

# EXPRESS

Intro, concepts, practice, HTTP, Postman

# Things to know -

- Node is a runtime environment – allowing JS to run **in the computer not just browser.**
- Node can be used from a variety of development from web to desktop app to IOT. VS code is made off of nodejs.
- Express makes creating **web backend quicker and easier.**

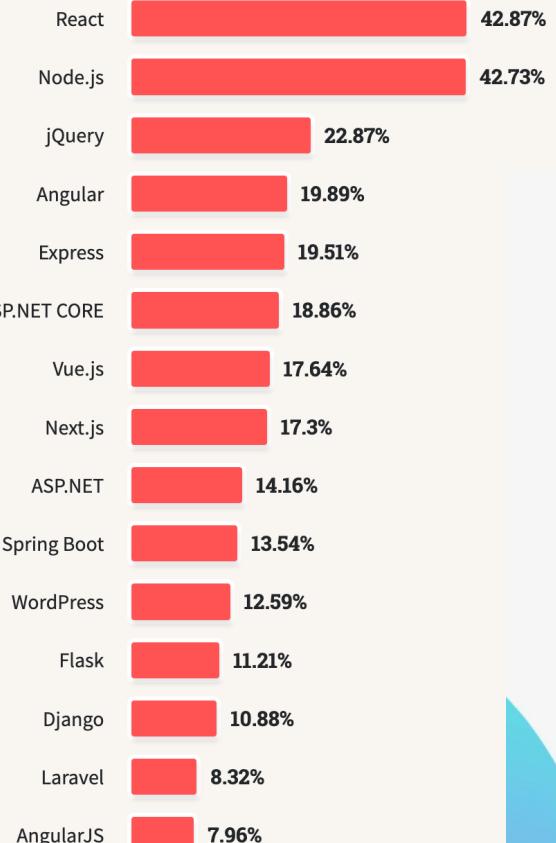
All Respondents

Professional Developers

Learning to Code

56,742 responses

Other Coders



## BACK-END FRAMEWORKS 💰

Frameworks for generating APIs and managing back-ends



6 steps -

## **Creating an Express Server**

- 1.** Create directory.
- 2.** Create index.js file.
- 3.** Initialise NPM.
- 4.** Install the Express package.
- 5.** Write Server application in index.js.
- 6.** Start server.

# 1 and 2 : cd, mkdir, touch

The screenshot shows the Visual Studio Code (VS Code) interface. The Explorer sidebar on the left lists open editors under 'OPEN EDITORS' (including 'Get Started') and files under 'BACKEND-FILES' (including '2.2 Native Modules', '3.1 Express Server', and '2.2 Native Modules....'). The 'TERMINAL' tab at the bottom is active, displaying a terminal session:

```
shoaibahamed@ZH0BM-G13FPQ05N Backend-files % "3.1 Express Server"  
zsh: command not found: 3.1 Express Server  
shoaibahamed@ZH0BM-G13FPQ05N Backend-files % mkdir "3.1 Express Server"  
shoaibahamed@ZH0BM-G13FPQ05N Backend-files % cd 3.1\ Express\ Server  
shoaibahamed@ZH0BM-G13FPQ05N 3.1 Express Server %
```

The 'Get Started' panel is open, showing options like 'New File...', 'Open...', and 'Clone Git Repository...'. A 'Walkthroughs' section highlights the 'Get Started with VS Code' guide, which says: 'Discover the best customizations to make VS Code yours.' Below it, another section titled 'Learn the Fundamentals' is partially visible.

# Express documentation

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

### 3. Initialize npm - `npm init -y`

The screenshot shows the Visual Studio Code interface with the following elements:

- EXPLORER** pane on the left, showing the file structure:
  - OPEN EDITORS**: `Get Started`
  - BACKEND-FILES**:
    - `2.2 Native Modules`
    - 3.1 Express Server**:
      - `index.js`
      - `package.json`
      - `2.2 Native Modules....`
- Get Started** pane in the center, containing links to "Start", "New File...", and "Walkthroughs".
- TERMINAL** pane at the bottom, showing the command line output:

```
/dev/fd/13:18: command not found: compdef
shoaibahamed@ZHOBM-G13FPQ05N Backend-files % pwd
/Users/shoaibahamed/Desktop/full-stack-learning/full-stack-classes/columbia-university /Backend-files
shoaibahamed@ZHOBM-G13FPQ05N Backend-files % "3.1 Express Server"
shoaibahamed@ZHOBM-G13FPQ05N 3.1 Express Server % npm init -y
Wrote to /Users/shoaibahamed/Desktop/full-stack-learning/full-stack-classes/columbia-university /Backend-files/3.1 Express Server/package.json:

{
  "name": "3.1-express-server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
}
```

#### 4. Install express package – `npm i express`

Add “type”:”module”,

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows the project structure with files like `package.json`, `node_modules`, `index.js`, `package-lock.json`, and `package.json` (the active editor).
- OPEN EDITORS**: Shows the current editor is `package.json`.
- BACKEND-FILES**: Shows files related to the backend.
- TERMINAL**: Shows the command `npm fund` run in the terminal.
- OUTPUT**: Shows the output of the command, indicating 0 vulnerabilities.

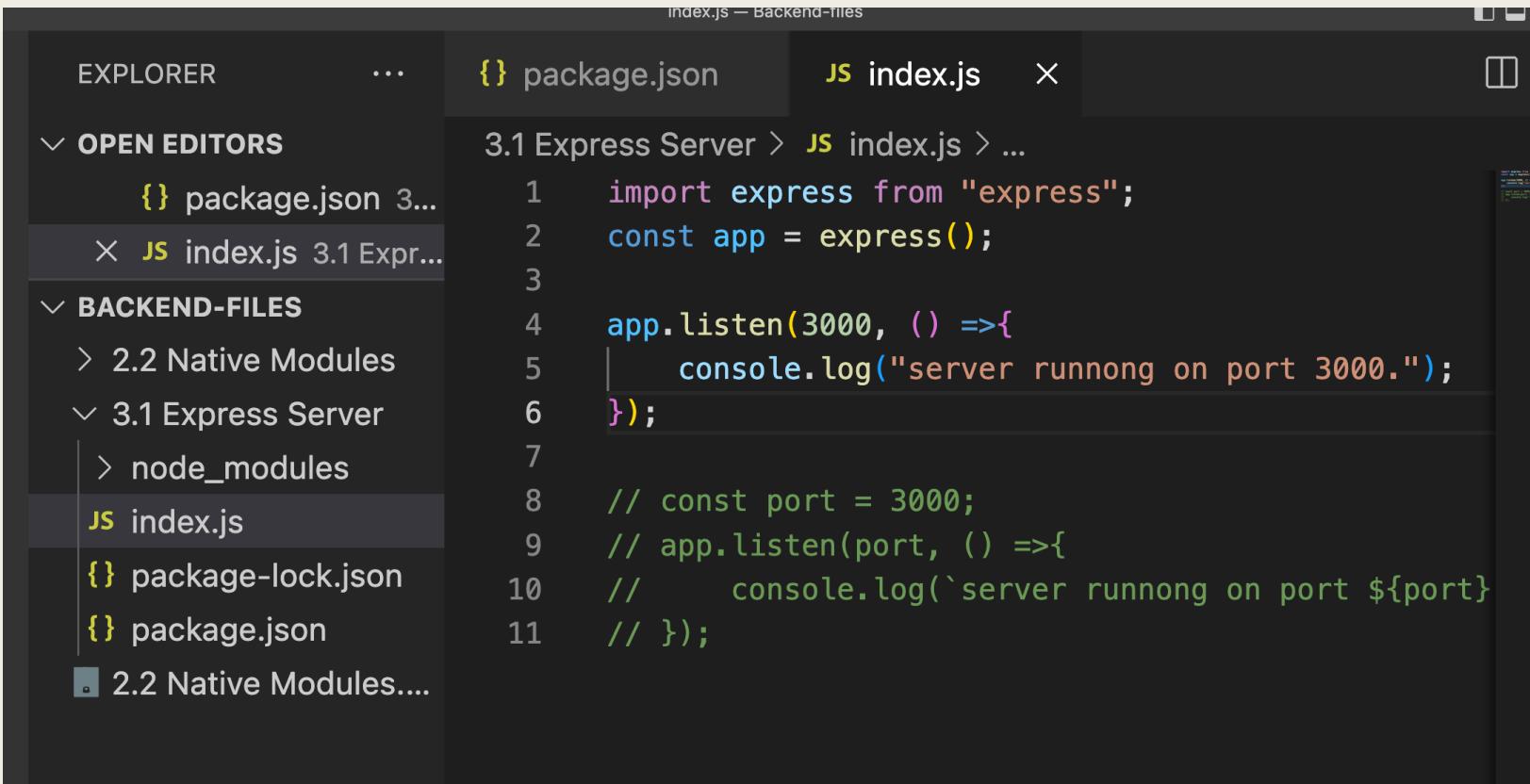
**package.json** content (highlighted in yellow):

```
3 "version": "1.0.0",
4 "description": "",
5 "main": "index.js",
6 "type": "module",
7 // Debug
8 "scripts": {
9   "test": "echo \\\"Error: no test specified\\\" &&
10 },
11 "keywords": [],
12 "author": "",
13 "license": "ISC",
14 "dependencies": {
15   "express": "^4.18.2"
16 }
```

