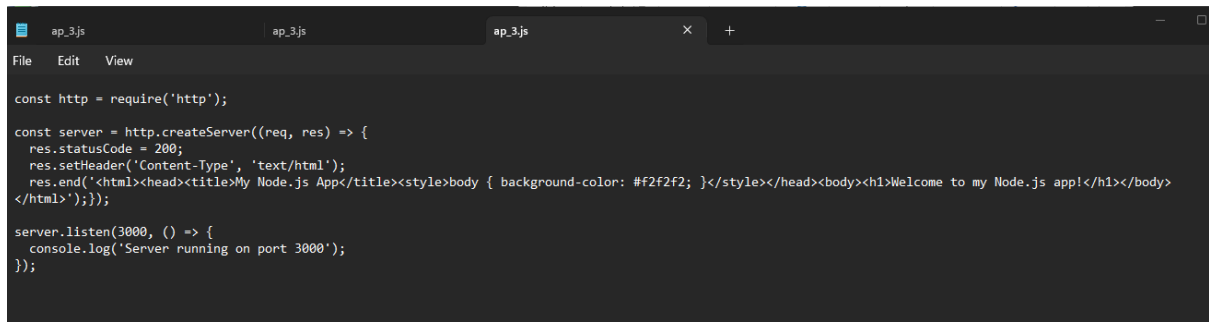
4. Pornim serverul folosind comanda `node ap_4.js` si apoi deschidem un browser → <http://localhost:3000>.



Bine ați venit pe pagina mea!

Salut NODE.JS

Aplicatia 4

A screenshot of a code editor window with three tabs, each labeled 'ap_3.js'. The editor shows a JavaScript file that sets up a simple HTTP server. The code includes the 'http' module, creates a server with a response handler that returns an HTML page with a pink background, and listens on port 3000. The HTML page has the title 'My Node.js App' and the body 'Welcome to my Node.js app!'.

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

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/html');
  res.end('<html><head><title>My Node.js App</title><style>body { background-color: #f2f2f2; }</style></head><body><h1>Welcome to my Node.js app!</h1></body></html>');
});

server.listen(3000, () => {
  console.log('Server running on port 3000');
});
```

<http://localhost:3000>

Obiectele request si response

Cand un client cere o anumita resursa de la server (de exemplu, o pagina web, o imagine, un xml etc) spunem ca face un *request*. Obiectul de tip *request* cuprinde toate datele cererii primite de la client. Din el putem prelua, de exemplu, datele introduse de utilizator intr-un formular. Request-ul este primit de catre web server, este procesat, si serverul raspunde cu ajutorul unui obiect de tip *response*. Obiectul de tip *response* este practic reprezentarea mesajului de tip HTTP oferit de serverul nostru. Putem seta anumite campuri ale acestuia, cum ar fi headerele raspunsului, sau codul introdus de server (statusCode), de exemplu 200 (pentru succes), 404 (pentru resursa negasita) etc.

Aplicatia 5

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

http.createServer((request, response) => {
  request.on('data', (date) => {
    console.log(date);
  }).on('error', (eroare) => {
    console.error(eroare);
  }).on('end', () => {
    console.log("Am primit o cerere");
    response.statusCode = 200;
    response.setHeader('Content-Type', 'text/html');

    response.write("<html><body>O pagina cu node</body></html>");
    response.end();

    console.log("Am trimis un raspuns");
  });
}).listen(8080);

console.log('Serverul a pornit pe portul 8080');
```

```
C:\node_js>node ap_5.js
Serverul a pornit pe portul 8080
Am primit o cerere
Am trimis un raspuns
Am primit o cerere
Am trimis un raspuns
^C
```

<http://localhost:8080/>

Aplicatia 6

aplicație de chat scrisă în Node.js, utilizând biblioteca Socket.IO pentru comunicarea în timp real:

`npm install express socket.io`

`node server.js`

```
const express = require('express');
const app = express();
const http = require('http').createServer(app);
const io = require('socket.io')(http);
```

```
app.use(express.static(__dirname + '/public1'));
```

```
io.on('connection', (socket) => {
  console.log('a user connected');
```

```
  socket.on('disconnect', () => {
    console.log('user disconnected');
  });
```

```
  socket.on('chat message', (message) => {
    console.log('message: ' + message);
    io.emit('chat message', message);
  });
});
```

```
http.listen(3000, () => {
  console.log('listening on *:3000');
});
```

`http://localhost:3000`