

# NodeJS Beginner Level Workshop

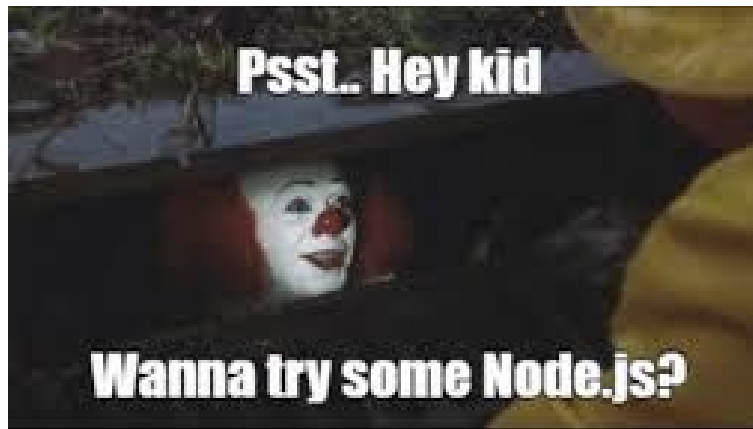
By Tarun Dhirendra Singh  
[10xtarun@gmail.com](mailto:10xtarun@gmail.com)

## **CONTENT**

1. Introduction
2. Career / Jobs in NodeJs
3. Trends with NodeJs
4. Mini Project - Todo API (Server and REST API)

## 1] INTRODUCTION

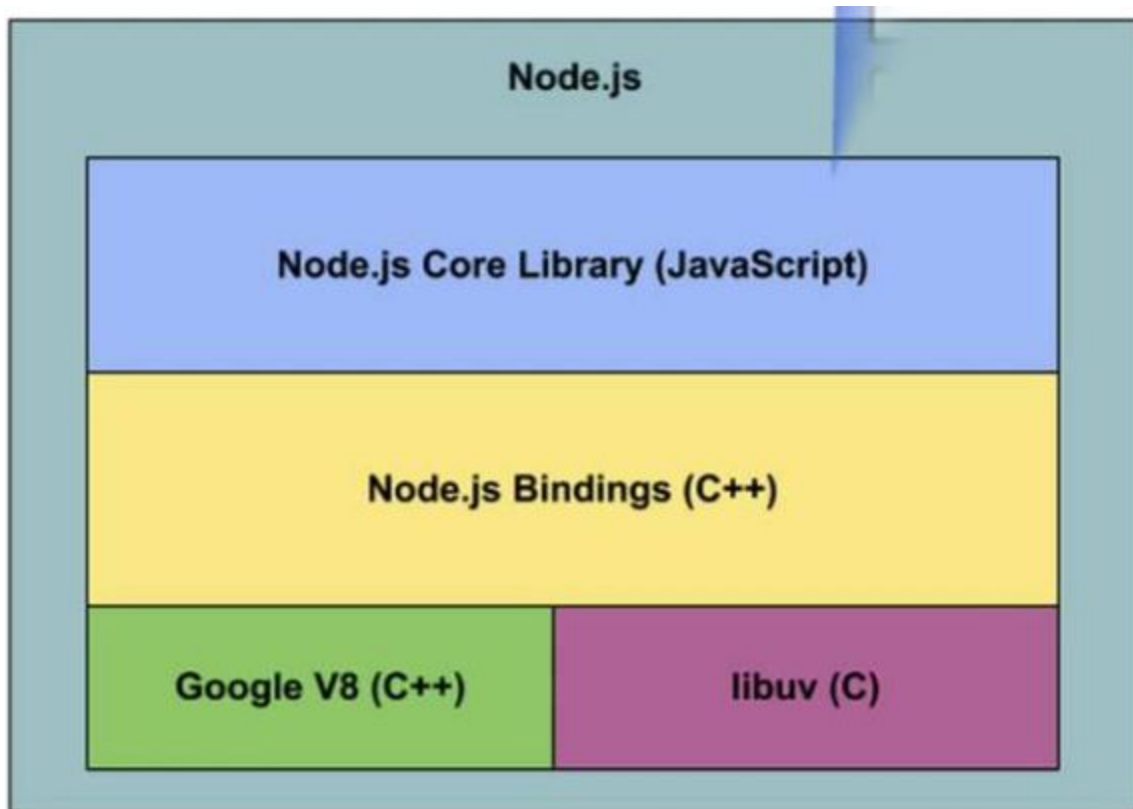
NodeJS is nothing but Javascript which does not run in a browser environment, runs out of the browser environment and hence can be used in the backend.