Bottom status bar: `zsh - 3.1 Express Server`

# Let's create a server using express!

# Index.js code should be this -



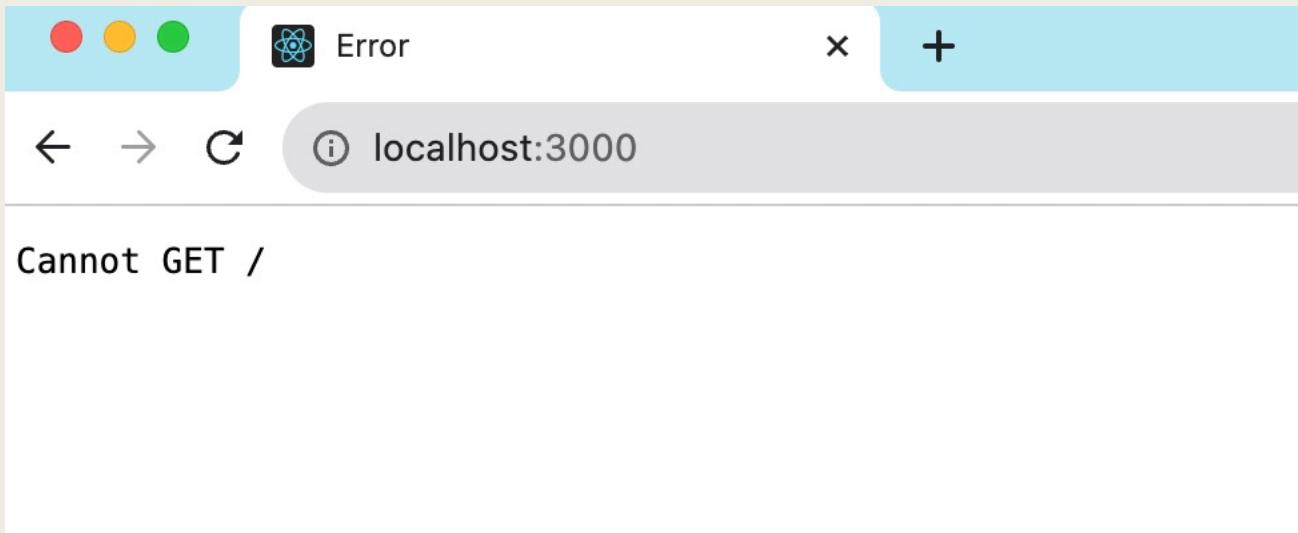
The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** sidebar:
  - OPEN EDITORS**: package.json, index.js (3.1 Expr...)
  - BACKEND-FILES**: 2.2 Native Modules, 3.1 Express Server
    - node\_modules
    - index.js**
  - package-lock.json, package.json
  - 2.2 Native Modules....- PACKAGE.JSON** tab: (empty)
- INDEX.JS** tab: JS index.js
- Content Area**:

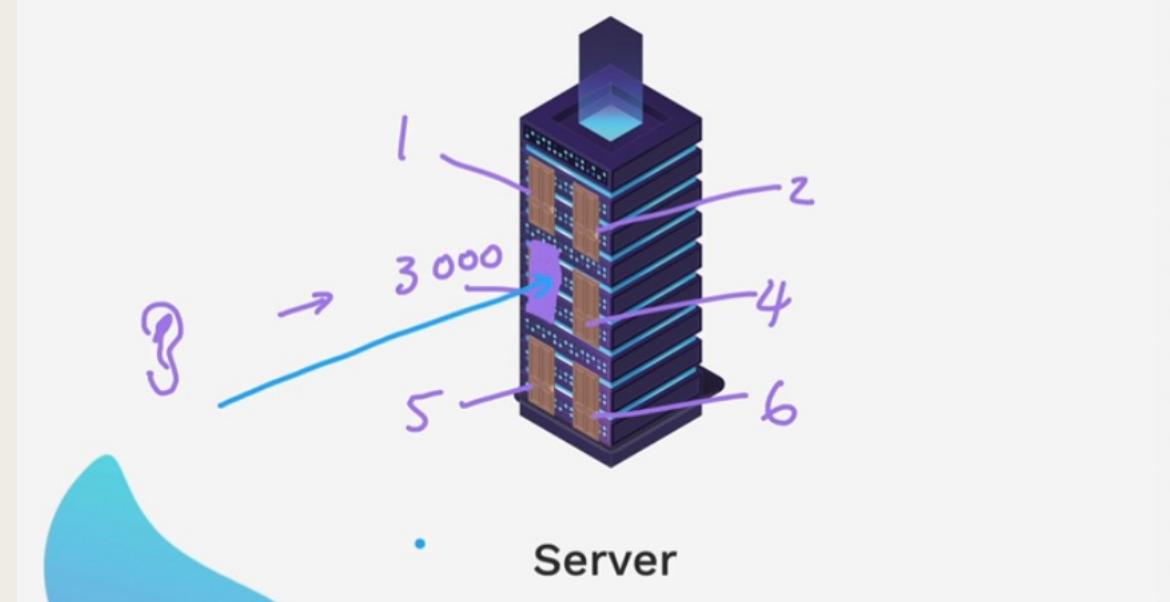
```
3.1 Express Server > JS index.js > ...
1 import express from "express";
2 const app = express();
3
4 app.listen(3000, () =>{
5   console.log("server runnong on port 3000.");
6 });
7
8 // const port = 3000;
9 // app.listen(port, () =>{
10 //   console.log(`server runnong on port ${port}
11 // });
```

