Who am I?

Humera Tariq

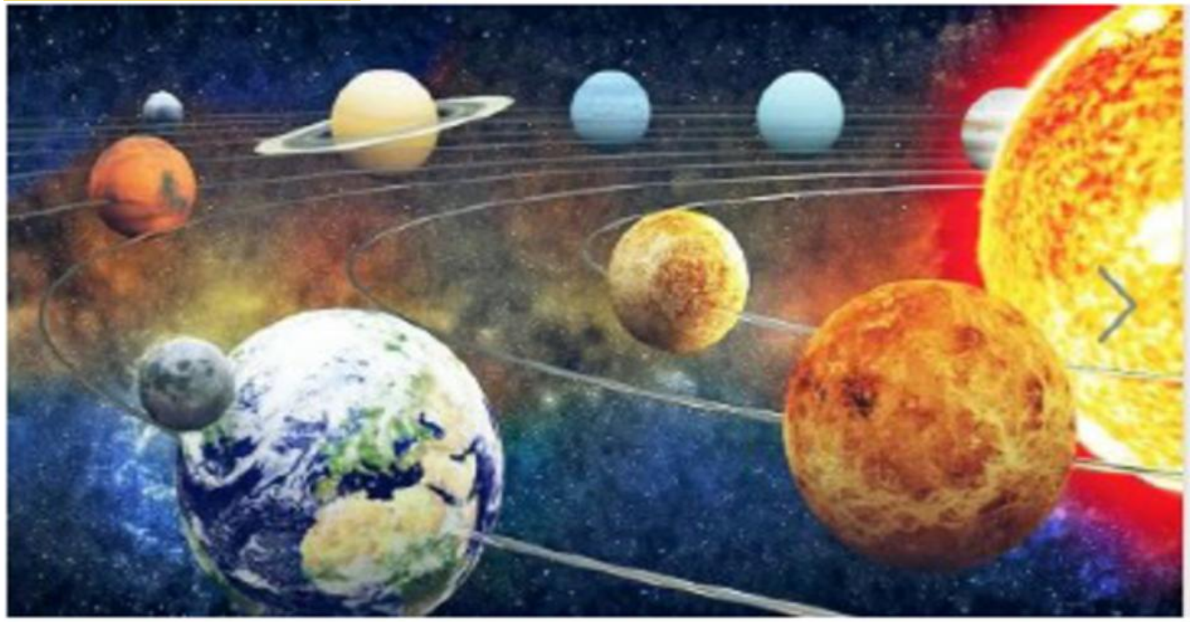
PhD, MS, MCS (Computer Science), B.E (Electrical)
Postdoc (Medical Image Processing, Deep Neural Networks)

Email: humera@uok.edu.pk
Web: https://humera.pk/

Discord: https://discord.gg/xeJ68vh9

Starting in the name of Allah,

the most beneficial,
the most merciful.



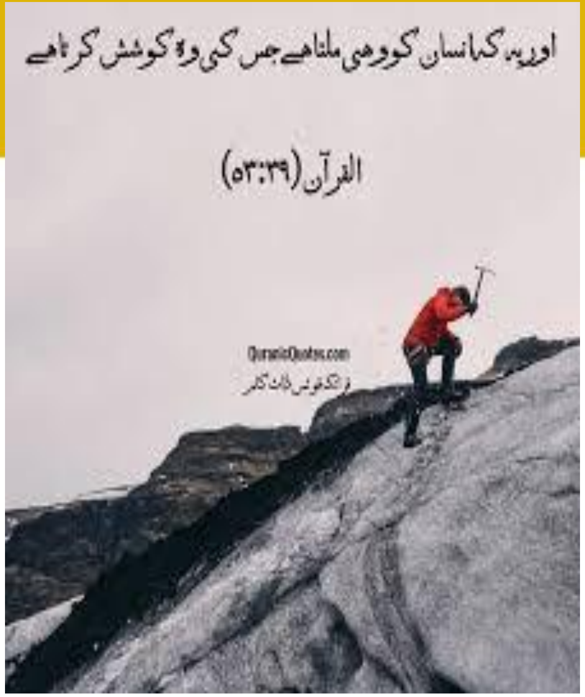
اَمْ لِلْاِنْسَانِ مَا تَمَنّٰى ۖ ﴿۲۴﴾

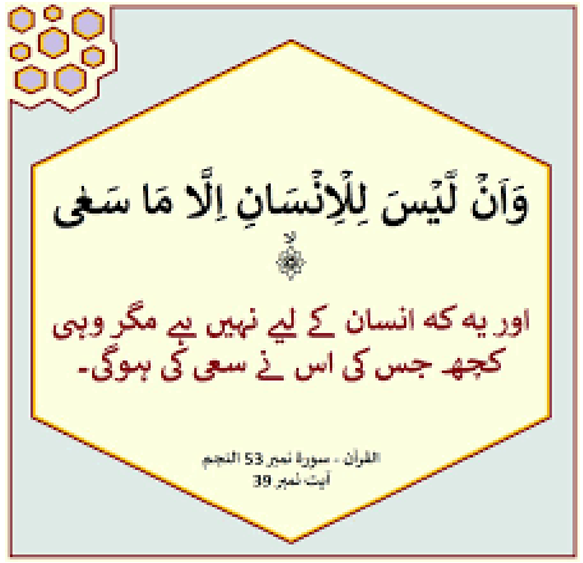
کیا انسان کو ہر وہ چیز حاصل ہے جس کی اس نے تمنا کی؟

UNIVERSITY OF KARACHI

Surah An-Najm Chapter 53 Verse 39

And there is not for man except that [good] for which he strives.





وَاَنۡ لَّيۡسَ لِلۡاِنۡسَانِ اِلَّا مَا سَعٰى

اور یہ کہ انسان کے لیے نہیں ہے مگر وہی کچھ جس کی اس نے سعی کی ہوگی۔

القرآن ، سورۃ نمبر 53 النجم
آیت نمبر 39

UNIVERSITY OF KARACHI

Discussion

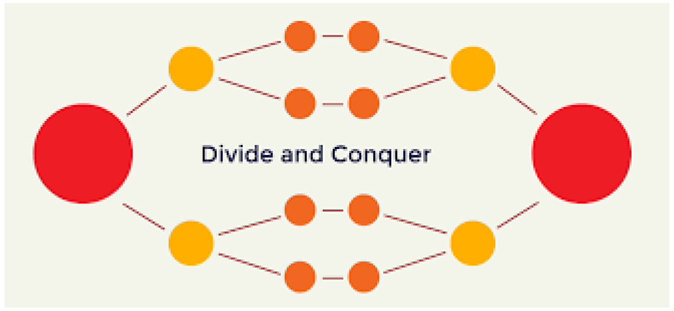
Week 07

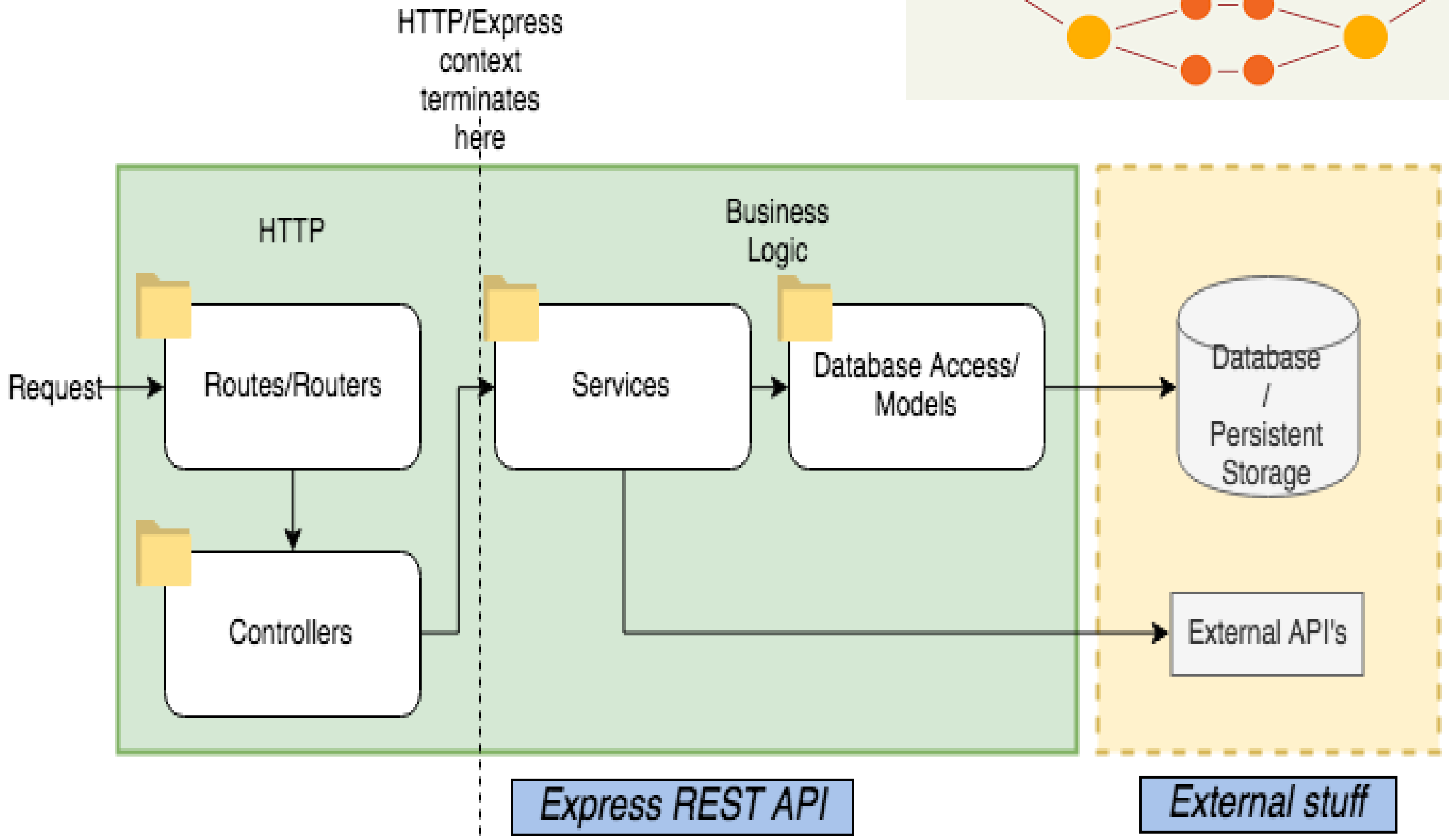
Internet Application Development

Department of Computer Science (DCS/UBIT)
University of Karachi
January 2025

Express as backend (REST API).

Mapping of architecture to folder structure



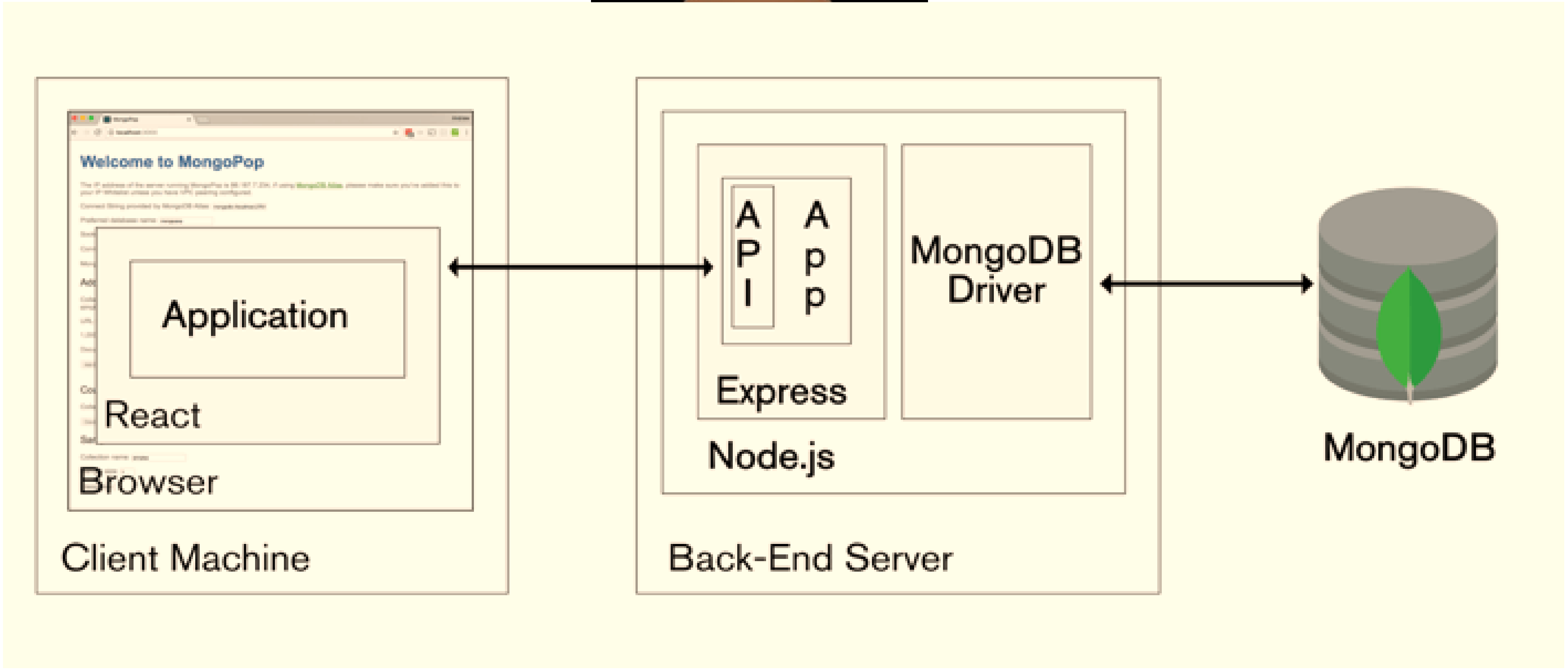


http (Handles Requests)
│—— routes/
│—— controllers/
│
business logic (Express REST API)
│—— services/
│—— models/

UNIVERSITY OF KARACHI

"For those who dare to dream, there is a whole world to win"
Dhirubhai Ambani

The MERN stack is a group of four technologies often used together to build web applications.

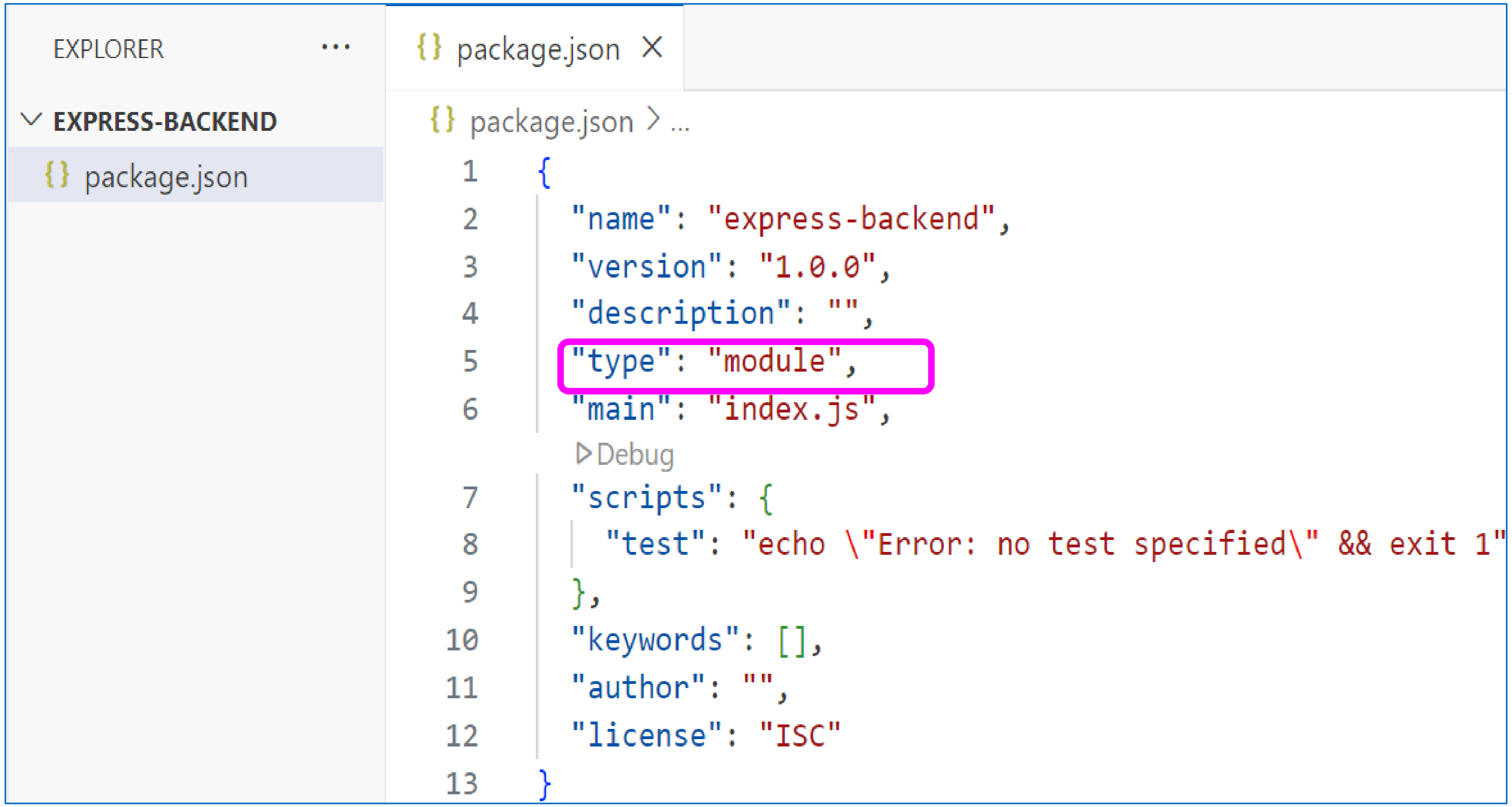


Request-Response Cycle (HTTP Communication)

Single file server.js

Let’s code a minimal restful API to show **how GET requests work** and how JSON responses are structured.

- ✓ mkdir first-server # Create a new directory
- ✓ cd first-server # Navigate into the directory
- ✓ npm init -y
- ✓ Modify package.json to Use ES Modules



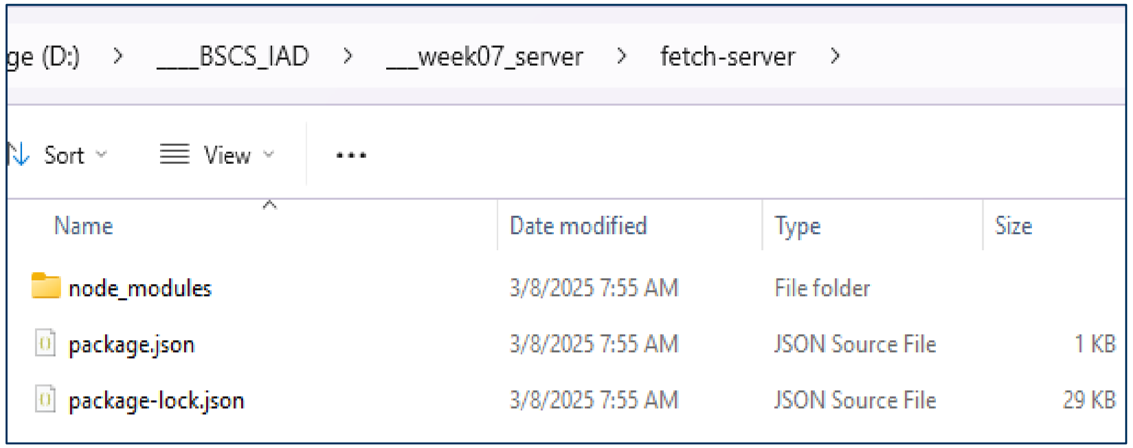
```json
{
  "name": "express-backend",
  "version": "1.0.0",
  "description": "",
  "type": "module",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

✓ npm install express cors dotenv

express → Backend framework to handle API requests.

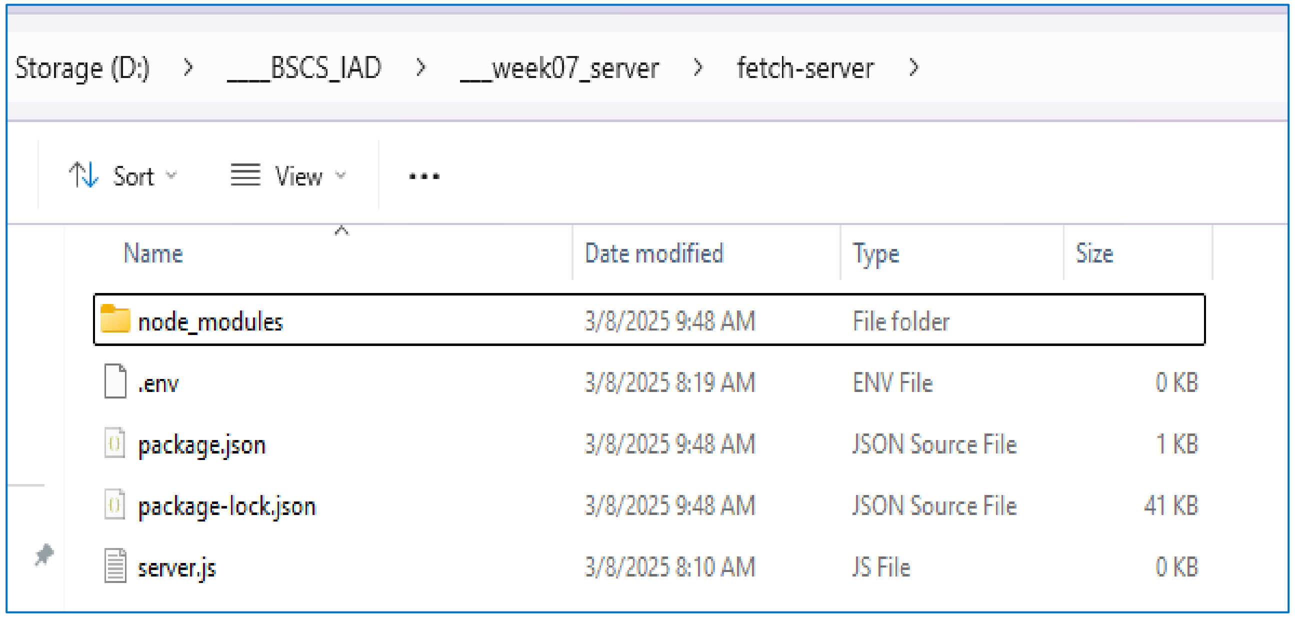
cors → Allows frontend to communicate with backend

dotenv → Helps manage environment variables.



ge (D:) > ___BSCS_IAD > __week07_server > fetch-server >

Name	Date modified	Type	Size
node_modules	3/8/2025 7:55 AM	File folder	
package.json	3/8/2025 7:55 AM	JSON Source File	1 KB
package-lock.json	3/8/2025 7:55 AM	JSON Source File	29 KB



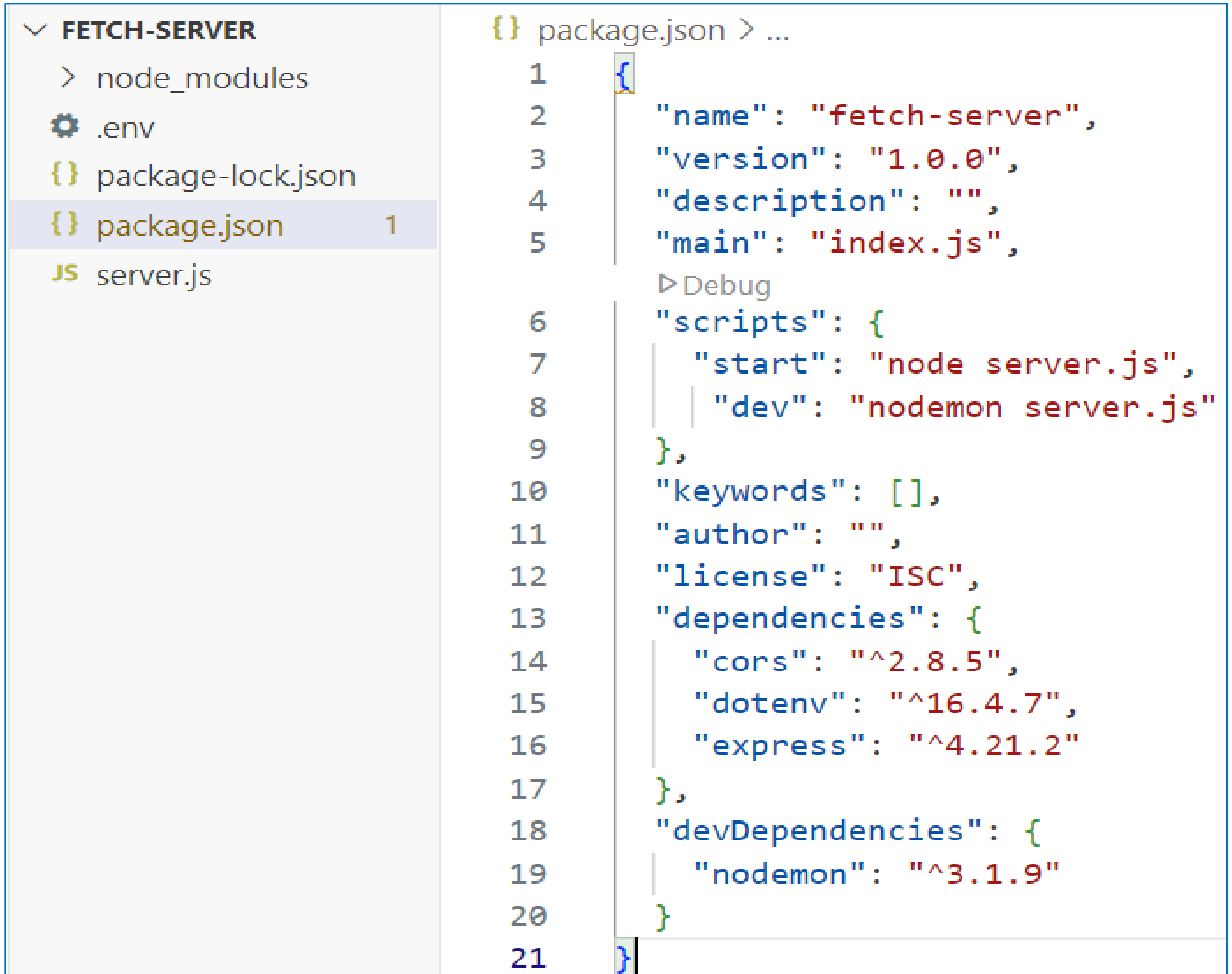
Storage (D:) > ___BSCS_IAD > __week07_server > fetch-server >

Name	Date modified	Type	Size
node_modules	3/8/2025 9:48 AM	File folder	
.env	3/8/2025 8:19 AM	ENV File	0 KB
package.json	3/8/2025 9:48 AM	JSON Source File	1 KB
package-lock.json	3/8/2025 9:48 AM	JSON Source File	41 KB
server.js	3/8/2025 8:10 AM	JS File	0 KB