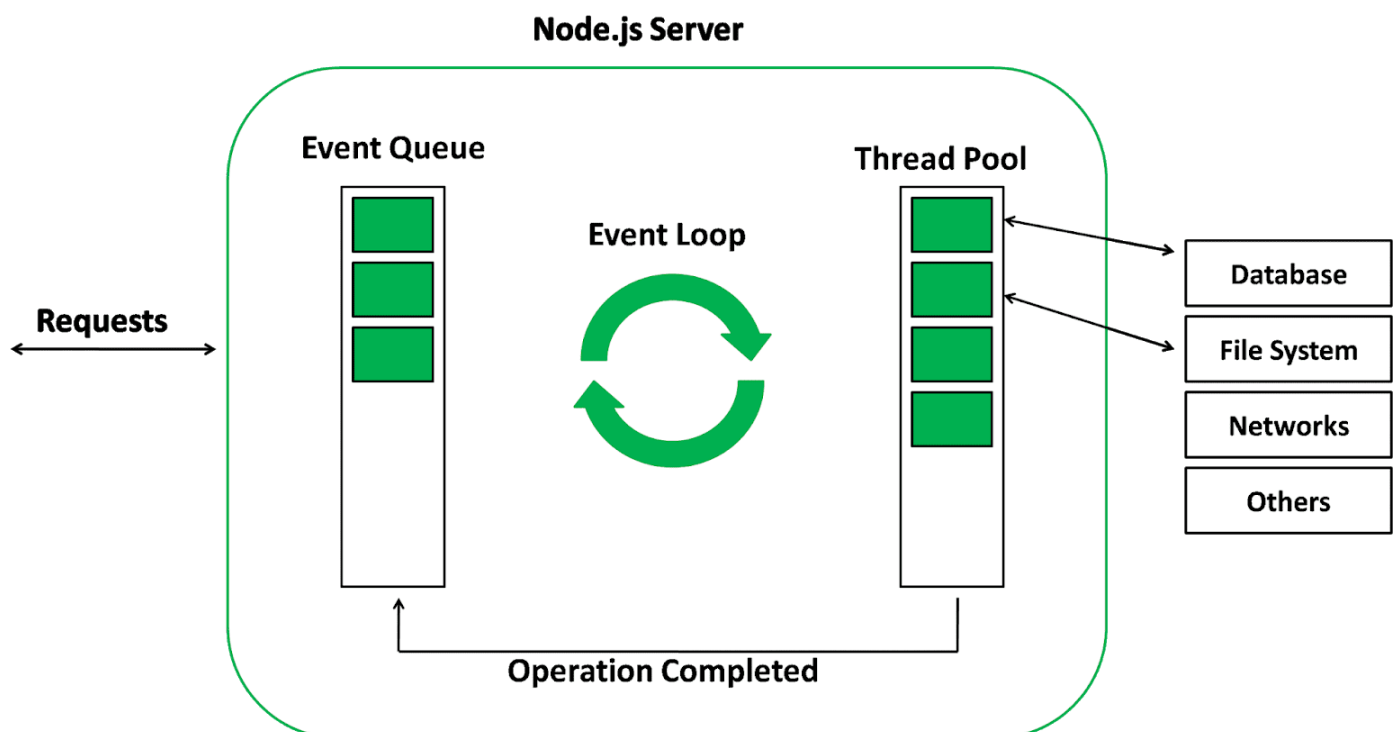
### What is NodeJS ?

- Usually Javascript is used in the Frontend along with the HTML and CSS but this cannot be used in backend where actual business logic is written for the application.
- This is where NodeJs comes into the picture. NodeJs is an open source, cross platform used for server side web development.
- Its main ability shines in IO based applications, which heavily involves input output of data.
- It is event driven, works in the fashion of receiving requests and returning responses.
- Lastly it runs on a V8 engine.