## 6. Start server – `node index.js`

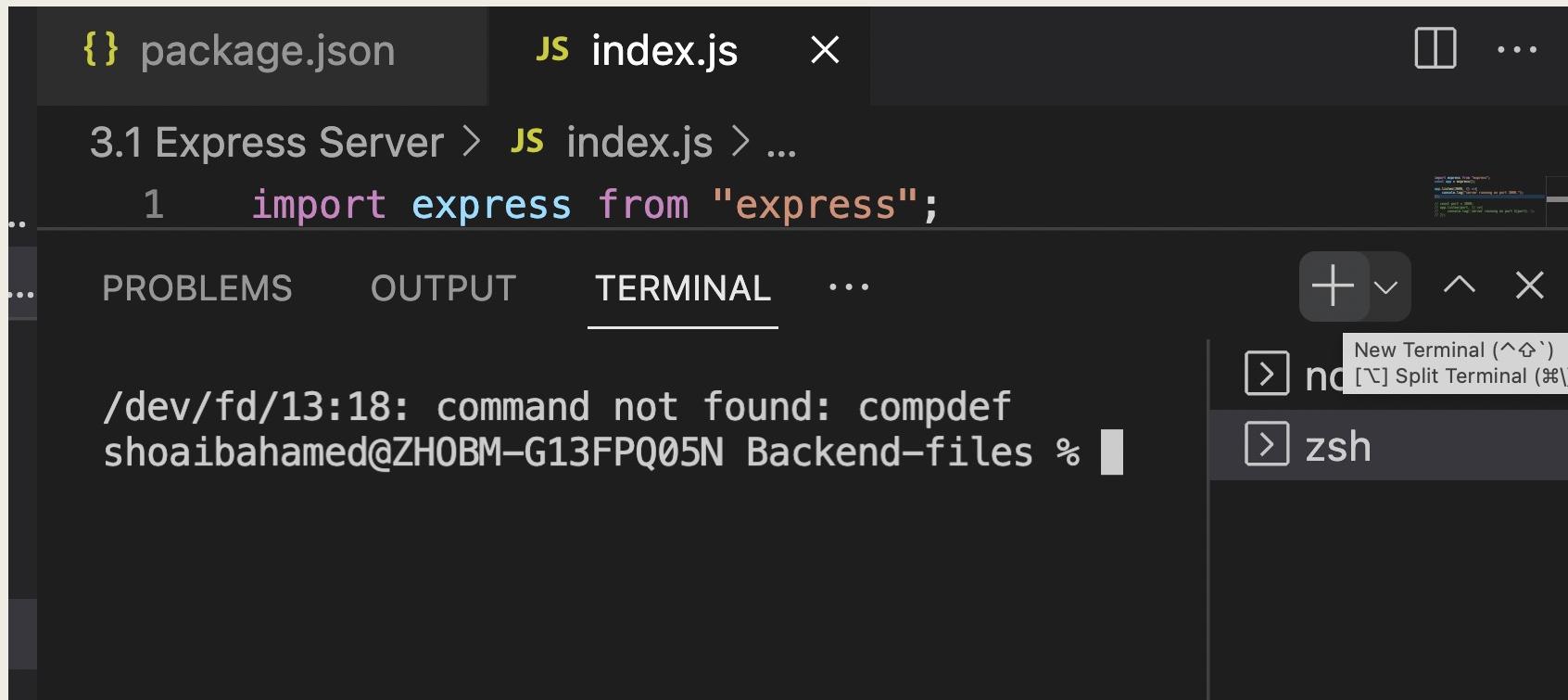
# Browser -



# What's localhost? What's port?



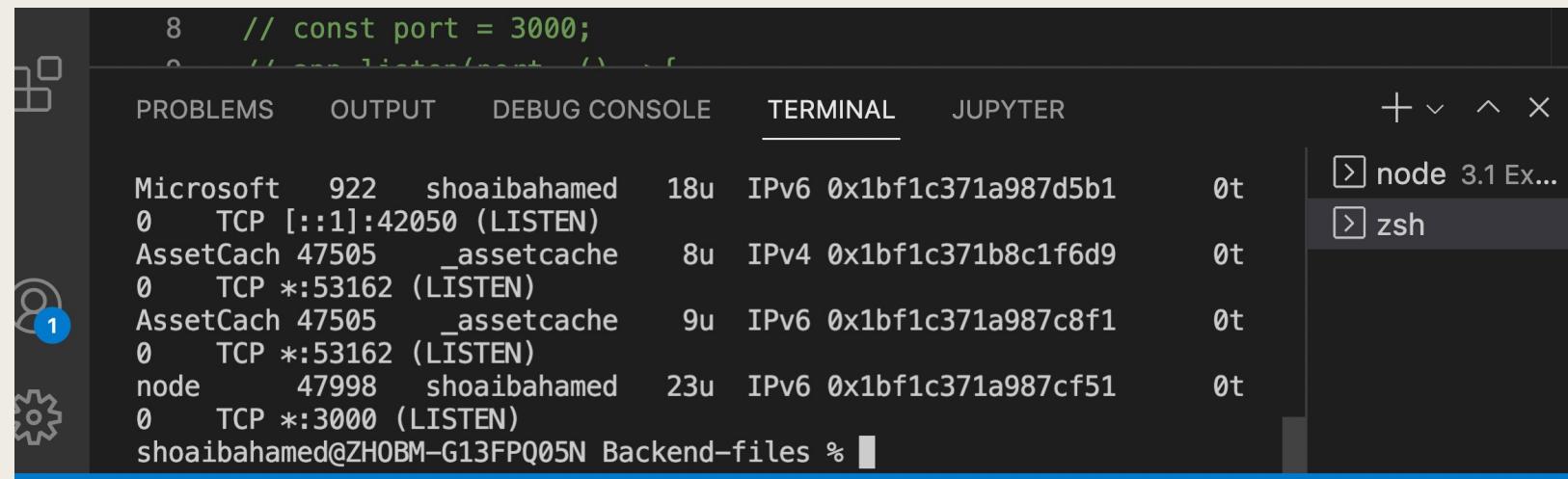
# Adding new terminal +



# List of ports our computer is currently listening to -

**Windows**

```
netstat -ano | findstr "LISTENING"
```



A screenshot of a terminal window titled "Windows Terminal". The window shows the command "netstat -ano | findstr "LISTENING"" being run. The output lists several ports that are currently listening:

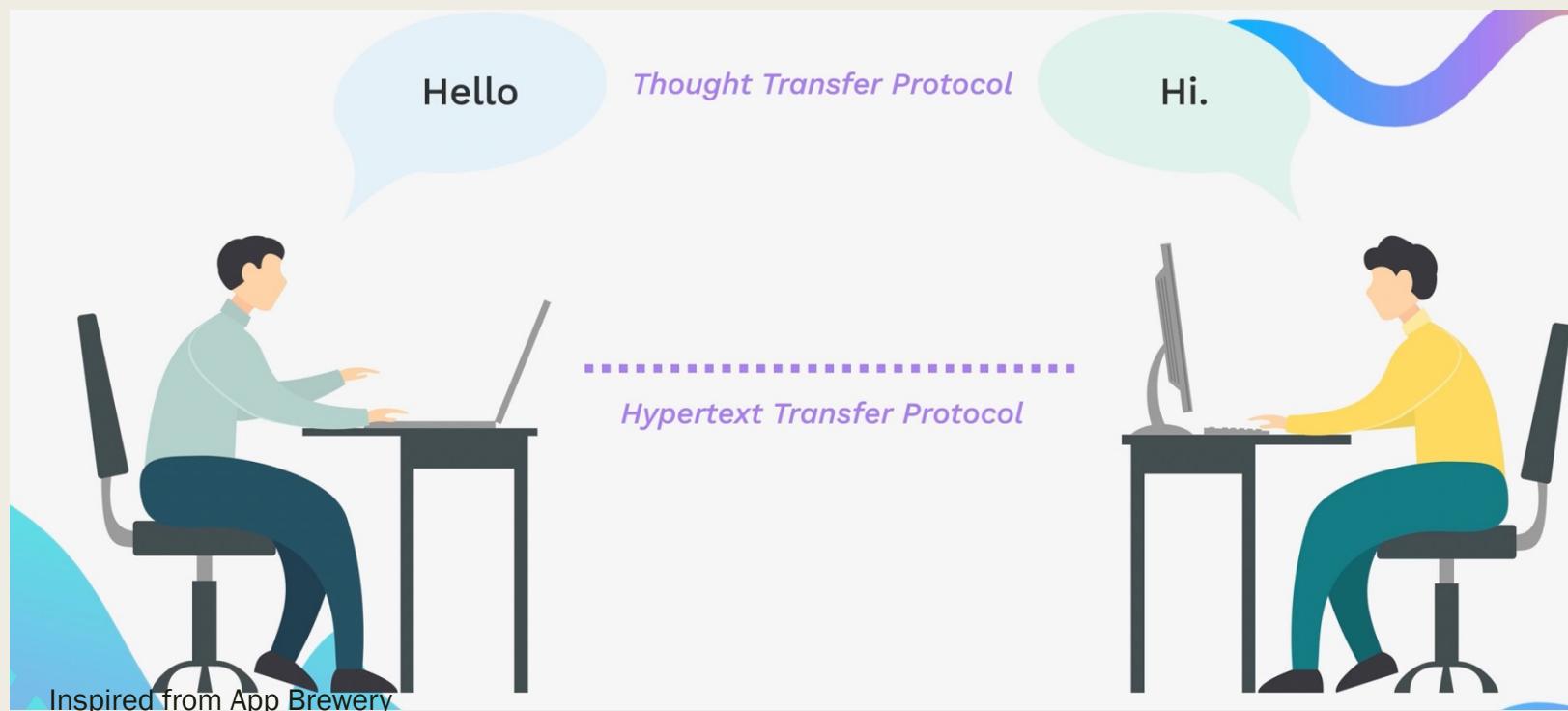
User	Process ID	Port	Type	Local Address	State	Foreign Address	State
Microsoft	922	shoaiyahamed	TCP	0.0.0.0:42050	(LISTEN)	IPv6 0x1bf1c371a987d5b1	0t
AssetCache	47505	_assetcache	TCP	0.0.0.0:53162	(LISTEN)	IPv4 0x1bf1c371b8c1f6d9	0t
AssetCache	47505	_assetcache	TCP	0.0.0.0:53162	(LISTEN)	IPv6 0x1bf1c371a987c8f1	0t
node	47998	shoaiyahamed	TCP	0.0.0.0:3000	(LISTEN)	IPv6 0x1bf1c371a987cf51	0t

**Mac/Linux**

```
sudo lsof -i -P -n | grep LISTEN
```

# HTTP request

- Hyper Text Transfer Protocol.
- Allows computers to talk to each other across the computer.

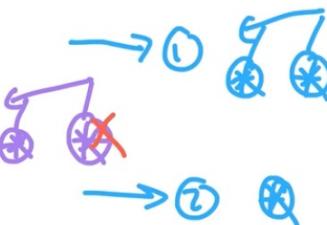


# Different kinds of request -

## Request Vocab

<u>GET</u>	→ Report resource
<u>POST</u>	→ Sending resource
<u>PUT</u>	→ Replace resource
<u>PATCH</u>	→ Patch up a resource
<u>DELETE</u>	→ Delete resource

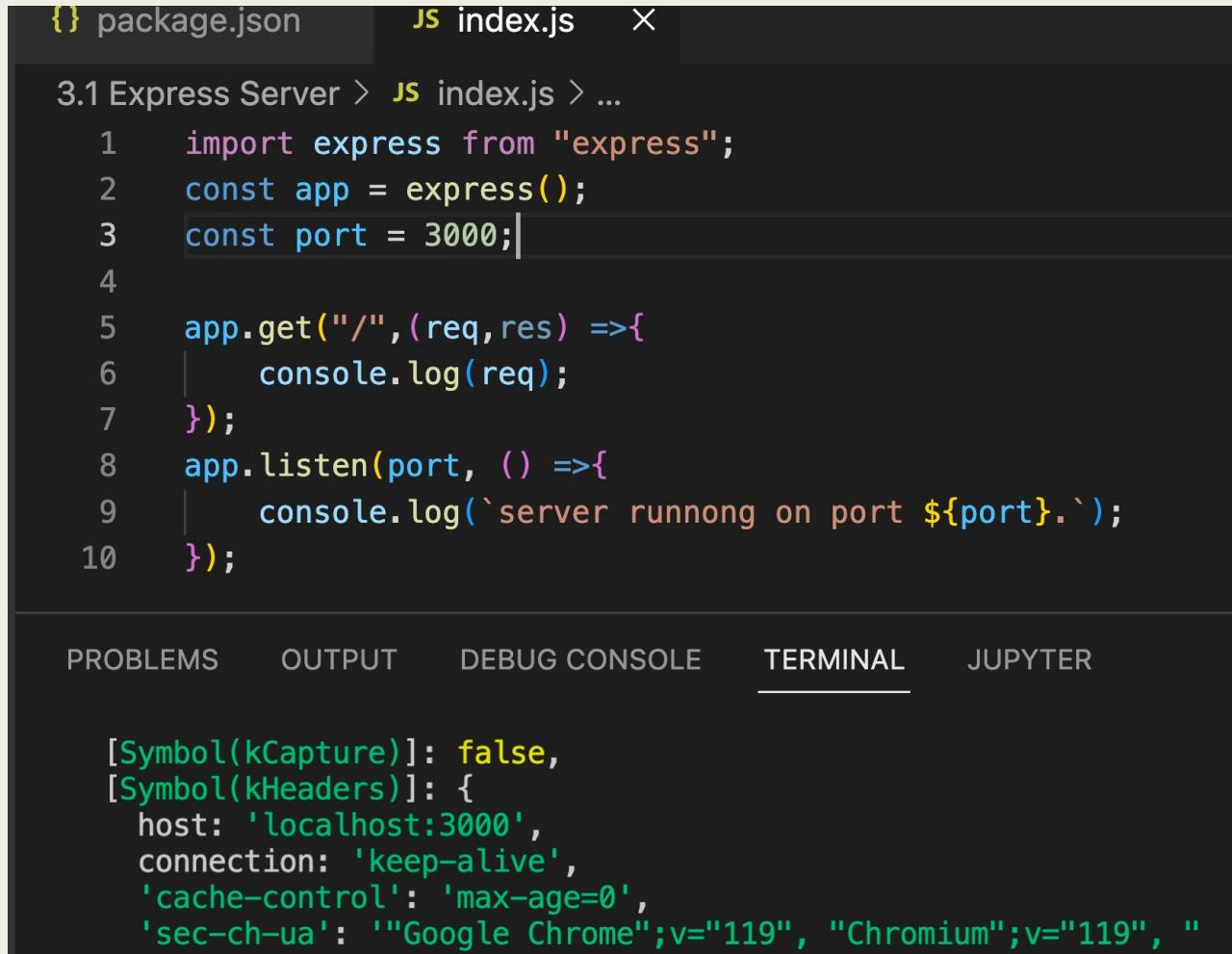
UPDATE



**Challenge : create a new server using  
the 6 steps.**

Now we will learn how to use the GET request!

