Build a Dynamic Web Site by using React

1. brief note on different key press and key down events
2. keypress event is fired when a key is pressed down and that key normally produces a character value
3. write a brief note on event delegation
4. write a program in JavaScript to count number of characters.
5. Write a program in JavaScript to limit the string with 12

Date: -22-01-2-25 Day 2

```````````````````````````````````````````````````````````````````````````````````````````````````````````

Web-vital module Not found issue at the Run time

* Npm install –save web-vitals

React structure

* Component is set of package
* Package is s set of functions
* Functions is set of instructions

**Mounting (Starting Phase)**

* **Constructor:**  
  When we create a constructor, it helps set up the object. The object is created only when the constructor runs.  
  Example: It’s like setting up a new house when you move in, arranging furniture (data).
* **Storing Data:**  
  After the constructor, the object is initialized with some data. This is called "mounting." It's like filling the house with your belongings.

**2. Updating (Changing Phase)**

When the data in the object is updated, it's called "updating." Think of it like rearranging or adding new furniture in the house.

**3. Unmounting (Ending Phase)**

When the object is removed or destroyed, it’s called "unmounting." It’s like moving out of the house and emptying it.

This process (Mounting → Updating → Unmounting) describes the lifecycle of an object or component in simple terms.

**What are Props?**

* **Props (short for Properties)** are used to pass data **from a parent component to a child component** in a one-way flow.
* **Immutable**: Once passed, the child component **cannot change** the value of props directly. The parent controls the data.

**Hooks (in React)**

Hooks are special functions in React that let you add features like **state**, **side effects**, and more to functional components (without needing classes).

**Common Hooks:**

* **useState:**  
  Used to manage state (data that changes over time).  
  Example: Keep track of a counter value.
* **useEffect:**  
  Handles side effects (e.g., fetching data, updating the DOM, setting up timers). Runs after rendering.
* **useContext:**  
  Helps share data (like theme, user info) between components without passing props manually.
* **useReducer:**  
  A more powerful way to manage complex state logic (like state actions and reducers).

**2. State Management**

State management is how you **control and share data** in your app efficiently, especially when many components need the same data.

**Approaches to State Management:**

* **React State (useState):**  
  Use it for simple, local state in a single component.
* **Context API:**  
  Share state globally across the app without passing props manually through every component.
* **Redux (or Redux Toolkit):**  
  A powerful library for managing global state with strict control over how state is updated. Ideal for large-scale apps.
* **MobX , Zustand, Recoil, etc.:**  
  Alternative libraries to manage state, offering more simplicity or flexibility depending on your project’s needs.