### React-JS HOL

### Hands-on-9

### 1. ****List the Features of ES6:****

ES6 (ECMAScript 2015) introduced many new features to JavaScript. Key features include:

-let and const for block-scoped variable declarations.

-Arrow functions (=>) for shorter function syntax and lexical this.

-Template literals using backticks (` ``) for multi-line strings and interpolation.

-Default parameters in functions.

-Destructuring assignment for arrays and objects.

-Enhanced object literals (shorter syntax, computed property names).

-Classes and class inheritance (class, extends, super).

-Promises for asynchronous programming.

-Modules using import and export.

-New data structures: Map, Set, WeakMap, and WeakSet.

-Spread (...) and Rest (...) operators.

### 2. ****Explain JavaScript**** let****:****

The let keyword in JavaScript declares variables that are **block-scoped**. This means they are only accessible within the block ({}) in which they are defined.

Unlike var, let does not allow re-declaration within the same scope and avoids hoisting issues, improving code reliability and readability.

### 3. ****Identify the Differences Between**** var ****and**** let****:****

| **Feature** | **var** | **let** |
| --- | --- | --- |
| Scope | Function-scoped | Block-scoped ({}) |
| Re-declaration | Allowed within the same scope | Not allowed in the same scope |
| Hoisting | Hoisted (initialized as undefined) | Hoisted but not initialized |
| Global Object | Adds to window object | Does **not** add to window |
| Use case | Older code, legacy JS | Modern JS development |

4. **Explain JavaScript** const**:**

const declares block-scoped constants. Variables declared using const **must be initialized** at the time of declaration and **cannot be reassigned**.

However, const **does not make objects immutable**—only the binding is constant. You can still modify the properties of objects declared with const.

5. **Explain ES6 Class Fundamentals:**

ES6 introduced the class syntax to create objects using a cleaner, more readable structure.

**Syntax:**

Javascript:

class Person {

constructor(name) {

this.name = name;

}

greet() {

return `Hello, ${this.name}`;

}

}

constructor initializes object properties.

-Methods are defined directly inside the class.

-Classes use new to create instances.

-Syntactic sugar over prototypes.

### 6. ****Explain ES6 Class Inheritance:****

ES6 supports inheritance using the extends and super keywords.

**Example:**

Javascript:

class Animal {

constructor(name) {

this.name = name;

}

speak() {

return `${this.name} makes a sound.`;

}

}

class Dog extends Animal {

speak() {

return `${this.name} barks.`;

}

}

-extends allows a class to inherit from another.

-super() calls the parent class constructor.

-Child class can override parent methods.

### 7. ****Define ES6 Arrow Functions:****

Arrow functions (=>) provide a shorter syntax for writing functions and **do not bind their own** this.

**Syntax:**

Javascript:

const add = (a, b) => a + b;

Features:

-Concise syntax.

-Lexical this binding (inherits from parent scope).

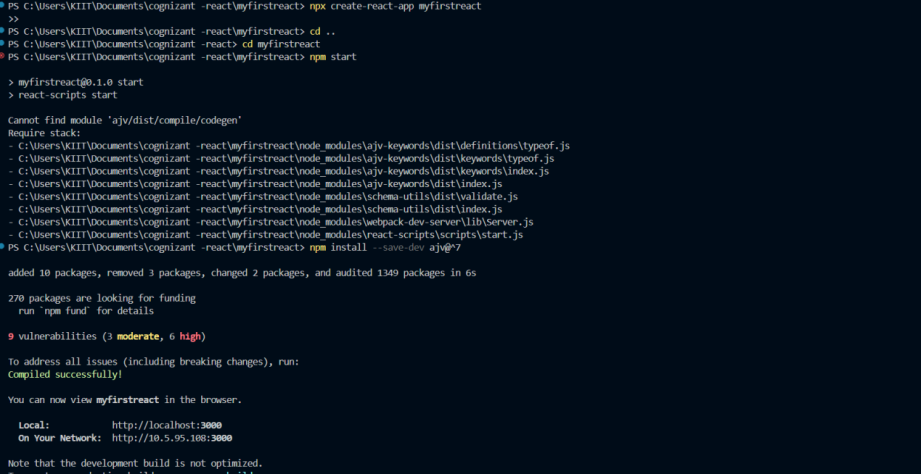
-Cannot be used as constructors.

