What is React

1. Javascript library to build dynamic user interface
2. Developed and managed by facebook
3. Used to create single page application (SPA)
4. Working on the basis of DOM (document object model)

Creating React APP

1. Setup Environment
2. Installing React
3. Understand directory structure

What is required

* Node JS (<https://nodejs.org/en>)
* VS Code (<https://code.visualstudio.com/>)
* System should be updated for better coding experience

Let’s dive in to creating app

1. The standard way to create react app ([https://react.dev/](https://react.dev/learn/start-a-new-react-project) )
   1. npx create-next-app@latest
2. More efficient way using Vite (<https://vitejs.dev/guide/>)
   1. npm create vite@latest

Understanding Directory Structure

* node\_modules
* public 🡺 contains static files
* src 🡺 main folder that contains react files
* package.json 🡺 contains package dependencies
* index.html 🡺 render first

What are the Components

1. File Extension
   1. .js / .jsx
2. JSX files
3. Class vs Functions
4. Dynamic / Reusable Components

function App() {

  return <h1>

    This is first react component

  </h1>

}

export default App;

Add new component

MyButton.jsx

function MyButton() {

    return <button>From Button Component</button>

}

export default MyButton;

Import it where you want to use

Multiple Exports

export function MyButton1() {

    return <button>Button One</button>

}

export function MyButton2() {

    return <button>Button Two</button>

}

Dynamic Component

export default function Welcome() {

    let name = "Farhan";

    let fullName = () => {

        return "Muhammad Farhan"

    }

    return <h1>Hey there i am {fullName()}</h1>

}

Installing Bootstrap

<https://getbootstrap.com/>

npm i [bootstrap@5.3.2](mailto:bootstrap@5.3.2)

main.jsx

import 'bootstrap/dist/css/bootstrap.min.css'

Fragment

<React.Fragment> </ React.Fragment>

<></>

Rendering List/Array Data

Array.map()

let daysList = ["mon", "tue", "wed", "thr"];

  return (

  <>

    <ul>

    {daysList.map( (dl, i) => {

      return <li>{dl}</li>

  } )}

    </ul>

  </>

  )

Conditional statements

  let daysList = ["mon", "tue", "wed", "thr"];

  if(daysList.length === 0) {

    return "data not found";

  }

  return (

  <>

    {daysList.length === 0 ? <h1>No data found</h1> : null }

    {daysList.length === 0 && <h2>data not found </h2>}

    <ul>

    {daysList.map( (dl, i) => {

      return <li>{dl}</li>

  } )}

    </ul>

  </>

  )

Props 🡺 properties

In ReactJS, the props are a type of object where the value of attributes of a tag is stored. The word “props” implies “properties”, and its working functionality is quite similar to HTML attributes.

Basically, these props components are read-only components. In ReactJS, the data can be passed from one component to another component using these props, similar to how the arguments are passed in a function. Inside the component, we can add the attributes called props; however, we cannot change or modify props inside the component as they are immutable.

Pass data from parent to child

function App() {

  let daysList = ["mon", "tue", "wed", "thr"];

  if(daysList.length === 0) {

    return "data not found";

  }

  return (

  <>

    <ShowList passList={daysList}></ShowList>

  </>

  )

}

export default function ShowList({ passList }) {

    return (

    <ul>

    {passList.map( (dl, i) => {

      return <li>{dl}</li>

    } )}

    </ul>

    )

}

**Note:** Props are actually used to pass data from parent to child’s

CSS module

import style from './AddTask.module.css';

<label className={`${style["my-style"]}`}>Enter Value</label>

Assignment 🡺 Task List

Assignment 🡺 Calculator

Child Props 🡺 interview question

export default function Container(props) {

  return (

    <div>

      {props.children}

    </div>

  )

}

function App() {

  return (

  <Container>

  <AddTask></AddTask>

  </Container>

  )

}

Events

An event is an action that could be triggered as a result of the user action or system generated event. For example, a mouse click, loading of a web page, pressing a key, window resizes, and other interactions are called events.

