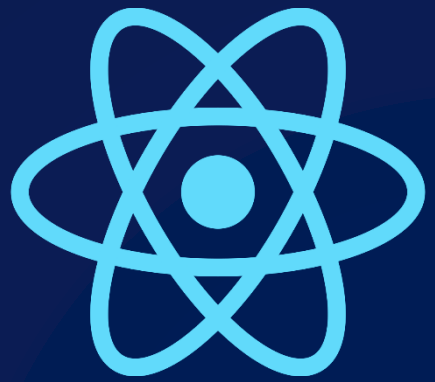
Day – 1

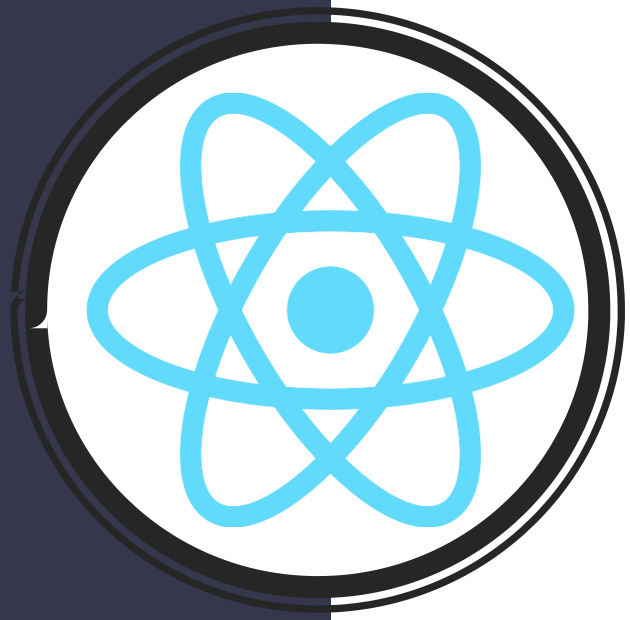
React JS Tutorials

తెలుగులో

Introduction

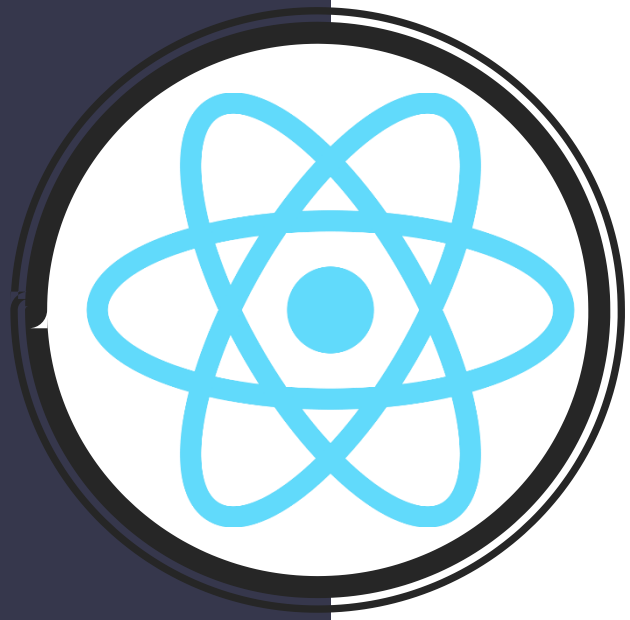


What is React JS



- JavaScript Library (Free & open Source)
- Devolved by Facebook Engineer
- Create Single Page Applications
- Web & Mobile Applications
- Latest Version 18.2

Why React JS



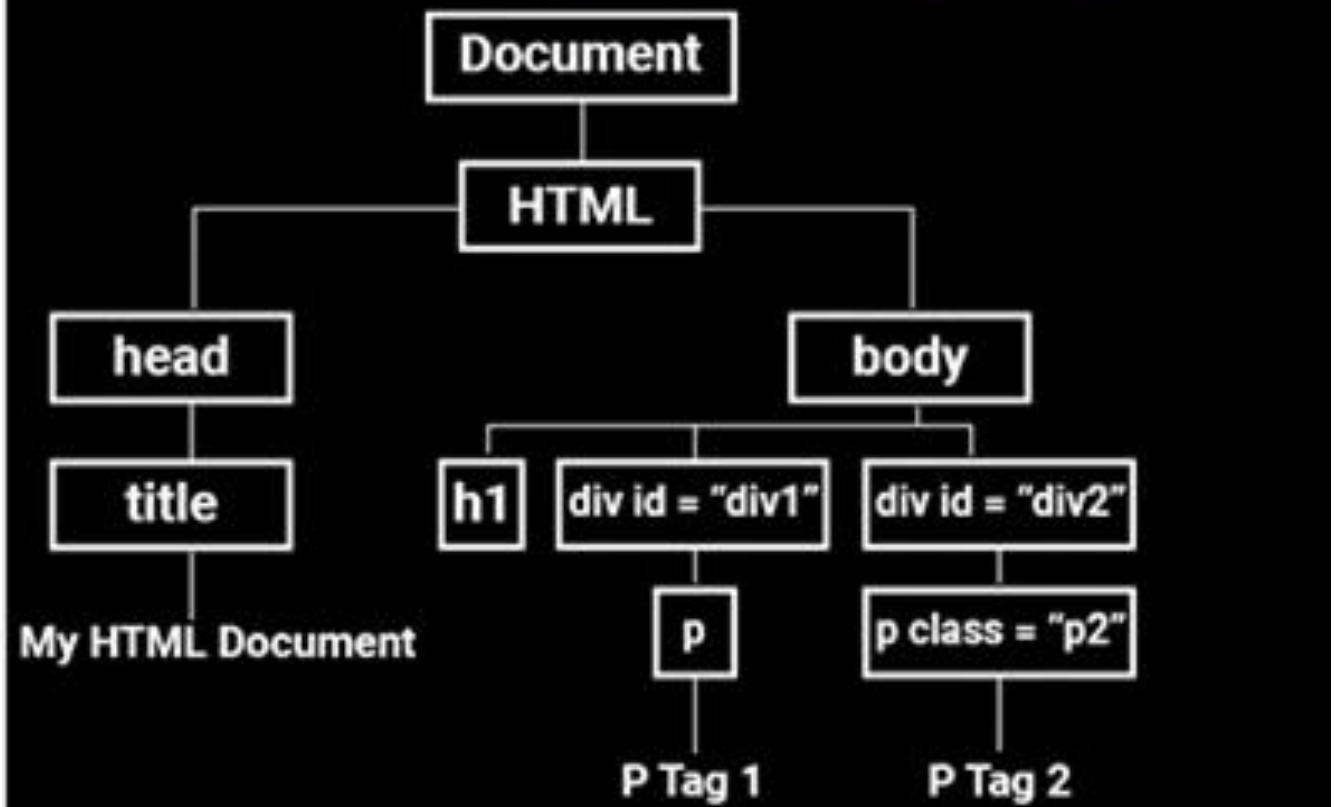
- Component Based Architecture
- Uses Virtual DOM for fast rendering
- Easy To Learn