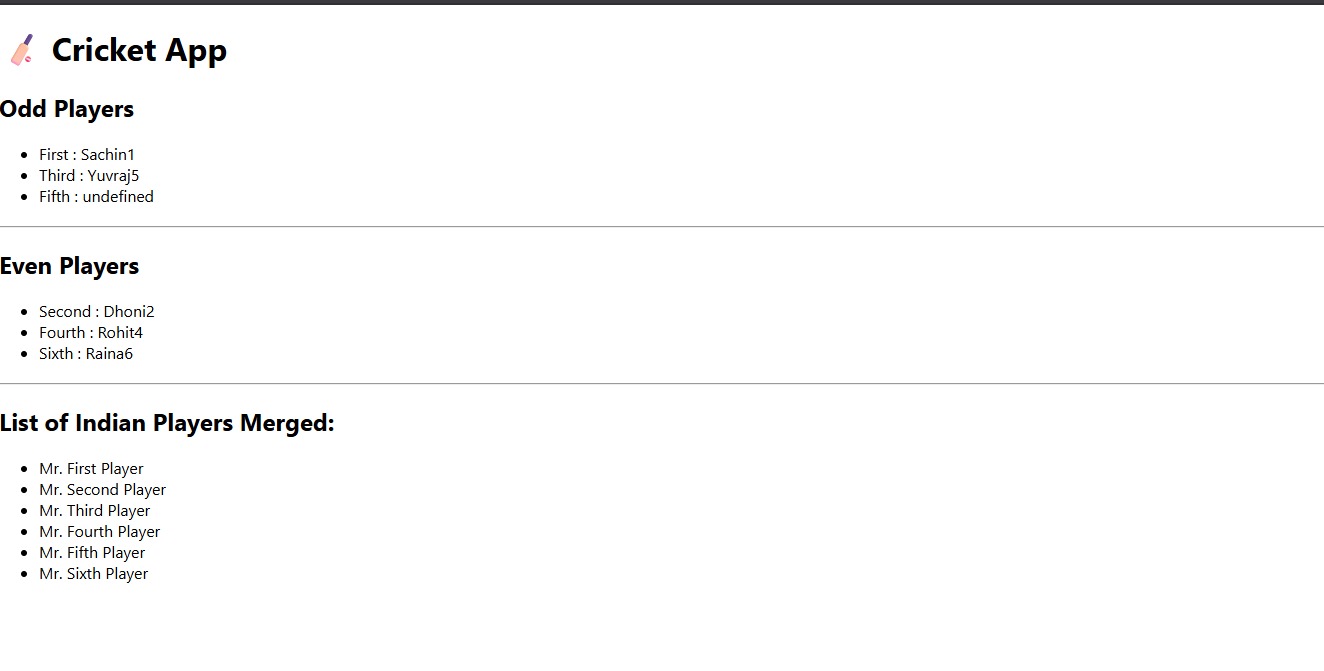
-No arguments object.

### 8. ****Identify**** set() ****and**** map()****:****

#### Set: A collection of **unique values**. Maintains insertion order. Supports operations like add(), has(), delete().

#### Map: A collection of **key-value pairs**. Keys can be any type (not just strings). Maintains insertion order. Methods: set(), get(), has(), delete().





### Hands-on-10

### 1. ****Define JSX:****

JSX (JavaScript XML) is a syntax extension for JavaScript used with React to describe what the UI should look like. It allows writing HTML-like code within JavaScript and is transpiled to standard JavaScript using tools like Babel.

### 2. ****Explain about ECMAScript:****

ECMAScript is the standard specification on which JavaScript is based. It defines the rules, syntax, and features of the language to ensure consistency across different implementations and platforms.