React has its own event handling system which is very similar to handling events on DOM elements. The react event handling system is known as Synthetic Events. The synthetic event is a cross-browser wrapper of the browser's native event.

<input type="button" onClick={getVal} value="Click Me" className="btn btn-info" />

let getVal = () => {

    console.log('submit button clicked');

  }

Getting input value

Pass functions as props from child to parent and parent to child

Input in child called getVal function as props that is defined in parent

<input type="text" name="putValue" className="form-control" onChange={getVal}/>

Manage States

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. The change in state can happen as a response to user action or system-generated events and these changes determine the behavior of the component and how it will render.

import { useState } from "react";

let [inputValue, setInputValue] = useState("Default");

  let getVal = (event) => {

    setInputValue(event.target.value);

    console.log(event.target.value);

  }

Assignment

Make Calculator Full Functional

CRUD Operations using with Array

React Icons

<https://www.npmjs.com/package/react-icons>

<https://react-icons.github.io/react-icons/>

npm install react-icons --save

Handling Forms

import {React, useRef} from 'react'

export default function FormHandling() {

    let fullName = useRef();

    let email = useRef();

    let getFormValues = (event) => {

        event.preventDefault();

        console.log(fullName);

    }

  return (

    <>

        <form action="" onSubmit={getFormValues}>

            <div className="form-group">

                <label htmlFor="">Enter Name</label>

                <input type="text" className='form-control' ref={fullName} />

            </div>

            <div className="form-group">

                <label htmlFor="">Enter Email</label>

                <input type="email" className='form-control' ref={email} />

            </div>

            <input type="submit" className='btn btn-primary' value="Enter Value" />

        </form>

    </>

  )

}

Context API

Context API is a (kind of) new feature added in version 16.3 of React that allows one to share state across the entire app (or part of it) lightly and with ease. Let's see how to use it.

Creating Context

import { createContext } from "react";

export const inputValueContext = createContext();

Importing to App

import { inputValueContext } from './context/inputValue';

<inputValueContext.Provider value={inputValue}>

    <Container>

      <Input/>

      <Buttons buttons={buttons} getBtnValue={getBtnValue}/>

      <FormHandling />

    </Container>

</inputValueContext.Provider>

Using Context

import { useContext } from 'react';

import { inputValueContext } from '../context/inputValue';

export default function Input() {

  const inputValue = useContext(inputValueContext);

  console.log(inputValue);

  return (

    <div className="col-xs-6">

        <div className="form-group">

        <input type="text" className="form-control" value={inputValue}/>

        </div>

    </div>

  )

}

**useReducer**

const reducer = (state, action) => {

  console.log(state, action);

  return state + 1;

};

function App() {

  const [state, dispatcher] = useReducer(reducer, 0);

  const updater = (t) => {

    dispatcher({ type: t });

  };

  return (

    <>

      <div className="main-container">

        <div className="main-content">

          <button onClick={() => updater("inc")}>Click Me</button>

          <h1>{state}</h1>

        </div>

      </div>

    </>

  );

}

**useEffect**

useEffect is a React Hook that lets you synchronize a component with an external system.

useEffect(() => {

    fetch("https://dummyjson.com/posts")

      .then((res) => res.json())

      .then((data) => {

        getPostFromApi(data.posts);

      });

  }, []);

React Router v5 & v6

<https://reactrouter.com/en/main>

npm install react-router-dom

With V5

import { BrowserRouter, Routes, Route } from "react-router-dom";

function App() {

  return (

    <>

      <div className="main-container">

        <div className="main-content">

          <BrowserRouter>

            <Routes>

              <Route path="/" element={<Sidebar />}>

                <Route path="/" element={<PostList />} />

                <Route path="/add-post/" element={<CreatePost />} />

              </Route>

            </Routes>

          </BrowserRouter>

        </div>

      </div>

    </>

  );

