■ Event of the Day Date:...../...../.....

MERN STACK

REACT.JS :

Javascript: used for making web pages interactive & bring the web-applications to life. React.js is a library.

MEAN	MERN
⇒ Speeds up the process of web & mobile app development.	It ensures a very smooth development process.
⇒ Angular is framework	React is a library.
⇒ Flow of data is bi-directional.	Flow is uni-directional

REACT.JS:

⇒ It is a Javascript library for building fast & interactive U.I for web as well as mobile apps.

■ Event of the Day Date:...../...../.....

⇒ React divides the U.I into multiple components, which makes the code easier to debug & each component has it's own property & function.

Advantages of React.JS:

1) Easy creation of dynamic web applications.
2) Reusable components.
3) Small learning curve
4) Can be used for mobile apps.
5) Easy debugging tools.

Features of React.JS:

1) JSX: (Syntax extention to Javascript)

⇒ Combination of Javascript + HTML

⇒ By using JSX, you can write HTML structures in the same file that contains JS code.

eg. `const simple = <h1> Hello World </h1>;`

Javascript HTML

■ Event of the Day Date:...../...../.....

2) Virtual DOM:

⇒ Document Object Model (DOM) treats an HTML file doc as a tree structure in which each node is an object representing a part of the document.

⇒ Virtual Dom is the exact replica of the Real Dom. It is a lightweight representation of the Real Dom Manipulating Real DOM is slower than Virtual Dom.

⇒ When an object changes, V. DOM changes only that object in the Real Dom instead of updating all the objects.

3) Performance: Uses multiple components to speed up the dev. time.

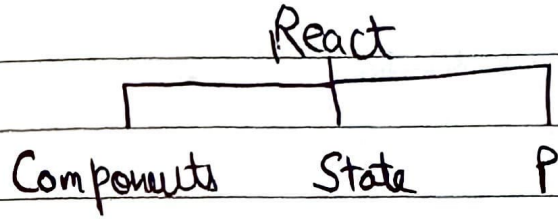
4) One-way Data Binding: Info flows in one direction.

5) Extensions: Many extensions like React Native for mobile applications, provides server-side rendering

eg. React Native, Flux

6) Debugging: easier to debug.

■ Event of the Day Date:...../...../.....



1) Components:

⇒ There are multiple components.
⇒ It splits the U.I into independent, reusable pieces.
⇒ A component is implemented as a Javascript class having some state & render method.

eg.

```
import React from "react";

class Simple extends React.component {
    state = {};
    render() {
        return (
            <div>
                <h1> Hello </h1>
            </div>
        );
    }
}
```

⇒ State is the data that we want the component to render.

⇒ Render method is responsible for how the U.I looks & feels to the user.

avantor

■ Event of the Day Date:...../...../.....

2) State :

⇒ It is an object that holds some data.

⇒ This data influences the output of the component.

⇒ Everytime the state of an object changes, the component is rerendered.

3) Props : (Properties)

⇒ Allow us to pass arguments or data to components.

Creating React App:

npm install -g create-react-app ∴ This command installs all the necessary React.js modules

create-react-app --version ∴ Version of React.js installed.

create-react-app myapp /(or name of your project)

cd app-name

npm start

■ Event of the Day Date:...../...../.....

⇒ App.js is the main component that gets rendered to the DOM.

⇒ In Index.js, you pass App.js which is a component that gets rendered to the ReactDOM.

Starting with App.js:

⇒ All HTML tags have to be enclosed within a div tag.

Creating a functional component:

⇒ Go to src, create a components folder, & create a file called FunctionalComp.js.

⇒ import React from 'react';

⇒ Import & Export are used for implementing the components from other files.

Exports are of 2 types:

1) Default: used to export only one object from file. A file can be renamed while importing it. There can be only one default export per file.

■ Event of the Day Date:...../...../.....

2) Named Export: Used to export multiple objects from a file. There can be several named export from a file. The name of the objects cannot be changed while importing

Higher Order Component:

It is a function that takes a component and returns a new component.

Const NewComponent = higherOrderComponent (originalComponent)

Pure Components:

⇒ React.Component is the base class for React Components. React.PureComponent is a variation of React.Component class & does a shallow comparison of props & state.

⇒ A react component can be considered pure if it renders the same output for the same state & props.

Props:

⇒ Allow users to pass arguments or data to components

⇒ Helps make components more dynamic

⇒ Passed to components in a way similar to HTML Tag attributes.

⇒ Props in a component are Read only

avantor

■ Event of the Day Date:...../...../.....

State :

⇒ It is an object that stores the values of properties belonging to a component that could change over time.

