Task 7: REST API Development

Objective:

Create a RESTful API with CRUD operations for a simple resource (e.g., users, products). Ensure all API endpoints work correctly by testing them using Postman.

Prerequisites:

- Basic understanding of JavaScript and Node.js.
- Node.js and NPM installed.

Concepts:

- **Creating a Basic HTTP Server:**
  - Node.js can be used to create an HTTP server without any additional frameworks.
  - Use the built-in `http` module to handle HTTP requests and responses.

  Example:

```
// server.js
const http = require('http');
const server = http.createServer((req, res) => {
  // Handle requests and responses
});

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

- **Defining CRUD Operations:**
  - **Create:** Use the `POST` method to add a new resource.
  - **Read:** Use the `GET` method to retrieve a resource.
  - **Update:** Use the `PUT` method to update an existing resource.
  - **Delete:** Use the `DELETE` method to remove a resource.

Setup:

1. **Install Node.js:**

   Ensure Node.js is installed on your machine. You can access the instructions here:

   - 
     Detailed Instructions for Installing Node.js and NPM and integration...
     For Windows: — Download Node.js Installer: — Visit the official Node.js website:...
     w3o NFThing Last Edited 7/3/2024

Tasks:

1. **Setting Up the Server:**
   - **Task:**

        - Create a file named `server.js` and write your own code that:
            - Sets up a basic HTTP server using Node.js.
            - Listens on port 3000 and logs a message when the server is running.
    - **Outcome:**
        - Ensure the server starts and logs the message correctly.
2. **Creating the Resource:**
    - **Task:**
        - In `server.js`, define an array to hold your resource data (e.g., users or products).
        - Create a function to handle `POST` requests to add a new resource to the array.
    - **Example:**

```
// server.js
let users = [];
if (req.method === 'POST' && req.url === '/users') {
  // Handle adding a new user
}
```

    - **Outcome:**
        - Ensure new resources can be added correctly.
3. **Reading the Resource:**
    - **Task:**
        - Define functions to handle `GET` requests to retrieve all resources and a specific resource by ID.
    - **Example:**

```
// server.js
if (req.method === 'GET' && req.url === '/users') {
  // Handle retrieving all users
}
if (req.method === 'GET' && req.url.startsWith('/users/')) {
  // Handle retrieving a user by ID
}
```

    - **Outcome:**
        - Ensure resources can be retrieved correctly.
4. **Updating the Resource:**
    - **Task:**
        - Create a function to handle `PUT` requests to update an existing resource in the array.
    - **Example:**

```
// server.js
if (req.method === 'PUT' && req.url.startsWith('/users/')) {
  // Handle updating a user by ID
}
```

- **Outcome:**
  - Ensure resources can be updated correctly.

5. **Deleting the Resource:**
   - **Task:**
     - Create a function to handle `DELETE` requests to remove a resource from the array.
   - **Example:**