# Using get request, console.log (req)



The screenshot shows a code editor interface with a dark theme. At the top, there are tabs for 'package.json' (disabled), 'index.js' (selected), and 'X'. Below the tabs, the file content is displayed:

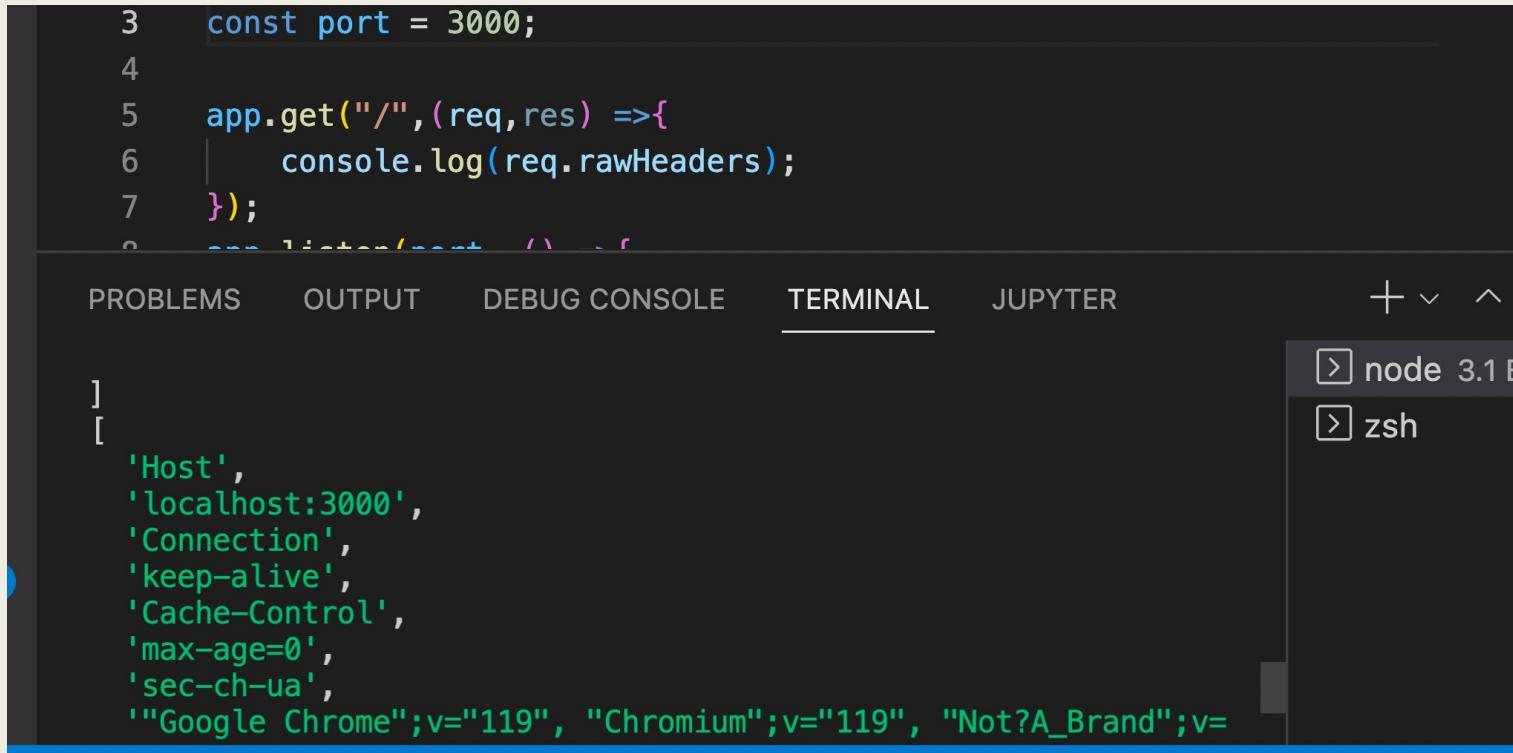
```
3.1 Express Server > JS index.js > ...
1 import express from "express";
2 const app = express();
3 const port = 3000;
4
5 app.get("/", (req, res) =>{
6   console.log(req);
7 });
8 app.listen(port, () =>{
9   console.log(`server runnong on port ${port}.`);
10});
```

Below the code editor, there is a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (selected), and JUPYTER.

The 'TERMINAL' tab shows the output of the application's execution:

```
[Symbol(kCapture)]: false,
[Symbol(kHeaders)]: {
  host: 'localhost:3000',
  connection: 'keep-alive',
  'cache-control': 'max-age=0',
  'sec-ch-ua': '"Google Chrome";v="119", "Chromium";v="119", "
```

# Console.log(req.rawHeaders)



A screenshot of the Visual Studio Code interface. The code editor shows a snippet of Node.js code:

```
3 const port = 3000;
4
5 app.get("/", (req, res) =>{
6   console.log(req.rawHeaders);
7 })
8
```

The 'TERMINAL' tab is selected, displaying the raw HTTP headers captured by the server:

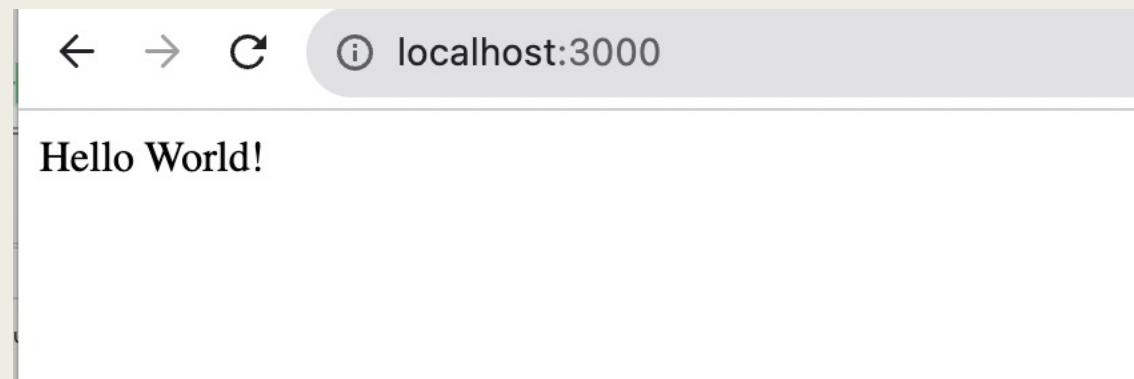
```
]
[
  'Host',
  'localhost:3000',
  'Connection',
  'keep-alive',
  'Cache-Control',
  'max-age=0',
  'sec-ch-ua',
  '"Google Chrome";v="119", "Chromium";v="119", "Not?A_Brand";v=
```

To the right of the terminal, there is a dropdown menu with two items: 'node 3.1 E' and 'zsh'.

Remember to close the server, start over and refresh the browser every time code changes