```
touch server.js      # For macOS/Linux/bash shell
nul > server.js      # For Windows (Command Prompt)

touch .env           # macOS/Linux
nul > .env           # Windows
```

npm install --save-dev nodemon

Modify package.json for Automatic Reload



```json
{
  "name": "fetch-server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node server.js",
     "dev": "nodemon server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^16.4.7",
    "express": "^4.21.2"
  },
  "devDependencies": {
    "nodemon": "^3.1.9"
  }
}
```

The "main": "index.js" in package.json tells **Node.js** that the main entry file of your project is index.js

Change "index.js" to "server.js"

Developer Dependencies (for better development experience)

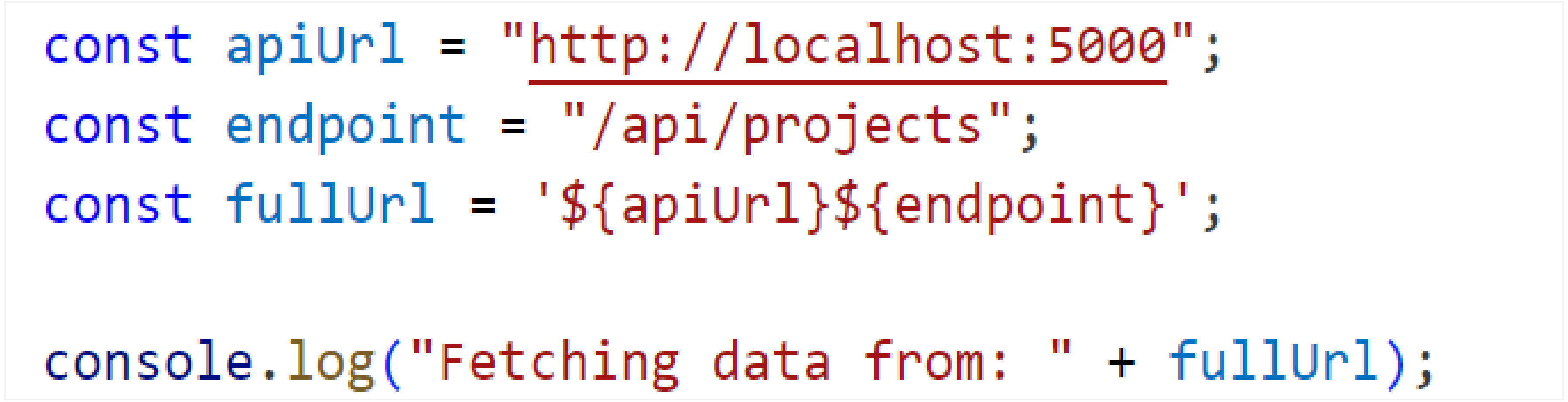
```
back_end> npm install --save-dev nodemon morgan eslint
```

- ✓nodemon → Automatically restarts the server on changes
- ✓morgan → Logs HTTP requests for debugging
- ✓eslint → Linting to enforce code quality

To be Run single file server.js

```
npm start

npm run dev
```

```
// Start the server
app.listen(PORT, () => console.log("Server running at http://localhost:${PORT}"));
```



```
const apiUrl = "http://localhost:5000";
const endpoint = "/api/projects";
const fullUrl = '${apiUrl}${endpoint}';

console.log("Fetching data from: " + fullUrl);
```

```
app.get("  ____??______  ", (req, res) => {

      res.json({ message:
              "Server is running! Welcome to the Capstone Project API." });
    });
```

```
app.get(“/api/projects", (req, res) =>
        res.json("Server is running! Welcome to the Capstone Project API." );
  );
```

Foundation for REST API

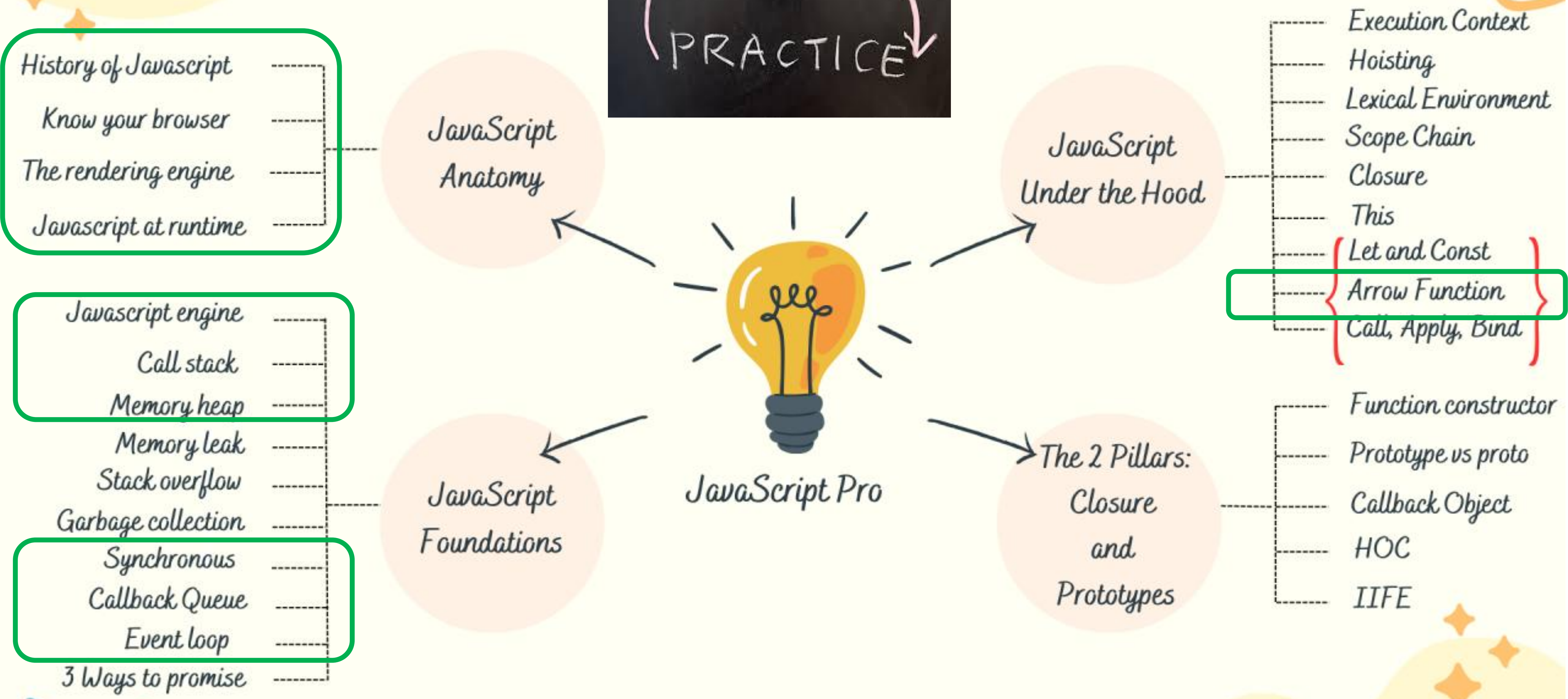
Exposes an API Endpoint → Clients can send a GET request to "/api/projects"

```
app.get("/api/projects", (req, res) => {
    res.json({ message:
                         "Server is running! Welcome to the Capstone Project API." });
                  });
```

To be Run single file server.js

```
npm start
```

```
npm run dev
```



Slimane Kadour

()=>{}

()=>{} as an Arrow Function

```
const add = (a, b) => { return a + b; };
```

()=>{} as a Callback (cb) Function

```
setTimeout(() => { console.log("Callback executed!"); }, 1000);
```

()=>{} as an Anonymous Function

UNIVERSITY OF KARACHI

Best choice depending on context? Why ?

OPTIONS

A. Arrow function **B.** call back function **C.** anonymous function

```
app.get("/api/projects", ____??________ );  // General structure
```

Why? → Because it is executed ________(when an ______________ is received).

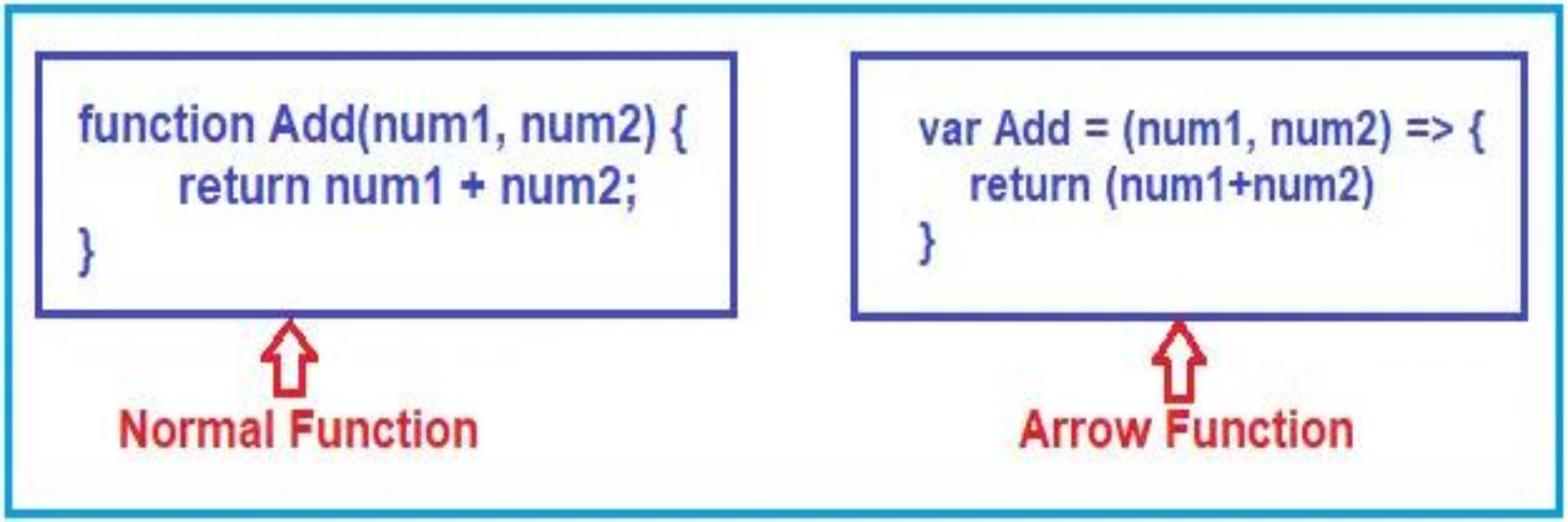
UNIVERSITY OF KARACHI

() => {}
ES6

The Arrow Functions in JavaScript helps us to create ________________ or methods i.e. functions without _________ As they do not have any names, the arrow makes the syntax ____________ .

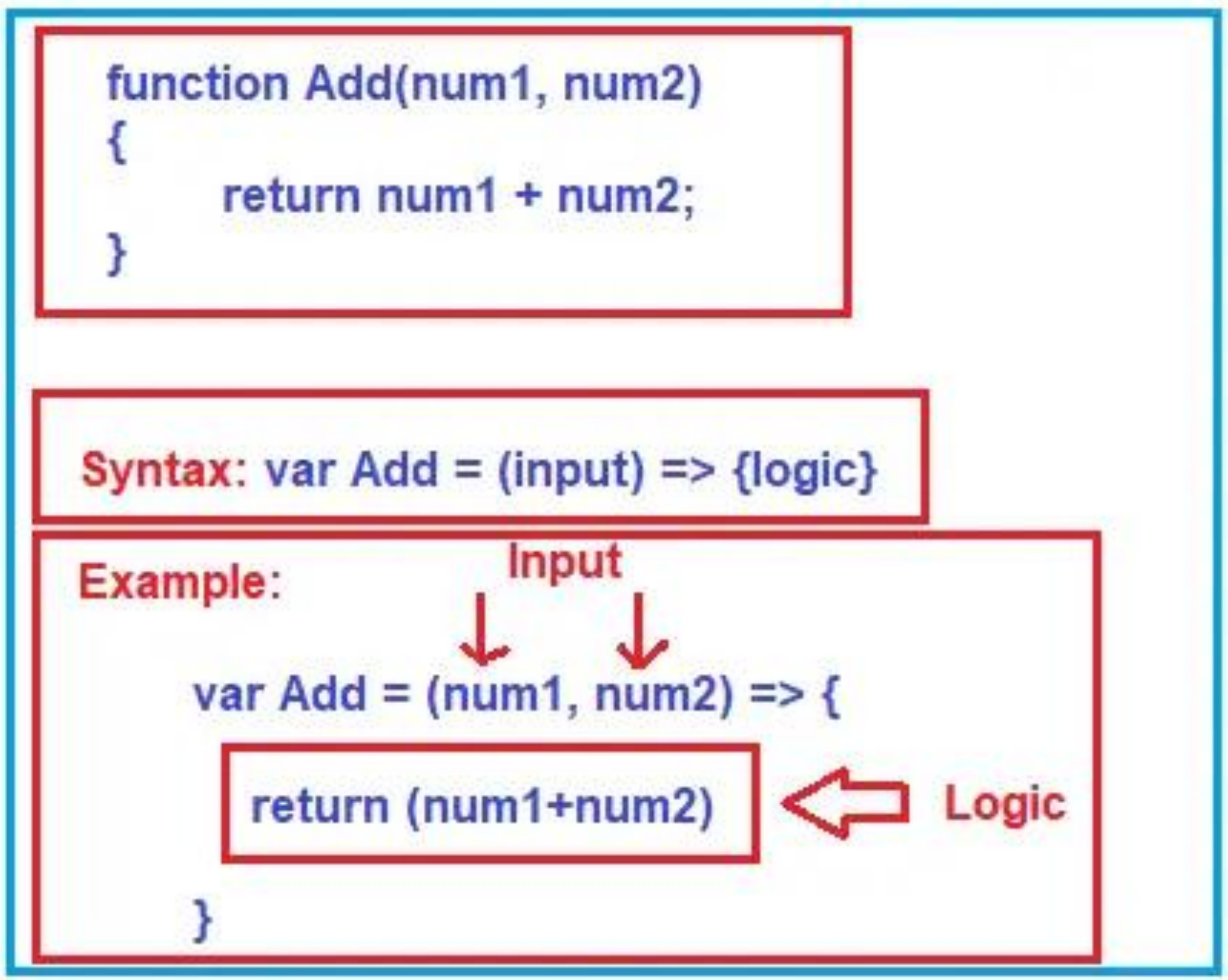
1. ()=>{} are a concise way of writing anonymous, lexically scoped functions in ES6.
2. The () =>{} can contain other ()=>{} or also __________ functions.
3. The () =>{} accomplishes the same result as regular function with fewer __________ .
4. The () =>{} automatically binds this object to the surrounding code's context.
5. The value of this keyword inside the () =>{} is not dependent on how they are called or how they are defined. It depends only on its enclosing context.
6. If the () =>{} is used as an __________(hint: inner/outer) this refers to the __________(hint: parent/global/surrounding)__ scope in which it is defined.

- ✓ Argument list () implies logic with in {}
- ✓ result of ()=>{} from R.H.S can be assigned to variable on L.H.S