HTML Document

index.html

```html
<html>
    <head>
        <title>My HTML Document</title>
    </head>

    <body>
        <h1>Heading</h1>
        <div id="div1">
            <p>P Tag 1</p>
        </div>
        <div id="div2">
            <p class="p2">P Tag 2</p>
        </div>
    </body>
</html>
```

Document Object Model (DOM)

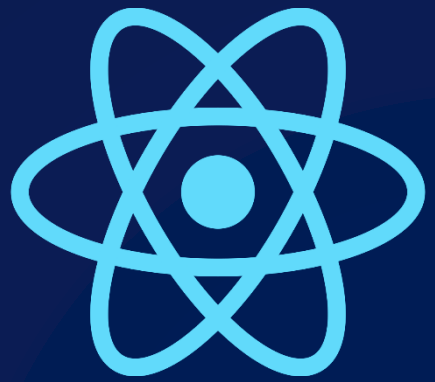


Day – 2

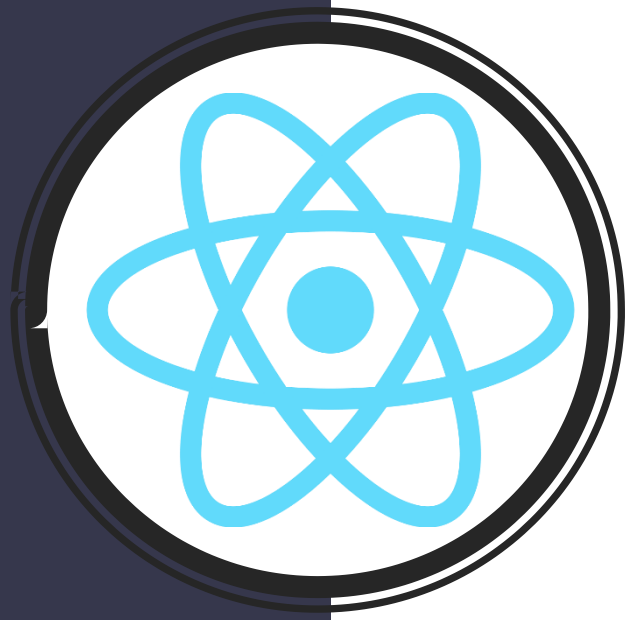
React JS Tutorials

తెలుగులో

Road Map



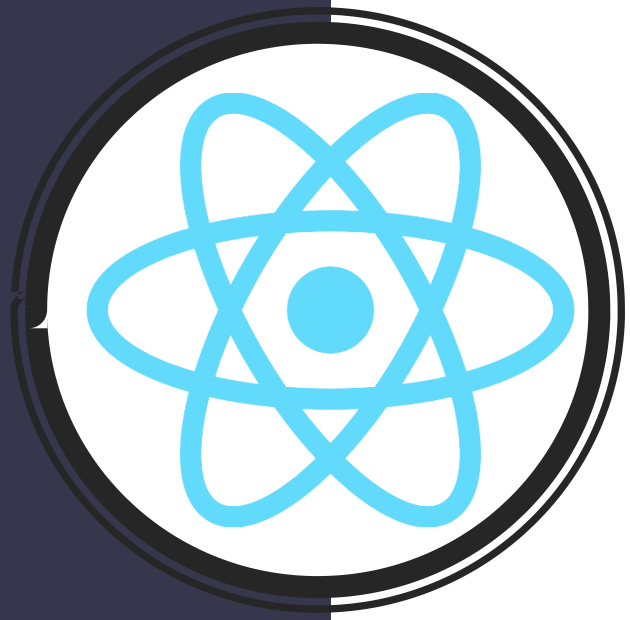
Pre requisite



- HTML
- CSS
- JavaScript

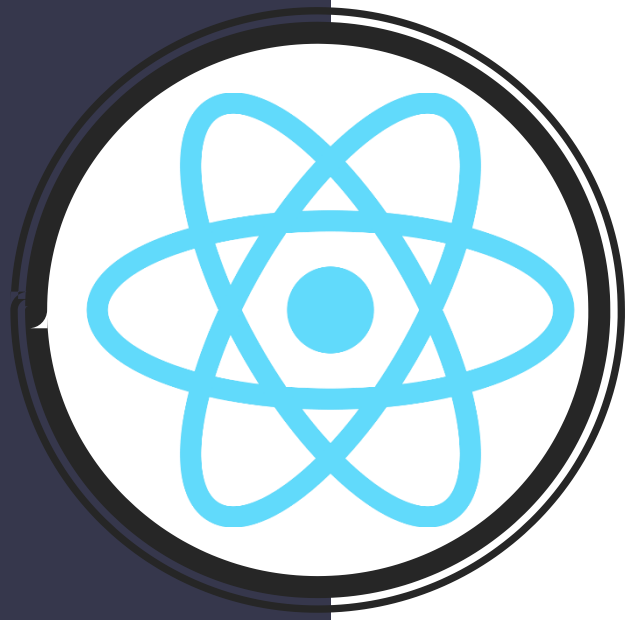
* Modules,Classes, Arrow functions, Classes,Map,Filter *

React Basics



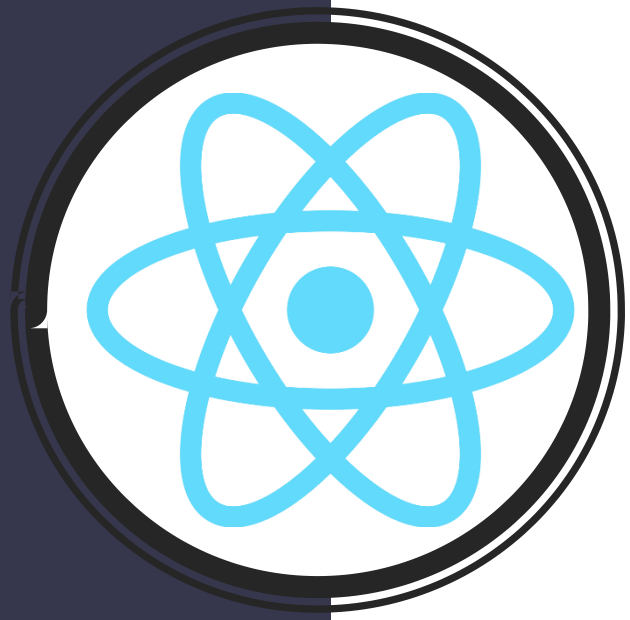
- How to create React Application
- Application Folder Structure
- What is JSX
- Components (Functional & Class)
- Ways to apply CSS
- Props

Intermediate



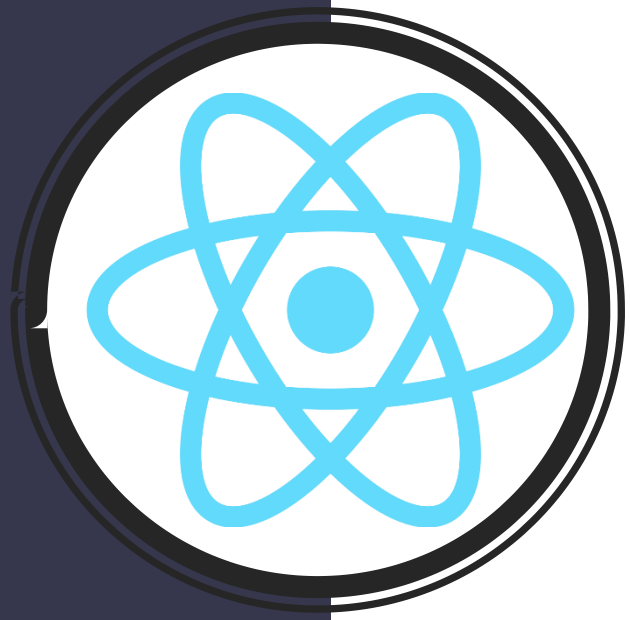
- Conditional Rendering
- React Events
- React List
- Components (Functional & Class)
- Form Handling
- Fragments