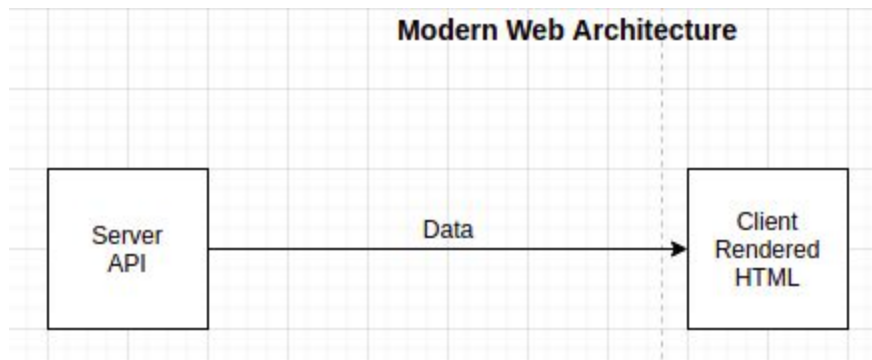
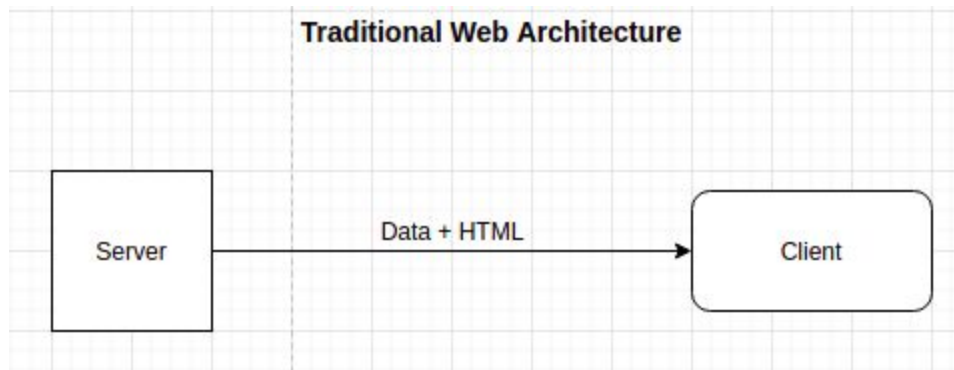
## Structure of V8 Engine:



## NodeJS event loop:




## Understanding Web Architecture



## 2] Careers and Jobs

### Check the number of Jobs

Data as per 30 Jan 2020



Jobs ▾

Date Posted ▾


Experience Level ▾

Compa

**Nodejs in Worldwide**  
85,833 results

Job Alert Off ☐

⚙



Jobs ▾

Date Posted ▾


Experience Level ▾

Compa

**Django in Worldwide**  
16,840 results

Job Alert Off ☐

⚙



Jobs ▾

Date Posted ▾

Experience Level ▾

Compa

**Laravel in Worldwide**  
21,028 results

Job Alert Off ☐

⚙

## **What to do to get a job in NodeJS ?**

LEVEL 0 : Learn fundamentals and perfect the Javascript.

LEVEL 1 :

- Switch to NodeJs and understand the difference between Javascript and NodeJS.
- Learn Fundamentals and understand the NodeJS event loop structure.
- Try to build simple server and APIs using Vanilla NodeJS.

LEVEL 2 :

- Understand ExpressJS (a popular backend framework for NodeJS)
- Code in ExpressJS along with NodeJS.
- Build some good projects.

LEVEL 3 :

- Time to focus on real world application.
- Understand SQL or NoSQL and try to integrate with NodeJS and ExpressJS.
- Build at least 3 projects.

LEVEL 4 : Apply for Internships or Jobs.

