INDEX

REACT

It's a JavaScript frontend library. It is a SPA [Single Page Application], so it's faster. It's developed by facebook. It is widely used for creating fast, dynamic, and scalable front-end applications.

-> npx create-react-app appName

When to use

- Dynamic web applications
- SPA
- Reusable UI elements

PROJECT STRUCTURE

➔ README.md = Basic detail structure.
➔ package.json = To manage packages and libraries which is installed using npm tools.
➔ package-lock.json = To install the same version of packages which is used to create a project after a long time, so here we are locking all packages.
➔ .gitignore = files/folders to be ignored during pushing
➔ node_modules = To store copies of all installed packages and libraries. node_modules is ignored in .gitignore as it has large files, so inorder to install it, npm i is used . We can check files in package.json
➔ public = If we want to access a file from public, no need to provide a path, instead a "/ " is enough.
➔ setupTests.js = To test the app which we created
➔ reportWeb vitals = to measure speed and performance

npm install = npm i [i – install]

Component :

A section of html page, component name must starts with capital letter.

--- Styles applied globally to components is given in index.css

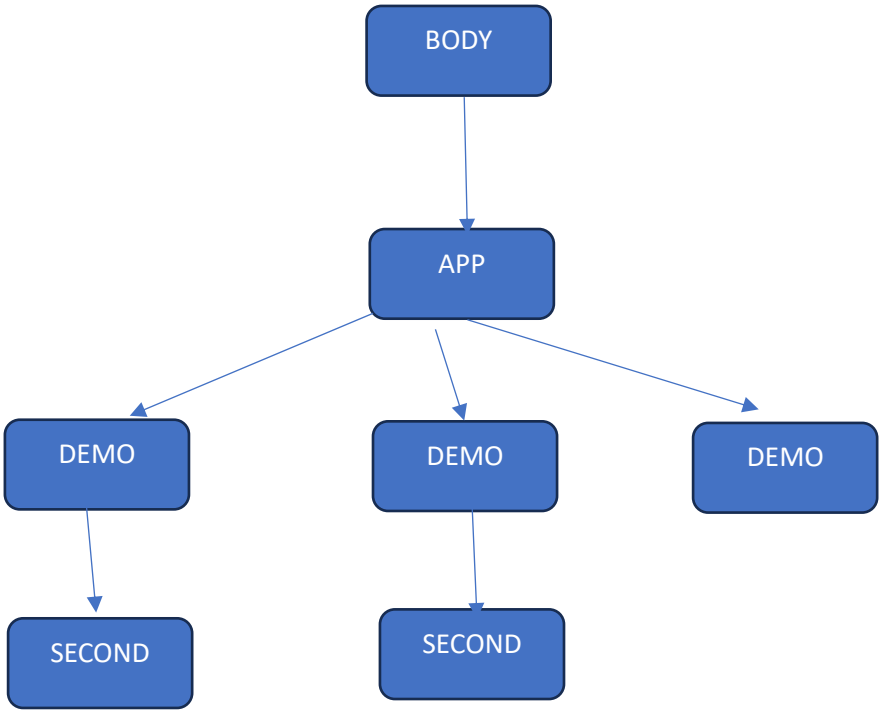
--- To execute a component individually, app.js is used

--- 2 types of components

- Class based components (Angular uses) : class is considered as component
- Function based components (React uses) : function is considered as component

--- Component needs html, css and js, so it uses jxs(javascript and xml) ; as it contains both html and js.

DOM STRUCTURE OF App.js



DOM

1. Real DOM = It has a physical DOM , if a component has any changes added, entire DOM will rerun
2. Virtual DOM = It does'nt have any physical DOM, but has virtual DOM. If any changes added to a component, only that component DOM will rerun. React uses it and is comparatively faster.

Browser will run only html,css and js, since react uses jsx, it needs to be converted using transpiling (splitting

Transpiling:

The process of converting a pgmming lang to another one

babel is the library used for transpiling.

STATE :- object used to create a piece of data

Variables can only be used inside a block, so state needs to be used so that it can used in any other files.

The state is a built-in React object that is used to contain data or information about the component. A component's state can change over time; whenever it changes, the component re-renders.

- Class based components – It's ***stateful,*** as it has inbuilt state on it
- Function based components – it's ***stateless,*** It does'nt have states inbuilt on it. To create a state, **hooks** is used.

Hooks:

Hooks are the new feature introduced in the React 16.8 version. It allows you to use state and other React features. All hooks starts with "use".

- useState() – a hook which is used to create state in react function based components.

Syntax;

const[state name, func name to update the state] = useState()

Eg: const[uname,setUname] = useState("anu")

Here, const value can be changed.

- useEffect() – a hook which is used to take effect when we open a component[function-useEffect() , class – container]

Syntax;

useEffect(()=>(fetchData()),[ ])

useEffect will work at the starting and will continue to rerun, to avoid it a second argument is need to be given, a null array. If you give an variable, array in the null array, useEffect will work in the starting and also when the content inside the variable changes.

