Written Assignment 2

Roll No: 71 Batch T14

Q.1] What are Refs? When to use Refs and when not to use refs

Ans:

In React, Refs (short for references) are a way to interact with the DOM (Document Object Model) directly. They provide a way to access and manipulate DOM elements or React components directly from your React code. Refs are typically used when you need to perform actions like focusing an input element, triggering animations, or integrating with third-party libraries that aren't designed for React.

Creating Ref:

```
class MyComponent extends Component {
  constructor(props) {
    super(props);
    this.myRef = React.createRef();
  }

  // ...
}
```

When to use Ref:

1. **Accessing DOM Elements:** When you need to access a DOM element directly, such as focusing an input field, playing/pausing a video, or measuring the dimensions of an element. Refs allow you to avoid relying on the virtual DOM and work with the actual DOM.

2. **Integrating with Third-party Libraries**: When you're using non-React libraries that need to interact with specific DOM elements, like charts, maps, or rich text editors. Refs provide a way to bridge the gap between React's virtual DOM and these external libraries.

3. **Managing Focus and Text Selection**: For implementing features like auto-focusing on input fields when a component mounts or selecting text within an element programmatically.

When Refs should be avoided:

1. **Avoid Excessive Use**: Refs should not be the default approach for managing component state or interacting with the DOM. React's data flow and state management (using state and props) should be preferred whenever possible, as they promote a more predictable and maintainable codebase.

2. **Think in React**: In many cases, you can achieve the same goals using React's state and props without resorting to refs. It's often better to follow the "thinking in React" paradigm and design your components in a way that leverages React's component lifecycle and data flow.

3. **Consider Functional Components:** If you're using a functional component, you can use the useRef hook instead of the older createRef API. Functional components with hooks are the recommended approach in modern React development.

4. **Use Refs Sparingly**: Overusing refs can lead to code that's harder to understand and maintain, as well as potential issues with React's reconciliation process. It's essential to carefully evaluate whether using a ref is the most appropriate solution for your specific use case.

Refs in React are a powerful tool for interacting with the DOM and integrating with non-React code when necessary. However, they should be used sparingly and as a last resort, as React's core philosophy encourages managing state and UI updates through component props and state for cleaner, more maintainable code.

Q.2] Write short note on

a. NPM b.REPL

Ans:

a.) NPM

Ans:

NPM, or Node Package Manager, is a widely used package manager for JavaScript and Node.js development. It plays a crucial role in managing the dependencies, scripts, and packages used in a JavaScript project.

Here are the feature of npm:

Dependency Management: NPM allows developers to define and manage project dependencies efficiently. By creating a package.json file in your project, you can specify the packages and their versions required for your application. NPM then installs these dependencies from the npm registry, making it easy to share and collaborate on projects.

Package Installation: Developers can easily install packages using the npm install command. NPM fetches the required packages and stores them in the node_modules directory of your project. These packages can be libraries, frameworks, or tools that enhance your development workflow.

Scripts and Automation: NPM also provides a way to define and run custom scripts in your project. By adding scripts to your package.json, you can automate various tasks such as building, testing, and starting your application. Common scripts include npm start for running the project and npm test for executing tests.

Version Management: NPM allows you to manage package versions easily. You can specify package versions with semantic versioning (e.g., "^1.0.0" means any compatible version above 1.0.0) to ensure that your project always uses compatible package versions while allowing for updates.

Global Packages: In addition to project-specific packages, NPM also allows you to install packages globally on your system. Global packages are typically command-line tools that you want to access from anywhere in your terminal.

Community and Ecosystem: NPM boasts a vast and active community of developers and an extensive ecosystem of open-source packages. This makes it easy to find, use, and contribute to existing packages, accelerating the development process and fostering collaboration.

Security: NPM takes security seriously and provides features like package vulnerability scanning and advisories. Developers are encouraged to keep their project dependencies up to date to mitigate security risks.

NPM is a vital tool in the JavaScript ecosystem that simplifies package management, facilitates dependency tracking, and enables automation in your projects. Whether you're building web applications, server-side applications with Node.js, or any JavaScript-based project, NPM is an essential part of modern JavaScript development.

b) REPL

Ans:

The Node.js REPL, which stands for "Read-Eval-Print Loop," is an interactive command-line environment that allows developers to experiment with and execute JavaScript code in an interactive and immediate manner.

Features of Node.js REPL:

Interactive Environment: The Node.js REPL provides an interactive environment where developers can enter JavaScript code and see the results instantly. It's particularly useful for quick code testing, prototyping, and experimenting with language features.

Read-Eval-Print Loop: The term "REPL" describes the core functionality of the environment. It reads user input (JavaScript code), evaluates the code, prints the result to the console, and then waits for the next input. This loop continues until the user exits the REPL.

Built-In to Node.js: The Node.js REPL is built directly into the Node.js runtime, so you don't need to install any additional software to use it. You can access the REPL by opening your terminal or command prompt and typing node followed by the Enter key.