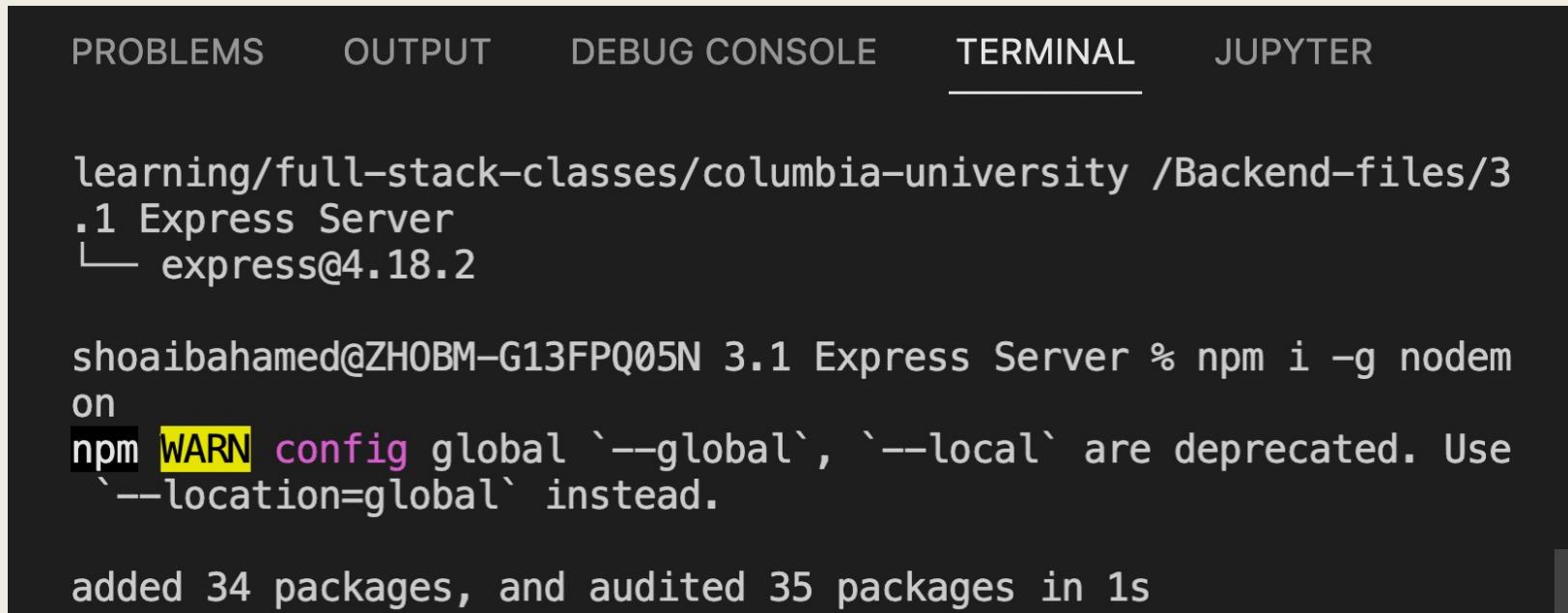
# Sending text to browser

```
4  
5  app.get("/", (req, res) =>{  
6    |    res.send("Hello World!")  
7  });  
8  app.listen(port, () =>{
```



# Stopping and restarting server is annoying!

- Use nodemon. First install: `npm i -g nodemon`. Then `nodemon index.js`. Reference : <https://www.npmjs.com/package/nodemon>



The screenshot shows a terminal window with several tabs at the top: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is currently selected), and JUPYTER. The terminal content displays the following:

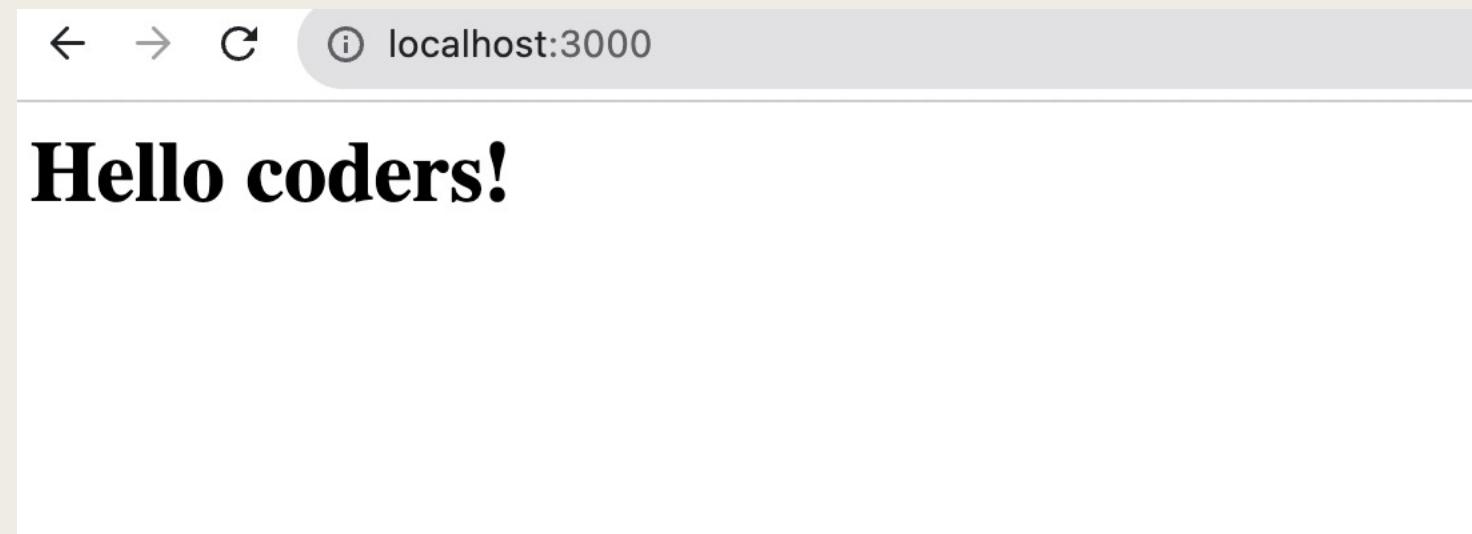
```
learning/full-stack-classes/columbia-university /Backend-files/3
.1 Express Server
└─ express@4.18.2

shoaibahamed@ZHOBM-G13FPQ05N 3.1 Express Server % npm i -g nodem
on
npm WARN config global `--global`, `--local` are deprecated. Use
`--location=global` instead.

added 34 packages, and audited 35 packages in 1s
```

# You can change in code files -

```
app.get("/",(req,res) =>{  
    res.send("<h1>Hello coders!</h1>");  
});  
  
app.get("/about",[req, res]=>{  
    res.send("About page");  
});
```



# /Endpoints



# Challenge: create about and contact endpoints.

- Hint: have multiple GET request.
- Example: localhost:3000/about should say “I am your about page”.  
localhost:3000/contact should say “This is my contact info.”

# Tips: Learn about different server response/HTTP status code (404)

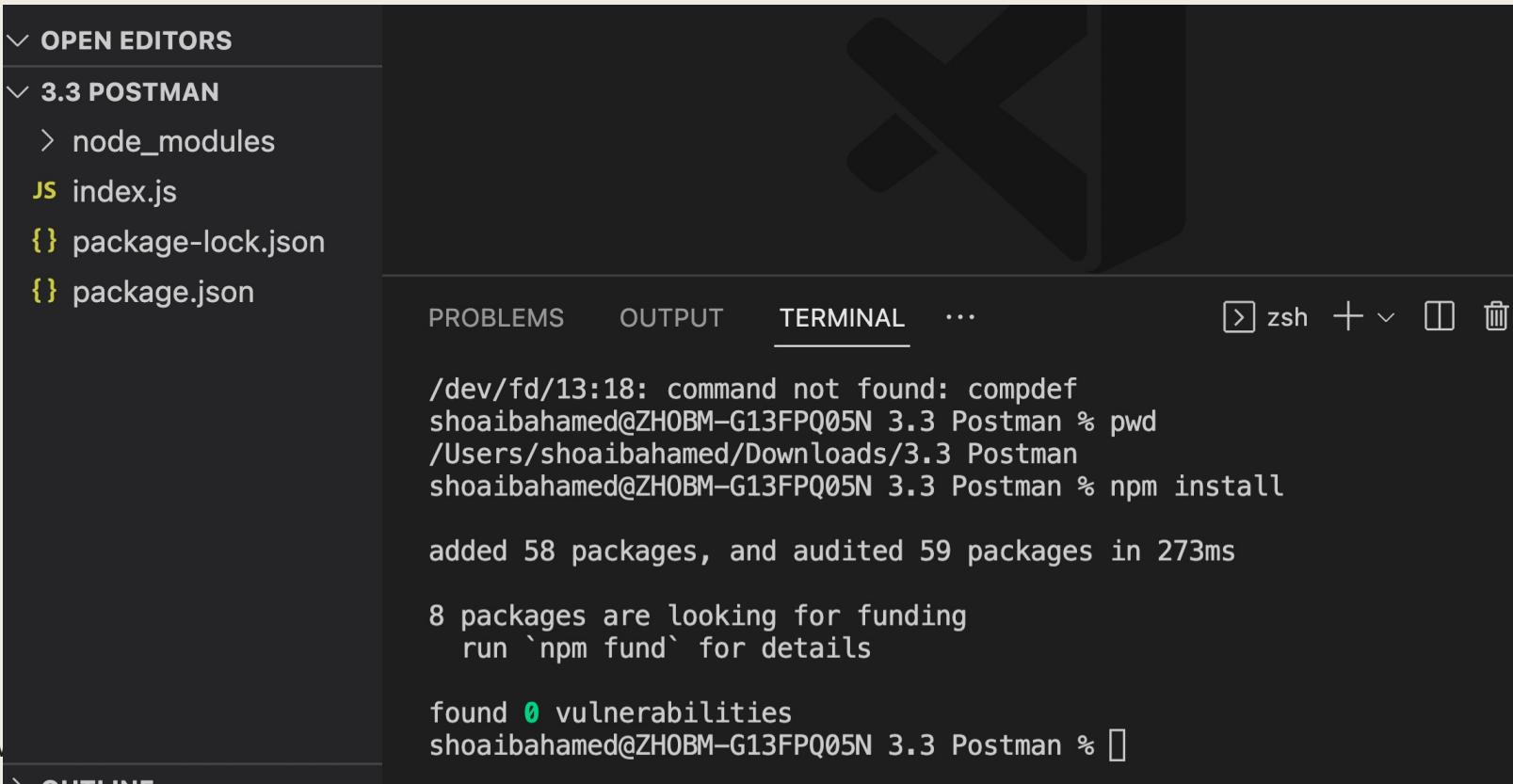
- Here - <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

