Animesh Parab T2-T21 88

Assignment No. 2

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

Ans. Refs in React are a way to access and interact with the DOM (Document Object Model) or to reference a React component instance directly. They provide a means to escape the usual data flow and can be used sparingly for specific tasks that require direct interaction with the DOM or other React components. Here's an overview of when to use refs and when not to use them:

When to Use Refs:

Accessing DOM Elements: Refs are commonly used to access and manipulate DOM elements directly. For example, you might use refs to focus an input field, scroll to a specific part of a page, or trigger animations associated with a particular element.

Integrating with Third-Party Libraries: When working with third-party libraries, especially those that rely on direct DOM manipulation, refs can be useful. They allow you to bridge React with these external libraries, accessing and controlling their elements.

Managing Media Playback: Refs can be employed to manage media elements like audio or video. You can play, pause, or manipulate the media directly using the ref.

Custom Component Communication: In some advanced scenarios, you might need to reference child components to communicate or trigger specific actions. Refs can be used for this purpose.

When Not to Use Refs

State Management and Data Flow: Refs should not be used for managing application state or data flow. Instead, rely on React's built-in state management (such as setState) and props to pass data and trigger updates between components. Proper data flow helps maintain the predictability of your application.

Conditional Rendering: Avoid using refs to conditionally render or hide components. React's declarative nature allows you to handle conditional rendering through state or props, making your code more understandable.

Functional Components and Hooks: While you can use refs with class components as shown above, in functional components, it's recommended to use the useRef hook to achieve similar functionality. Be cautious with the usage of useRef in functional components, and consider if the same functionality can be achieved with state and props, which align better with React's functional paradigm.

Layout and Styling: Refs should not be used to directly modify the layout or styling of elements. React is designed to handle rendering efficiently, and altering styles directly through refs can lead to unexpected results. Instead, use React state and CSS styles.

2. Write short note on:

a) NPM (Node Package Manager):

NPM is the default package manager for Node.js, a popular runtime environment for running JavaScript on servers and for building web applications. NPM serves as a central repository for thousands of open-source packages and libraries that can be easily installed and managed in your Node.js projects. Here are some key points about NPM:
Package Management: NPM is primarily used for managing and installing Node.js packages. Developers can use it to install packages, manage dependencies, and update libraries in their Node.js applications.
Command-Line Tool: NPM provides a command-line interface for interacting with packages. You can use commands like npm install, npm init, and npm update to handle various aspects of package management.
Package.json: Each Node.js project typically includes a package.json file, which contains metadata about the project, as well as a list of its dependencies. NPM uses this file to manage project dependencies and scripts.
Version Control: NPM allows you to specify version ranges for your dependencies to ensure compatibility and stability within your project. It also provides semantic versioning (SemVer) for package version management.
Global and Local Installation: NPM can install packages globally (available across the system) or locally (within a specific project directory) depending on your needs.
Scripts: NPM lets you define custom scripts in your package.json file. These scripts can be executed using the npm run command and are often used for tasks like building, testing, and deploying applications.
b) REPL (Read-Eval-Print Loop):

A REPL is an interactive programming environment commonly used for interpreted programming languages like JavaScript. It provides a simple way to interactively execute code, see immediate results, and experiment with language features. Here's a brief overview of REPL:
Interactive Environment: The REPL is a command-line interface or tool where you can enter code, which is immediately executed, and the results are displayed. It's a great way to quickly test and prototype code.
Debugging and Testing: Developers often use a REPL for debugging code snippets, testing small code fragments, and exploring language features. It's particularly useful for learning and experimenting with a programming language.
Immediate Feedback: The primary advantage of a REPL is the immediate feedback it provides. As you enter code, it's evaluated and the results are displayed, allowing you to make adjustments and see the impact in real-time.
Available for Many Languages: While commonly associated with languages like Python and JavaScript, REPLs exist for a wide range of programming languages, each tailored to the specific language's features and syntax.

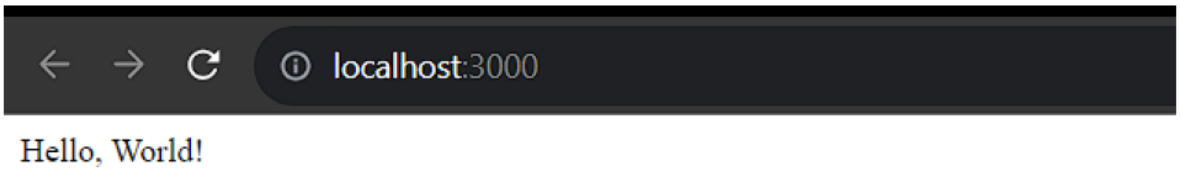
3. Explain Routing in ExpressJS along with an example

Routing in Express.js is a fundamental concept that allows you to define how your application responds to different HTTP requests. It helps you map specific URLs (or routes) to particular functions or handlers, enabling you to create a structured and organized web application with various endpoints. Express.js provides a flexible and easy-to-use routing system.

Example of Routing in Express.js:

```
const express = require('express');
const app = express();
const port = 3000;
// Route for the homepage
app.get('/', (req, res) => {
res.send('Hello, World!');
});
// Route for echoing a user-provided message
app.get('/echo/:message', (req, res) => {
const message = req.params.message;
res.send(`You said: ${message}`);
});
app.listen(port, () => {
console.log(`Server is running on port ${port}`);
});
```

Output :



localhost:3000

```
Hello, World!
```