Advanced Concepts



- Component Life cycle Methods
- React Hooks
- Redux

Software's Required

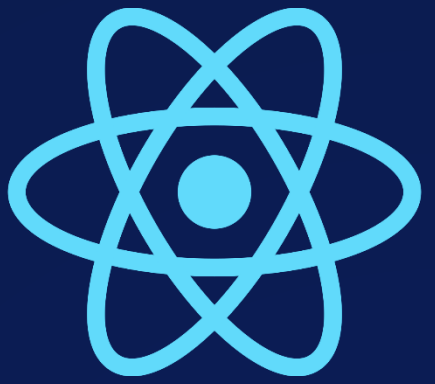


- Node JS
- Any Code Editor
  - VS Code
  - Atom
  - Sublime Text
  - Vim Editor

Day – 3

React JS Tutorials

తెలుగులో

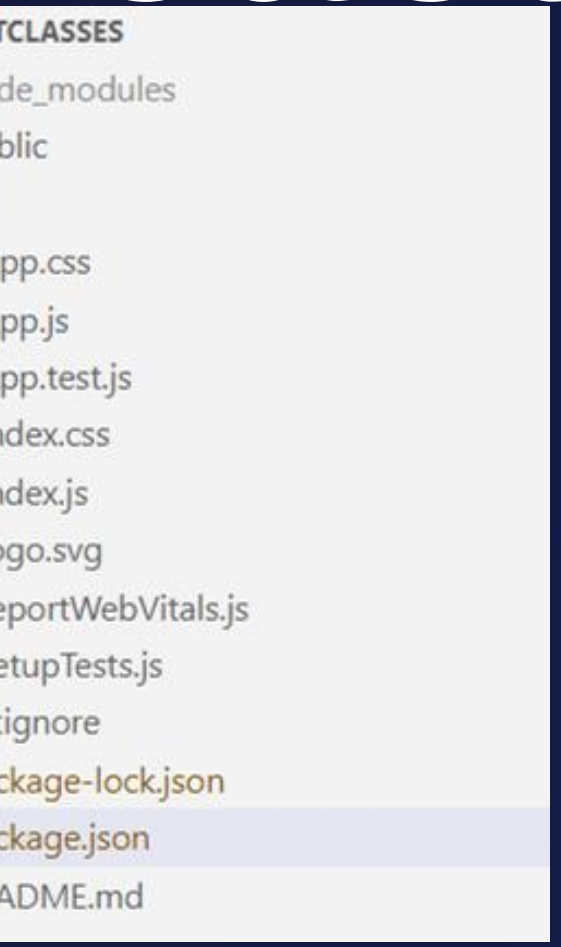


Create React App

Day – 4

React JS Tutorials

తెలుగులో

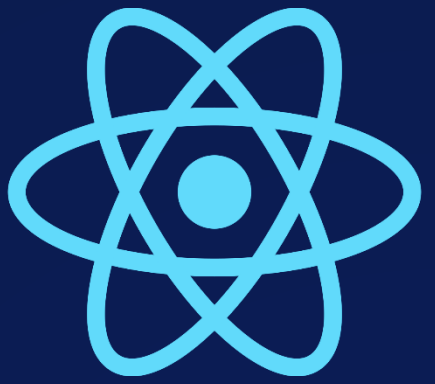


React App Folder Structure

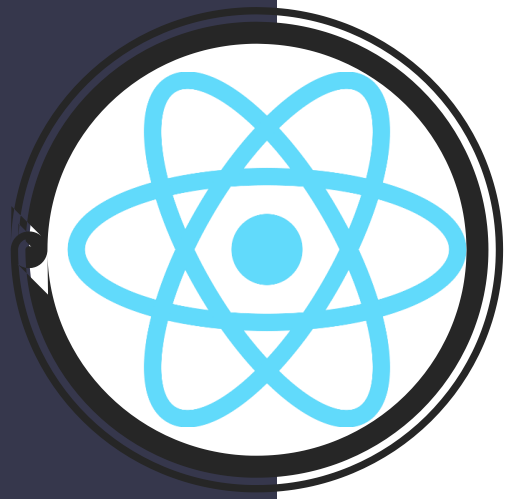
Day – 5

React JS Tutorials

తెలుగులో



JSX in detail



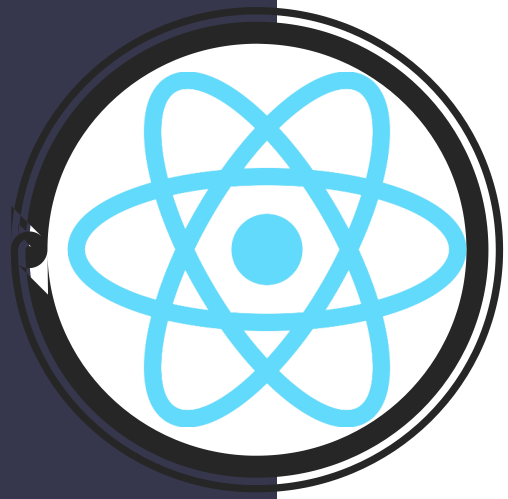
In Html

```
<ul>
  <li>Home</li>
  <li>About</li>
</ul>
```

With JS

```
let ul = document.createElement('ul');

let li1 = document.createElement('li');

li1.innerText = "Home";

let li2 = document.createElement('li');

li2.innerText = "About“

ul.appendChild(li1);

ul.appendChild(li2);

document.body.appendChild(ul);
```

Using React



```
let li1 = React.createElement("li", {}, "Home")

let li2 = React.createElement("li", {}, "About")

let ul = React.createElement("ul", {}, [li1, li2]);

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(ul);
```

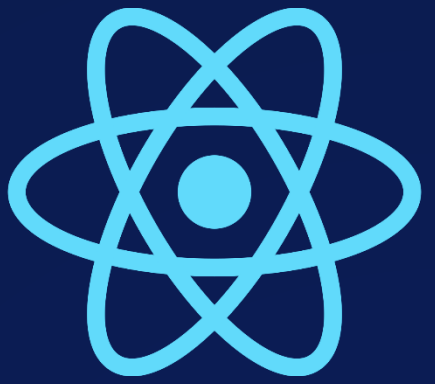
Using JSX

```
let menu = (<ul>
             <li>Home</li>
             <li>About</li>
            </ul>)
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(menu);
```