An api cannot be given directly to useEffect, it should be called in another function and that function should be given in useEffect

Event:

1. With target value – onChange [when button is clicked, a data is acquired]
2. Without target value – onClick [no data]
3. onSubmit - Triggered on form submission

DATA SHARING In REACT

Data can be shared from parent component to child component (child to parent is not possible).Props(Properties) concept is used for data sharing in react.Props is a special keyword in React that stands for properties and is used for passing data from one component to another. Data with props are passed in a unidirectional flow from parent to child.

Props is created as object, eg; props={data:uname}

Asynchronous code handle

- promise = then, catch
- async , await = It's used to reduce chaining [chaining - increasing use of 'then']
  Can use an asynchronous code in synchronous manner.

Fetch

Drawbacks

- Only provide response data , not getting other informations (status,url…) about api response.
- Not supported to all browsers
- Very low error handling capacity

Axios [library] = react uses this library for handling above drawbacks.

Advantages

- Provides all types of data [status, url..]
- Supports all web browsers
- High error handling capacity

To install a library = npm i library name

Redirection in React

React Router is a popular library used for implementing routing in React applications. It allows you to create a single-page application (SPA) with multiple views or pages, enabling seamless navigation without requiring a full page reload.

To apply redirection, 3 components is required

- Route
- Routes
- BrowserRouter

```
<BrowserRouter>
    <App />
    </BrowserRouter>
      <Routes>
        <Route path='/' element={<Landing></Landing>}></Route>
      </Routes>
```

useNavigate : A hook for programmatic navigation.

State Management Technology

- Redux
- Context Api

REDUX :

Rather than storing data in a component by react, redux stores data in a store file.

Props drilling : Sharing data in depth

When props drilling happens, if there is an issue in intermediate components data will get stuck. This is a common problem faced in react. So in order to solve this issue Redux is introduceCreate d.

useSelector : hook used to access state from redux store

useDispatch : hook used to dispatch action.

ContextApi :

It provides a centralized way to manage state across components.It share specific information like state or functions with multiple components without props drilling.

Steps :

1. Create a context by createContext() method
2. Providing the context by Provider
3. Consuming the context by useContext()

Conditional Rendering

- If =
  condn && (code)
- If – else =
  condn?(code):(code)

React lifecycle methods

Each component in React has a lifecycle which you can monitor and manipulate during its three main phases.

- **Mounting**: When a component is being inserted into the DOM for the first time.

```
import React from "react";
import ReactDOM from 'react-dom';

class Header extends React.Component {
    constructor(props) {
      super(props);
      this.state = {favoritecolor: "red"};
    }
    componentDidMount() {
      setTimeout(() => {
        this.setState({favoritecolor: "yellow"})
      }, 1000)
    }
    render() {
      return (
        <h1>My Favorite Color is {this.state.favoritecolor}</h1>
      );
    }
  }

  ReactDOM.render(<Header />, document.getElementById('root'));

export default Header
```

- **Updating**: When a component is being re-rendered due to changes in state or props.

```
class Header extends React.Component {
  constructor(props) {
    super(props);
    this.state = {favoritecolor: "red"};
  }
  componentDidMount() {
    setTimeout(() => {
      this.setState({favoritecolor: "yellow"})
    }, 1000)
  }
  componentDidUpdate() {
    document.getElementById("mydiv").innerHTML =
    "The updated favorite is " + this.state.favoritecolor;
  }
  render() {
    return (
      <div>
      <h1>My Favorite Color is {this.state.favoritecolor}</h1>
      <div id="mydiv"></div>
      </div>
    );
  }
}

ReactDOM.render(<Header />, document.getElementById('root'));

export default Header
```

- **Unmounting**: When a component is being removed from the DOM.

```
class Container extends React.Component {
  constructor(props) {
    super(props);
    this.state = {show: true};
  }
  delHeader = () => {
    this.setState({show: false});
  }
  render() {
    let myheader;
    if (this.state.show) {
      myheader = <Child />;
    };
    return (
      <div>
      {myheader}
      <button type="button" onClick={this.delHeader}>Delete Header</button>
      </div>
    );
  }
}

class Child extends React.Component {
  componentWillUnmount() {
    alert("The component named Header is about to be unmounted.");
  }
  render() {
    return (
      <h1>Hello World!</h1>
    );
  }
}

ReactDOM.render(<Container />, document.getElementById('root'));
```

Lists

In React, you will render lists with some type of loop.

The JavaScript map() array method is generally the preferred method.

```
<ul>
      {fruits.map((fruit, index) => (
        <li key={index}>{fruit}</li>
      ))}
    </ul>
```

Forms

Just like in HTML, React uses forms to allow users to interact with the web page.

```
<form>
    <label>Enter your name:
      <input type="text" />
    </label>
  </form>
```