```
function Add(num1, num2) {
        return num1 + num2;
}
```

```
var Add = (num1, num2) => {
    return (num1+num2)
}
```



```
function Add(num1, num2)
{
        return num1 + num2;
}
```

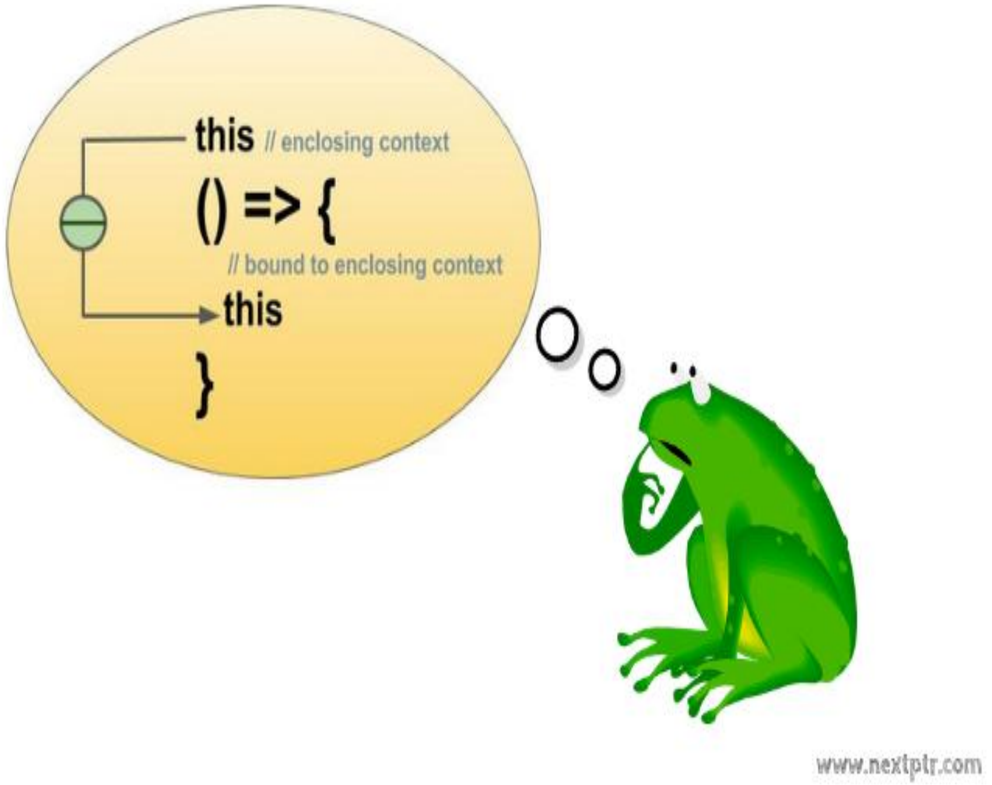
```
Syntax: var Add = (input) => {logic}
```

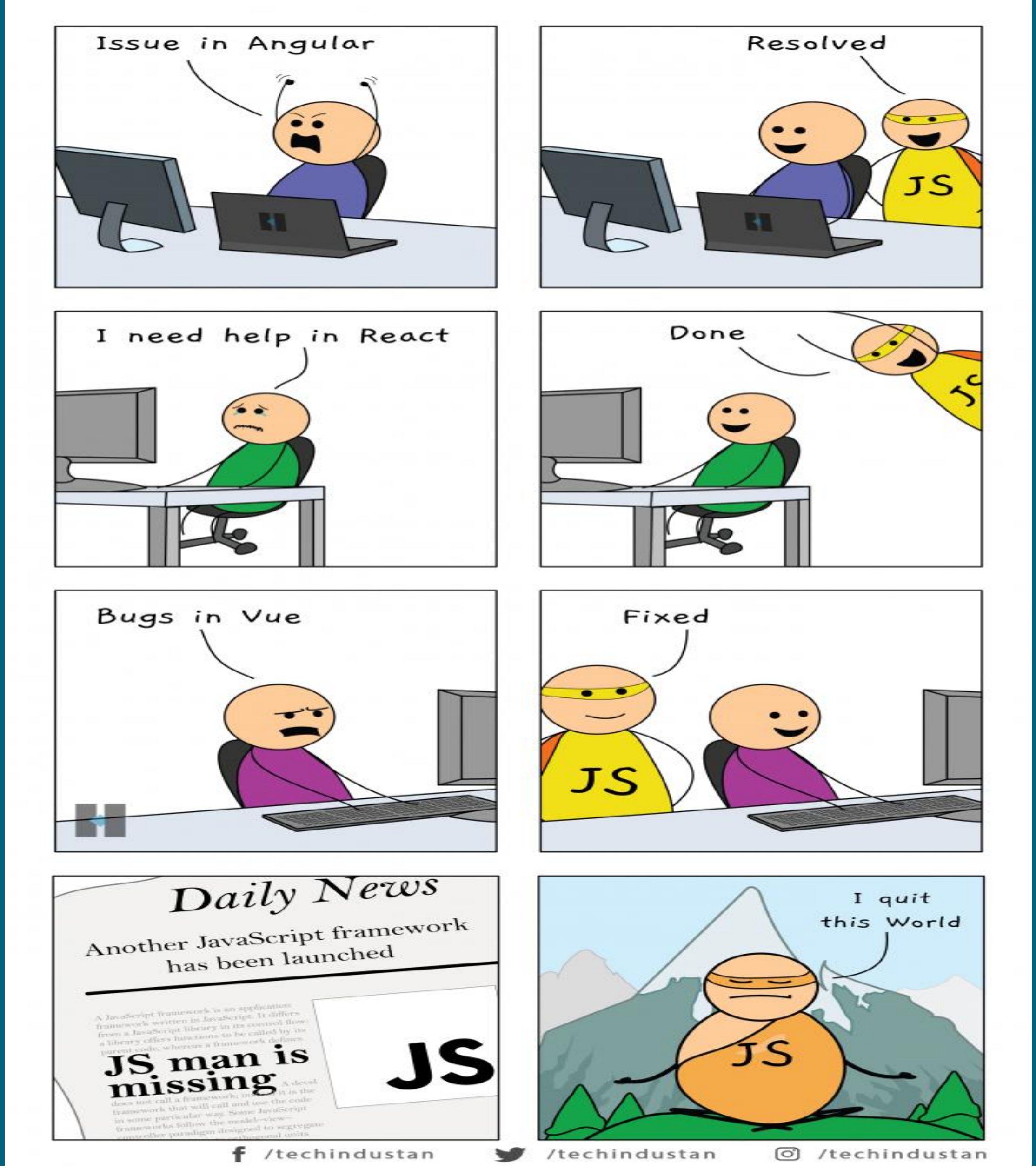
Example:

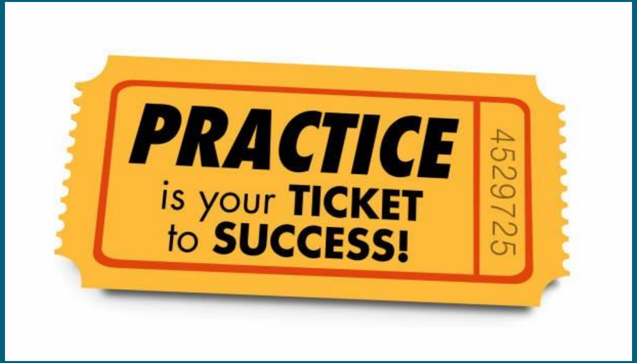
```
      var Add = (num1, num2) => {
          return (num1+num2)
      }
```

Input
Logic

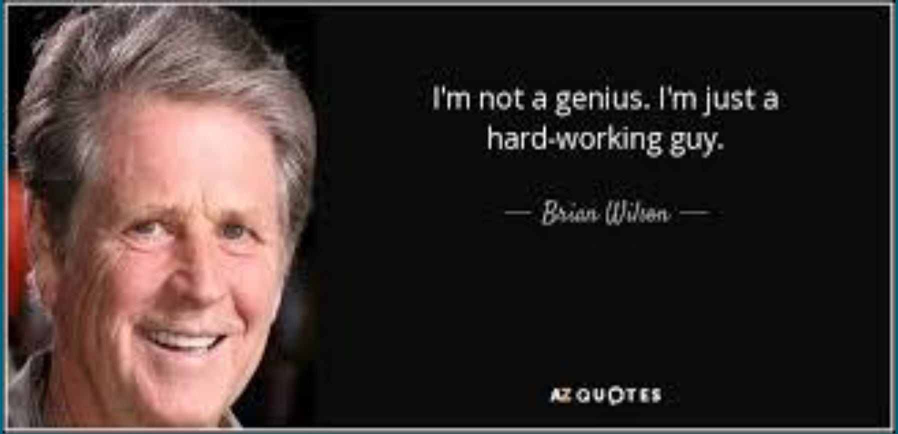
Comment on discord what frog is thinking in his head/bubble



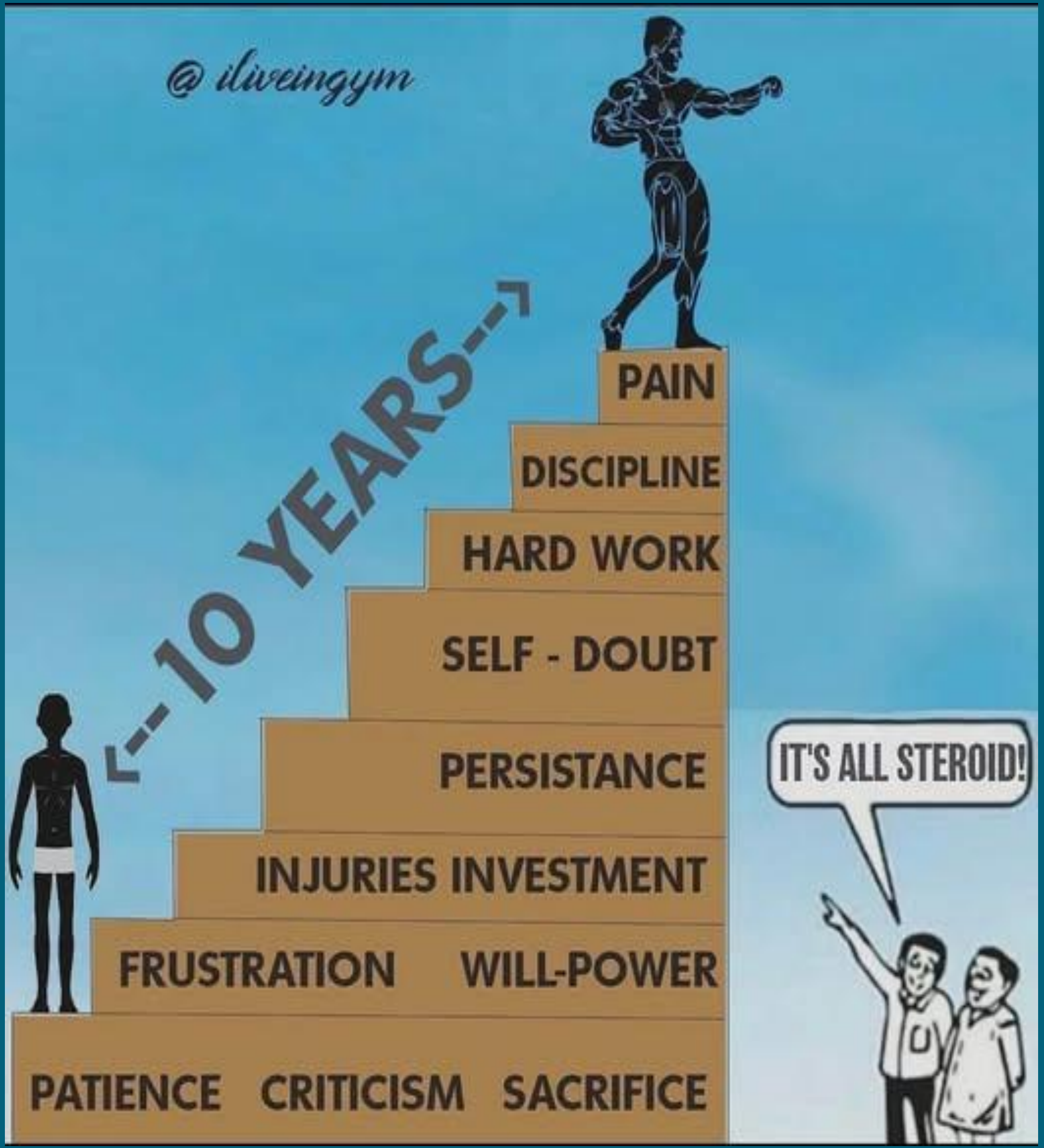




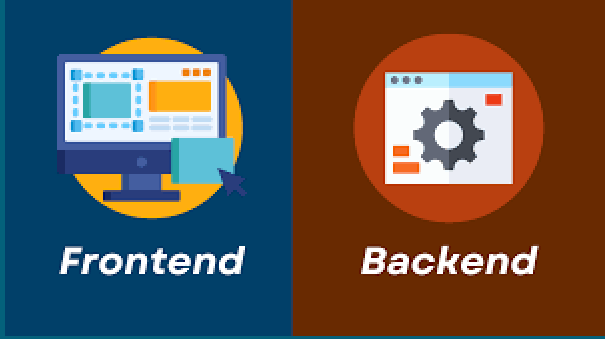
Java script Practice Time


I'm not a genius. I'm just a hard-working guy.

— Brian Wilson —



Java script practice time



The **primary use of arrow functions in the frontend** is to attach functionality to UI interactions, such as

______________ , ______________ , and

______________ .

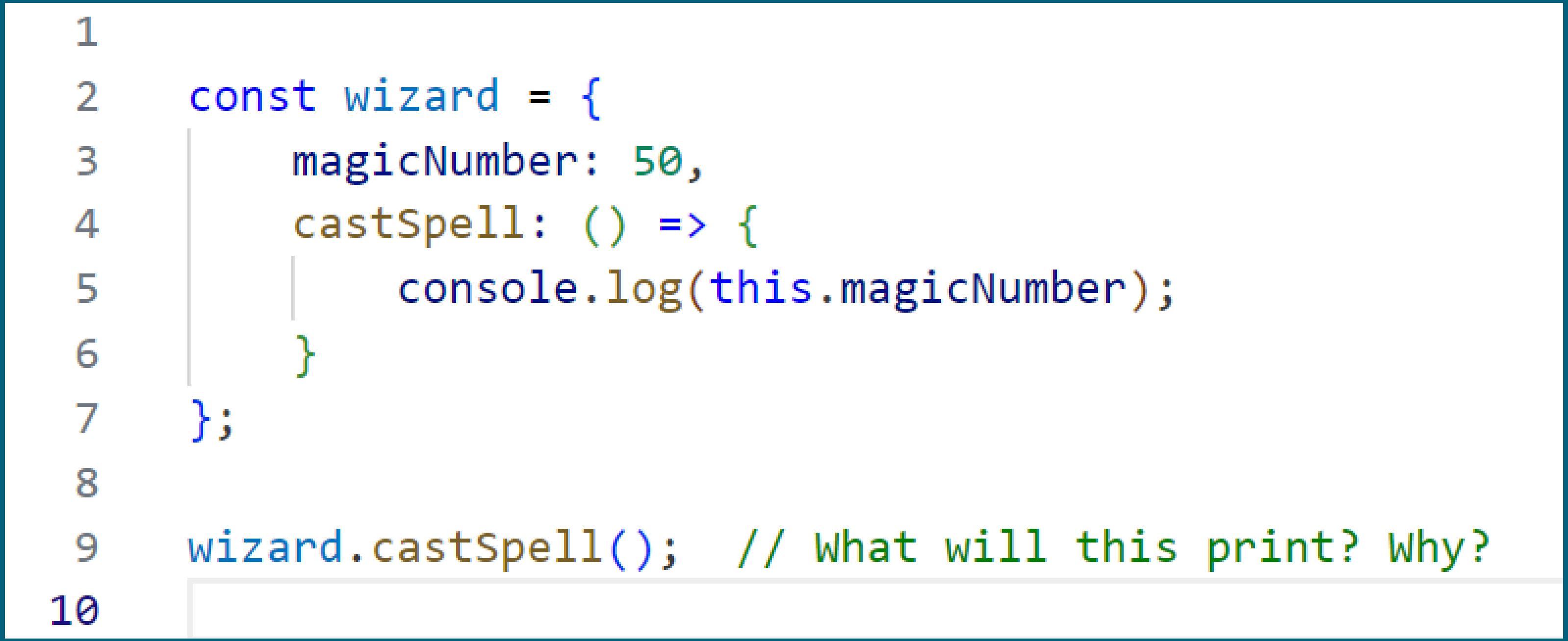
However, that's not the only use case.

Transform regular function to Arrow functions (Individual)

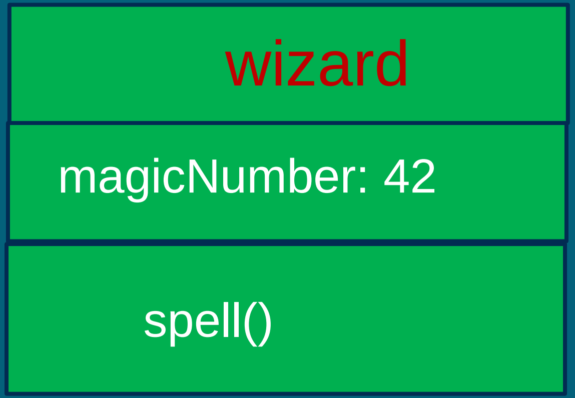
- ✓ Share and explain about 3 trickiest possible code snippet (plus .js/.jsx files) with practical use case relevant to this course

- ✓ Write original hand-written brief argument about it, put it in individual folder. Ask me on discord group to have a look by Tomorrow .

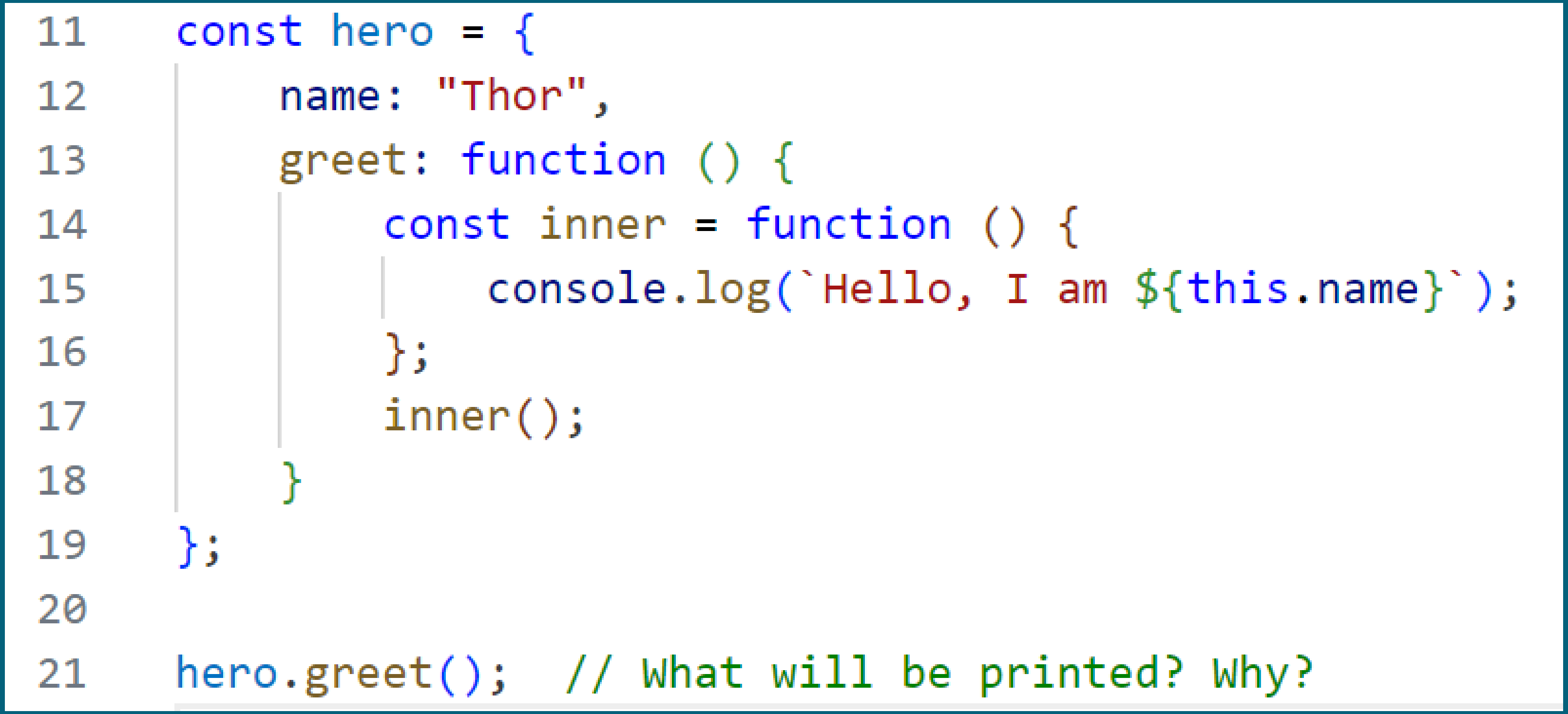
What is the Output?



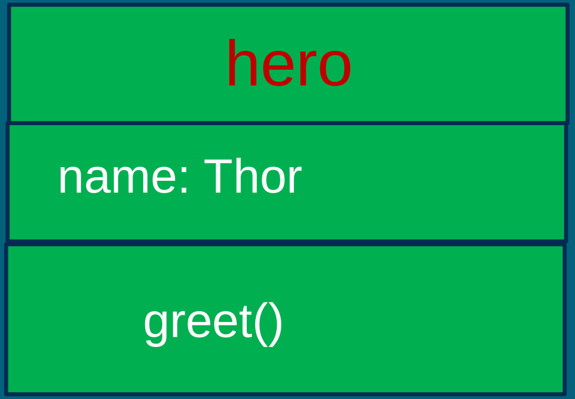
```
const wizard = {
    magicNumber: 50,
    castSpell: () => {
        console.log(this.magicNumber);
    }
};

wizard.castSpell();  // What will this print? Why?
```



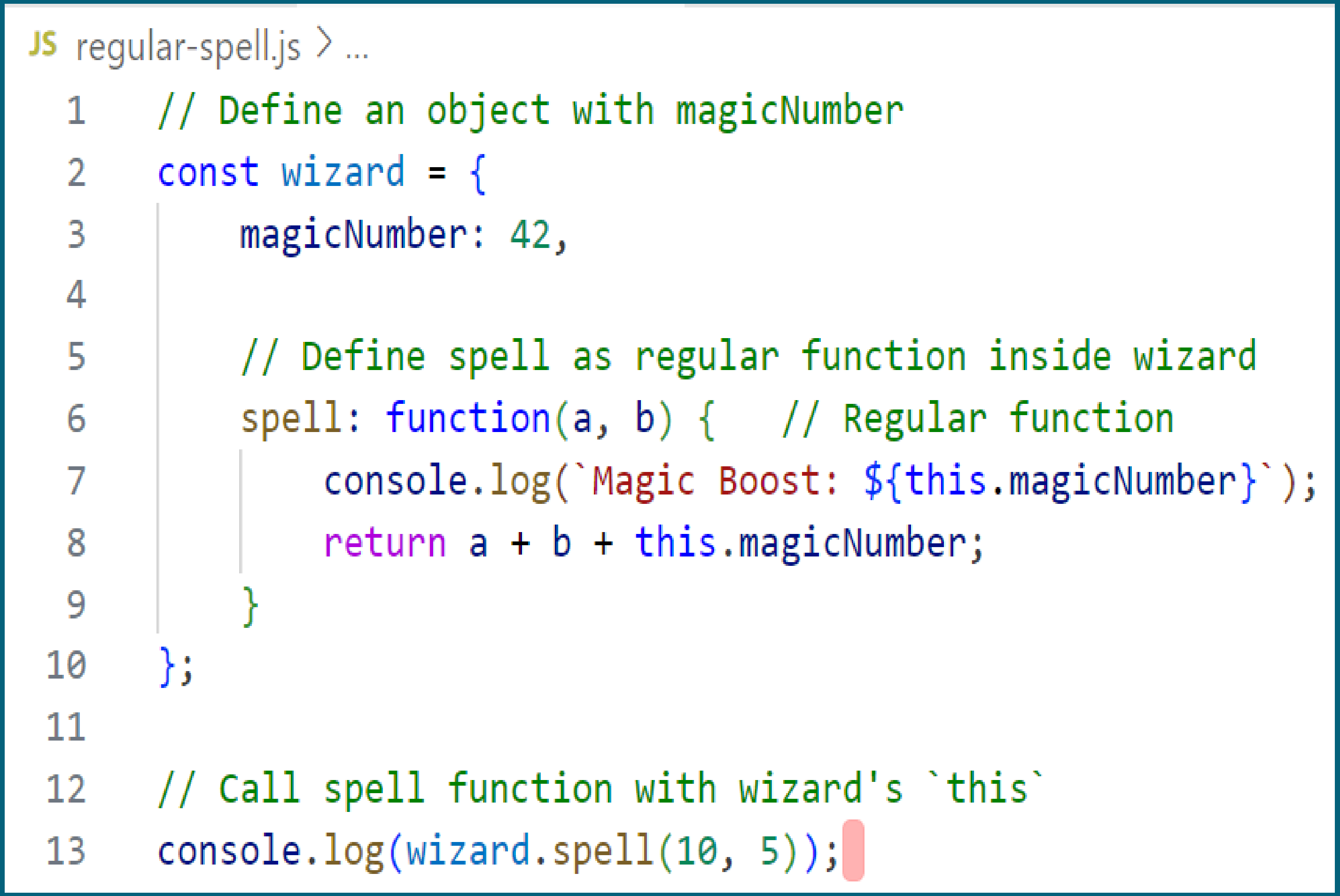
Practice time



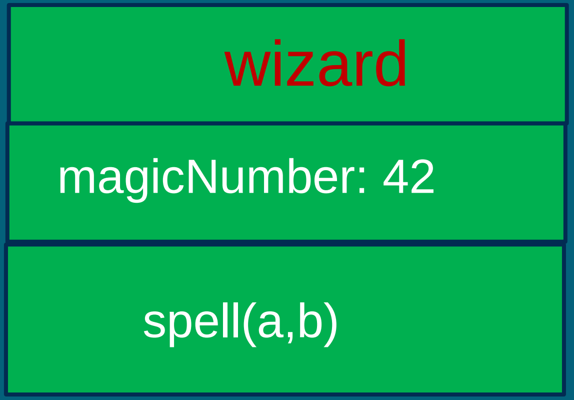
```
const hero = {
    name: "Thor",
    greet: function () {
        const inner = function () {
            console.log(`Hello, I am ${this.name}`);
        };
        inner();
    }
};

hero.greet();  // What will be printed? Why?
```



Practice time


regular-spell.js > ...

```javascript
// Define an object with magicNumber
const wizard = {
    magicNumber: 42,

    // Define spell as regular function inside wizard
    spell: function(a, b) {   // Regular function
        console.log(`Magic Boost: ${this.magicNumber}`);
        return a + b + this.magicNumber;
    }
};

// Call spell function with wizard's `this`
console.log(wizard.spell(10, 5));
```



Practice time

Feature	**Regular Function (wizard.spell)**	**Arrow Function (mathWizard.add)**
this behavior	Dynamic (this depends on how it's called)	Lexical (this is inherited from the surrounding function)
Needs .bind(this) / .call(this) / .apply(this)??	Sometimes, especially if passed as a callback	No, this is automatically inherited
Works when used as an event handler?	No, unless .bind(this) is used	Yes, inherits the correct this

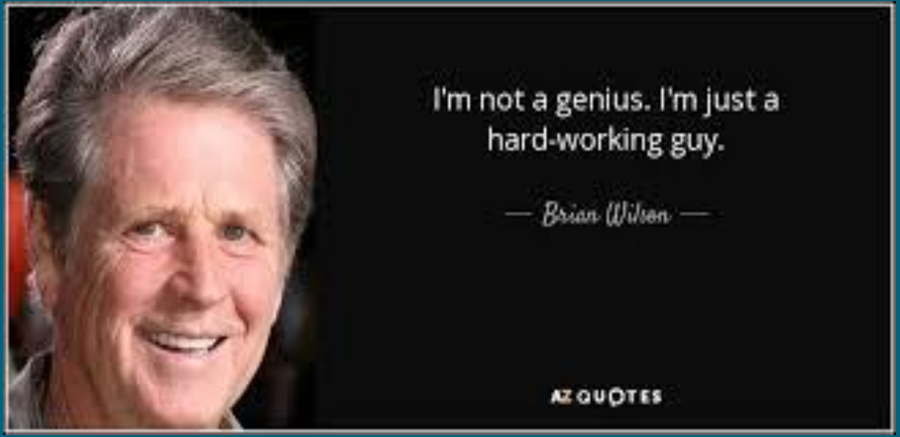
Practice time

JS Practice : Inheritance & this Demo (Individual)

You are encouraged to designand share your own problem along with demo

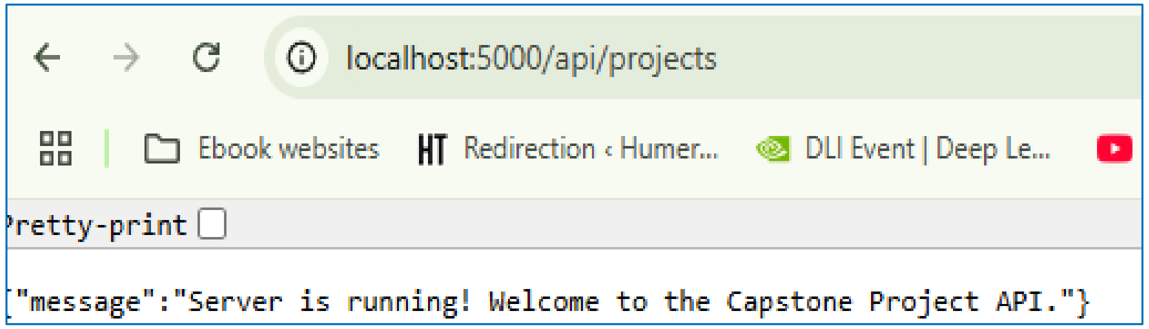
Create a **Wizard** class with:
- magicNumber property.
- spell() (regular function) → logs magicNumber.
- castSpell() (arrow function) → logs magicNumber.

- Create a **DarkWizard** class (inherits from Wizard).
- Overrides spell() with a modified magicNumber.
- **Frontend:**
  - Two buttons: "Regular Spell" & "Arrow Spell".
  - Clicking triggers respective methods.
- **Event Handling:**
  - spell() needs .bind(this).
  - castSpell() works without .bind(this)

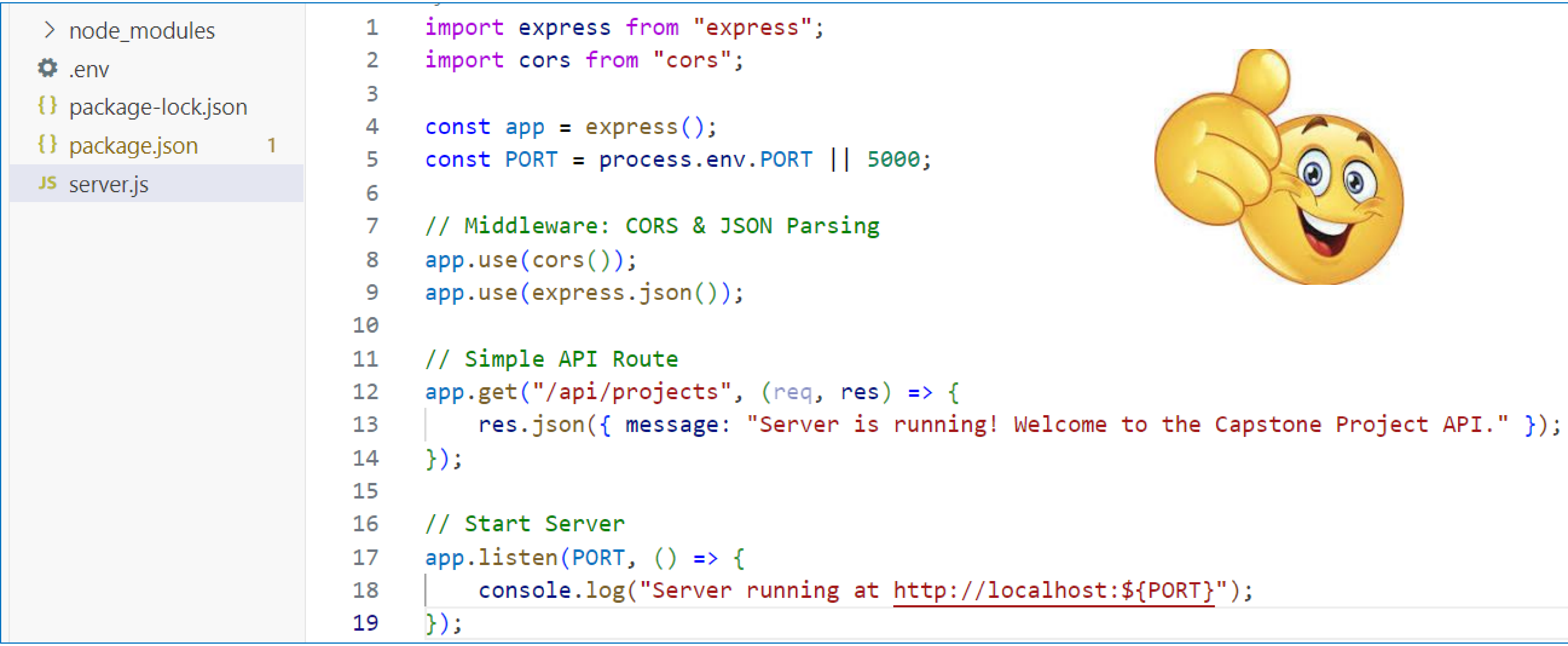


Practice time

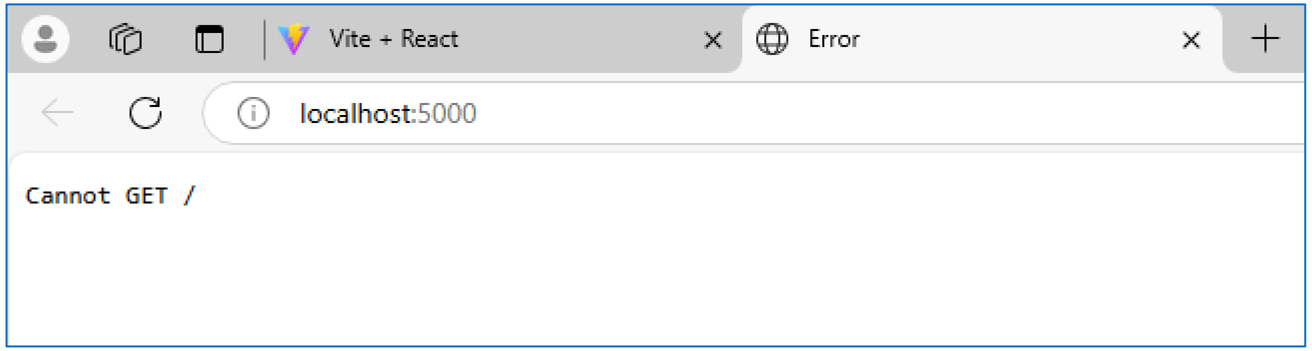
```
npm start