```
// server.js
if (req.method === 'DELETE' && req.url.startsWith('/users/')) {
  // Handle deleting a user by ID
}
```

   - **Outcome:**
     - Ensure resources can be deleted correctly.

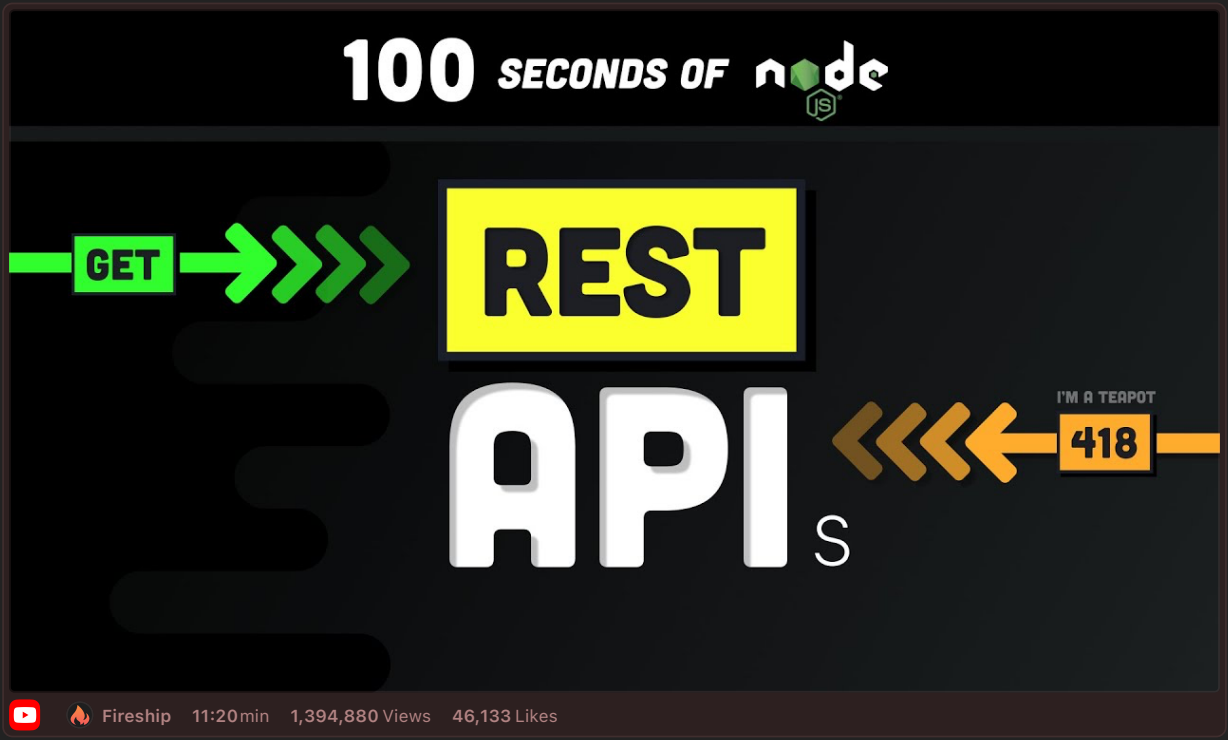
Instructions:

- **Perform the following tasks:**
  - Write the required code in `server.js`.
  - Run the server using Node.js and test the endpoints using Postman to ensure the code executes without errors and demonstrates the correct handling of CRUD operations.

Resources:

- HTTP | Node.js v22.4.0 Documentation
  This module, containing both a client and server, can be imported via — require('node:http') (Commo...
  https://nodejs.org/api/http.html
- Learn REST API Design - REST API Tutorial
  Learn REST API Design
  Author **Todd Fredrich**

videos:



GitHub Instructions:

1. **Open in Visual Studio Code:**
   - After clicking on the "Open in Visual Studio Code" button from the GitHub Classroom confirmation page, Visual Studio Code (VSCode) will open the repository directly.
   - If prompted, select "Open" or "Allow" to open the repository in VSCode.
2. **Complete the Task:**
   - Write your solution in `server.js`.
3. **Run and Test Your Code:**
   - Run your code to ensure it works correctly. Use the following command:

```
node server.js
```

4. **Commit Your Changes:**
   - Commit your changes with a meaningful message:

```
git commit -m "Completed REST API Development task"
```

5. **Push Your Changes to Your Forked Repository:**
   - Push your changes to your forked repository:

```
git push origin main
```

6. **Create a Pull Request:**
    - Go to your forked repository on GitHub.
    - Click on the "Pull Requests" tab.
    - Click the "New Pull Request" button.
    - Ensure the base repository is the original template repository and the base branch is `main`.
    - Ensure the head repository is your forked repository and the compare branch is `main`.
    - Click "Create Pull Request".
    - Add a title and description for your pull request and submit it.

Summary of Commands:

```
# Fork the repository on GitHub

# Clone the forked repository
git clone https://github.com/your-github-username/repository-name.git
cd repository-name

# Complete the task by writing and running the code in server.js

# Add, commit, and push your changes
git commit -m "Completed REST API Development task"
git push origin main

# Create a pull request on GitHub
```