⇒ Everytime the state of an object changes, React re-renders the component to the browser.

⇒ this.setState() is used to change the value of the state object.

⇒ setState() function performs a shallow merge b/w the new & previous state.

Props	State
1) Used to pass data ~~and~~ to ~~event trans~~ it's children comp.	Used to store the data of the components
2) Props cannot be changed.	Can change over time.
3) Can be used in both functional & class components	Can only be used in class components.

avantor

■ Event of the Day Date:...../...../.....

React	Angular
1) Uses Virtual DOM, so faster.	1) Uses Real DOM.
2) Library	2) Framework
3) Small in size & lightweight	3) Bigger in size
4) Have to write many lines of code and make use of ext. libraries	4) Framework, so built in support & only fewer lines of code.

2) Node modules: Contains all the dependencies of the project, including React libraries.

3) Public Folder:
Contains static assets that are served directly to the users browser such as images, fonts & index.HTML file.

4) Src Folder:
used to store all the source code of the app.

5) Index.html:
entry point to the application. This is the first page that will be loaded when an app opens.

■ Event of the Day Date:...../...../.....

6) App.js :
Root component responsible for rendering all of the other components

7) Role of function & return in App.js:
Function is a js func. that returns a React element. Both of them work together to return the output to the SPA (single page app).

8) Role of Index.js:
Consists of ReactDOM, which is a js library that converts your components to actual browser DOM. ReactDOM.createRoot means we are trying to replace the root in the index.html. root.render() will set the root components & it's child components.

React Component Lifecycle:
Series of events that happen from the mounting of a React component to it's unmounting.

1) Mounting : Birth of component
2) Update : Growth " "
3) Unmount : Death " "

■ Event of the Day Date:...../...../.....

Methods:

1) **Render()**: Used to render HTML of the component. It should be pure, cannot modify state in it. Runs during Mounting & updating of component.

2) **ComponentDidMount()**: runs after the component output has been rendered to the DOM. Used to fetch data.

3) **ComponentDidUpdate()**: invoked when updating happens. eg. called when updating the DOM in response to prop or state changes. ∴ Props can be recieved again.

4) **ComponentWillUnmount()**: Called just before the component is unmounted. Usually used to perform clanups.

■ Event of the Day Date:...../...../.....

Hooks:

⇒ Features of class based components in function based components.

⇒ It allows you to use state & other React features w/o writing a class.

Commonly used React Hooks:

1) **useState()**: Allows you to add state to your functional components. It is used when you have to manage the state in the component.

2) **useEffect()**: Allows you to run side effects in functional components. It is used to fetch data from the API, manipulate DOM.

3) **useContext()**: Allows you to access data from a parent component in your child component w/o having to pass it down through props.

4) **useRef()**: Allows you to create a mutable value that remains same across re-renders. Used to access a DOM node or mounted component.

- **Error Boundaries:** Error Boundaries in React are components that catch JavaScript errors anywhere in their child component tree and log those errors. They help prevent entire React applications from crashing due to unhandled errors.

- **Ref:** Refs provide a way to access DOM nodes or React elements created in the render method. They are used to interact imperatively with a DOM element or a React component, like focusing an input field or triggering animations.

- **Portals:** Portals in React provide a way to render children into a DOM node that exists outside of the parent component's DOM hierarchy. They're useful for scenarios like modal dialogs or tooltips where the content needs to visually "break out" of its containing component.

- **Redux and Redux Cycles:** Redux is a predictable state container for JavaScript apps, commonly used with React for managing application state. Redux cycles refer to the flow of actions dispatched by components, which are processed by reducers to update the state, providing a predictable data flow.

- **React Patterns:** React patterns are established practices or solutions to common problems encountered in React development. They include techniques like render props, higher-order components (HOCs), and context API for managing state and component composition effectively.

- **Types of Web Apps:** Web apps can be categorized into single-page applications (SPAs), which load once and dynamically update content, and multi-page applications (MPAs), which load new pages from the server in response to user actions or navigation.

- **Cookies:** Cookies are small pieces of data stored in the user's browser by websites. They're commonly used for session management, tracking user behavior, and storing user preferences.

- **Browser Storage:** Browser storage refers to mechanisms like localStorage and sessionStorage, which allow web applications to store data locally in the user's browser. They provide a way to persist data across sessions without using cookies.

- **Cache:** The browser cache stores copies of web resources like HTML pages, CSS stylesheets, and JavaScript files. When a user revisits a web page, the browser can load these resources from the cache instead of downloading them again, which improves page load times and reduces server load.