}

<Link to="/" className="nav-link text-white">Home</Link>

<Outlet />

With V6 🡺 advanced and efficient way to use routing

import {

  RouterProvider,

  BrowserRouter,

  createBrowserRouter,

} from "react-router-dom";

import CreatePost from "./components/CreatePost.jsx";

const router = createBrowserRouter([

  { path: "/", element: <App /> },

  { path: "/add-post", element: <CreatePost /> },

]);

ReactDOM.createRoot(document.getElementById("root")).render(

  <React.StrictMode>

    <RouterProvider router={router} />

  </React.StrictMode>

);

Child Routing

const router = createBrowserRouter([

  {

    path: "/",

    element: <App />,

    children: [

      { path: "/", element: <PostList /> },

      { path: "/add-post", element: <CreatePost /> },

    ],

  },

]);

In App.jsx

  return (

    <>

      <div className="main-container">

        <Sidebar

          selectedMenu={selectedMenu}

          setSelectedMenu={setSelectedMenu}

        />

        <div className="main-content">

          <Header />

          <Outlet />

          <Footer />

        </div>

      </div>

    </>

  );

Use with anchor tag

<link to="/add-post" className= “nav-link text-white”>Create Post</link>

Navigate to link

import { useNavigate } from "react-router-dom";

const navigate = useNavigate();

navigate("/");

Load data before rendering page

const router = createBrowserRouter([

  {

    path: "/",

    element: <App />,

    children: [

      { path: "/", element: <PostList />, loader: getList },

      { path: "/add-post", element: <CreatePost /> },

    ],

  },

]);

export const getList = () => {

  return fetch("https://dummyjson.com/posts")

    .then((res) => res.json())

    .then((data) => {

      return data.posts;

    });

};

const postList = useLoaderData();

Form action with route

<Form method="POST">

  <div className="mb-3">

    <label>Title</label>

    <input type="email" name="title" />

  </div>

  <div className="mb-3">

    <label >Description</label>

    <textarea name="description"></textarea>

  </div>

  <button type="submit" className="btn btn-primary">

    Submit

  </button>

</Form>

const router = createBrowserRouter([

  {

    path: "/",

    element: <App />,

    children: [

      { path: "/", element: <PostList />, loader: getList },

      { path: "/add-post", element: <CreatePost />, action: createPostForm },

    ],

  },

]);

export const createPostForm = (data) => {

  return data.request.formData().then((getData) => {

    const postData = Object.fromEntries(getData);

    console.log(postData);

    return redirect("/");

  });

};

Dynamic Routing

What dynamic routing is ?

const router = createBrowserRouter([

  {

    path: "/",

    element: <App />,

    children: [

      { path: "/", element: <PostList />, loader: getList },

      {

        path: "/add-post/:flag",

        element: <CreatePost />,

        action: createPostForm,

      },

    ],

  },

]);

import { useParams } from "react-router-dom";

export default function CreatePost() {

  const params = useParams();