# Learn HTTP POST, PUT, PATCH and DELETE using postman

- Download postman : <https://www.postman.com/downloads/>
- Sample video - <https://www.youtube.com/watch?v=wmz1sGZp814>

# Exercise

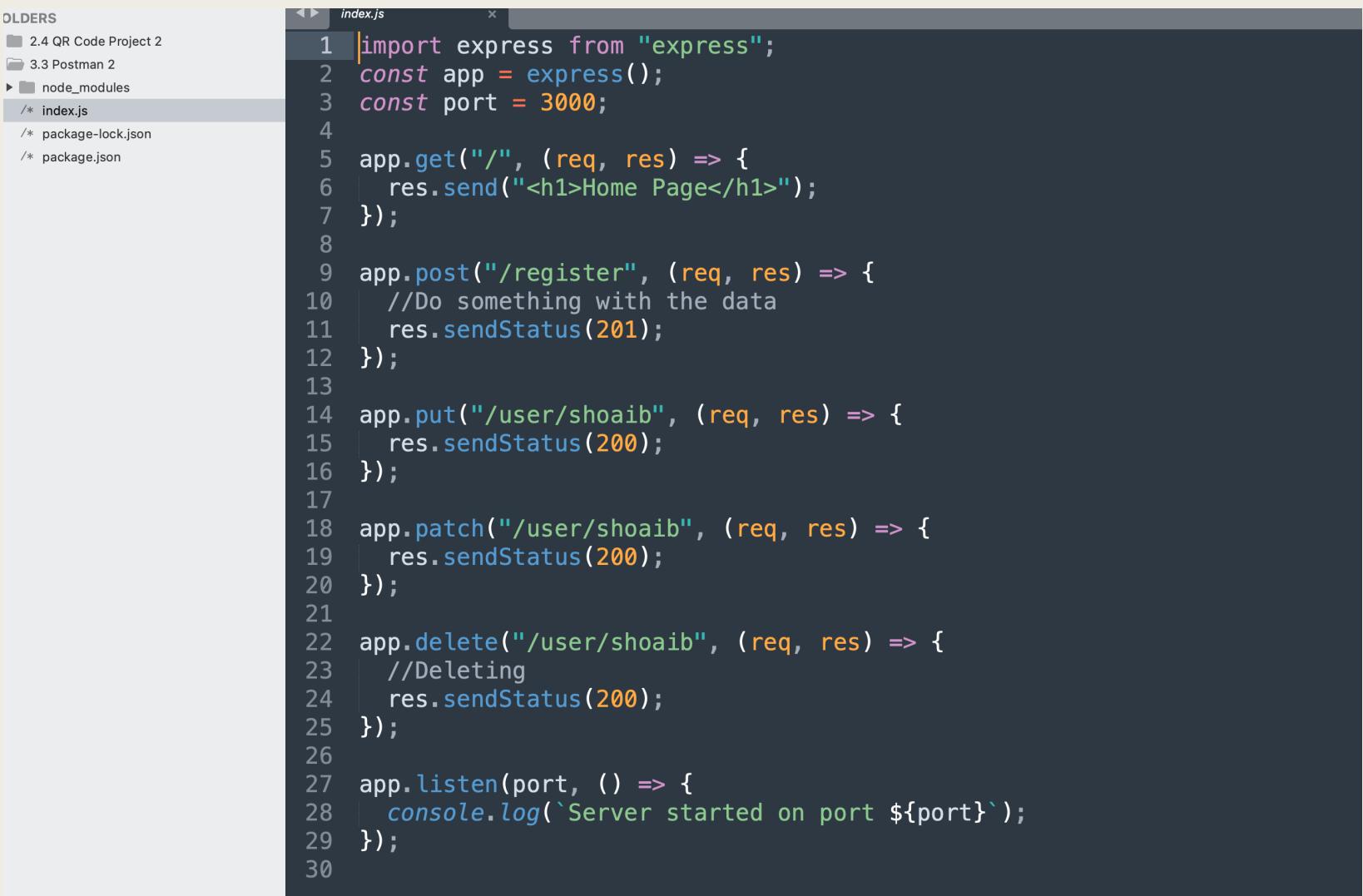
- Download the starter files -  
[https://drive.google.com/file/d/1ckJ7gd3eZIESW\\_HRUEdxpBVS4ZiiszBL/view?usp=sharing](https://drive.google.com/file/d/1ckJ7gd3eZIESW_HRUEdxpBVS4ZiiszBL/view?usp=sharing)
- Open with VS code and cd into it. Run **npm install**. It will add node\_modules folder.
- Run **nodemon index.js**



The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays a project structure under 'OPEN EDITORS' for '3.3 POSTMAN'. The files listed are 'node\_modules', 'index.js' (highlighted in yellow), 'package-lock.json', and 'package.json'. On the right, the Terminal tab is active, showing the following command-line session:

```
/dev/fd/13:18: command not found: compdef
shoaibahamed@ZH0BM-G13FPQ05N 3.3 Postman % pwd
/Users/shoaibahamed/Downloads/3.3 Postman
shoaibahamed@ZH0BM-G13FPQ05N 3.3 Postman % npm install
added 58 packages, and audited 59 packages in 273ms
8 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
shoaibahamed@ZH0BM-G13FPQ05N 3.3 Postman %
```

# Index.js



The image shows a screenshot of a code editor with the file "index.js" open. The file contains Node.js code for a RESTful API. The code includes imports for express, defines a port, and sets up routes for GET, POST, PUT, PATCH, and DELETE methods on the "/user/shoaib" endpoint. It also includes a listen call and a log statement.

```
index.js
1 import express from "express";
2 const app = express();
3 const port = 3000;
4
5 app.get("/", (req, res) => {
6   res.send("<h1>Home Page</h1>");
7 });
8
9 app.post("/register", (req, res) => {
10   //Do something with the data
11   res.sendStatus(201);
12 });
13
14 app.put("/user/shoaib", (req, res) => {
15   res.sendStatus(200);
16 });
17
18 app.patch("/user/shoaib", (req, res) => {
19   res.sendStatus(200);
20 });
21
22 app.delete("/user/shoaib", (req, res) => {
23   //Deleting
24   res.sendStatus(200);
25 });
26
27 app.listen(port, () => {
28   console.log(`Server started on port ${port}`);
29 });
30
```

# Exercise -

- Open new tabs in postman for each of the requests listed in index.js file
- Hit the right endpoints listed in index.js file.
- Send these requests to get back a successful response status codes.

