Unit 4:-

1.what is react.write a step of installation.

- React is a JavaScript Library known for front-end development
- ReactJS is an open-source JavaScript library for building dynamic, single-page applications and creating reusable interactive UI components.
- React is developed and managed by META
- ReactJS was developed by Jordan Walke
- Feature:-
    - **Virtual DOM:**
      React uses a copy of the real DOM called **Virtual DOM**. When something changes on the page, React compares the new Virtual DOM with the old one and updates **only the changed part** in the real DOM. This makes React **fast and efficient**.

    - **Reusable Components:**
      In React, we write small parts of UI called **components**. These components can be used **again and again** in different parts of the application, which saves time and reduces code repetition.

    - **One-Way Data Binding:**
      React follows **one-way data flow**, meaning data moves from **parent to child**. This makes it easier to understand how data is changing and helps in **managing the app's state** properly.

    - **JSX (JavaScript XML):**
      JSX allows us to write **HTML-like code** inside JavaScript. It makes the code easier to write and understand. Though not required, JSX is **highly recommended** in React.

- **Components:**
  React apps are made of **multiple components**, and each component has its **own logic and UI**. These components are easy to **maintain and reuse**, especially in large projects.

- **Simplicity:**
  React's use of JSX and components makes the code **simple, readable, and easy to learn**. Its structure helps in building large applications with **less effort**.

- Step 1:-install node js
  - After installation to verify
    - Node -v
    - Npm -v
- Step 2:- install create react app
  - Npm install -g create-react-app
  - To verify create-react-app –version
- Step 3:- to select the folder
- Step 4:-create react app
  - Npx create-react-app demo
- Step 5:-move to demo.
  - cd demo
- Step 6:- run the program.
  - Npm start

2.what is reactdom.

- ReactDOM is a core react package that provides methods to interact with the Document Object Model or DOM.
- This package allows developers to access and modify the DOM.
- To use reactdom then use ReactDom module.
- There are four methods in reactdom.

- ReactDOM.render():-
  - This method was used to render a React component into a specified DOM node.
  - Syntax:
    ReactDOM.render(element, container, [callback])
- ReactDOM.createRoot():-
  - ReactDOM.createRoot() is used to initialize a root for the app.
  - It is used for rendering and handling state updates efficiently.
  - const root = ReactDOM.createRoot(document.getElementById('root'));
    root.render(<App />);
- ReactDOM.hydrate():
  - This is used to hydrate an app on the client side when server-side rendering (SSR) is involved.
  - ReactDOM.hydrate(<App />, document.getElementById('root'));
- ReactDOM.unmountComponentAtNode():
  - This is used to unmount a React component from the DOM, removing it and its event listeners.
  - Syntax:-
    - ReactDOM.unmountComponentAtNode(container)
  - Ex:-
    - ReactDOM.unmountComponentAtNode(document.getElementById('root'));

3.what is the component.

- A component is a piece of the UI (user interface) that has its own logic and appearance.
- React component names must always start with a capital letter, while HTML tags must be lowercase.
- Types of components.
    - In ReactJS, mainly two types of components. They are
      Function Components
      Class Components
- Function component:-
    - These are **simple JavaScript functions**.
    - They **accept props** (data) and **return JSX**.
    - Syntax:-
        - function function_name(argument_name) {
          function_body;
          }
    - Ex:-
    - import React from 'react';
      import ReactDOM from 'react-dom/client';
      import './index.css';
      import App from './App';

      function Car()
      {
      return(

      <h1>This is a car</h1>
      It is BMW

      );
      }

```
const root =
ReactDOM.createRoot(document.getElementById('root'))
;
root.render(<Car />);
```

- Class component:-
  - Class components are ES6 classes that extend from React.Component or React.PureComponent.
  - They have a render() method where you define the structure of your component's UI using JSX.
  - Syntax:-
    - ```
      class class_name extends React.Component {
          render() {
              return <<element>>
          }
      }
      ```
  - Ex:-
    - ```
      import React from 'react';
      import ReactDOM from 'react-dom/client';
      class Car extends React.Component {
      render() {
      return <h2>Hi, I am a Car!</h2>;
      }
      }
      const root =
      ReactDOM.createRoot(document.getElementById('
      root'));
      root.render(<Car />);
      ```

4.what is constructor?

- constructor() is a function in your component and this function will be called when the component gets initiated.
- Syntax:-

```
constructor(){  }
```

- Ex:-
- ```
  import React from 'react';
  import ReactDOM from 'react-dom/client';
  class Car extends React.Component {
  constructor() {
  super();
  this.state = {color: "red"};
  }
  render() {
  return <h2>I am a {this.state.color} Car!</h2>;
  }
  }
  const root =
  ReactDOM.createRoot(document.getElementById('root'));
  root.render(<Car />);
  ```

5.what are props?