### 3. ****Explain**** React.createElement()****:****

React.createElement() is a method provided by React to create virtual DOM elements. It returns a JavaScript object representing a DOM node which React uses to render UI.

4. **Explain How to Create React Nodes with JSX:**

React nodes can be created using JSX by writing HTML-like syntax inside JavaScript, which React converts into virtual DOM elements for rendering UI components.

### 5. ****Define How to Render JSX to DOM:****

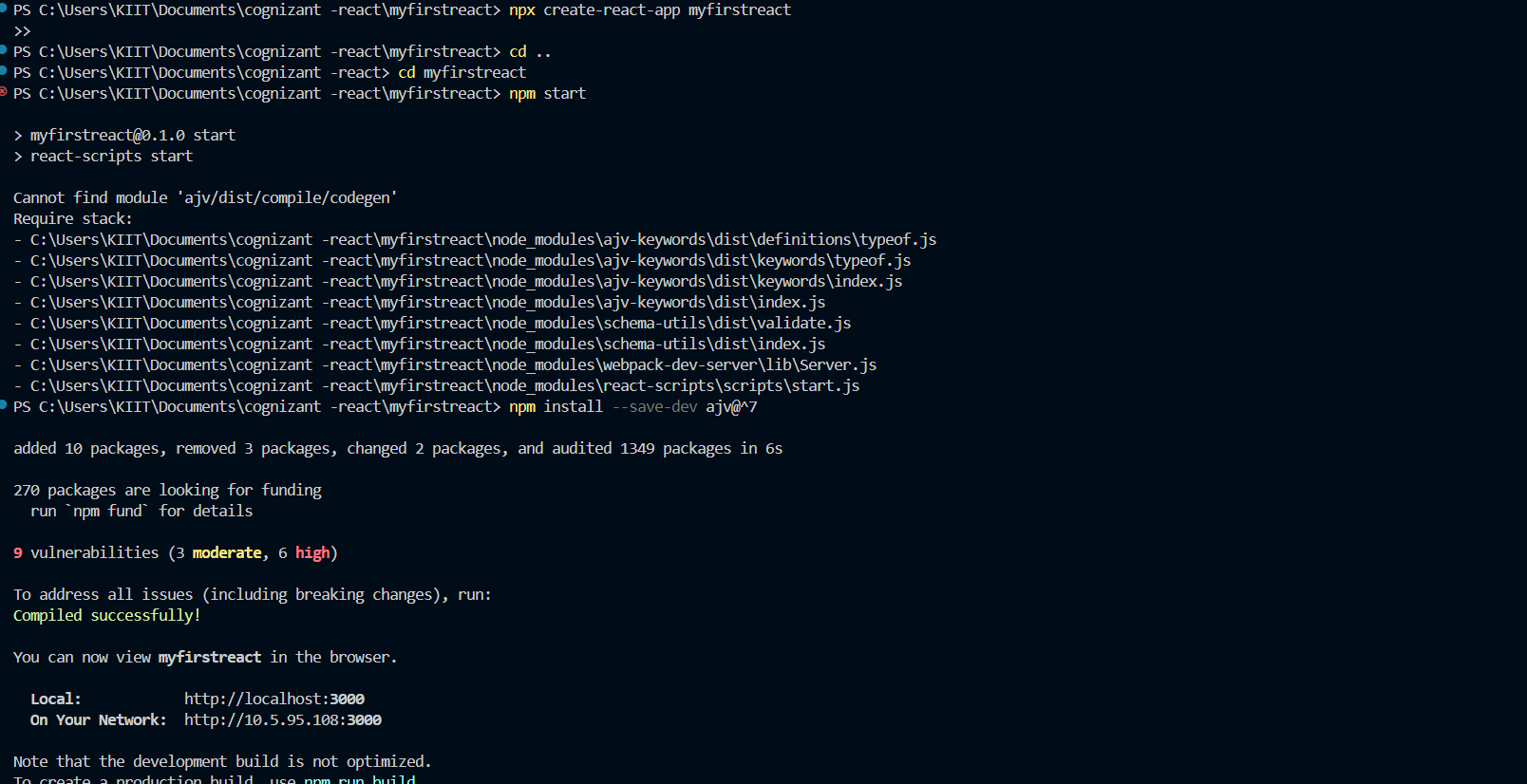
JSX is rendered to the DOM using React’s rendering methods, which attach the virtual DOM elements to a real DOM node in the browser, enabling the display of UI components.