Day – 6

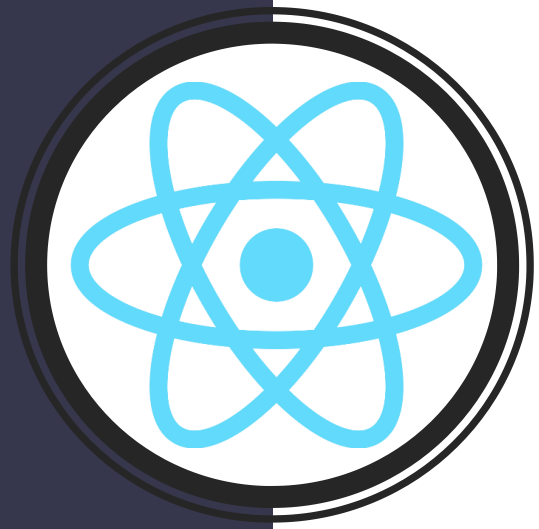
React JS Tutorials

తెలుగులో



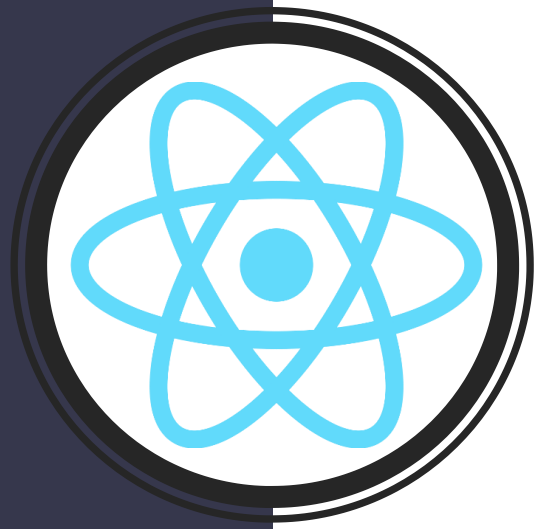
Components in React

Components in React



- Any React Applications is made up of Components
- Component is some code (html+css+JS)