npm run dev
```



```
{"message":"Server is running! Welcome to the Capstone Project API."}
```

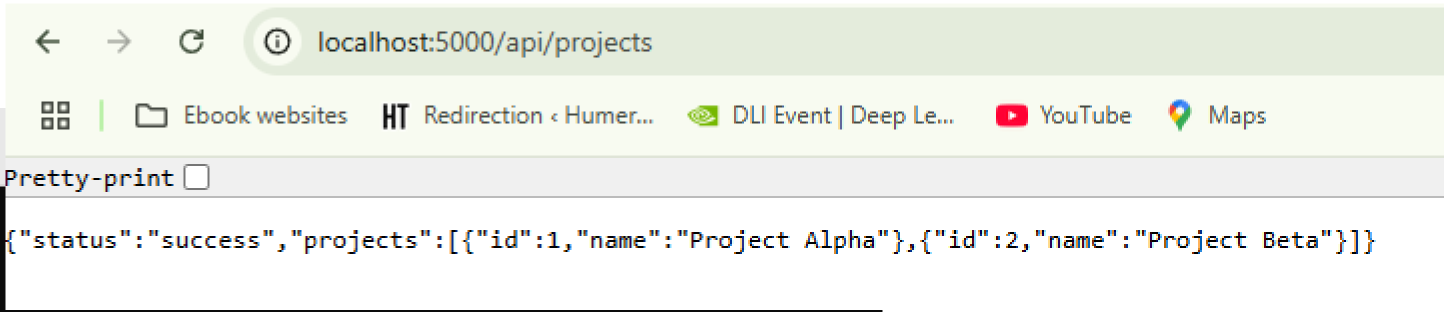


```javascript
import express from "express";
import cors from "cors";

const app = express();
const PORT = process.env.PORT || 5000;

// Middleware: CORS & JSON Parsing
app.use(cors());
app.use(express.json());

// Simple API Route
app.get("/api/projects", (req, res) => {
    res.json({ message: "Server is running! Welcome to the Capstone Project API." });
});

// Start Server
app.listen(PORT, () => {
    console.log("Server running at http://localhost:${PORT}");
});
```

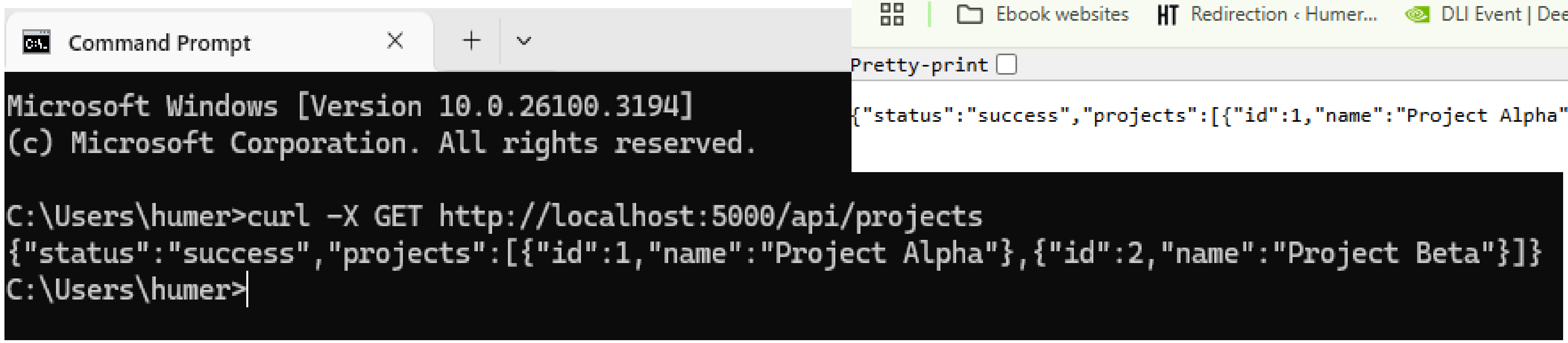


```
Cannot GET /
```

What have I done wrong and how to improve server.js to avoid this ??



```
{"status":"success","projects":[{"id":1,"name":"Project Alpha"},{"id":2,"name":"Project Beta"}]}
```



```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\Users\humer>curl -X GET http://localhost:5000/api/projects
{"status":"success","projects":[{"id":1,"name":"Project Alpha"},{"id":2,"name":"Project Beta"}]}
C:\Users\humer>
```

"Cannot GET /"

The error "Cannot GET /" happens because your **server does not define a route for the ______________ .**

```
app.get( _______??_______ , (req, res) => {
    res.send("Welcome to the Projects API! Use /api/projects to fetch data.");
});
```

To be Run single file server.js

```
npm start

npm run dev
```

The **Request-Response Cycle** in action:

- ✓ The **client (browser/Postman/frontend app)** sends a **request**.
- ✓ The **server (Express backend) processes** it and sends a **response**.

•Base URL + Endpoint = Full API Route

http://localhost:5000 /api/projects http://localhost:5000/api/projects

```
(req, res) => {}
```

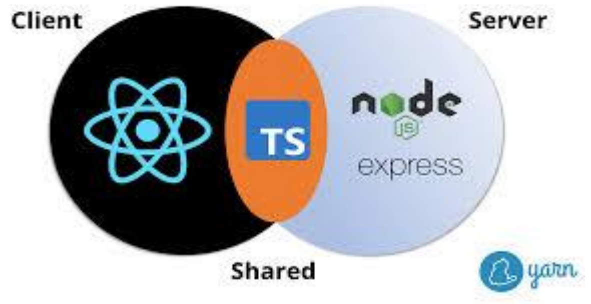
This is the ______________ (also called a request handler) that gets

executed when a ___________ , such as a frontend app making a request

to “____________??____________/api/projects", _hits the API.

Hint:(base url,endpoint, full url)

Say a React frontend makes a request like this:



```
fetch() (“ ____________________ “)
    .then( _________ =>__________)  // is a cb
    .then( _________ =>__________) // is a cb
```

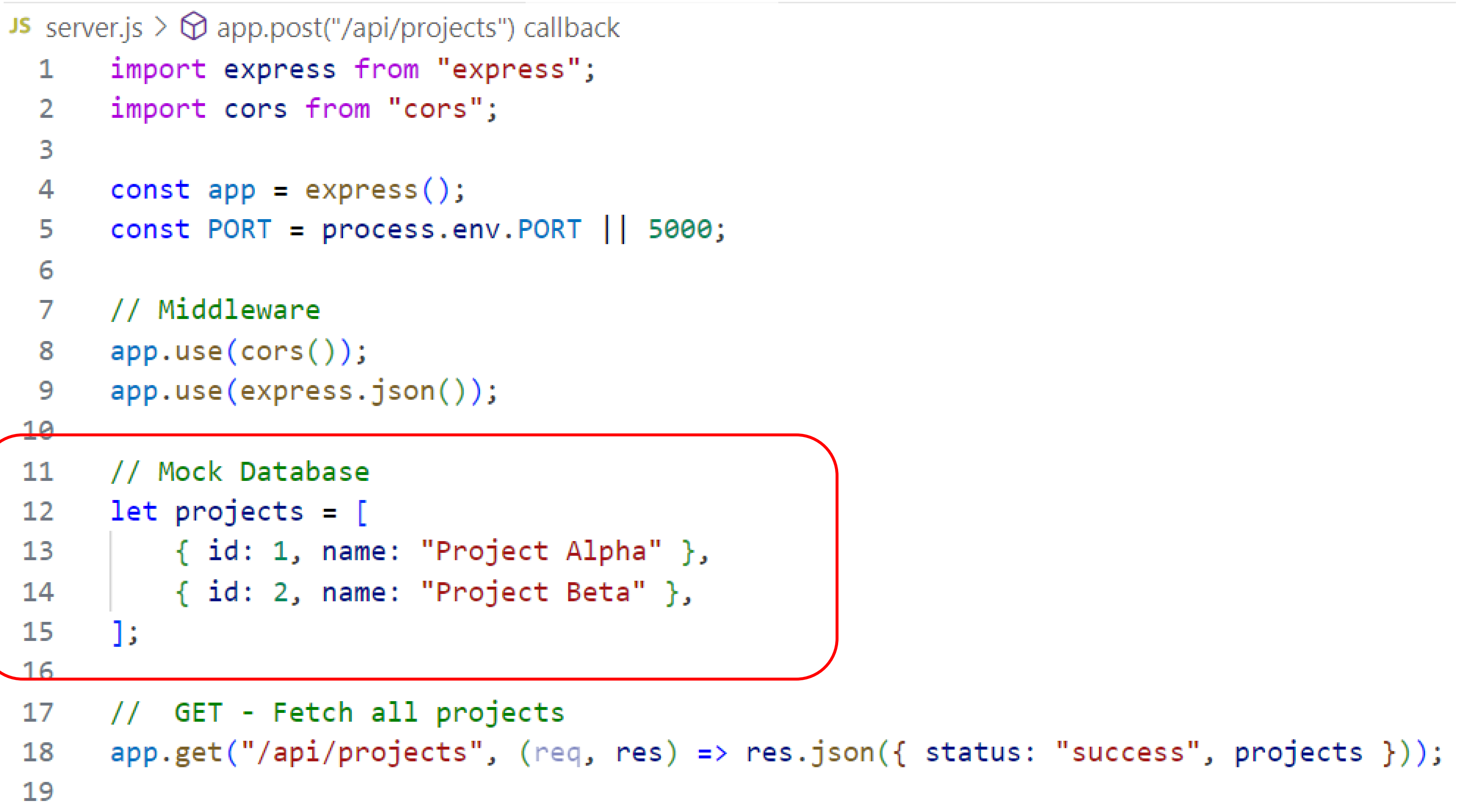
Then, when the request reaches the backend, the callback function (request handler) in

app.get("/api/projects", (req, res) => {...}) executes and returns data.

How Do They All Connect?

API	Any interface allowing communication between software (backend & frontend).
RESTful API	A type of API that follows REST rules (uses HTTP methods like GET, POST, DELETE).
app.get() in Express	A method that defines an API route that handles HTTP GET requests.
HTTP GET	A request sent by a client to retrieve data from a server.
Base URL	The main API address (http://localhost:5000).
Endpoint	A specific path (/api/projects).
Request	The client's call to the backend (e.g., GET /api/projects).
Response	The data sent back by the server ({ message: "Server is running! Welcome to the Capstone Project API." }

The projects array in server acts as a Mock database



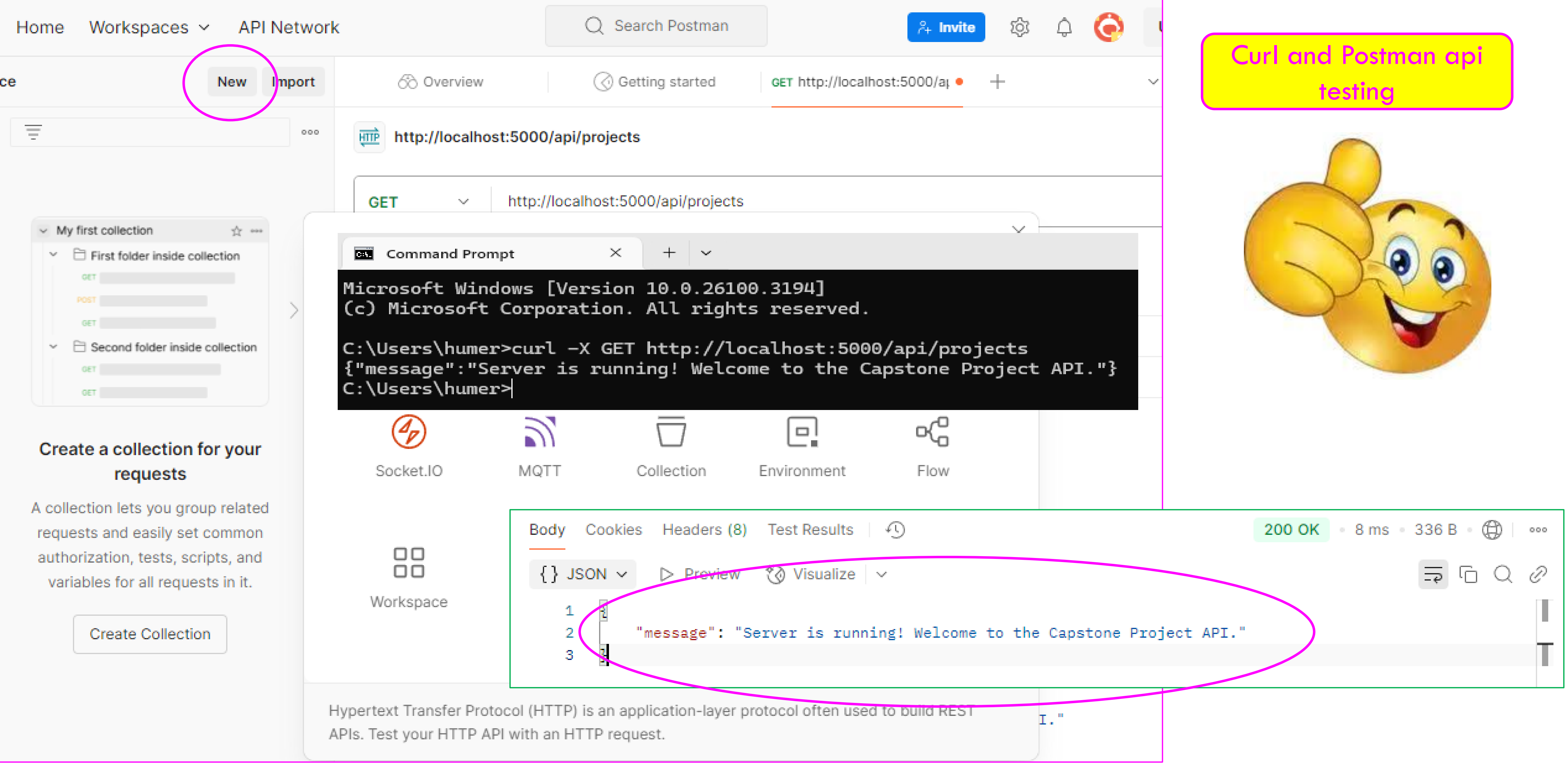
server.js > app.post("/api/projects") callback

```
import express from "express";
import cors from "cors";

const app = express();
const PORT = process.env.PORT || 5000;

// Middleware
app.use(cors());
app.use(express.json());

// Mock Database
let projects = [
    { id: 1, name: "Project Alpha" },
    { id: 2, name: "Project Beta" },
];

//  GET - Fetch all projects
app.get("/api/projects", (req, res) => res.json({ status: "success", projects }));

```

Request-Response Cycle (HTTP Communication)

Single file server.js

Test API
in
Postman



http://localhost:5000/api/projects

GET http://localhost:5000/api/projects

```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\Users\humer>curl -X GET http://localhost:5000/api/projects
{"message":"Server is running! Welcome to the Capstone Project API."}
C:\Users\humer>
```

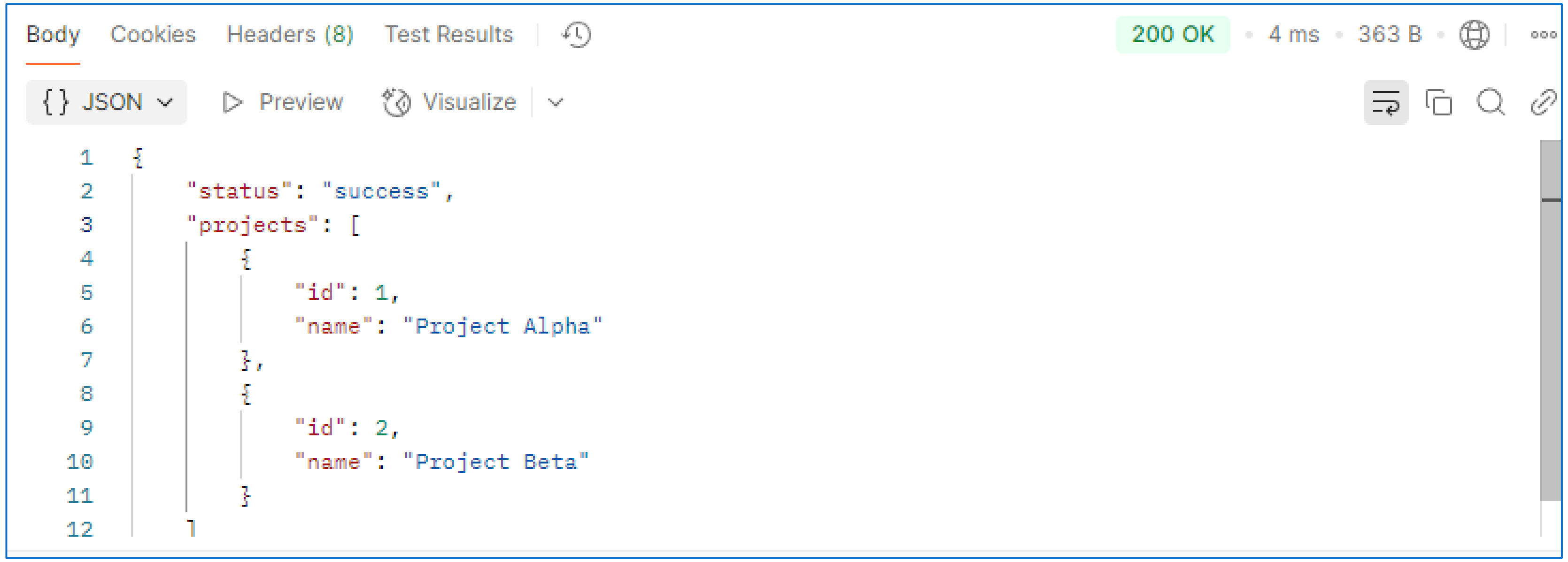
```
{
    "message": "Server is running! Welcome to the Capstone Project API."
}
```

Create a collection for your requests

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

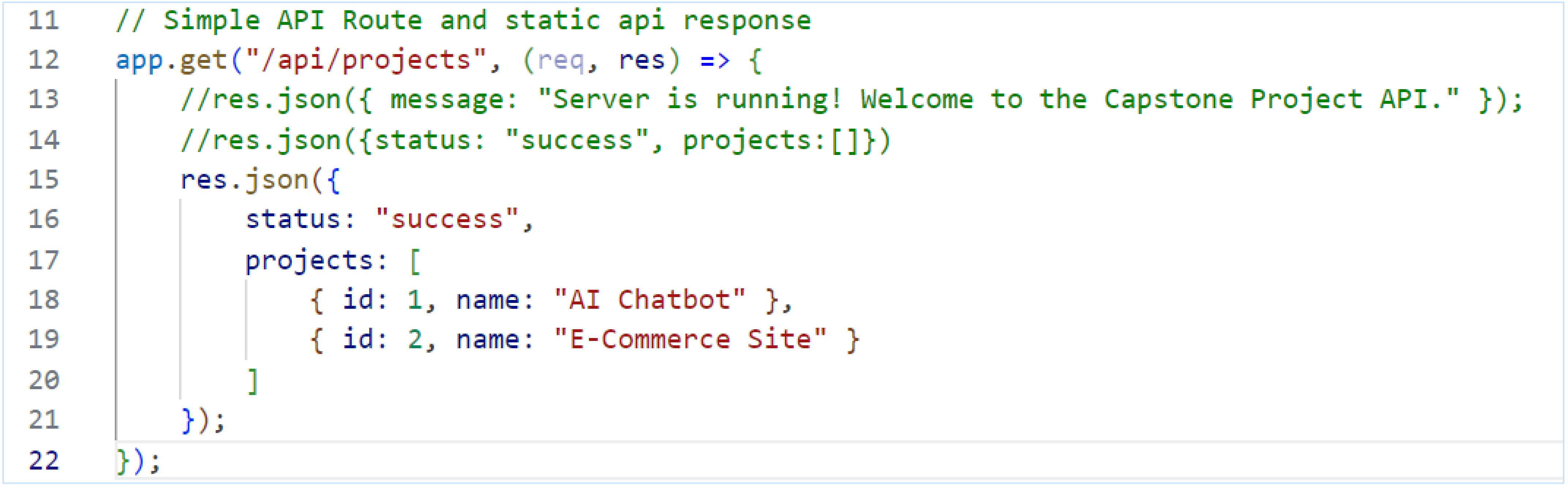
Hypertext Transfer Protocol (HTTP) is an application-layer protocol often used to build REST APIs. Test your HTTP API with an HTTP request.

Curl and Postman api testing