Access to Node.js Features: In addition to standard JavaScript functionality, the Node.js REPL provides access to Node.js-specific modules and features. This means you can experiment with file I/O, networking, and other Node.js capabilities directly from the REPL.

Multi-Line Input: The Node.js REPL allows you to enter multi-line code blocks. You can start a multi-line block by typing .editor and then pressing Enter. This mode provides a more convenient way to write and test complex JavaScript code.

Tab Completion: The REPL supports tab completion, which can be helpful for exploring available functions, methods, and object properties. Simply press the Tab key to see the available options.

Documentation and Help: You can access documentation and help directly from the REPL by typing help or . followed by the method or object you want to learn more about. This is a valuable resource for quickly understanding JavaScript and Node.js functionality.

Educational Tool: The Node.js REPL is often used as an educational tool for teaching and learning JavaScript. It allows beginners to experiment with code snippets and see immediate results, which can aid in the learning process.

The Node.js REPL is a valuable tool for developers working with JavaScript and Node.js. It offers an interactive environment for testing code, exploring language features, and quickly trying out ideas without the need for a separate development environment. Whether you're a beginner learning JavaScript or an experienced developer debugging and experimenting, the Node.js REPL is a handy resource in your toolkit.

Q.3] Explain Routing in ExpressJS along with an example.

Ans:

Routing in Express.js is the process of defining how an application responds to client requests to different endpoints or URLs. Express.js provides a flexible and powerful routing system that allows you to define routes for your web application and specify how each route should handle different HTTP methods (such as GET, POST, PUT, DELETE, etc.). Here's an explanation of routing in Express.js and some commonly used HTTP methods:

Routing in Express.js:

A router can be created using the express.Router() function. This router can be used to define routes and their associated handlers.

Defining Routes: We can define routes for different URLs by using the router's HTTP methods (get, post, put, delete, etc.) and specifying a callback function (handler) that executes when a request to that route is made

Middleware: Can also attach middleware functions to routes. Middleware functions are executed before the route's handler and can be used for tasks like authentication, logging, or data validation. Middleware can be applied globally or to specific routes. For example:

Route Parameters: Express.js allows you to define routes with parameters, which capture values from the URL and make them accessible in the request object.

HTTP Methods in Express.js:

GET: The GET method is used to request data from a specified resource. It is commonly used for fetching data and rendering web pages.

POST: The POST method is used to send data to the server for processing. It is commonly used for submitting forms or creating new resources.

PUT: The PUT method is used to update an existing resource or create one if it doesn't exist. It typically updates the entire resource.

DELETE: The DELETE method is used to request the removal of a resource. It is used for deleting specific resources on the server.

PATCH: The PATCH method is used to apply partial modifications to a resource. It is often used for updating only specific fields of a resource.

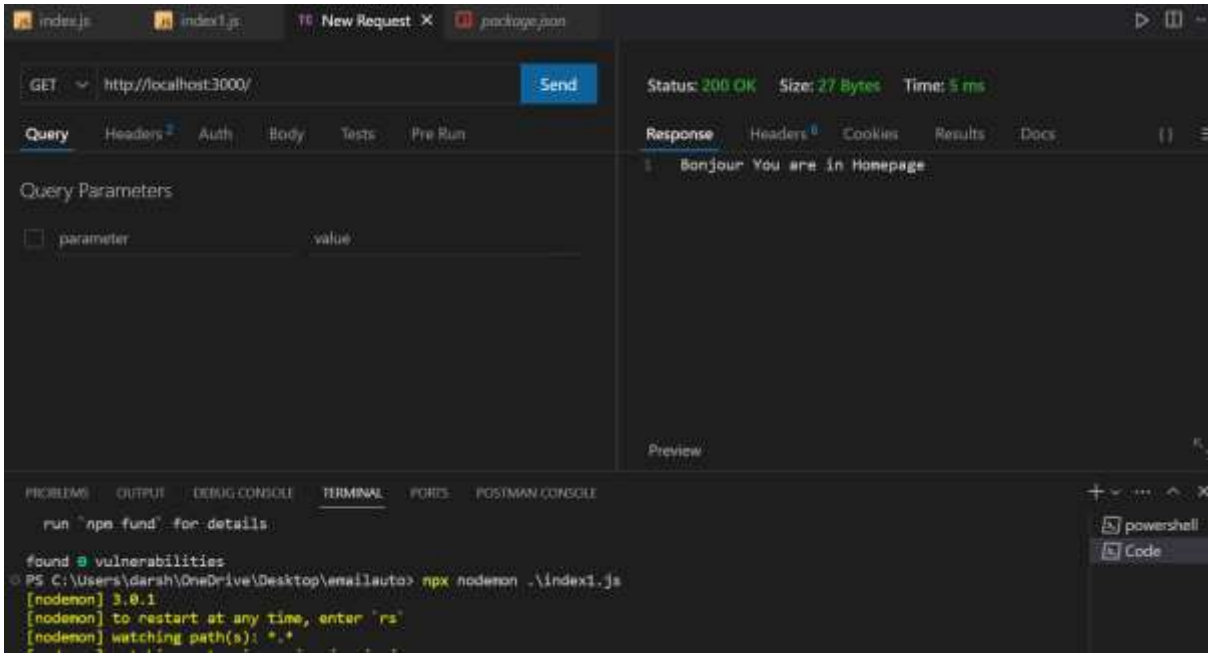
HEAD: The HEAD method is similar to GET but returns only the headers of the response, without the actual data. It is useful for checking the existence and metadata of a resource.