Types Components

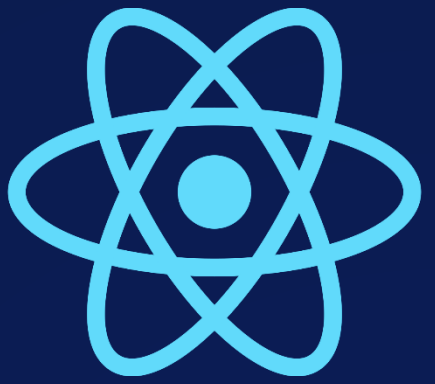


- Functional Components
- Class Components

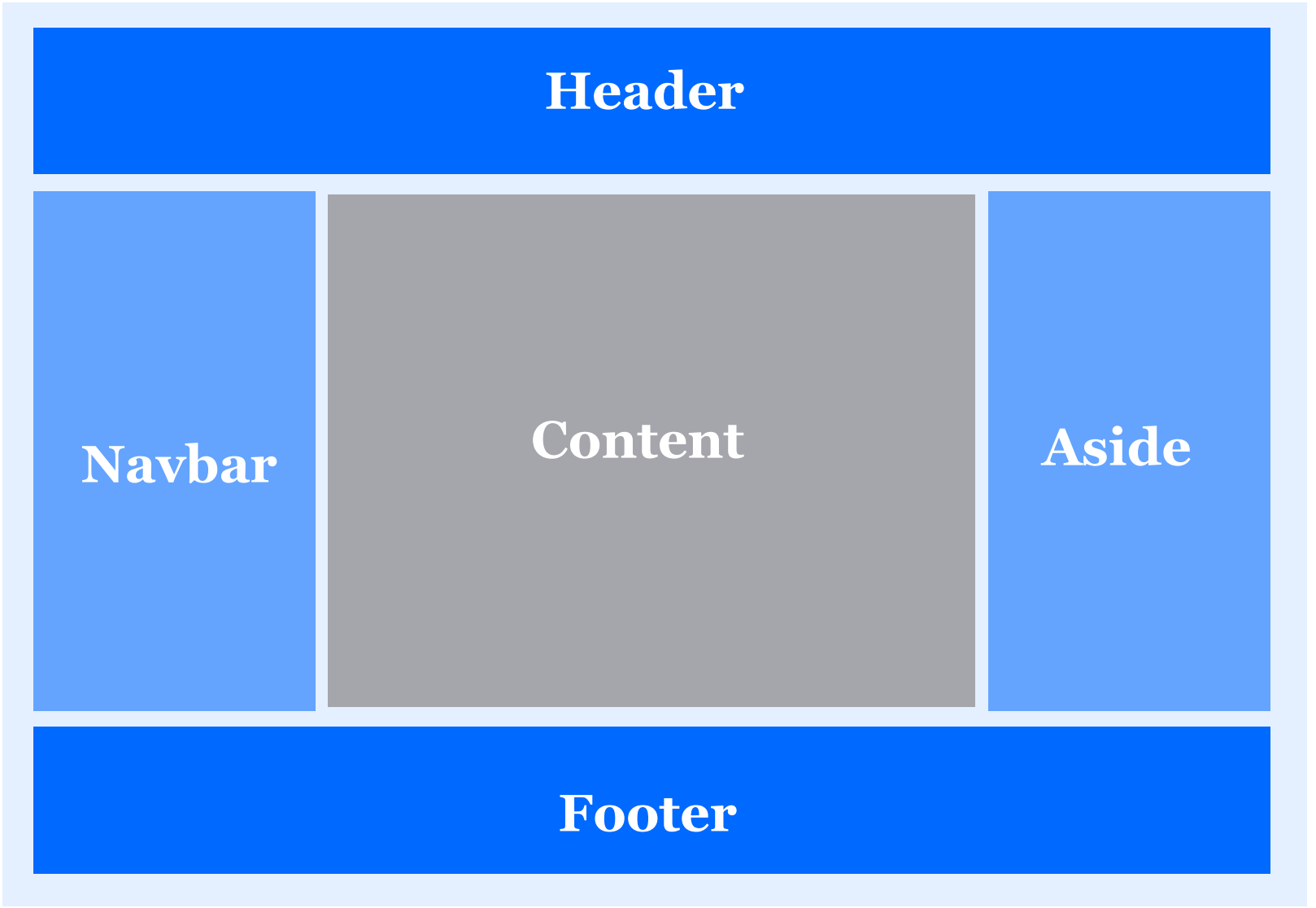
Day – 7

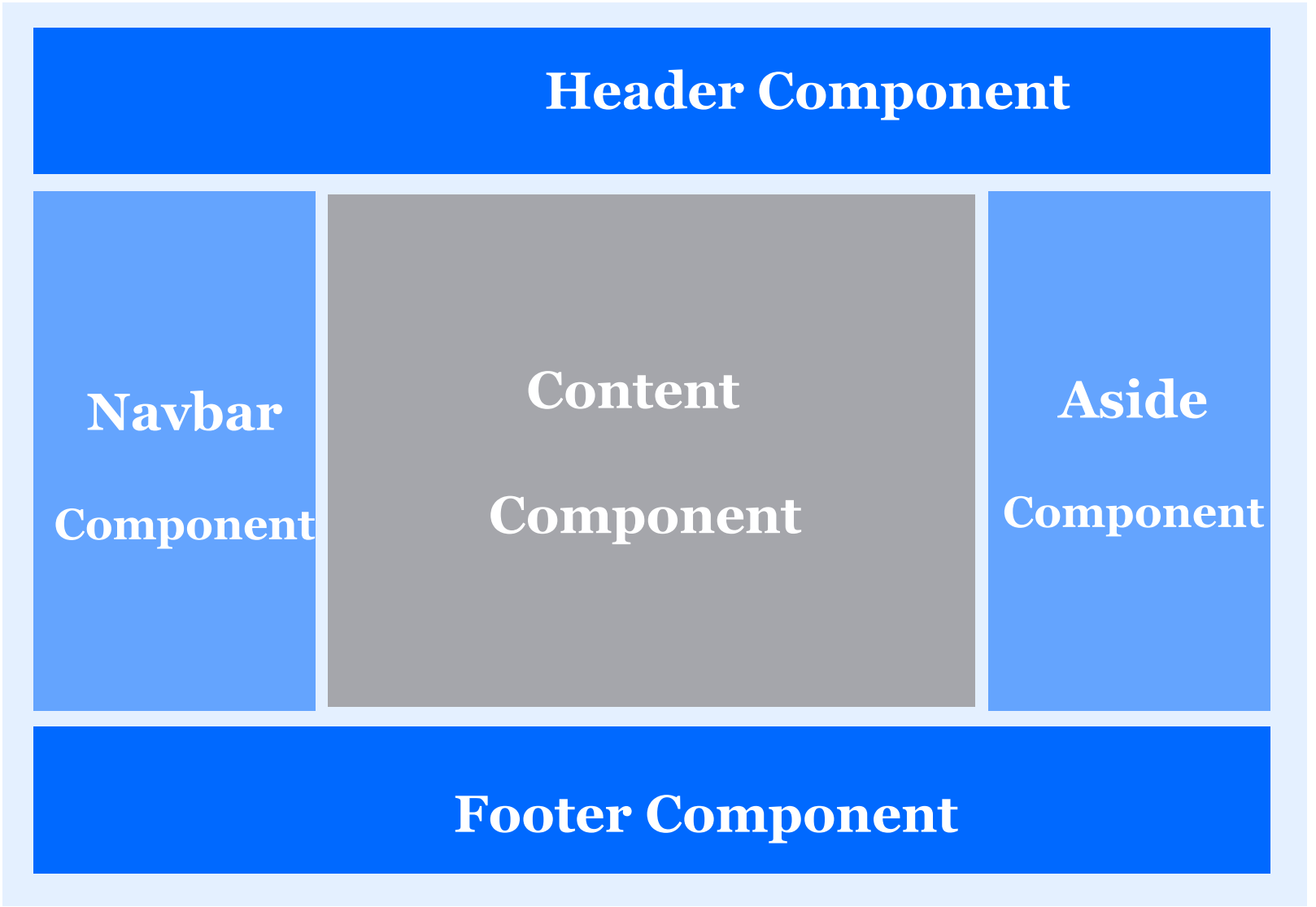
React JS Tutorials

తెలుగులో



Exports& Imports in React

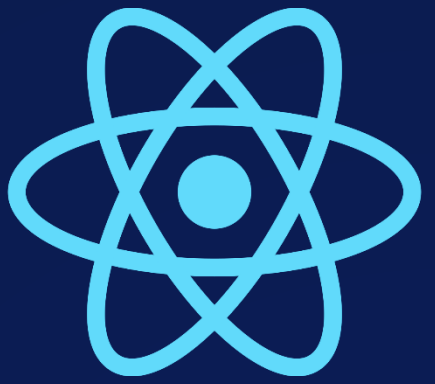

Header
Navbar
Content
Aside
Footer


Header Component
Navbar
Component
Content
Component
Aside
Component
Footer Component