```
{
    "status": "success",
    "projects": [
        {
            "id": 1,
            "name": "Project Alpha"
        },
        {
            "id": 2,
            "name": "Project Beta"
        }
    ]
```

Will Modifying Response Structure Will affect Front End Design?



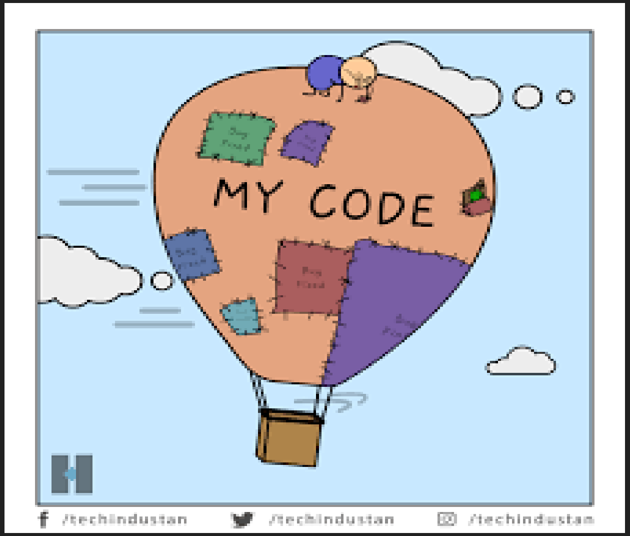
```
// Simple API Route and static api response
app.get("/api/projects", (req, res) => {
    //res.json({ message: "Server is running! Welcome to the Capstone Project API." });
    //res.json({status: "success", projects:[]})
    res.json({
        status: "success",
        projects: [
            { id: 1, name: "AI Chatbot" },
            { id: 2, name: "E-Commerce Site" }
        ]
    });
});
```

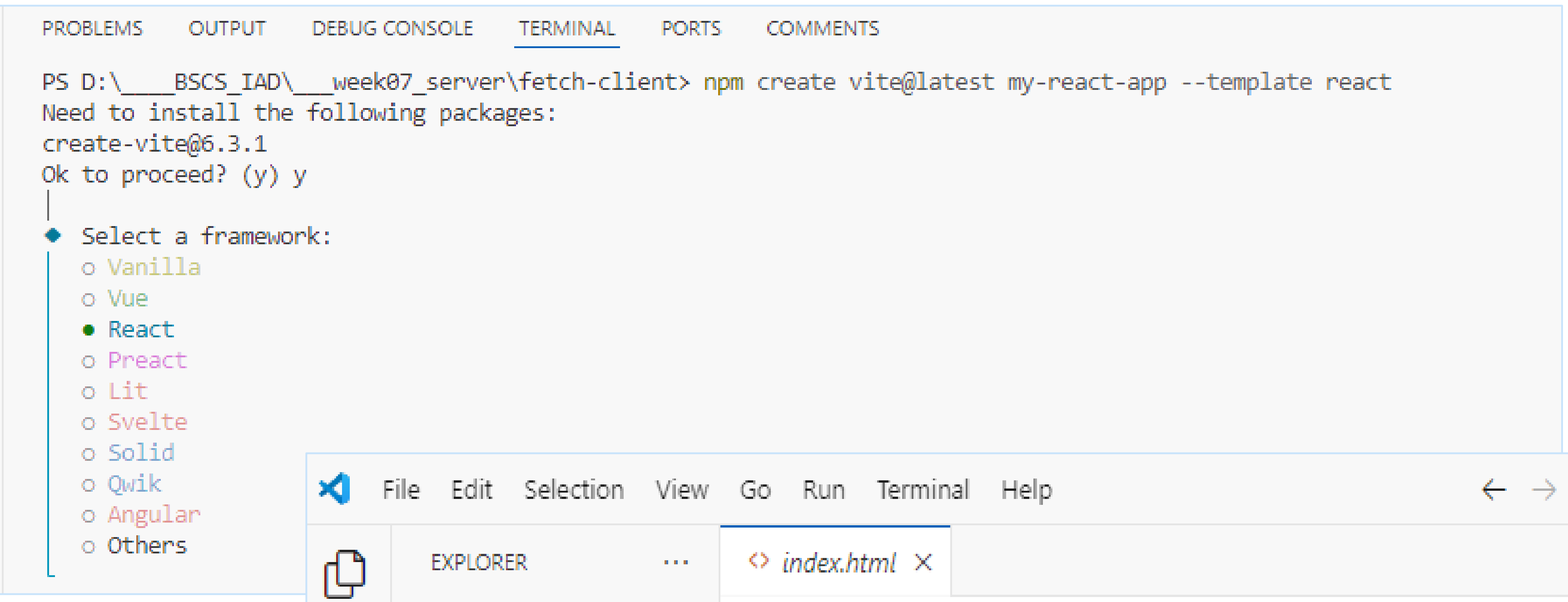
```
Server running at http://localhost:${PORT}
[nodemon] restarting due to changes...
[nodemon] starting `node server.js`
```

The endpoints are ready
Flex
useState
align-items
<div>
<span>
!important
Center
<Butt
useEffect
Time to work on the Frontend

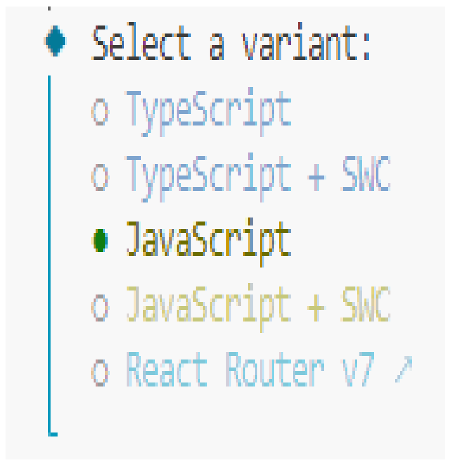
```
Server running at http://localhost:${PORT}
[nodemon] restarting due to changes...
[nodemon] starting `node server.js`
```

Implement Fetch API Call in React

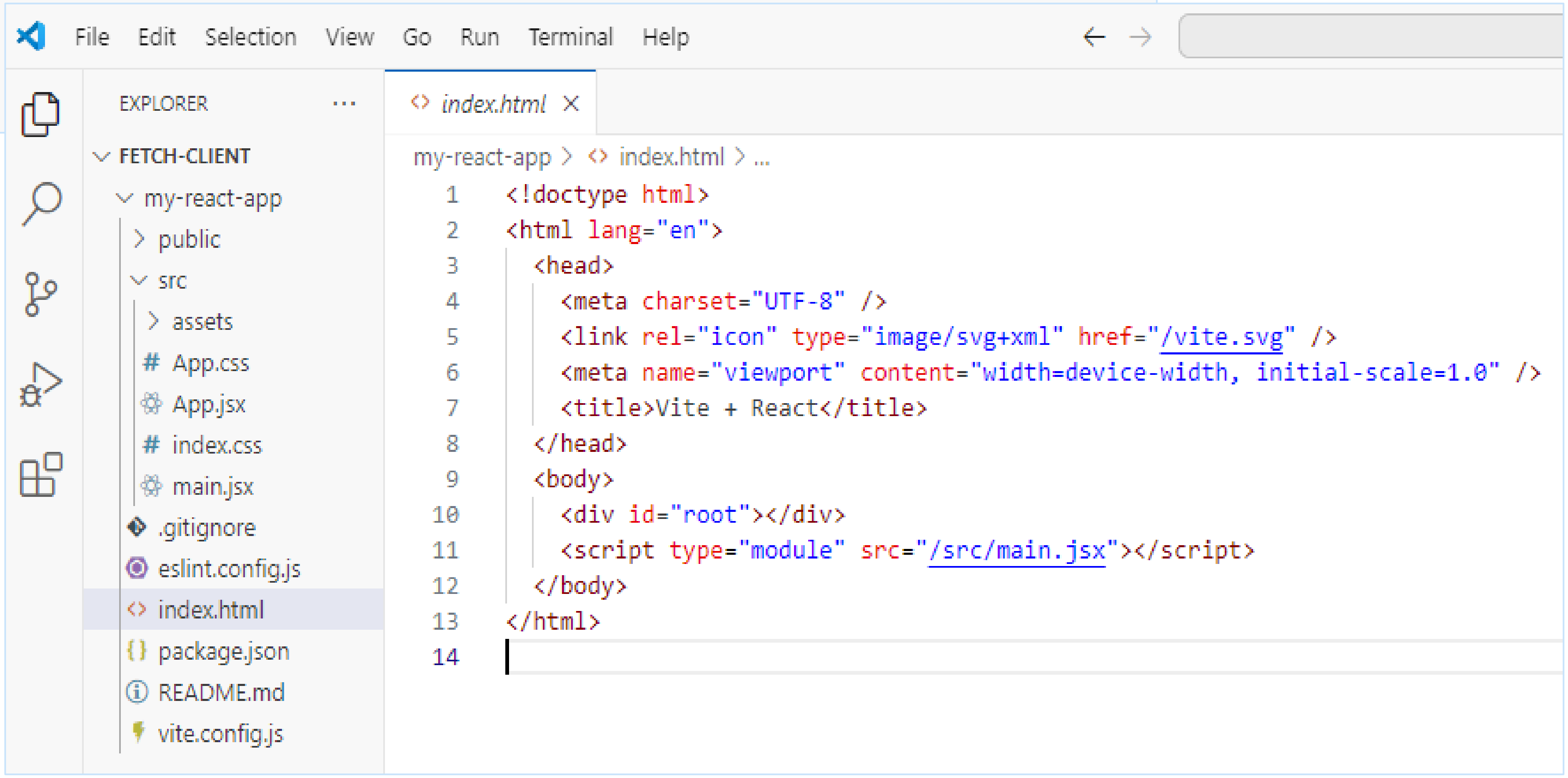




```
PS D:\____BSCS_IAD\___week07_server\fetch-client> npm create vite@latest my-react-app --template react
Need to install the following packages:
create-vite@6.3.1
Ok to proceed? (y) y
|
◆ Select a framework:
|  ○ Vanilla
|  ○ Vue
|  ● React
|  ○ Preact
|  ○ Lit
|  ○ Svelte
|  ○ Solid
|  ○ Qwik
|  ○ Angular
|  ○ Others
```

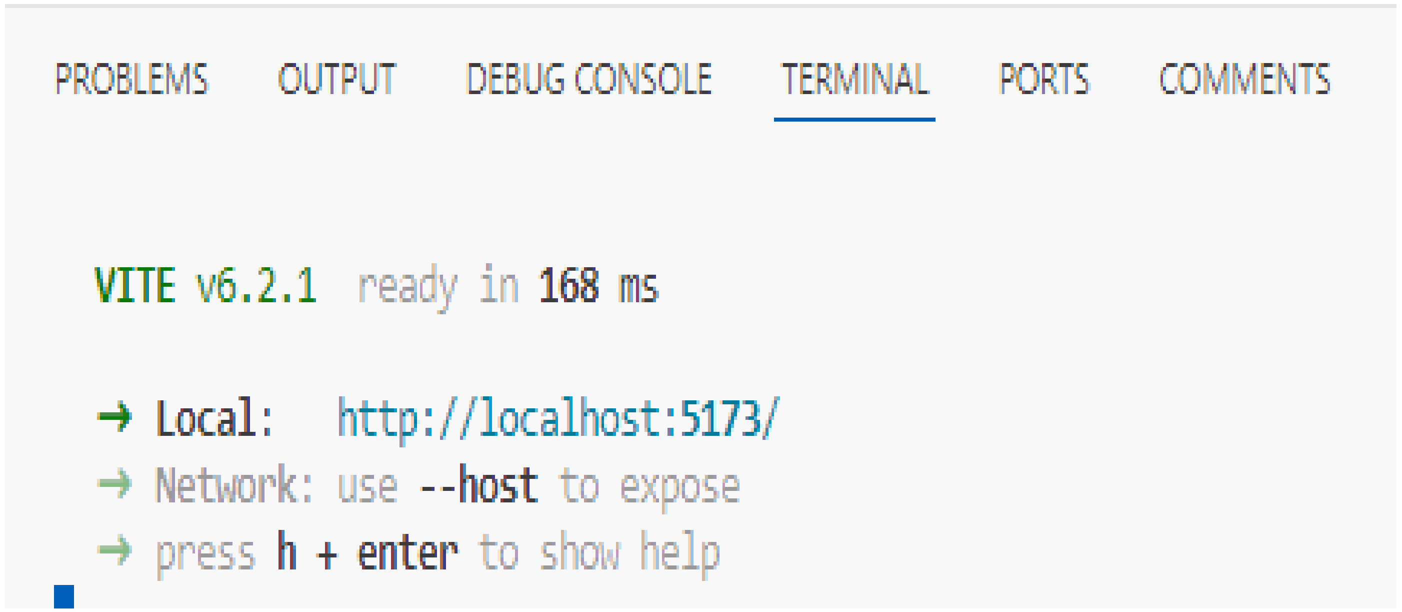


```
◆ Select a variant:
|  ○ TypeScript
|  ○ TypeScript + SWC
|  ● JavaScript
|  ○ JavaScript + SWC
|  ○ React Router v7 ↗
```

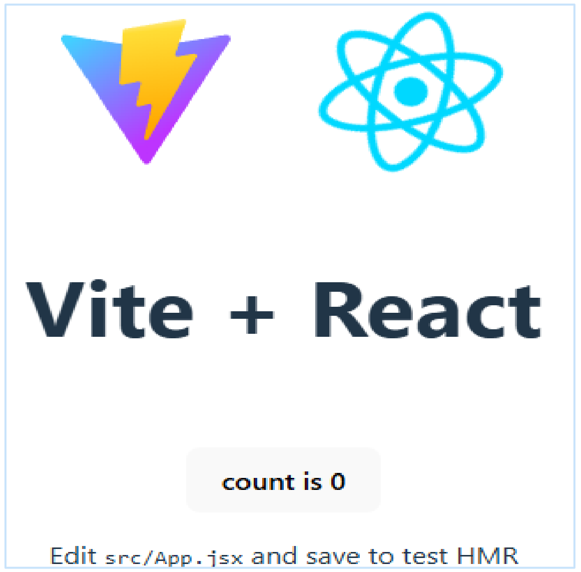


```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>

```

```
PS D:\BSCS_IAD\week07_server\fetch-client> cd my-react-app
PS D:\BSCS_IAD\week07_server\fetch-client\my-react-app> npm run dev
PS D:\BSCS_IAD\week07_server\fetch-client\my-react-app> npm install
```



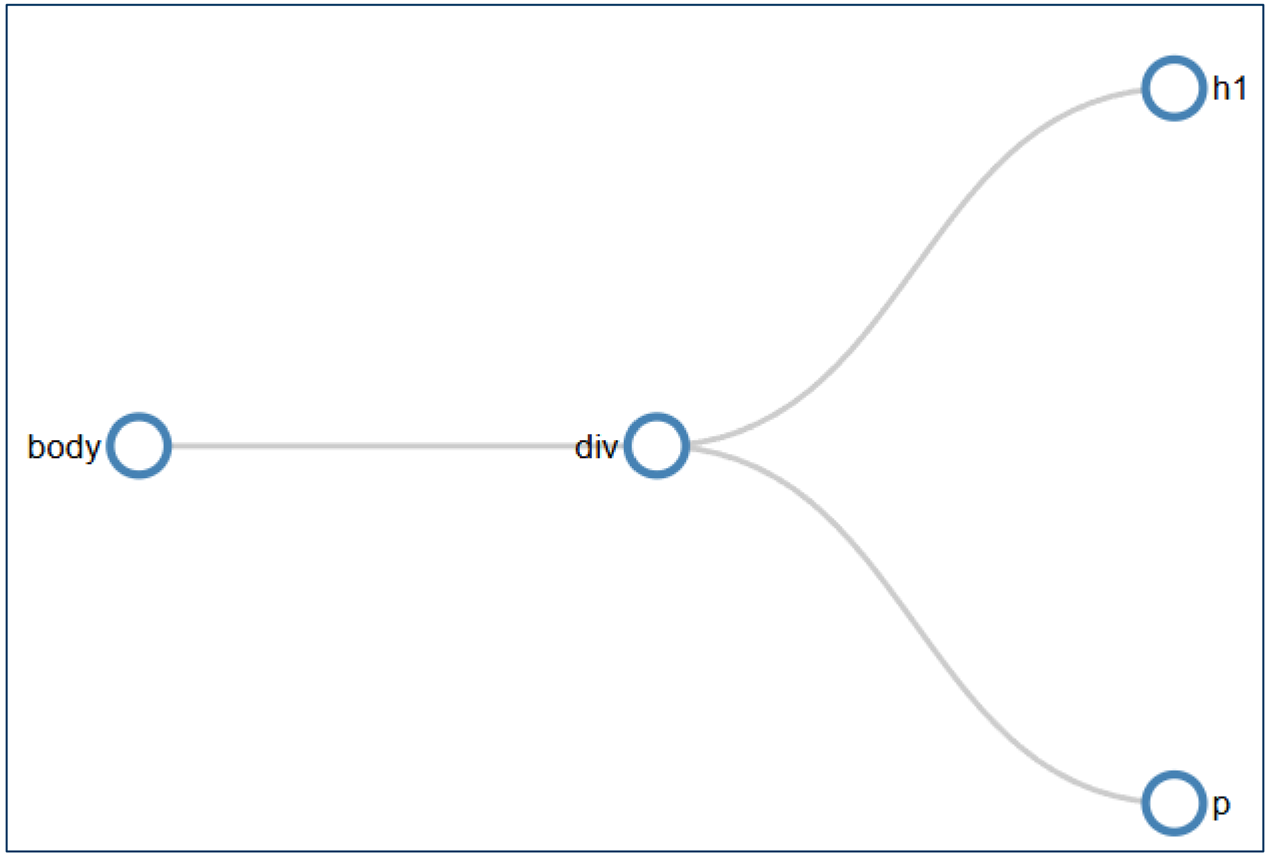
```
  VITE v6.2.1  ready in 168 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

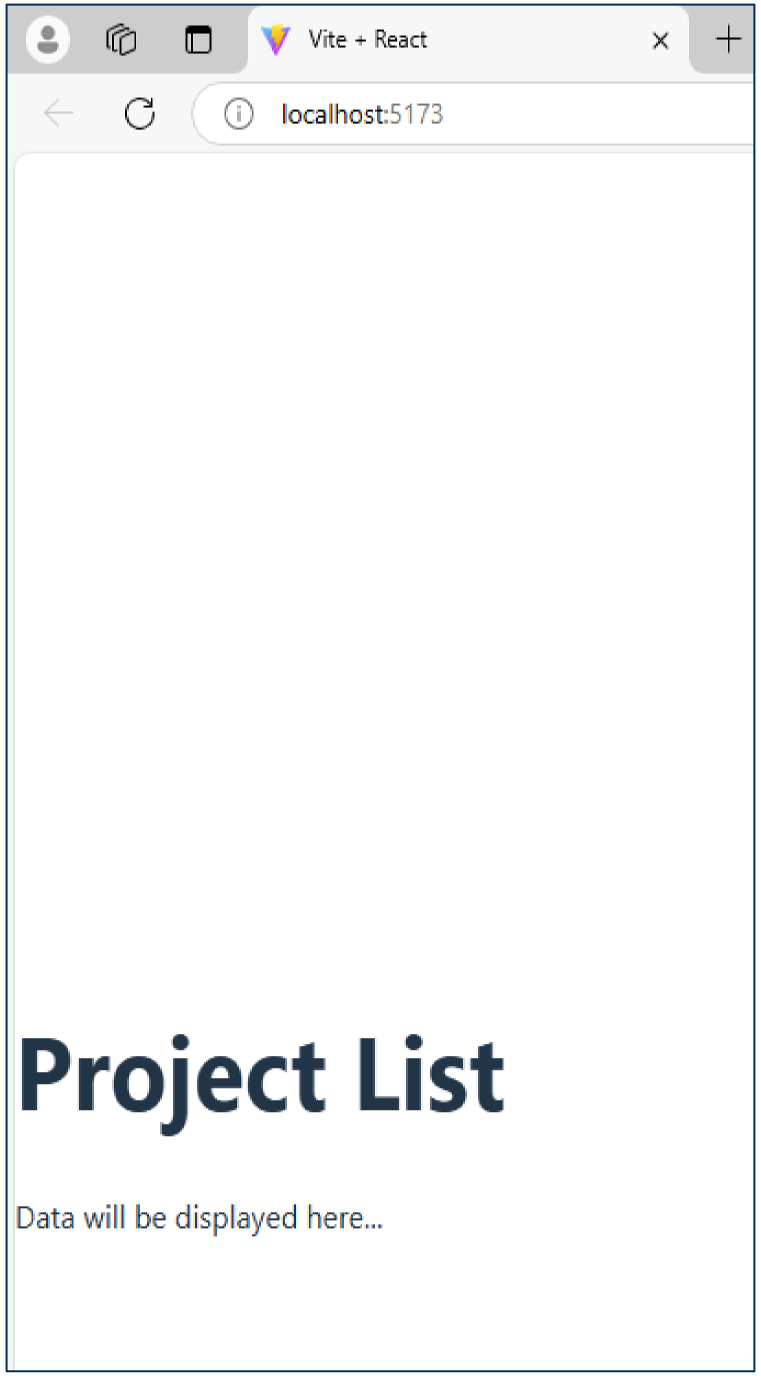


Vite + React

Edit `src/App.jsx` and save to test HMR

Let's, create and render a **simple** React component with a basic layout.

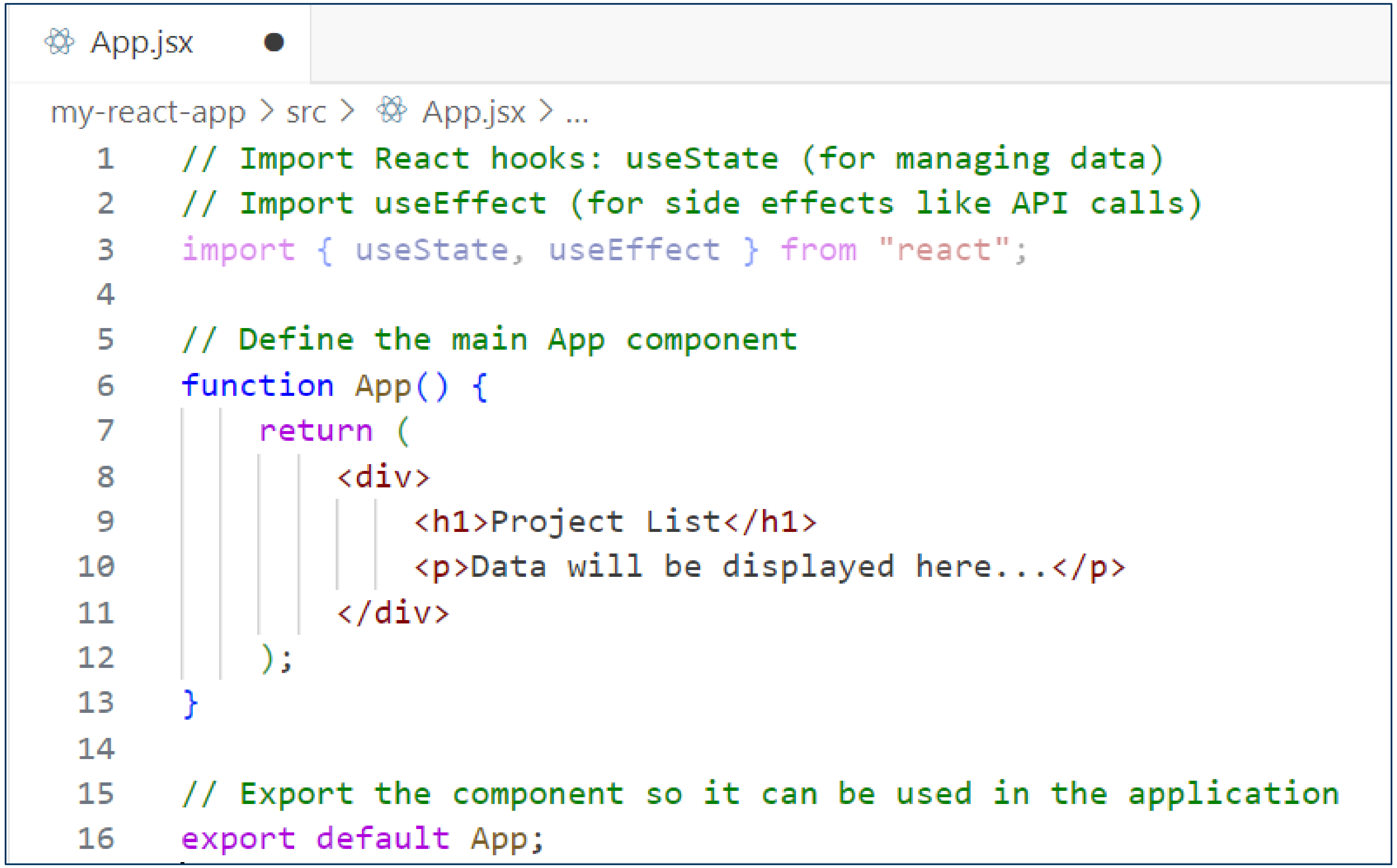




Project List

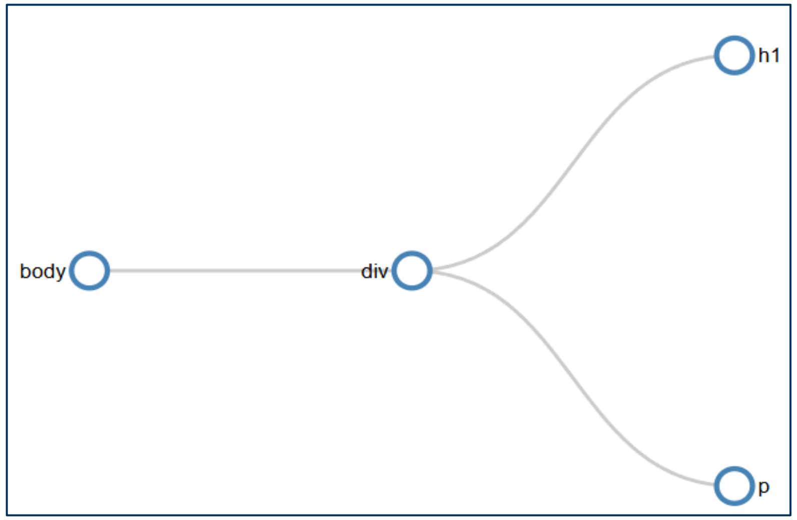
Data will be displayed here...