### **Bonus ?**

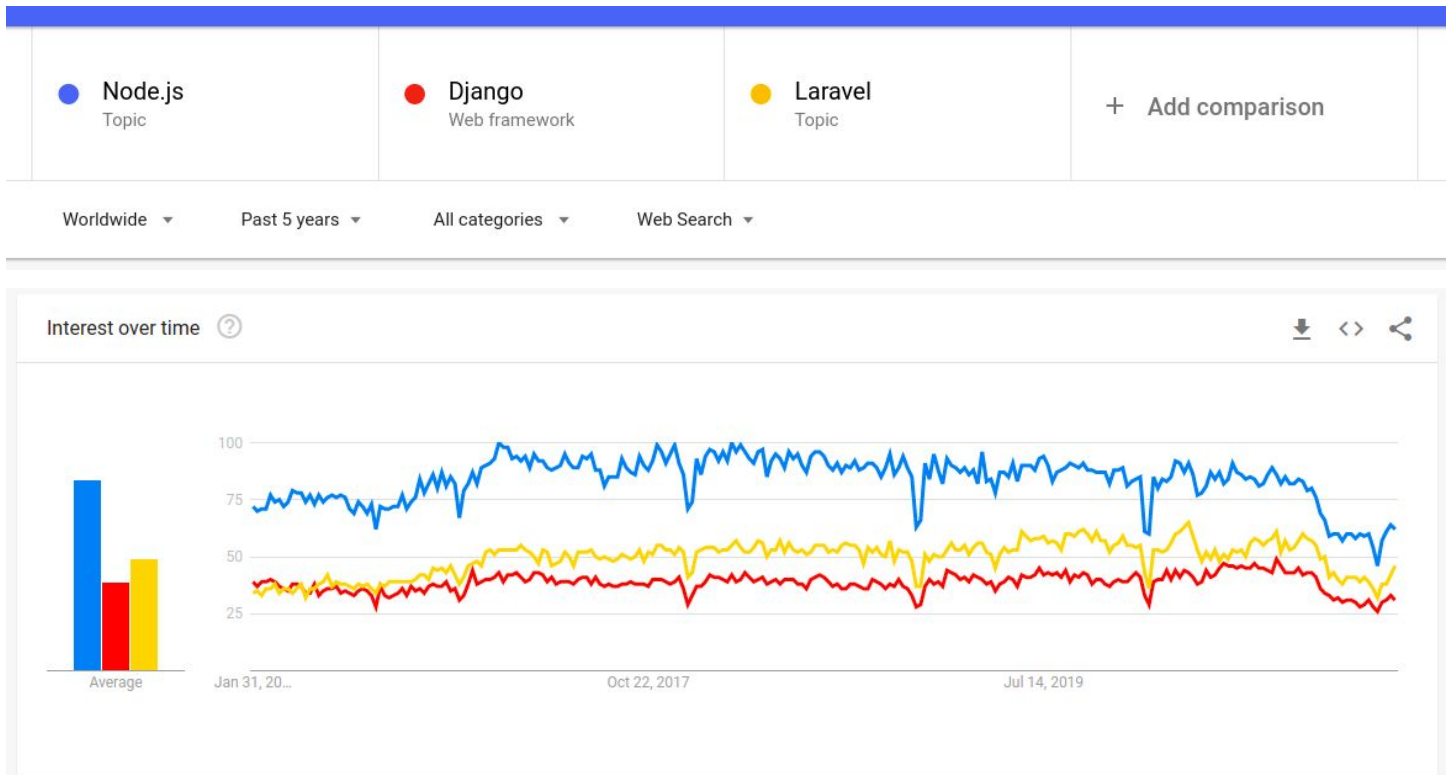
- Host your application on the server, AWS, GCP or Heroku, anything.
- Try to write Unit Test cases at least for 1 project (Hirers love it)
- These last 2 points will make sure that you stand out of the crowd.

### 3] Trends with NodeJS

- Because of the easy to learn curve and its capabilities NodeJS is quite in trend.
- With simple knowledge of Javascript anyone can learn NodeJS.
- NodeJs can handle many concurrent requests with straining load on the server.
- It is designed to handle scalable applications.