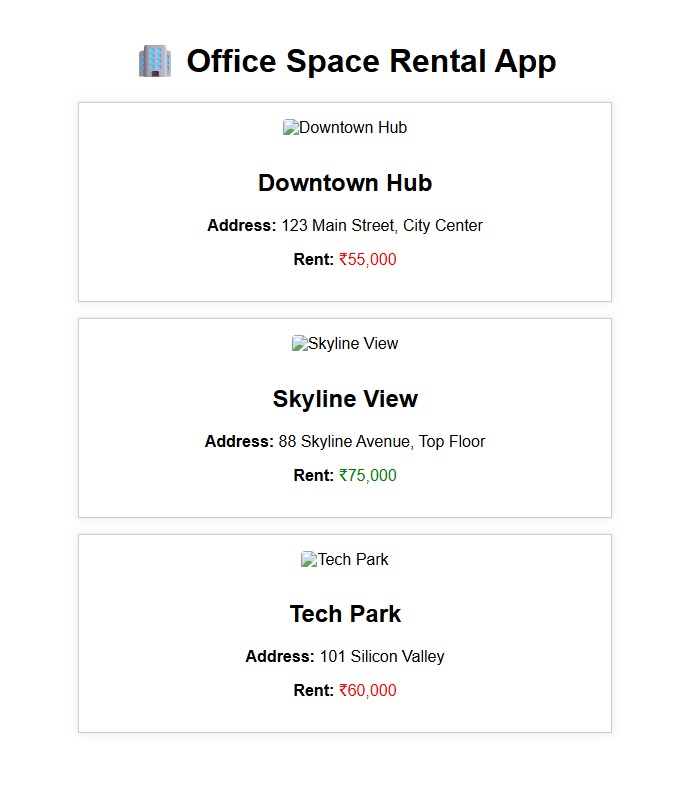
### 6. ****Explain How to Use JavaScript Expressions in JSX:****

JavaScript expressions can be embedded inside JSX using curly braces. This allows dynamic values and logic to be inserted directly within the markup.

### 7. ****Explain How to Use Inline CSS in JSX:****

Inline CSS in JSX is applied by passing a JavaScript object to the style attribute, with CSS property names written in camel Case format and values as strings.

****

****

### Hands-on-11

### 1. ****Explain React Events:****

React events are the way to handle user interactions such as clicks, form submissions, key presses, etc., in React components. They are similar to DOM events but follow the React-specific event system.