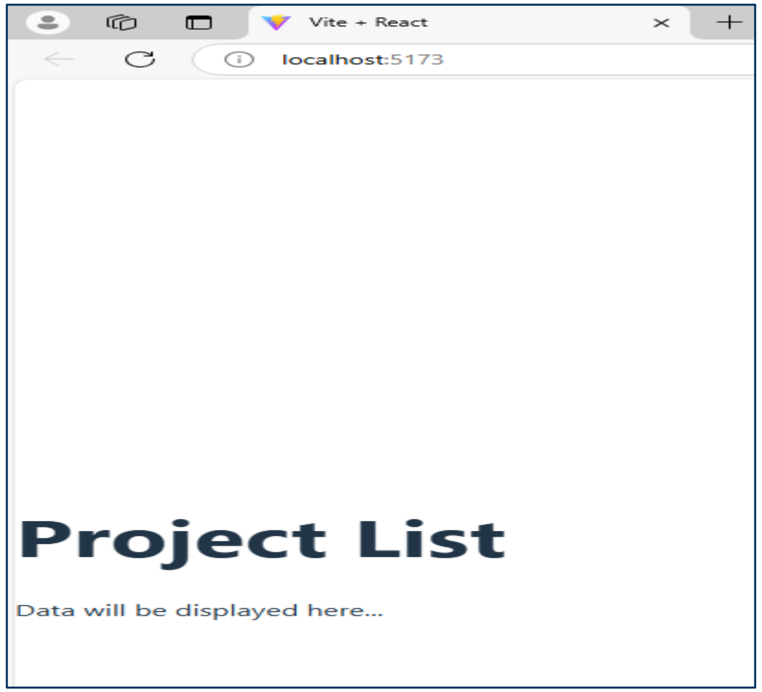
create and render a simple React component App()


App.jsx

my-react-app > src > App.jsx > ...

```
// Import React hooks: useState (for managing data)
// Import useEffect (for side effects like API calls)
import { useState, useEffect } from "react";

// Define the main App component
function App() {
    return (
        <div>
            <h1>Project List</h1>
            <p>Data will be displayed here...</p>
        </div>
    );
}

// Export the component so it can be used in the application
export default App;
```



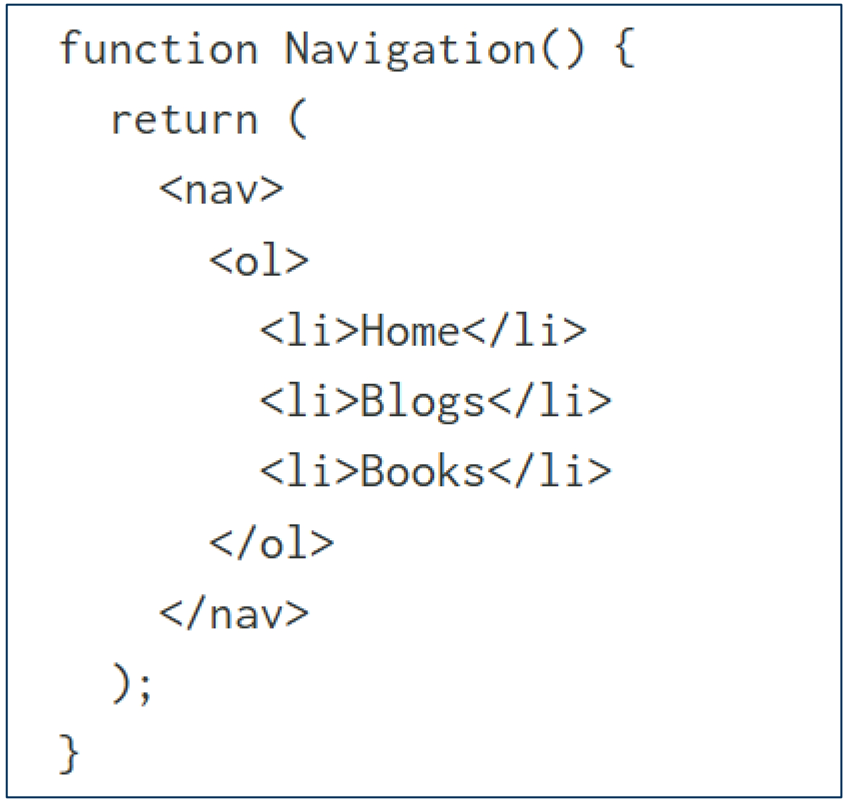


Project List

Data will be displayed here...

Recall what is a React Component?

A React component is a ______________ that returns a piece of _________ , which can be as straightforward as a fragment of __________ . Consider the creation of a __________ that renders a navigation bar.



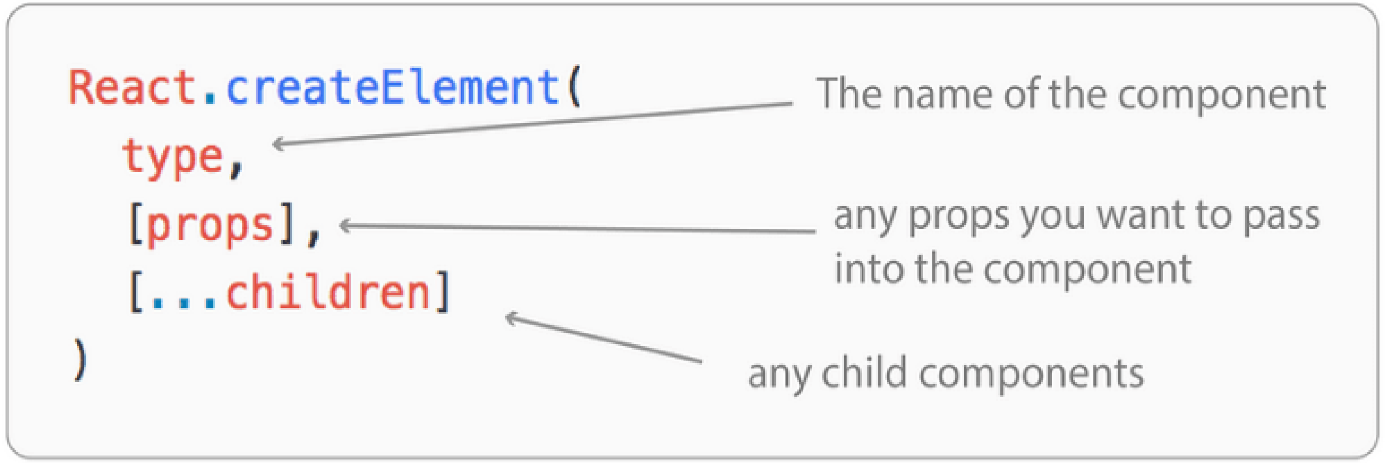
```
function Navigation() {
  return (
    <nav>
      <ol>
        <li>Home</li>
        <li>Blogs</li>
        <li>Books</li>
      </ol>
    </nav>
  );
}
```

The mixture of JavaScript with HTML tags might seem strange (it's called ______ , a syntax extension to JavaScript. For those using _________ , a similar syntax called TSX is used). To make this code functional, a compiler is required to translate the JSX into valid ____________ code.

After being compiled by Babel, the JSX code would roughly translate to the following:

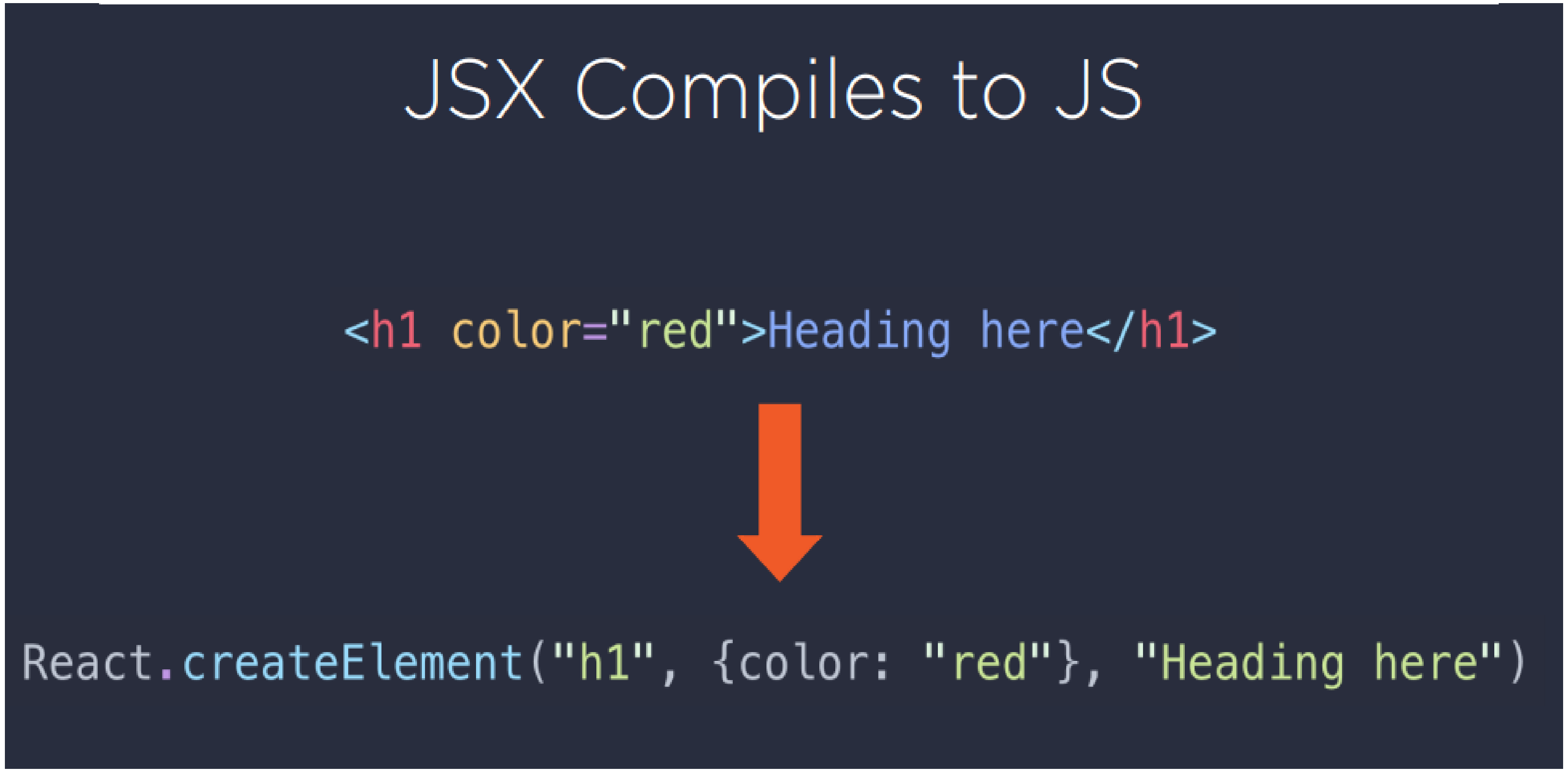
```
function Navigation() {
  return (
    <nav>
      <ol>
        <li>Home</li>
        <li>Blogs</li>
        <li>Books</li>
      </ol>
    </nav>
  );
}
```

```
function Navigation() {
  return React.createElement(
    "nav",
    null,
    React.createElement(
      "ol",
      null,
      React.createElement("li", null, "Home"),
                    ??
    )
  );
}
```



```
React.createElement(
  type,
  [props],
  [...children]
)
```

The name of the component
any props you want to pass into the component
any child components

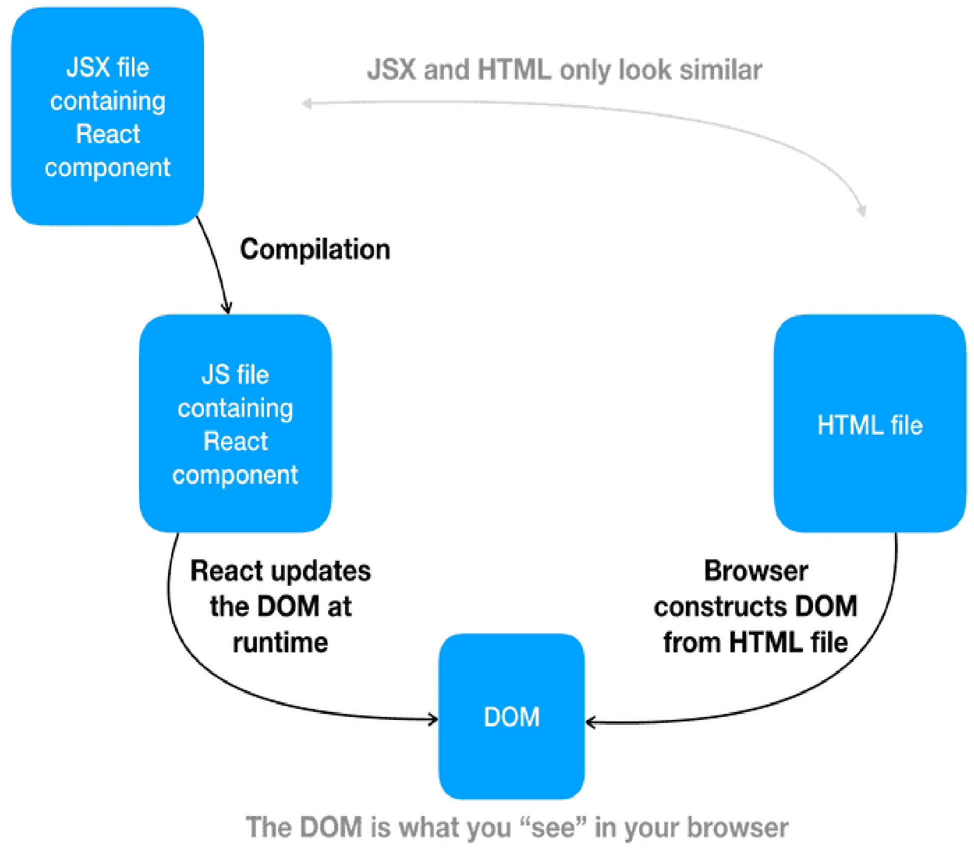


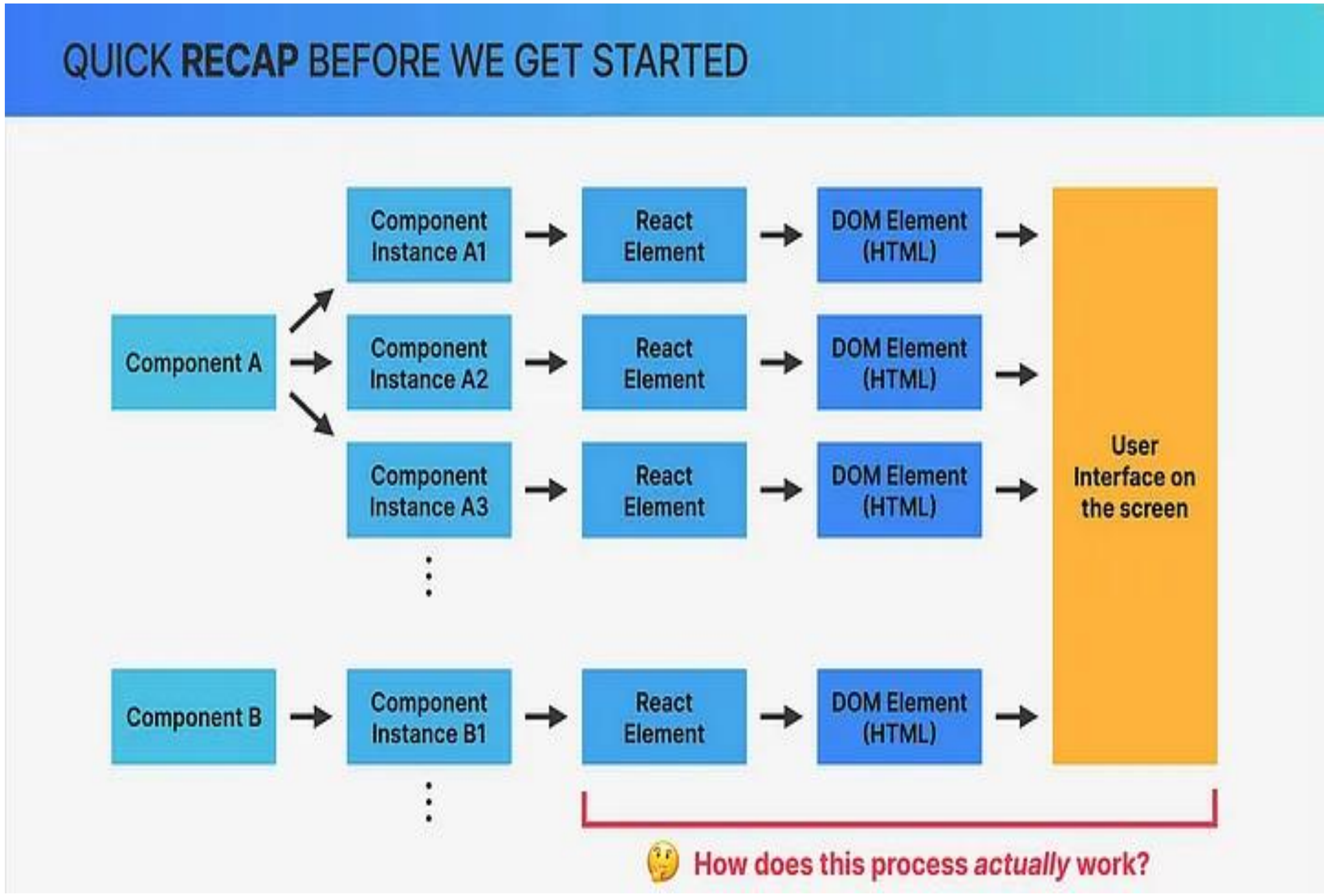
JSX Compiles to JS

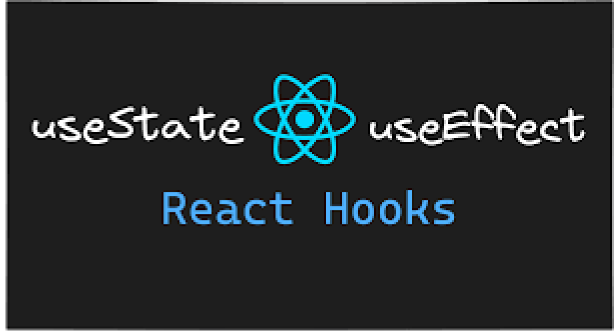
```
<h1 color="red">Heading here</h1>
```

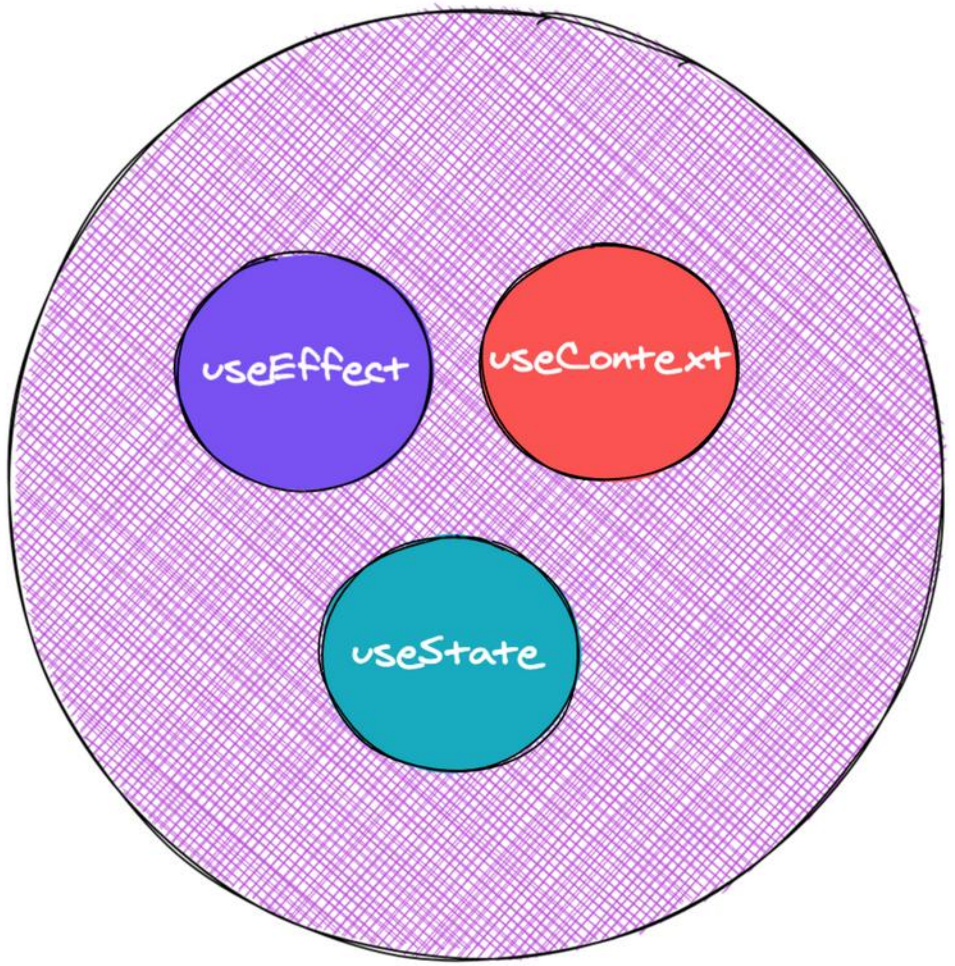
```
React.createElement("h1", {color: "red"}, "Heading here")
```

JSX gets compiled to this tree of JS function calls


JSX file containing React component
JSX and HTML only look similar
Compilation
JS file containing React component
HTML file
React updates the DOM at runtime
Browser constructs DOM from HTML file
DOM
The DOM is what you "see" in your browser