### Comparison between trending Web Technologies

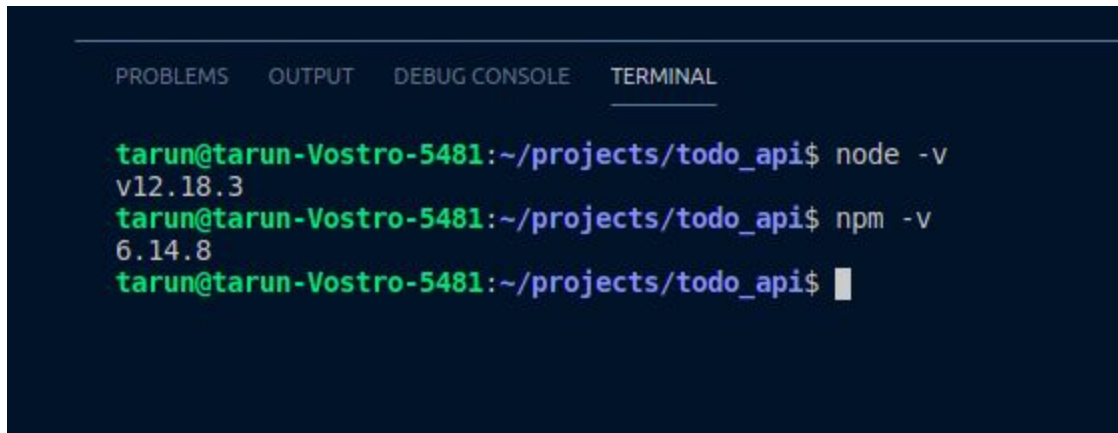
Data as per 30 Jan 2020





#### 4] Mini Project - TODO REST API

Check for the NodeJS version and npm installed on your system.

A screenshot of a terminal window with a dark blue background. At the top, there are four tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL', with 'TERMINAL' being the active tab. The terminal shows three lines of text: the first line is 'tarun@tarun-Vostro-5481:~/projects/todo\_api\$ node -v' followed by 'v12.18.3' on the next line; the second line is 'tarun@tarun-Vostro-5481:~/projects/todo\_api\$ npm -v' followed by '6.14.8' on the next line; and the third line is 'tarun@tarun-Vostro-5481:~/projects/todo\_api\$' followed by a white cursor bar.

```
tarun@tarun-Vostro-5481:~/projects/todo_api$ node -v
v12.18.3
tarun@tarun-Vostro-5481:~/projects/todo_api$ npm -v
6.14.8
tarun@tarun-Vostro-5481:~/projects/todo_api$
```

With the “npm init” command you are initiating the project for nodejs. It asks for various details, like the version, starter file, dependencies, etc.

```
tarun@tarun-Vostro-5481:~/projects/todo_api$ 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: (todo\_api)

version: (1.0.0)

description: A simple todo api using vanilla NodeJS

entry point: (index.js) server.js

test command:

git repository:

keywords: NodeJs REST API

author: Tarun Singh

license: (ISC)

About to write to /home/tarun/projects/todo\_api/package.json:

```
{
  "name": "todo_api",
  "version": "1.0.0",
  "description": "A simple todo api using vanilla NodeJS",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "NodeJs",
    "REST",
    "API"
  ],
  "author": "Tarun Singh",
  "license": "ISC"
}
```


Is this OK? (yes) ☐

## Folder Structure:

- Package.json
- Server.js - project source code

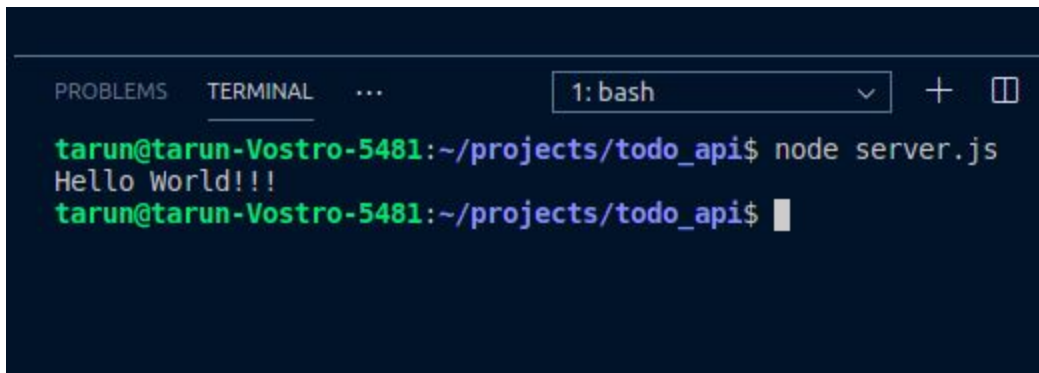


## Simple Hello World Code:

A screenshot of a code editor window with a dark theme. The title bar shows a file named 'server.js' with a JavaScript icon and a close button. The editor contains a single line of code: `1 console.log("Hello World!!!");`.

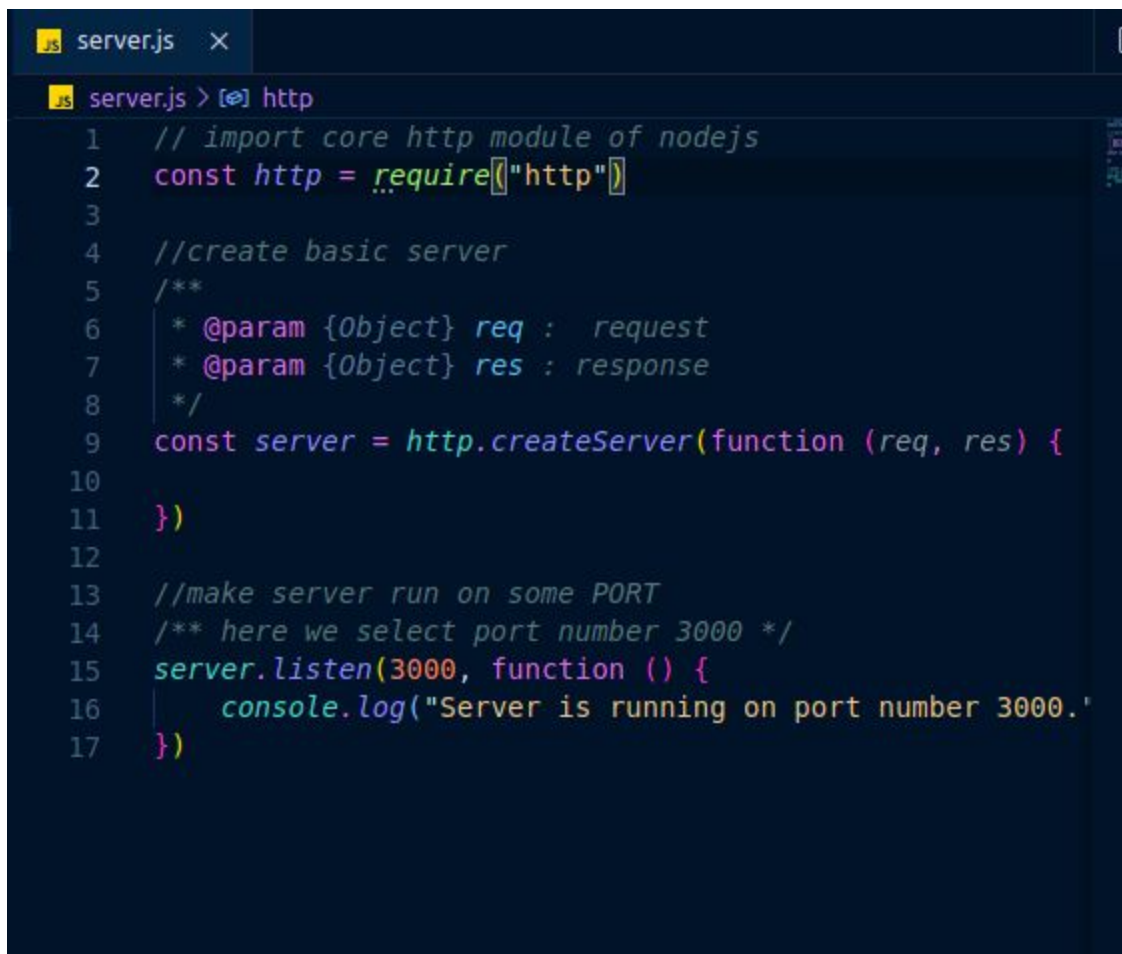
```
server.js
1 console.log("Hello World!!!");
```

## Output:

A screenshot of a terminal window with a dark theme. The terminal shows the command `node server.js` being executed, which results in the output `Hello World!!!`. The prompt is `tarun@tarun-Vostro-5481:~/projects/todo_api$`.

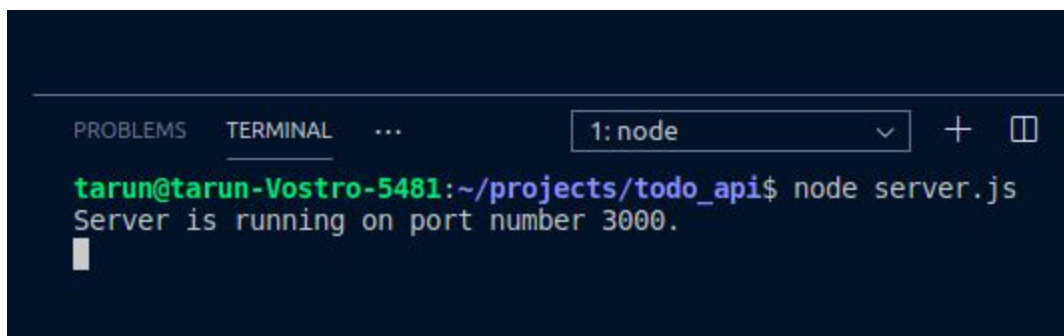
```
tarun@tarun-Vostro-5481:~/projects/todo_api$ node server.js
Hello World!!!
tarun@tarun-Vostro-5481:~/projects/todo_api$
```

## Writing basic server logic.