2. **Explain About Event Handlers:**

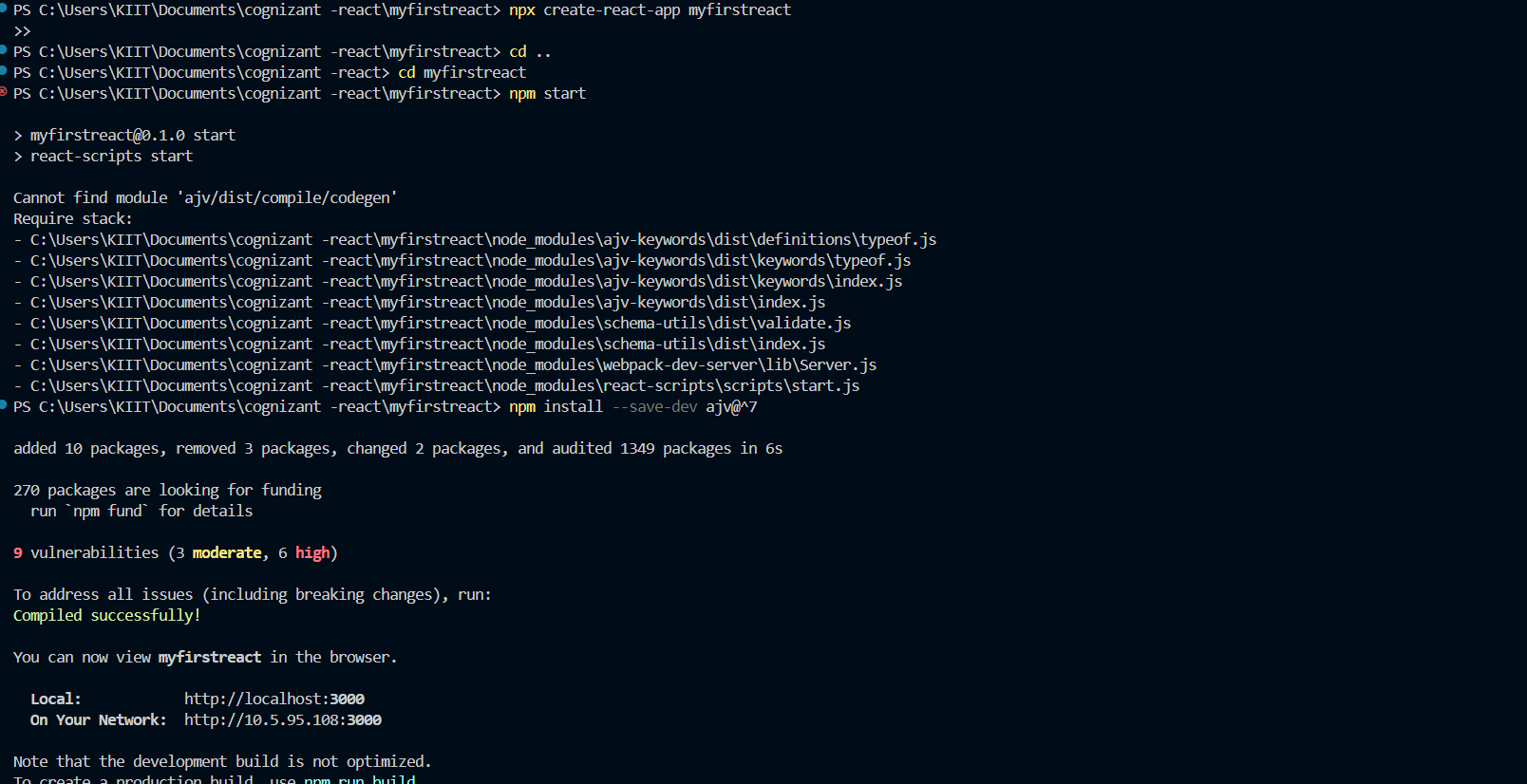
Event handlers in React are functions that are triggered when a specific event occurs. They define what action should be performed in response to user interactions or browser events.