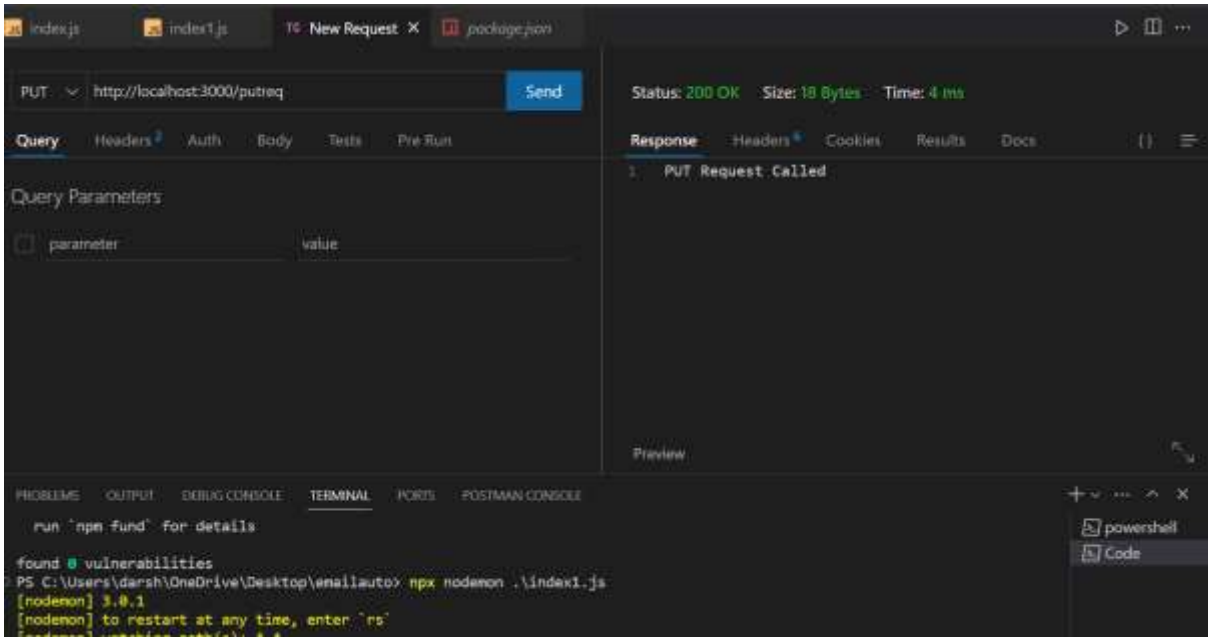
These are the fundamental concepts of routing and HTTP methods in Express.js. Express provides a robust and flexible framework for building web applications and APIs, allowing you to define routes and handle various HTTP methods to create dynamic and interactive web services.

The following code explains these methods:

```
const express = require('express');
const app = express();
const port = 3000;

let arr_darsh = [];

app.use(express.json());

// GET method
app.get('/', (req, res) => {
  res.send("Bonjour You are in Homepage");
});

// POST method
app.post('/enterdata', (req, res) => {
  const newTask = req.body;
  arr_darsh.push(newTask);
  res.status(201).json(newTask);
});

// PUT method
app.put('/putreq', (req, res) => {
        res.send("PUT Request Called")
})

// DELETE method
app.delete('/delete', (req, res) => {
  res.send("DELETE ROUTE")
});

// Start the Express server
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

Output:


index.js
index1.js
New Request
package.json
GET
http://localhost:3000/
Send
Status: 200 OK
Size: 27 Bytes
Time: 5 ms
Query
Headers 2
Auth
Body
Tests
Pre Run
Response
Headers 8
Cookies
Results
Docs
Query Parameters
parameter
value
1 Bonjour You are in Homepage
Preview
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
POSTMAN CONSOLE
run `npm fund` for details
found 0 vulnerabilities
PS C:\Users\darsh\OneDrive\Desktop\emailauto> npx nodemon .\index1.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
powershell
Code


index.js
index1.js
New Request
package.json
PUT
http://localhost:3000/putreq
Send
Status: 200 OK
Size: 18 Bytes
Time: 4 ms
Query
Headers 2
Auth
Body
Tests
Pre Run
Response
Headers 6
Cookies
Results
Docs
Query Parameters
parameter
value
1 PUT Request Called
Preview
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
POSTMAN CONSOLE
run `npm fund` for details
found 0 vulnerabilities
PS C:\Users\darsh\OneDrive\Desktop\emailauto> npx nodemon .\index1.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
powershell
Code