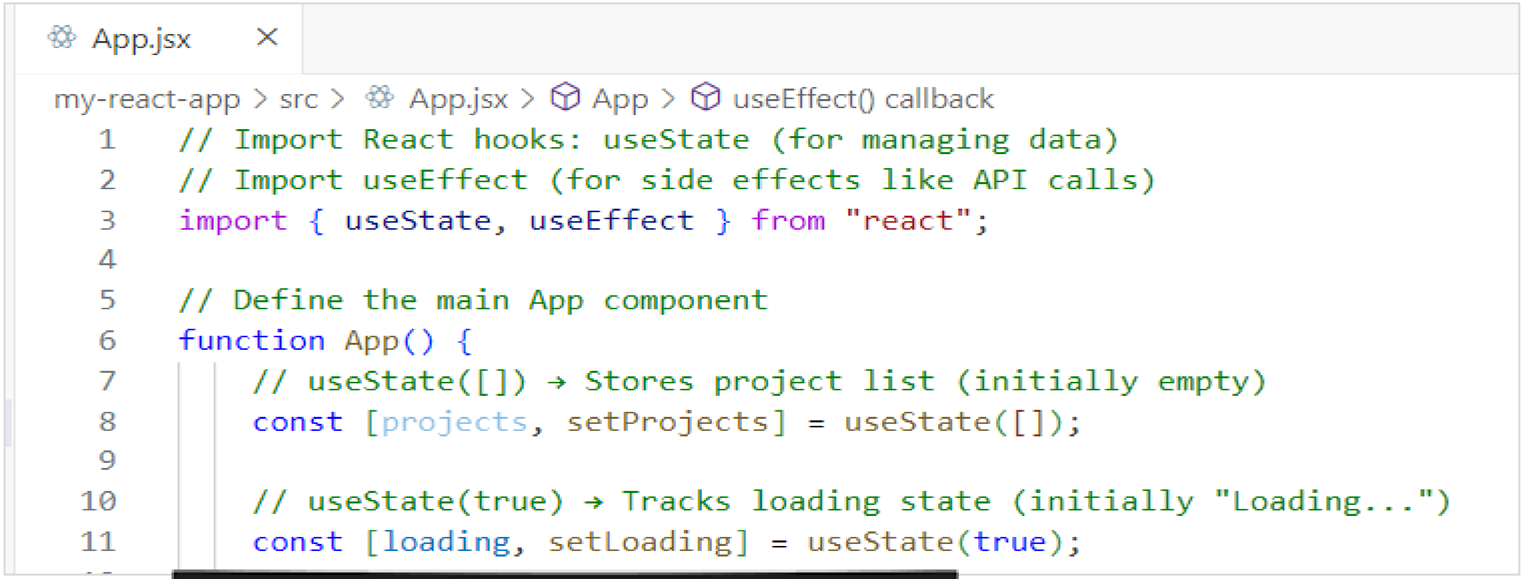



useState
useEffect
React Hooks


useEffect
useContext
useState

▨ Basic Hooks

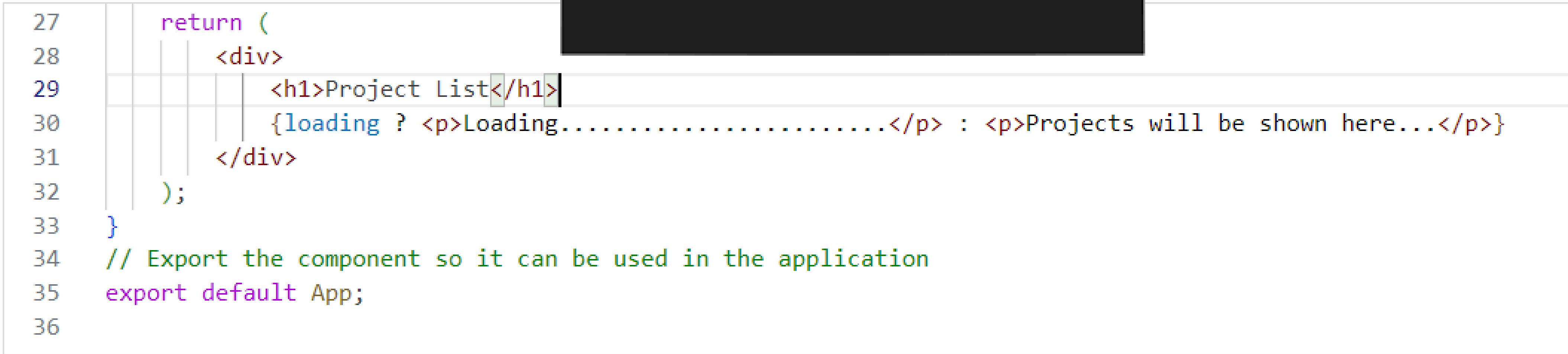
Implement Fetch API Call in React



App.jsx

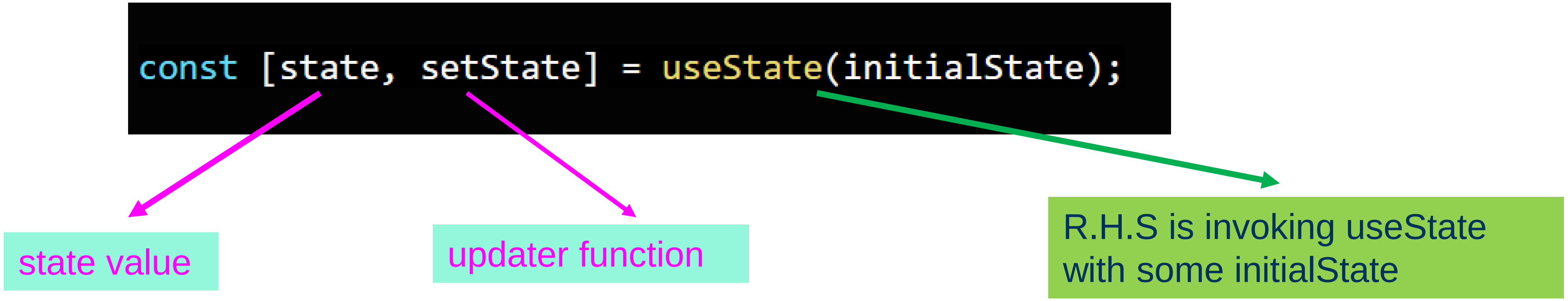
my-react-app > src > App.jsx > App > useEffect() callback

```jsx
// Import React hooks: useState (for managing data)
// Import useEffect (for side effects like API calls)
import { useState, useEffect } from "react";

// Define the main App component
function App() {
    // useState([]) → Stores project list (initially empty)
    const [projects, setProjects] = useState([]);

    // useState(true) → Tracks loading state (initially "Loading...")
    const [loading, setLoading] = useState(true);
```



```jsx
    return (
      <div>
        <h1>Project List</h1>
        {loading ? <p>Loading........................</p> : <p>Projects will be shown here...</p>}
      </div>
    );
}
// Export the component so it can be used in the application
export default App;
```

The signature for the useState is as follows:



```
const [state, setState] = useState(initialState);
```

where state and setState refer to the state value and updater function returned on <u>invoking useState</u> with some initialState

✓ Our React Front end is using useState twice

ProjectManager

ArrayList<String> projects

ProjectManager()

void addProject(String projectName)

ArrayList<String> getProjects()

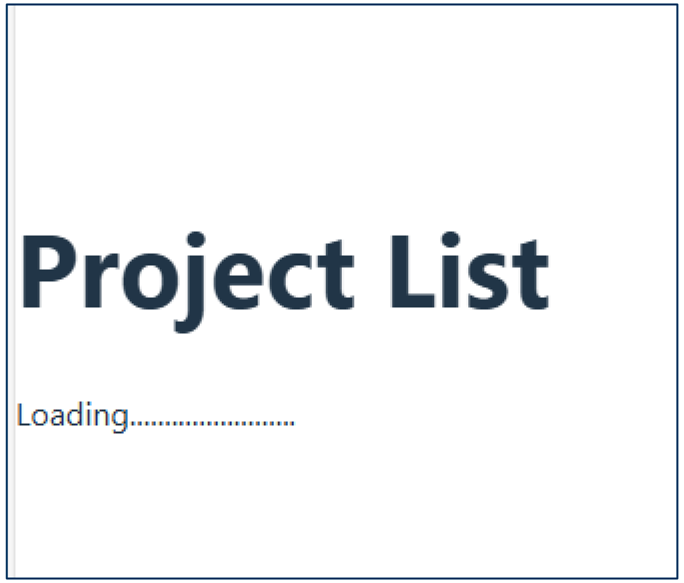
useState([]) for project management

```
// Define the main App component
function App() {
    // useState([]) → Stores project list (initially empty)
    const [projects, setProjects] = useState([]);

    // useState(true) → Tracks loading state (initially "Loading...")
    const [loading, setLoading] = useState(true);

    return (
        <div>
            <h1>Project List</h1>
            {loading ? <p>Loading.......................</p> : <p>Projects will be shown here...</p>}
        </div>
    );
}
```

UNIVERSITY OF KARACHI

```
const [loading, setLoading] = useState(true);
```

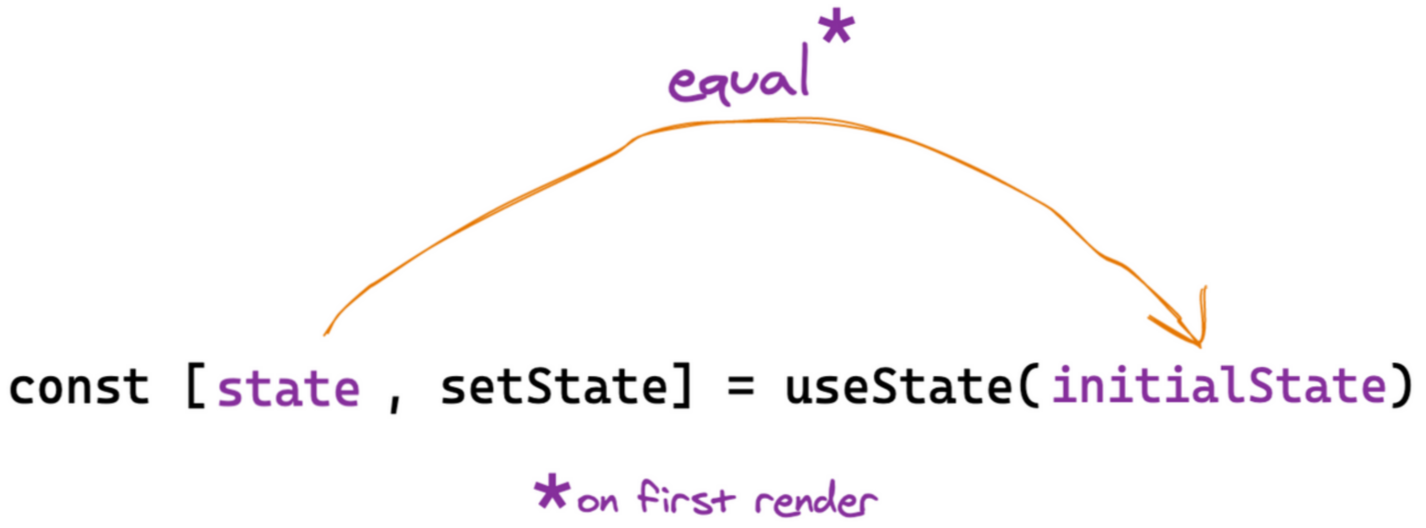
The second useState(true) simply **acts as a flag** to track loading state.
We **toggle this flag** (true → false) once data is fetched.


Project List

Loading........................

- ✓ If loading === true → UI shows "Loading.......".
- ✓ If loading === false → UI shows the project list.

It's important to note that when your component first renders and invokes useState, the initial State is the returned state from useState .

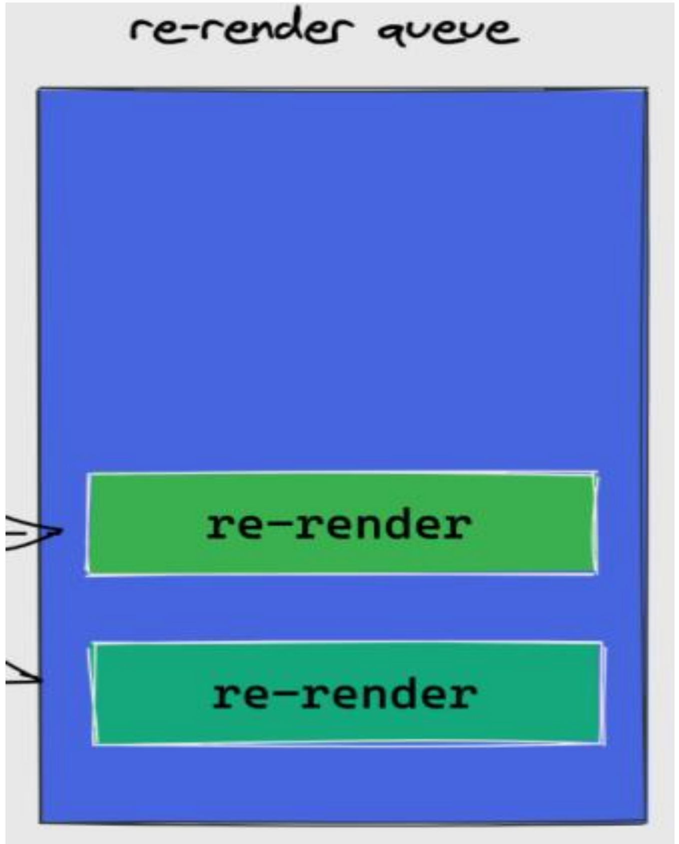


```
const [state , setState] = useState(initialState)
const [state, setState] = useState(initialState);
```

```
setState(newValue)
```

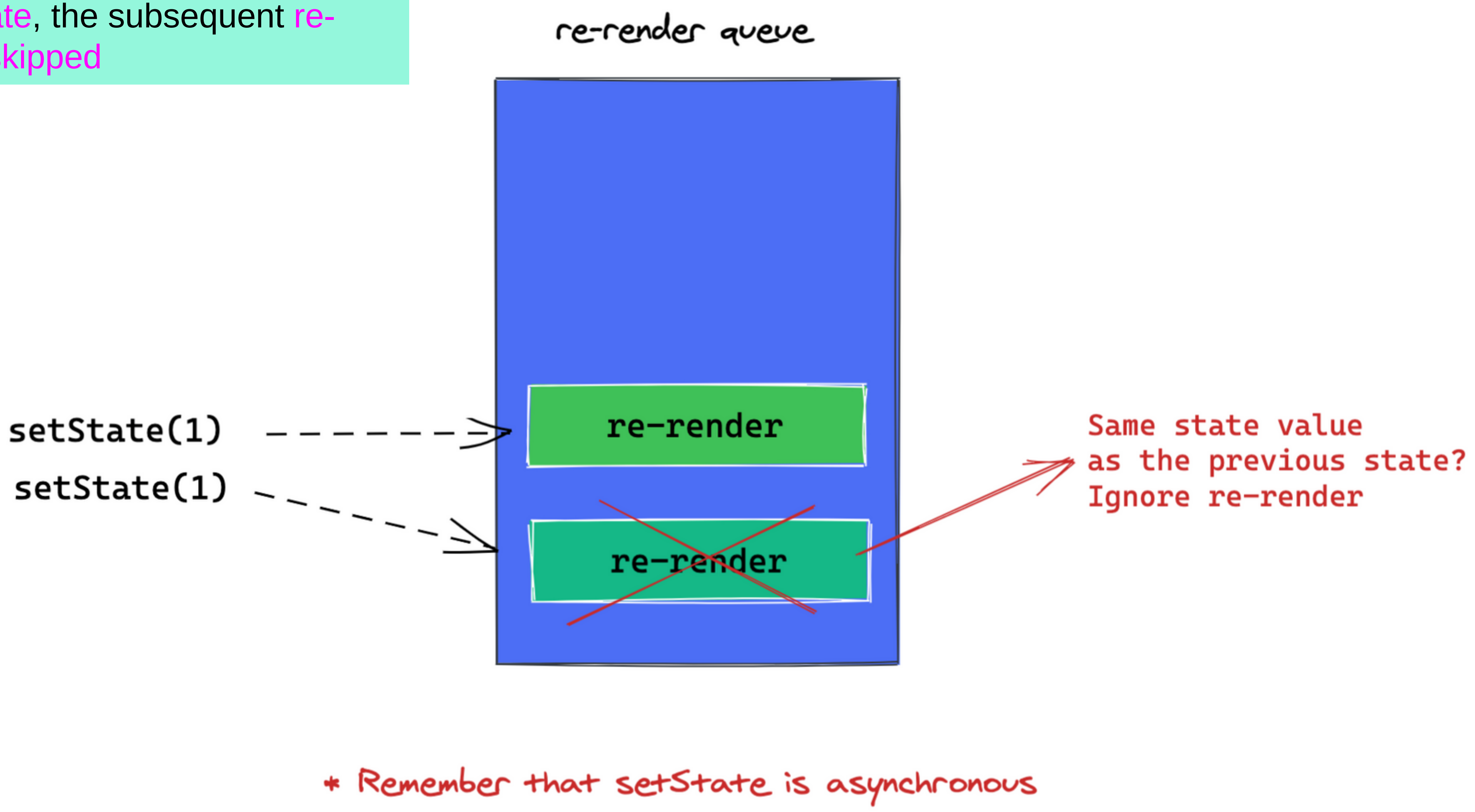
Also, to update state, the state updater function **`setState`** should be invoked with a new state value as shown below:

By doing this a new re-render of the component is queued.

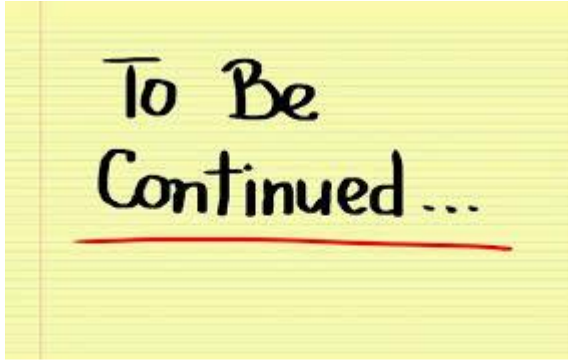


useState guarantees that the state value will always be the most recent after applying updates.

Note that if the updater function returns the exact same value as the current state, the subsequent re-render is skipped

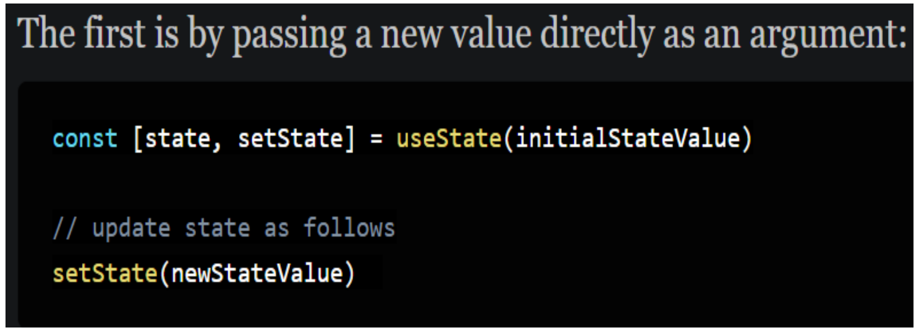


https://blog.ohansemmanuel.com/react-hooks-documentation-easy-to-read/



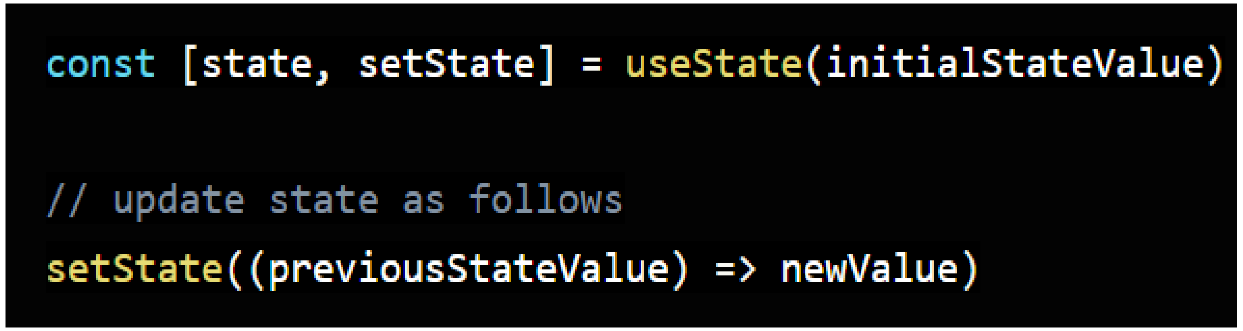
The state updater function returned by useState can be invoked in two ways.

setState(newStateValue)


The first is by passing a new value directly as an argument:

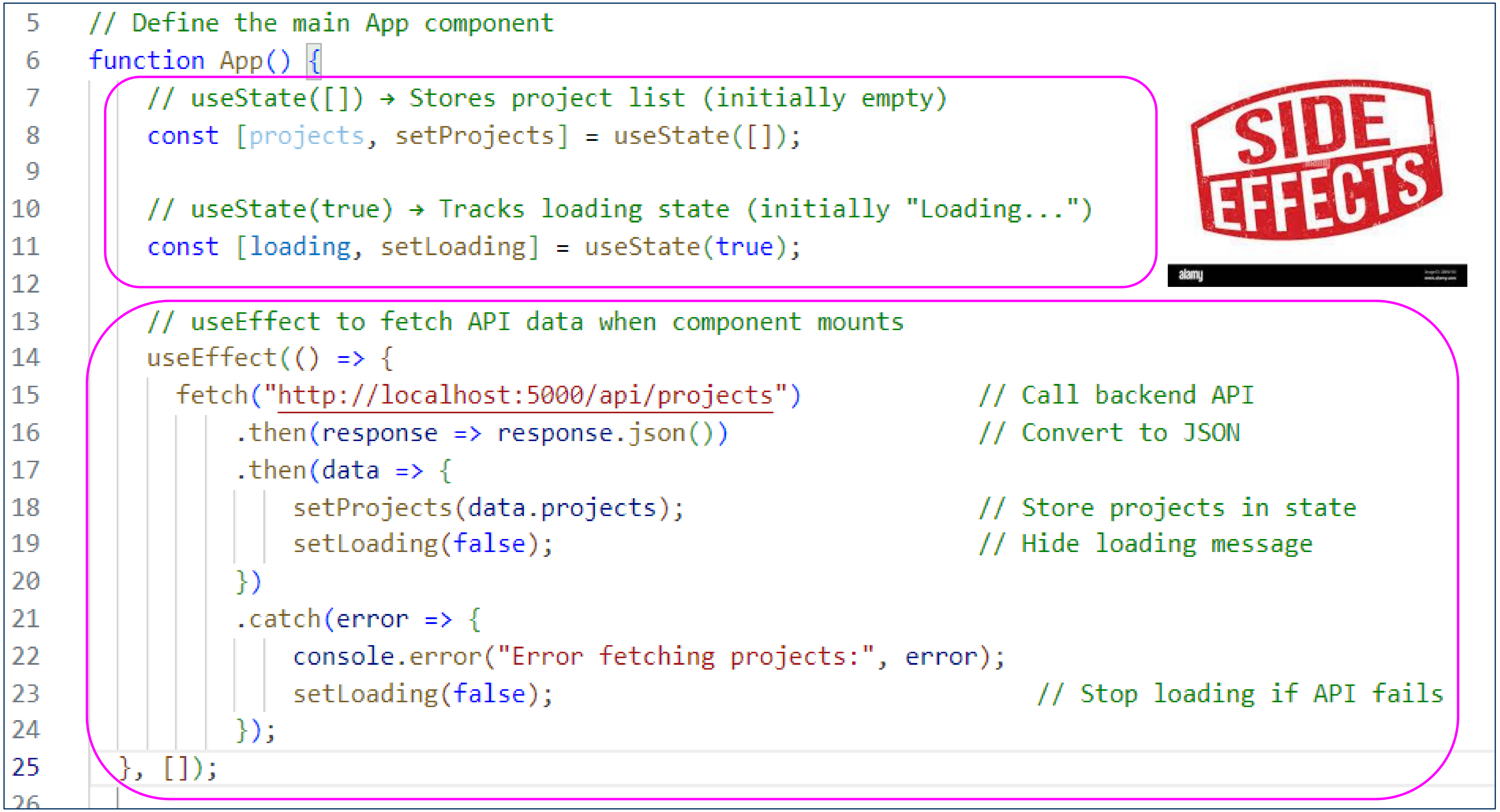
```
const [state, setState] = useState(initialStateValue)

// update state as follows
setState(newStateValue)
```

setState(() => {})

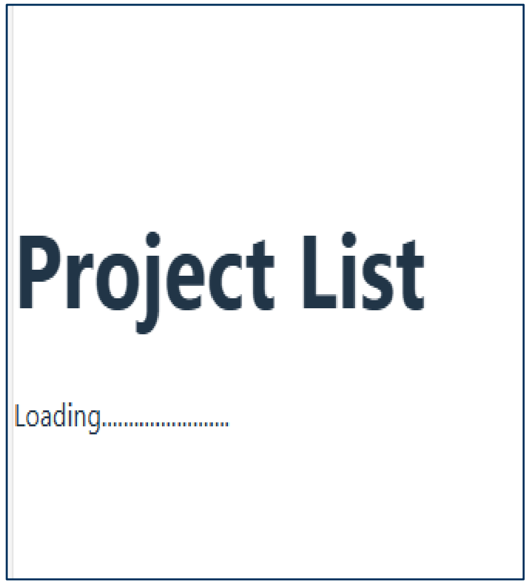


```
const [state, setState] = useState(initialStateValue)

// update state as follows
setState((previousStateValue) => newValue)
```