- props stands for properties.
- React Props are like function arguments in JavaScript and attributes in HTML.
- Ex:-
  - ```
    import React from 'react';
    import ReactDOM from 'react-dom/client';

    function Car(props)
    {
    return(
    <div>
    <h1>This is a {props.name} car</h1>
    <h3>It's color is {props.color}</h3>
    </div>
    );
    ```

```
}
const root =
ReactDOM.createRoot(document.getElementById('root'))
;
root.render(<Car name="BMW" color="red" />);
```

6.what is state?

- **State** is an object in React that stores **changing data** (like user input or clicks).

- It is used to make the **component dynamic**, meaning it can **update and show new data**.

- When the **state changes**, the component **automatically updates** (re-renders) on the screen.

- We use `this.setState()` to **change the value** of the state.

- `setState()` doesn't remove the old state — it just **adds or updates** the new values.
- Ex:-
    - ```
      import React from 'react';

      class Counter extends React.Component {
        constructor() {
          super();
          this.state = {
            count: 0
          };
        }

        increment = () => {
          this.setState({ count: this.state.count + 1 });
      ```
      ```

```
  };

  render() {
    return (
      <div>
        <h2>Count: {this.state.count}</h2>
        <button onClick={this.increment}>Increase</button>
      </div>
    );
  }
}

export default Counter;
```

7.what react lifecycle?

- React components go through different stages from **creation** to **deletion**. These stages are called the **component lifecycle**.
- **Mounting:-**
  - **Component is being created and added to the DOM**
  - **Methods:**
    - `constructor()` – Initializes the component.
    - `getDerivedStateFromProps()` – Sets state based on props (rarely used).
    - `render()` – Returns the JSX to be shown.
    - `componentDidMount()` – Runs after the component is added to the DOM. Good for API calls.

- **Updating**
  - **Component updates due to state/props change**
  - **Methods:**
    - `getDerivedStateFromProps()` – Again, runs before rendering.
    - `shouldComponentUpdate()` – Lets you decide whether to re-render.
    - `render()` – Renders updated JSX.
    - `getSnapshotBeforeUpdate()` – Captures info (like scroll position) before update.
    - `componentDidUpdate()` – Runs after the component is updated.

- **Unmounting**
  - **Component is removed from the DOM**
  - **Method:**
    - `componentWillUnmount()` – Runs before the component is destroyed. Used for cleanup like clearing timers or listeners.

- Ex:-
  - class Demo extends React.Component {
    constructor() {
    super();
    console.log("Constructor");
    }

    componentDidMount() {

```
    console.log("Component Mounted");
  }

  componentDidUpdate() {
    console.log("Component Updated");
  }

  componentWillUnmount() {
    console.log("Component Will Unmount");
  }

  render() {
    return <h1>Hello Lifecycle</h1>;
  }
}
```

-
-

8.what is local storage?

- localStorage is commonly used to store data locally in the user's browser.
- The localStorage object allows you to save key-value pairs in the browser.
- The localStorage object stores data with no expiration date.
- Operation of local storage:-
  - setItem():
    - This method is used to add a key and a value to localStorage.
    - Syntax
      - localStorage.setItem(‘key’, ‘value’);
    - Example:

        - localStorage.setItem('username', 'Jack');
- getItem():
    - This method is used to get an item from localStorage using the key.
    - Syntax:
        - localStorage.getItem(‘key’);
    - Example:
        - localStorage.getItem('username');

- removeItem():
    - This technique is used to delete an item from localStorage based on its key.
    - Syntax:
        - localStorage.removeItem(‘key’);
    - Example:
        - localStorage.removeItem('username');

- clear():
    - This technique is used to delete all instances of localStorage.
    - Syntax:
        - localStorage.clear();
    - Example:
        - localStorage.clear();

- key(): When you supply a number, it aids in the retrieval of a localStorage key.
-

8.what is the react event?

- In React, events are actions that happen in the browser, like clicking a button, submitting a form, or pressing a key.
- onClick : Triggered when mouse button clicked.
- onDoubleClick : Triggered when a mouse button double clicked.
- onMouseEnter : Triggered when the mouse pointer enters an element.
- onMouseLeave: Triggered when the mouse pointer leaves an element.
- onMouseOver: Triggered when the mouse pointer moves over an element.
- onMouseDown : Triggered when a mouse button is pressed down.
- onMouseOut: Triggered when the mouse pointer moves out of an element.
- onMouseUp : Triggered when a mouse button is released.
- onMouseMove : Triggered when the mouse is moved over an element (while inside the bounds
- of that element).

9.what is lifting state up?

- **Lifting State Up** means moving the **state** from a **child component to a common parent component** so that **multiple child components** can **share and sync data**.

Why It's Needed:

- When **two or more components** need to share the same data.

- Keeping state in **one common parent** makes data flow and updates easier and more consistent.

How It Works:

1. Create the **state** in the **parent component**.

2. Pass the **state and updater function** to child components via **props**.

3. Children can call the **updater function** to update the parent's state.

4. All child components using that state will **re-render** accordingly.

- Ex:- see online:-

10.what is composition and inheritance?

- Composition and inheritance are the approaches to use multiple components together in React.js .
- This helps in code reuse.
- Composition:-
  - Composition means to build complex UIs by combining simpler components.
- Inheritance:-
  - **Inheritance** is when one class inherits properties and methods from another class. While inheritance is common in object-oriented programming, **React avoids it** for UI components because it becomes hard to manage and reuse code.
- Ex:- to see

11. what fetch api and explain example.
    - The Fetch API is used to make HTTP requests to retrieve or send data from a server.
    - Syntax:-
        - fetch('https://api.example.com/data')
          .then(response => response.json())
          .then(data => console.log(data));
    -