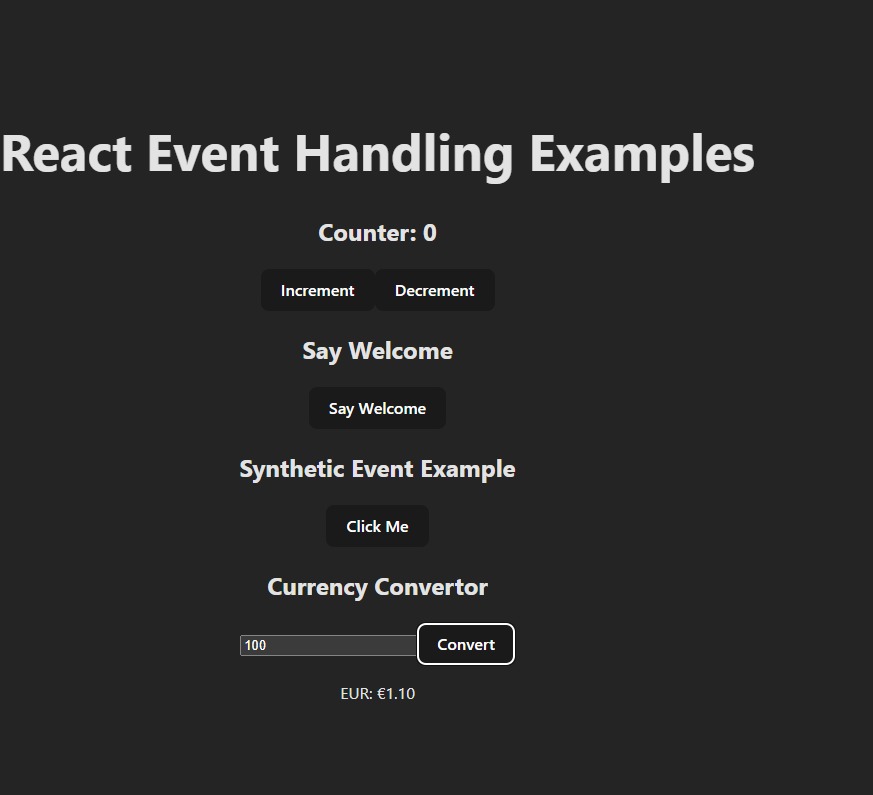
3. **Define Synthetic Event:**

A Synthetic Event is a cross-browser wrapper around the browser’s native event system provided by React. It normalizes events to ensure consistency across different browsers.

4. **Identify React Event Naming Convention:**

React uses camelCase for event names instead of lowercase. For example, onClick instead of onclick, and the event handler is passed as a function reference.

****

****

### Hands-on-12

### 1. ****Explain About Conditional Rendering in React:****

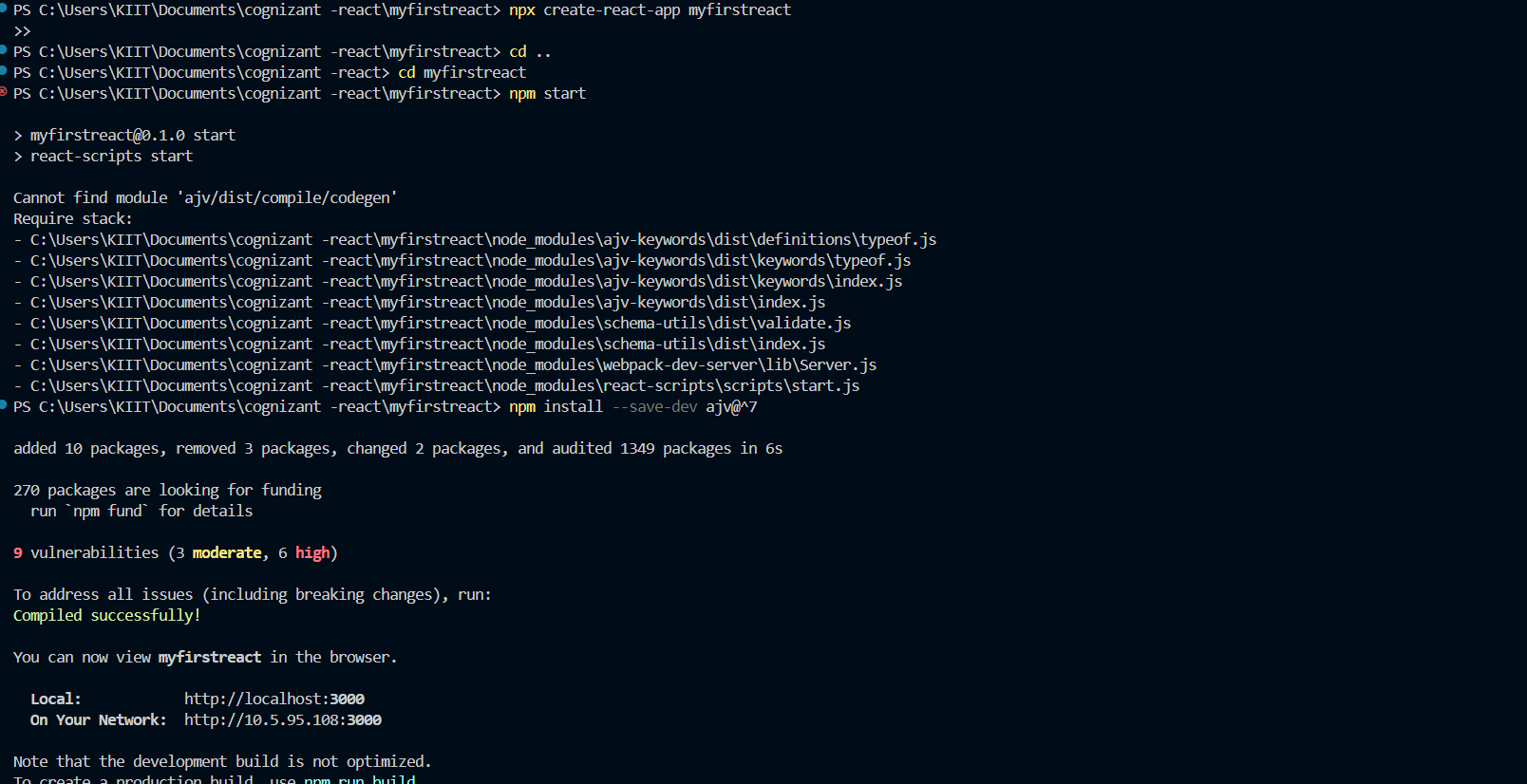
Conditional rendering in React refers to the technique of displaying different UI elements or components based on certain conditions or logic, using JavaScript constructs like if, ternary operators, or logical &&.