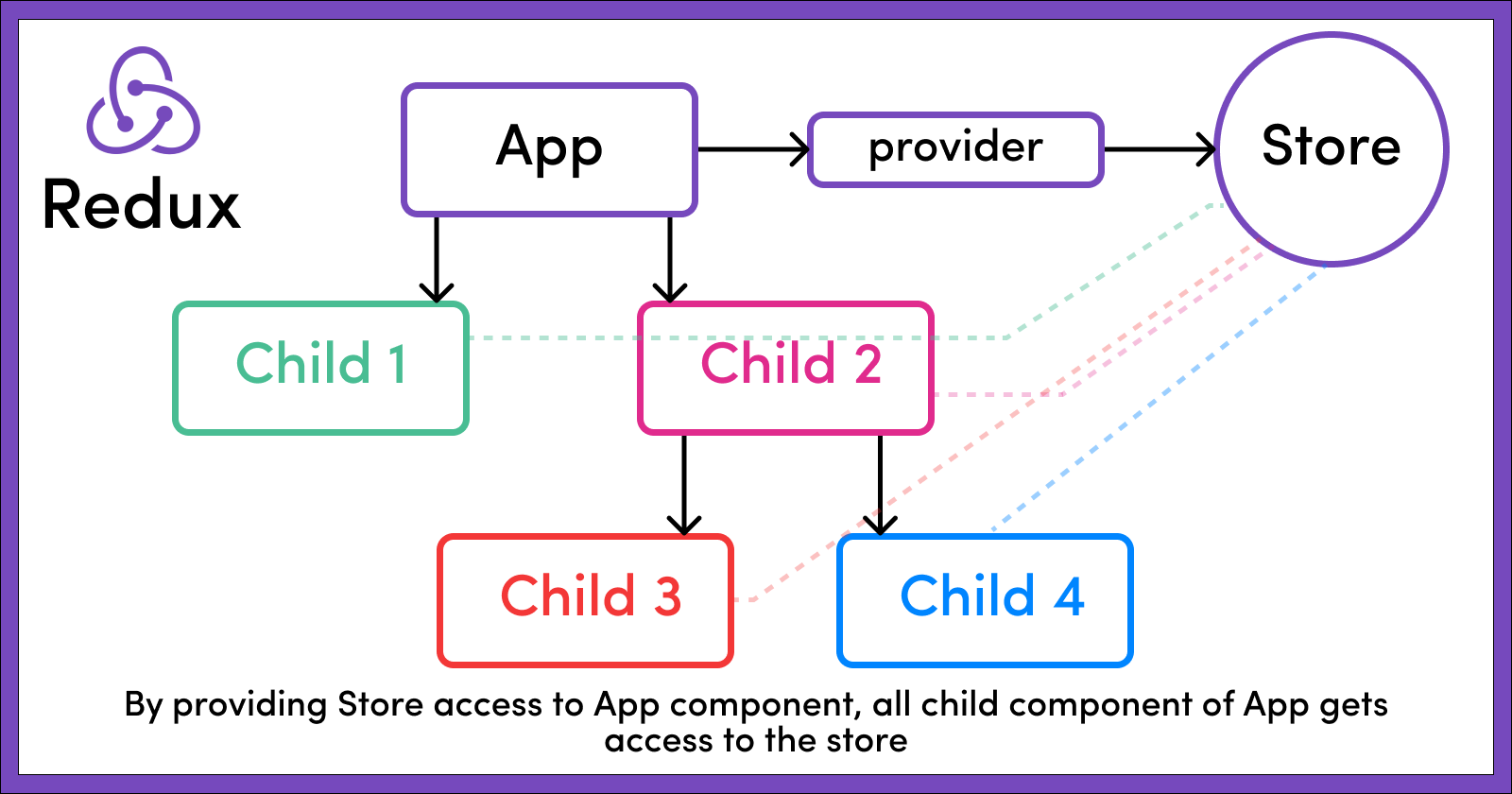
  console.log(params);

  return (

  );

}

REDUX



npm install redux

1. Store
2. Reducer
3. Subscriber
4. Actions

const redux = require("redux");

const INIT\_VAL = {

  counter: 0,

};

const reducer = (store = INIT\_VAL, action) => {

  return { counter: store.counter + 1 };

};

const store = redux.createStore(reducer);

const subscriber = () => {

  const state = store.getState();

  console.log(state);

};

store.subscribe(subscriber);

store.dispatch({ type: "INCR" });

Redux Setup in React

npm install redux

npm install react-redux

Create Store

import { createStore } from "redux";

const INIT\_VAL = {

  counter: 0,

};

const counterReducer = (store = INIT\_VAL, action) => {

  if (action.type == "INC") {

    return { counter: store.counter + 1 };

  }

  if (action.type == "DEC") {

    return { counter: store.counter - 1 };

  }

  return store;

};

const counterStore = createStore(counterReducer);

export default counterStore;

Using With App Component

ReactDOM.createRoot(document.getElementById("root")).render(

  <Provider store={counterStore}>

    <App />

  </Provider>

);

useSelector Hook

const counter = useSelector((store) => store.counter);

useDispatch Hook

const dispatch = useDispatch();

dispatch({type: "action"})

React Toolkit

npm install @reduxjs/toolkit

<https://redux-toolkit.js.org/tutorials/quick-start>