Functional updates

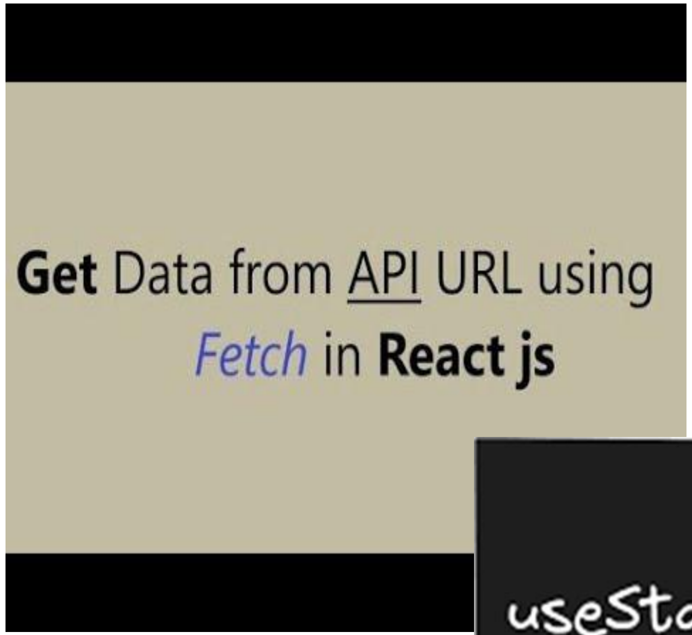


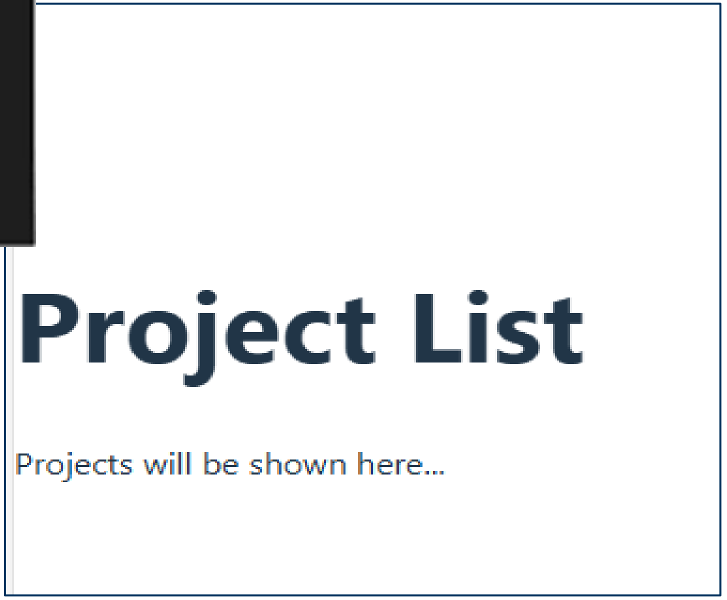
```
// Define the main App component
function App() {
    // useState([]) → Stores project list (initially empty)
    const [projects, setProjects] = useState([]);

    // useState(true) → Tracks loading state (initially "Loading...")
    const [loading, setLoading] = useState(true);

    // useEffect to fetch API data when component mounts
    useEffect(() => {
      fetch("http://localhost:5000/api/projects")            // Call backend API
        .then(response => response.json())                   // Convert to JSON
        .then(data => {
          setProjects(data.projects);                         // Store projects in state
          setLoading(false);                                  // Hide loading message
        })
        .catch(error => {
          console.error("Error fetching projects:", error);
          setLoading(false);                                      // Stop loading if API fails
        });
  }, []);
```



Project List

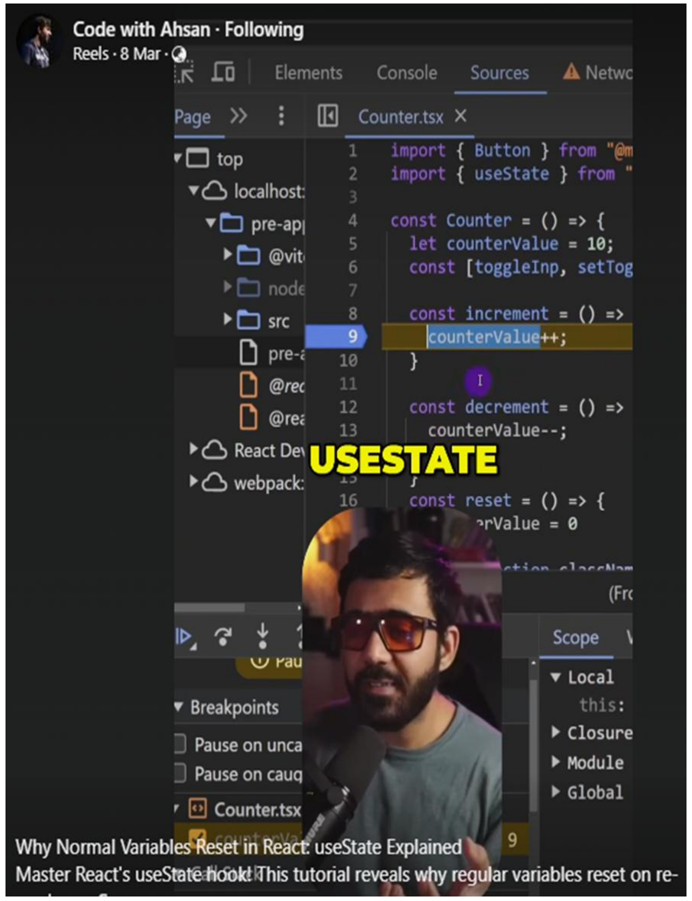
Loading......................





Project List

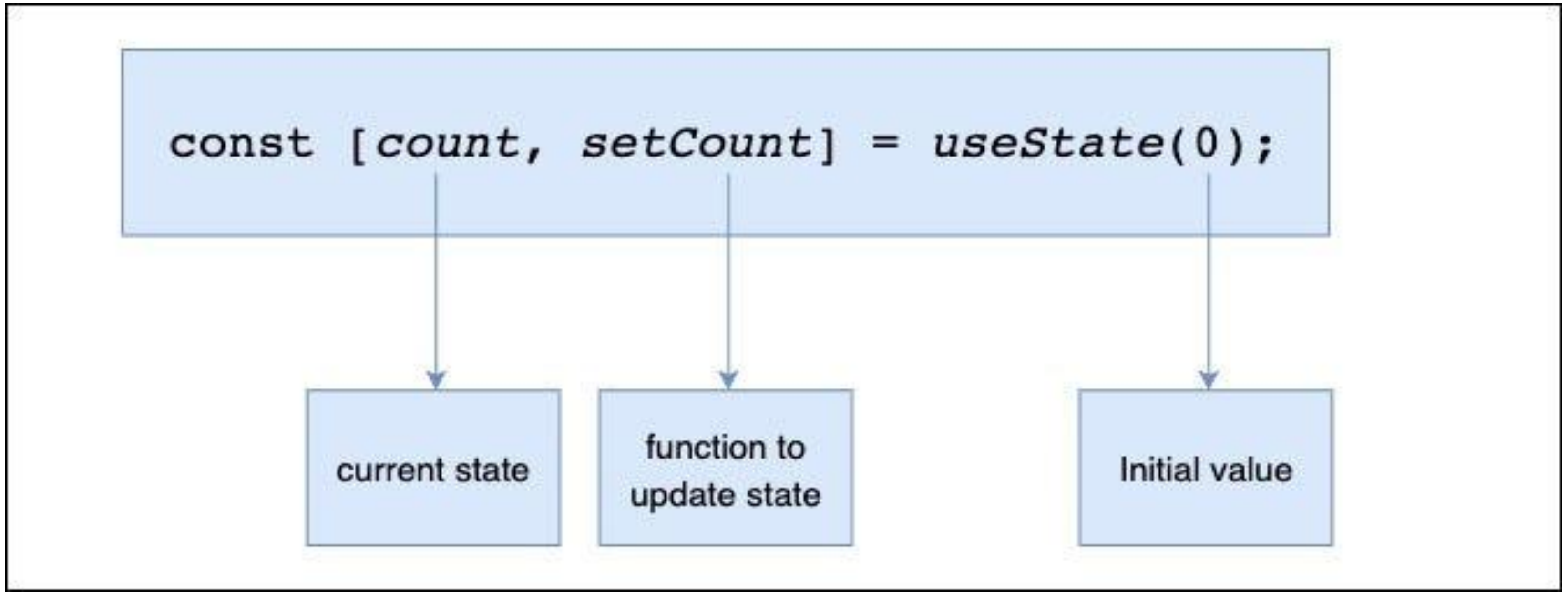
Projects will be shown here...



Code with Ahsan · Following
Reels · 8 Mar ·

```
import { Button } from "@m
import { useState } from "

const Counter = () => {
  let counterValue = 10;
  const [toggleInp, setTog

  const increment = () =>
    counterValue++;
  }

  const decrement = () =>
    counterValue--;

  const reset = () => {
      rValue = 0
```

Why Normal Variables Reset in React: useState Explained
Master React's useState hook! This tutorial reveals why regular variables reset on re-

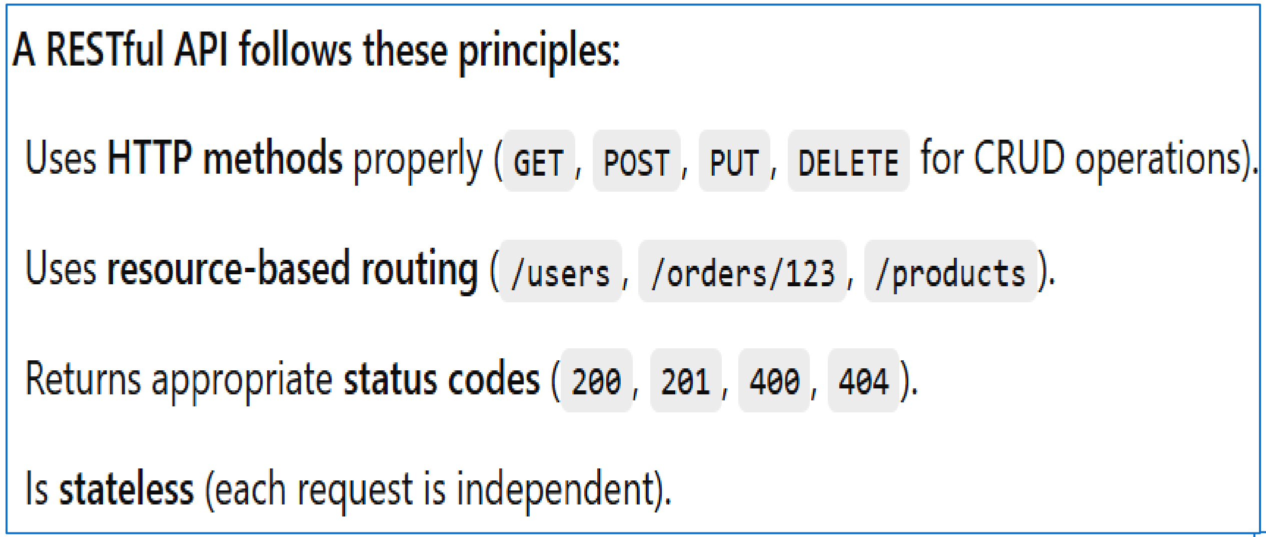


Request-Response Cycle (HTTP Communication) Done ???

There is no elevator to success.

You have to take the stairs! 🙂

HTTP Method	Server Route	Frontend fetch() Call	Purpose
GET	/api/projects	fetch("http://localhost:5000/api/projects")	Fetch all projects
POST	/api/projects	fetch("http://localhost:5000/api/projects", { method: "POST", body: JSON.stringify({...}) })	Add new project
PUT	/api/projects/:id	fetch("http://localhost:5000/api/projects/1", { method: "PUT", body: JSON.stringify({...}) })	Update project
DELETE	/api/projects/:id	fetch("http://localhost:5000/api/projects/1", { method: "DELETE" })	Remove project



A RESTful API follows these principles:

- Uses **HTTP methods** properly (`GET`, `POST`, `PUT`, `DELETE` for CRUD operations).
- Uses **resource-based routing** (`/users`, `/orders/123`, `/products`).
- Returns appropriate **status codes** (`200`, `201`, `400`, `404`).
- Is **stateless** (each request is independent).

Advanced Routing Techniques

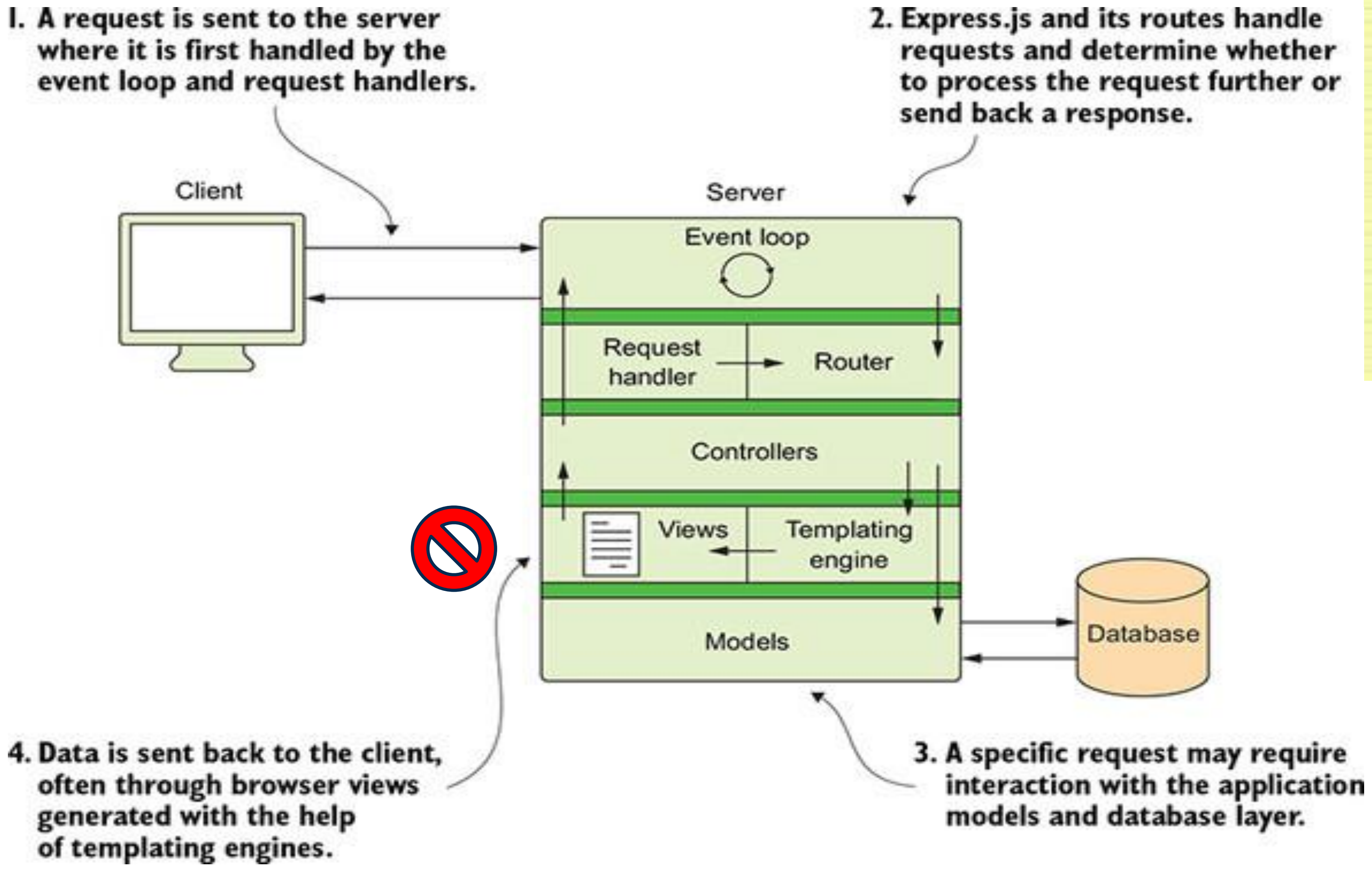
Express Router: Use express.Router to create modular, mountable route handlers.

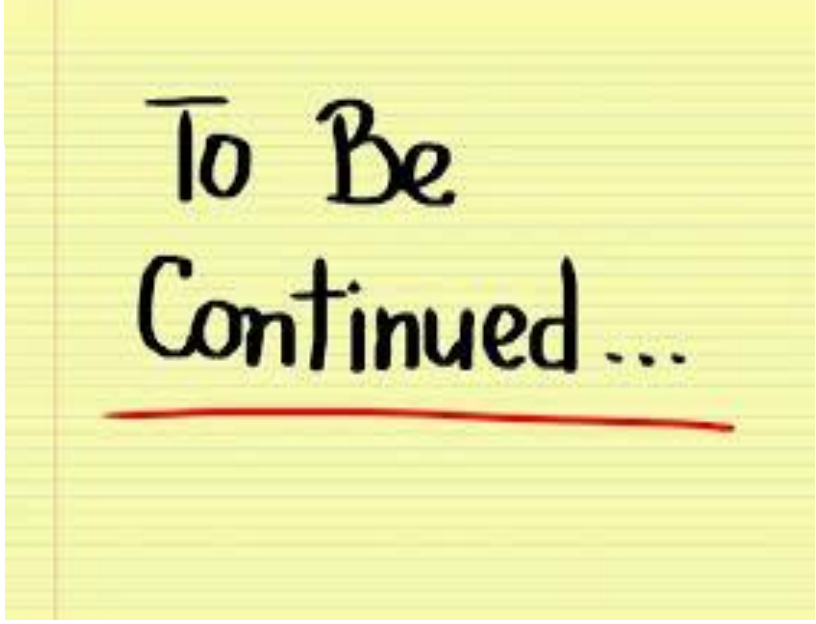
Route Methods: Define routes using methods like app.get(), app.post(), app.put(), and app.delete().

Route Paths: Can be strings, string patterns, or regular expressions.

Route Parameters: Capture values specified at their position in the URL using named segments.

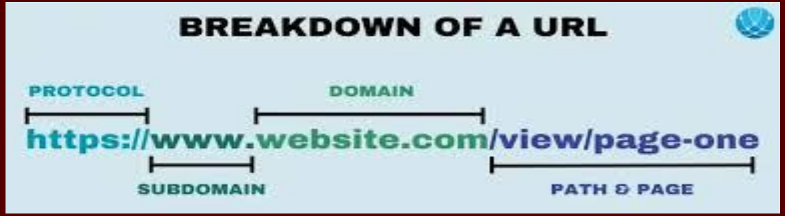
Chained Route Handlers: Use app.route() to create chainable route handlers for a route path.


1. A request is sent to the server where it is first handled by the event loop and request handlers.
2. Express.js and its routes handle requests and determine whether to process the request further or send back a response.
Client
Server
Event loop
Request handler
Router
Controllers
Views
Templating engine
Models
Database
3. A specific request may require interaction with the application models and database layer.
4. Data is sent back to the client, often through browser views generated with the help of templating engines.


To Be Continued ...

Request-Response Cycle (HTTP Communication)

Deadline: 25 March 2025



```
const app = express();
const PORT = process.env.PORT || 5000;

// Middleware

// Mock Database

// GET - Fetch all projects
app.get("/api/projects", (req, res) => res.json({
status: "success", projects }));

// POST - Add a new project
app.post("/api/projects", (req, res) => {
    if (!req.body.name) return res.status(400).json({
error: "Project name is required" });

    const newProject = { id: projects.length + 1,
name: req.body.name };
    projects.push(newProject);
    res.status(201).json({ status: "success", project:
newProject });
});

// Start the server
app.listen(PORT, () => console.log(`Server running at
http://localhost:${PORT}`));
```

Backend (Express.js)

ASSIGNMENT

1) Add a **new endpoint** /api/projects/count that returns the total number of projects.

2) Try DELETE. Then Modify DELETE /api/projects/:id to return the deleted project details instead of just a message.

3) Ensure project names are at least 3 characters long in both POST and PUT requests. If validation fails, return { status: "error", message: "Project name must be at least 3 characters" }

```
function App() {
         // useState
         //  useEffect
  return (<div> …. </div>
}
```

Frontend (React)

ASSIGNMENT

(1) Convert API URL into a prop so the component can fetch from dynamic endpoints.

(2) Display the total number of projects (using the /api/projects/count endpoint).

(3) Add a simple **form with a button** to submit new projects.

(4) Each project should have a **delete button** that removes it via DELETE /api/projects/:id.

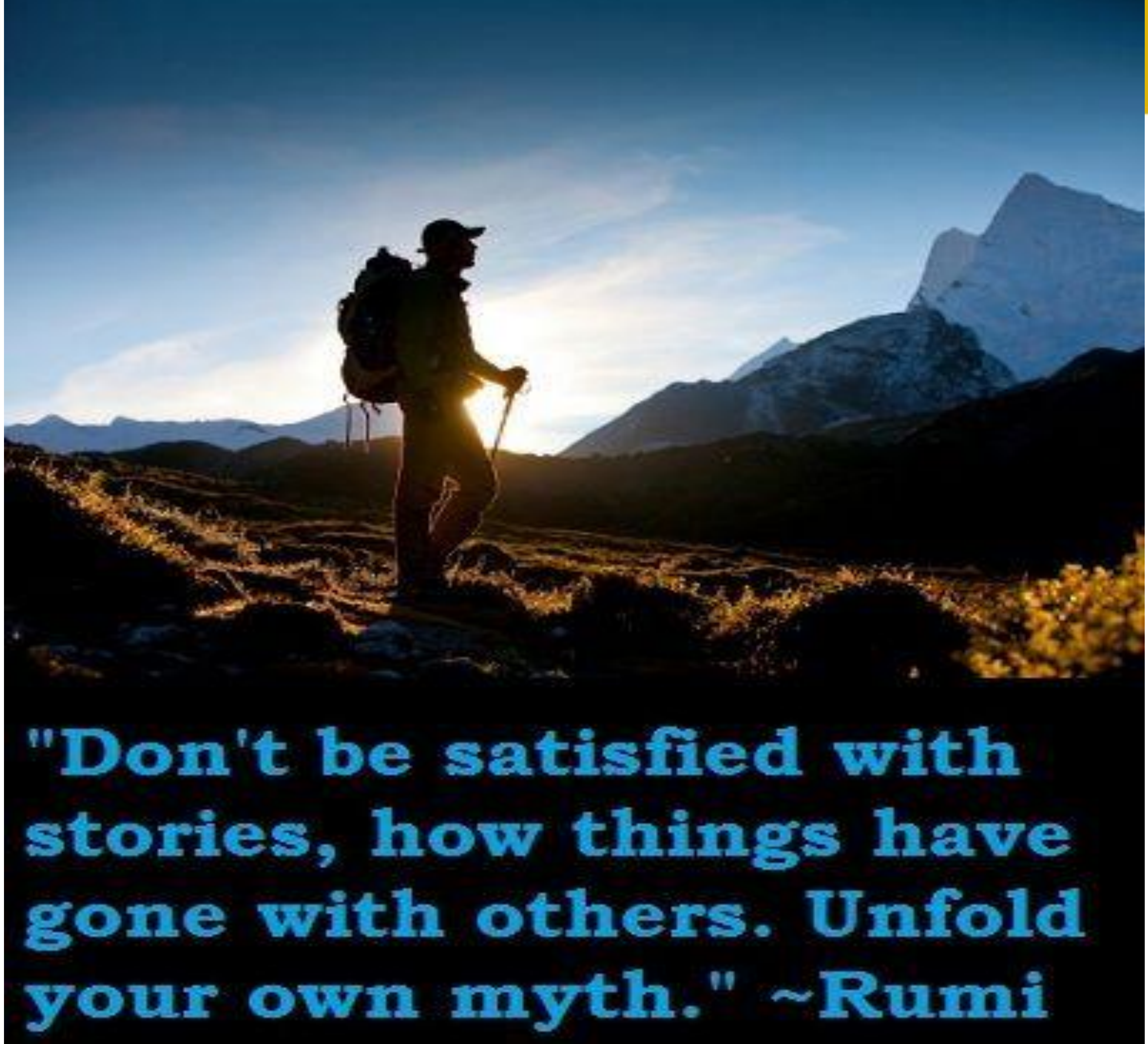
(5) Update the UI immediately after adding or deleting a project.

(6) Prevent duplicate project names on the backend (case-insensitive).

(7) Display **error messages** properly in React when API validation fails.

(8) : **Add a "Refresh Projects" button in the frontend (App.jsx)**
that re-fetches data from the backend (server.js) using an arrow function.

UNIVERSITY OF KARACHI


"Don't be satisfied with stories, how things have gone with others. Unfold your own myth." ~Rumi

UNIVERSITY OF KARACHI

Department of Compute Science (UBIT Building), Karachi, Pakistan.

1200 Acres (5.2 Km sq.)
53 Departments
19 Institutes
25000 Students

My Homeland Pakistan