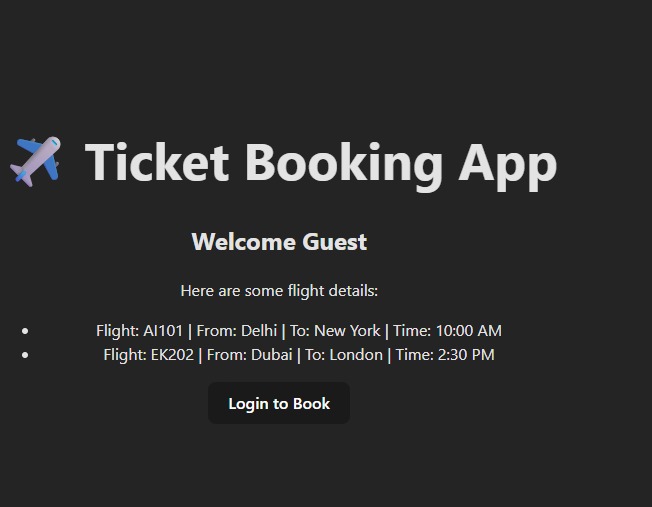
2. **Define Element Variables:**

Element variables in React are variables used to store JSX elements. They allow dynamic rendering by assigning different JSX based on conditions or logic within the component.

3. **Explain How to Prevent Components from Rendering:**

Components in React can be prevented from rendering by returning null or using conditional logic to exclude them from the JSX output. Returning null means the component renders nothing and has no effect on the DOM.

****

****

### Hands-on-13

### 1. ****Explain Various Ways of Conditional Rendering:****

Conditional rendering in React can be done using several methods, such as if-else statements, ternary operators (condition ? true : false), logical && operators, and switch statements to control which elements or components should be displayed.

2. **Explain How to Render Multiple Components:**

Multiple components in React can be rendered by placing them together inside a parent component, using fragments (<> </>), arrays of elements, or wrapping them in a common container element like div.

3. **Define List Component:**

A list component in React is a component used to display a collection of data items by iterating over arrays and rendering individual items, typically using the map() function.