Day – 8

React JS Tutorials

తెలుగులో

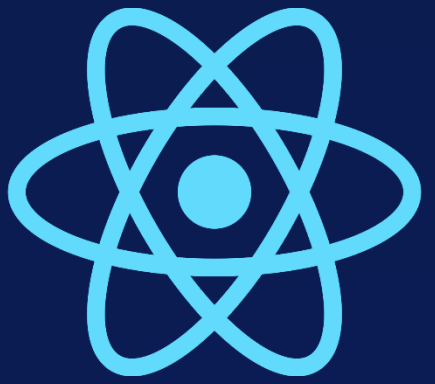


JSX Attributes & Expressions

Day – 9

React JS Tutorials

తెలుగులో

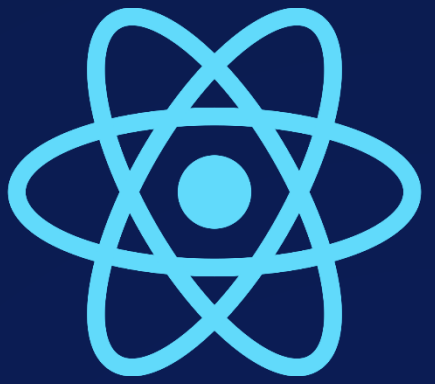


Ways to Apply CSS to components

Day – 10

React JS Tutorials

తెలుగులో

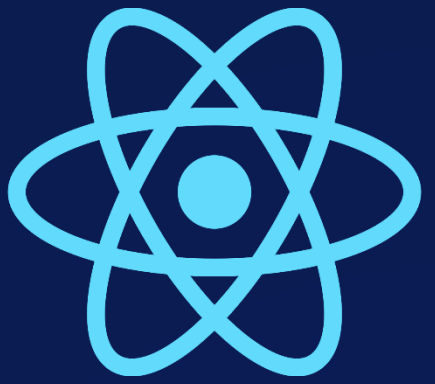


Props in React

Day – 11

React JS Tutorials

తెలుగులో

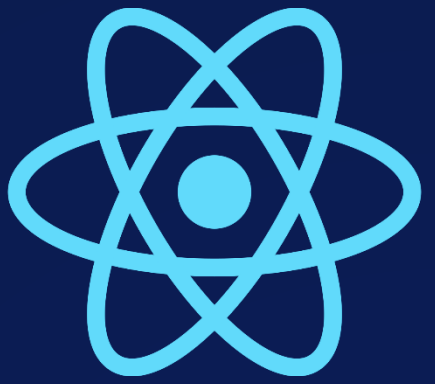


How to Use Map Method with Props in React

Day – 12

React JS Tutorials

తెలుగులో

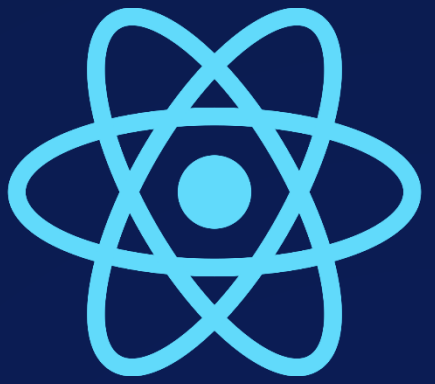


Different Ways to Create &Call Functional Components

Day – 13

React JS Tutorials

తెలుగులో

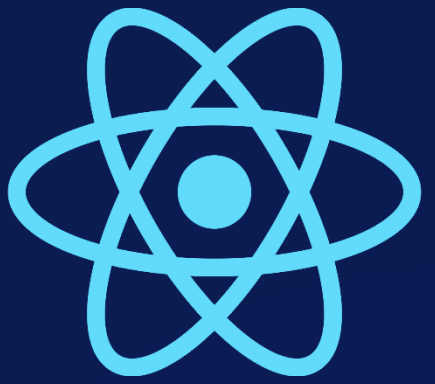


Child Props in React (props.children)

Day – 14

React JS Tutorials

తెలుగులో

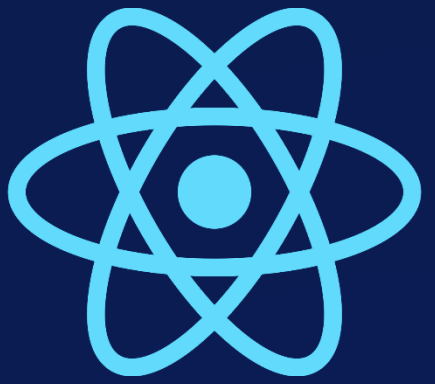


React Fragments

Day – 15

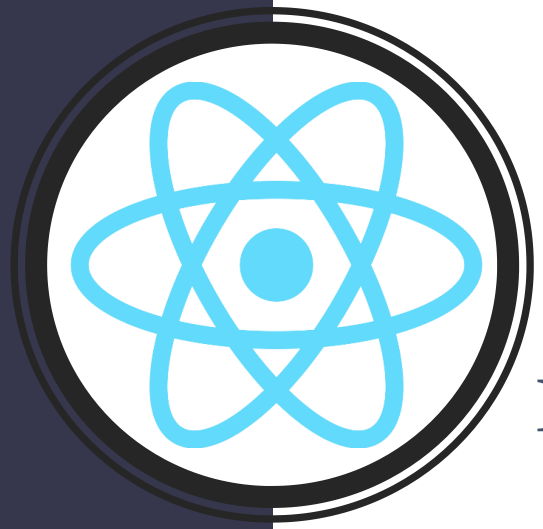
React JS Tutorials

తెలుగులో



Event Handling in React

Event Handling



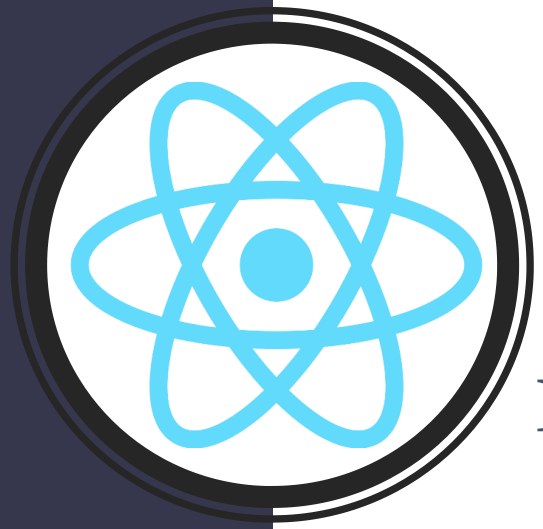
In HTML :

```
<button onclick="displayMessage()">click</button>
```