# Get Request

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Import', 'Overview' (selected), 'GET localhost:3000' (the current request), a red error icon, a '+' button, and three dots. To the right is 'No Environment' and a dropdown arrow.

The main area shows a request card for 'localhost:3000'. It has a method dropdown set to 'GET' and a URL field containing 'localhost:3000'. To the right of the URL are 'Save' and 'Edit' buttons. Below the card is a toolbar with 'Send' and a dropdown arrow.

The request configuration section includes tabs for 'Params' (selected), 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', 'Settings', and 'Cookies'. Under 'Params', there's a table for 'Query Params' with columns: Key, Value, Description, and Bulk Edit. One row is present with 'Key' and 'Value' empty, and 'Description' as 'Description'.

At the bottom, there are tabs for 'Body', 'Cookies', 'Headers (7)', and 'Test Results' (selected). On the right side of the bottom bar, there's a globe icon, '200 OK', '26 ms', '246 B', a 'Save as example' button, and three dots.

Below the bottom bar, there are buttons for 'Pretty', 'Raw', 'Preview', and 'Visualize'. The word 'Home Page' is displayed prominently in large, bold, black font below the preview area.

At the very bottom, the text 'Inspired from App Brewery' is visible.

# POST request

The screenshot shows the Postman application interface for making a POST request to `localhost:3000/register`. The request body contains a single parameter `Name` with the value `Shoaib`.

**Request Headers:**

- Method: POST
- URL: `localhost:3000/register`
- Content-Type: `x-www-form-urlencoded`

**Body (x-www-form-urlencoded):**

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> Name	Shoaib			

**Response:**

- Status: 201 Created
- Time: 6 ms
- Size: 239 B
- Save as example

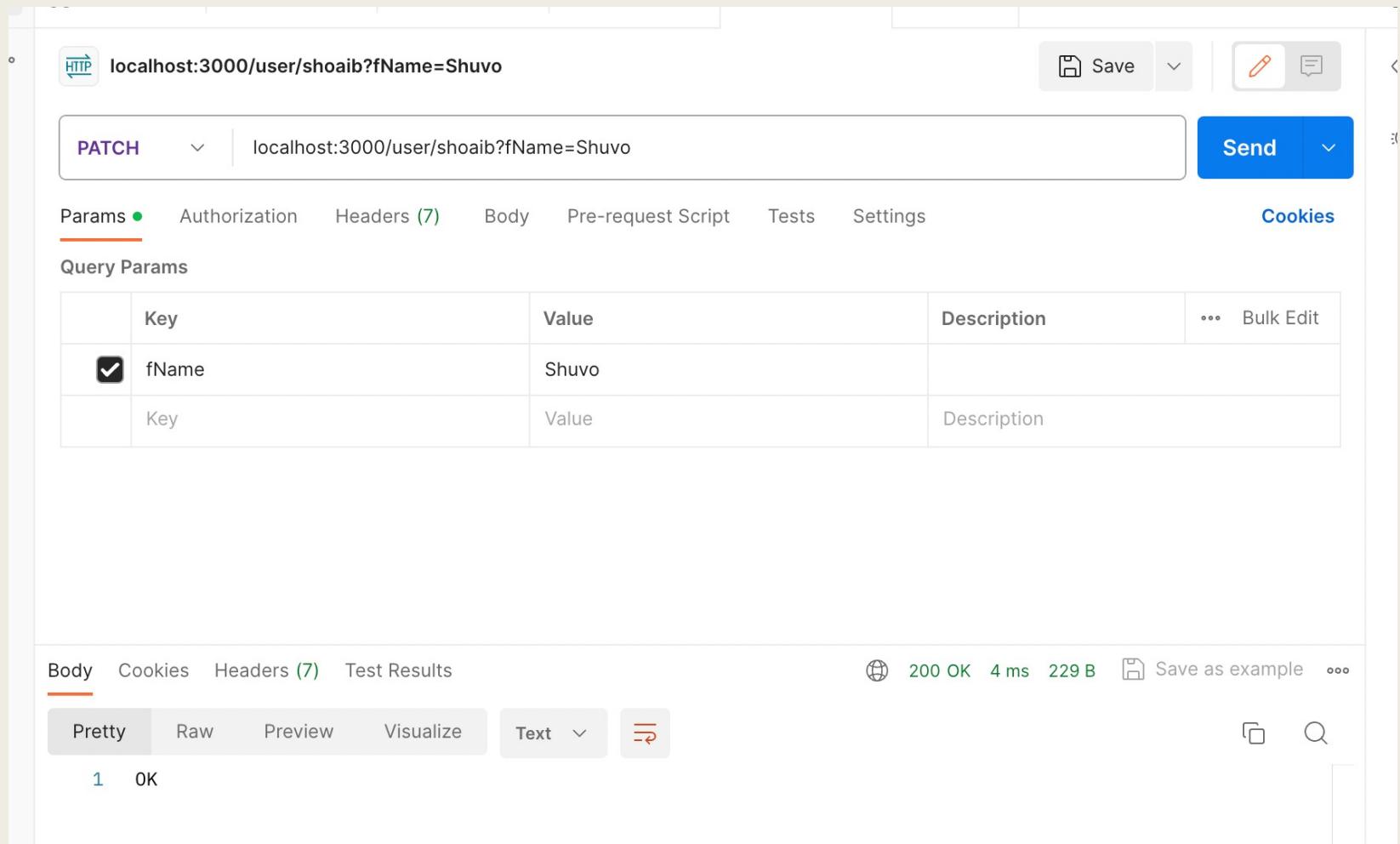
**Body Content:**

```
1 Created
```

# PUT request

The screenshot shows the Postman application interface. At the top, there are tabs for Overview, GET localhost:3000, POST localhost:3000, and PUT localhost:3000/. Below these, the URL is set to `localhost:3000/user/shoaib`. The method is selected as `PUT`. The Headers section shows 8 items. The Body tab is active, showing the content type as `x-www-form-urlencoded`. The body contains one key-value pair: `email` with the value `shoaib@gmail.com`. In the bottom right corner, the response is displayed as a `200 OK` status with the message: "Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request the response will contain an entity describing or containing the result of the action."

# PATCH request



The screenshot shows the Postman application interface with a PATCH request configuration.

**Request URL:** `localhost:3000/user/shoaib?fName=Shubo`

**Method:** PATCH

**Headers:** (7)

**Body:** (Empty)

**Tests:** (Empty)

**Settings:** (Empty)

**Cookies:** (Empty)

**Query Params:**

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	fName	Shubo			
	Key	Value	Description		

**Response Status:** 200 OK

**Response Time:** 4 ms

**Response Size:** 229 B

**Actions:** Save as example, Copy, Find

# DELETE request

The screenshot shows the Postman application interface. At the top, there's a navigation bar with tabs for Overview, GET, POST, PUT, PATCH, and a redacted URL. Below the tabs, a search bar contains the URL "localhost:3000/user/shoaib". To the right of the search bar are Save, Edit, and Delete buttons. The main area shows a request configuration for a DELETE method. The Headers tab is selected, showing six headers. The Body tab is also visible. Below the request area, the response section shows a status of 200 OK with a response body containing "1 OK".

# Homework

- Research and learn about Express middlewares. Uses of bodyParser.
- Sample link to look at - <https://expressjs.com/en/resources/middleware/body-parser.html>