4. **Explain About Keys in React Applications:**

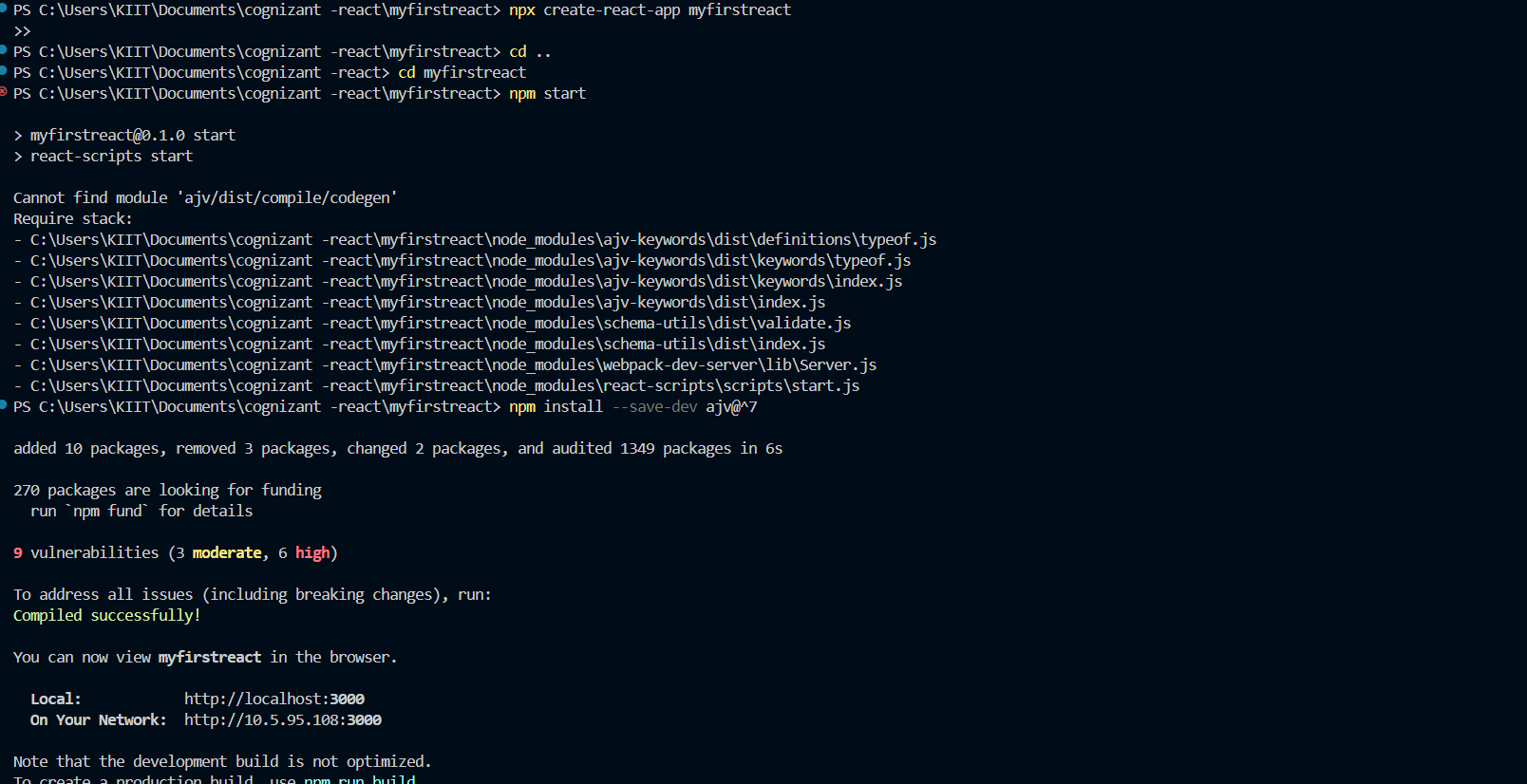
Keys are unique identifiers used in React lists to help React identify which items have changed, are added, or removed. They improve rendering performance and must be unique among siblings.

5. **Explain How to Extract Components with Keys:**

Components can be extracted inside the map() function when rendering lists, and each extracted component should be assigned a unique key prop to maintain identity across re-renders.

6. **Explain React Map,** map() **Function:**

In React, the map() function is used to iterate over arrays and return a new array of JSX elements. It helps render lists of components dynamically based on data collections.

****

****