```
server.js x
server.js > http
1 // import core http module of nodejs
2 const http = require("http")
3
4 //create basic server
5 /**
6  * @param {Object} req : request
7  * @param {Object} res : response
8  */
9 const server = http.createServer(function (req, res) {
10
11 })
12
13 //make server run on some PORT
14 /** here we select port number 3000 */
15 server.listen(3000, function () {
16     console.log("Server is running on port number 3000.")
17 })
```

Output:



```
PROBLEMS  TERMINAL  ...  1: node  +  □
tarun@tarun-Vostro-5481:~/projects/todo_api$ node server.js
Server is running on port number 3000.
```

Adding Get all Todo logic:  
Create a array of todos:

```
let TODO = [  
  {  
    name: "Task 1",  
    completed: false  
  },  
  {  
    name: "Task 2",  
    completed: false  
  },  
  {  
    name: "Task 3",  
    completed: false  
  }  
];
```

Server logic to GET route to retrieve all todos from Array:

```
const server = http.createServer(function (req, res) {  
  if (req.method === 'GET') {  
    return getTodos(req, res)  
  }  
});
```

Function to get all todos from :

```
//to get all todos  
function getTodos(req, res) {  
  const req_url = url.parse(req.url);  
  if (req_url.pathname !== '/todos') {  
    return handleError(res, 404)  
  }  
  res.setHeader('Content-Type', 'application/json; charset=utf-8')  
  return res.end(JSON.stringify(TODO));  
}
```

Output:

## NodeJS Beginner Level Workshop

```
1  [  
2    {  
3      "name": "Task 1",  
4      "completed": false  
5    },  
6    {  
7      "name": "Task 2",  
8      "completed": false  
9    },  
10   {  
11     "name": "Task 3",  
12     "completed": false  
13   },  
14   {  
15     "name": "new task",  
16     "completed": true  
17   }  
18 ]
```

Adding new Todo:

Server logic to add new Todo:

```
const server = http.createServer(function (req, res) {  
  if (req.method === 'GET') {  
    return getTodos(req, res)  
  }  
  else if (req.method === 'POST') {  
    return addTodo(req, res)  
  }  
})
```

Function to add a todo:

```
//to add a todo in the array
function addTodo(req, res) {
  const req_url = url.parse(req.url)
  if (req_url.pathname !== '/todos') {
    return handleError(res, 404)
  }
  let body = '';
  req.on('data', chunk => {
    body += chunk.toString();
  });
  req.on('end', () => {
    let parsed_body = qs.parse(body)
    if (parsed_body.completed === "false") {
      parsed_body.completed = false;
    }
    else {
      parsed_body.completed = true;
    }
    TODO.push(parsed_body);
  })
  res.setHeader('Content-Type', 'application/json;charset=utf-8')
  return res.end("Data added successfully.");
}
```

Output:

---

```
1 Data added successfully.
```