In React (JSX):

```
<button onClick={displayMessage}>click</button>
```

Event Handling



In HTML :

```
<button onclick="displayMessage()">click</button>
```

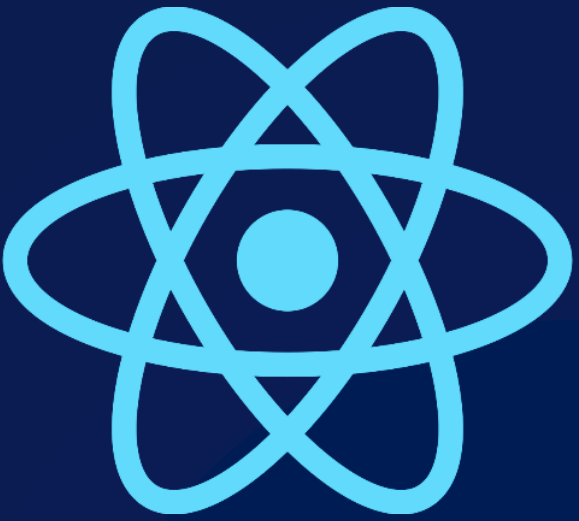
In React (JSX):

```
<button onClick={displayMessage}>click</button>
```

Day – 23

React JS Tutorials

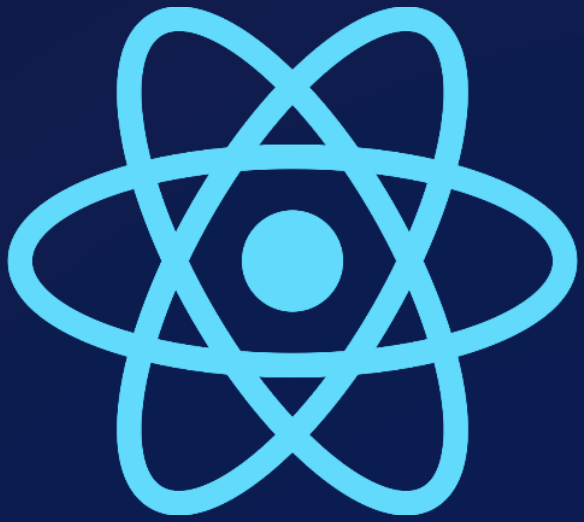
What is Two way data binding



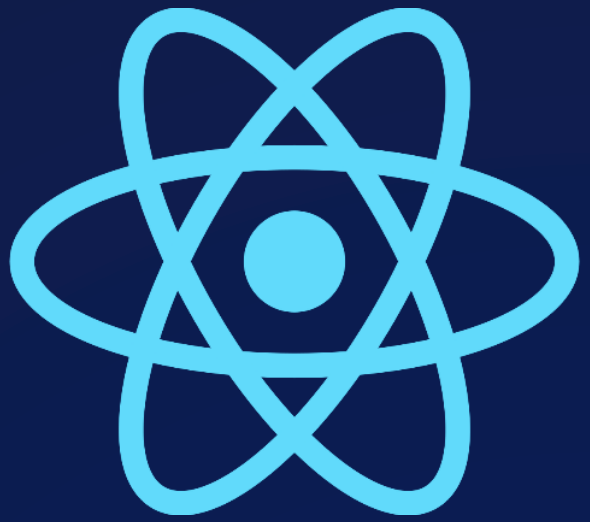
Types of data binding

One way data binding

Two way data binding



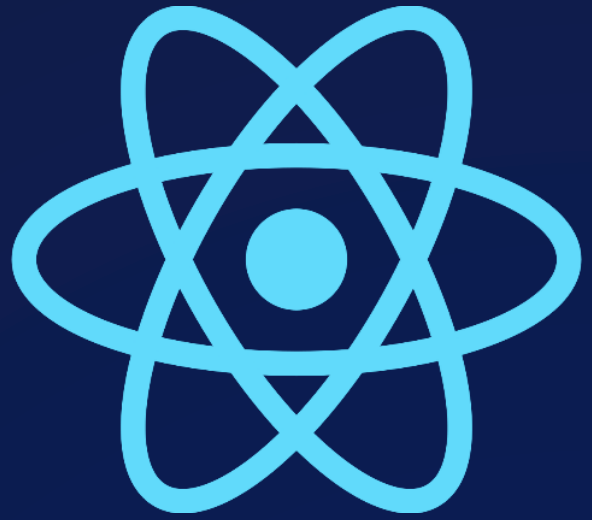
One way data binding



Model/ .js file → DOM/HTML

DOM/HTML → Model/ .js file

Two way data binding



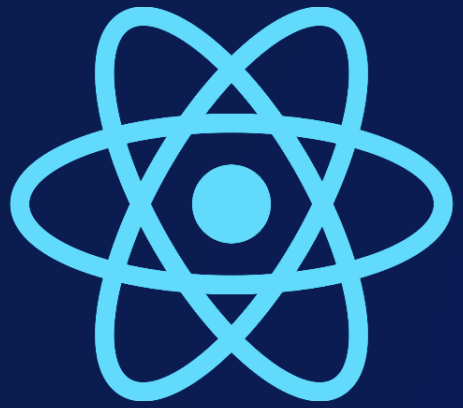
Model/ .js file

DOM/HTML

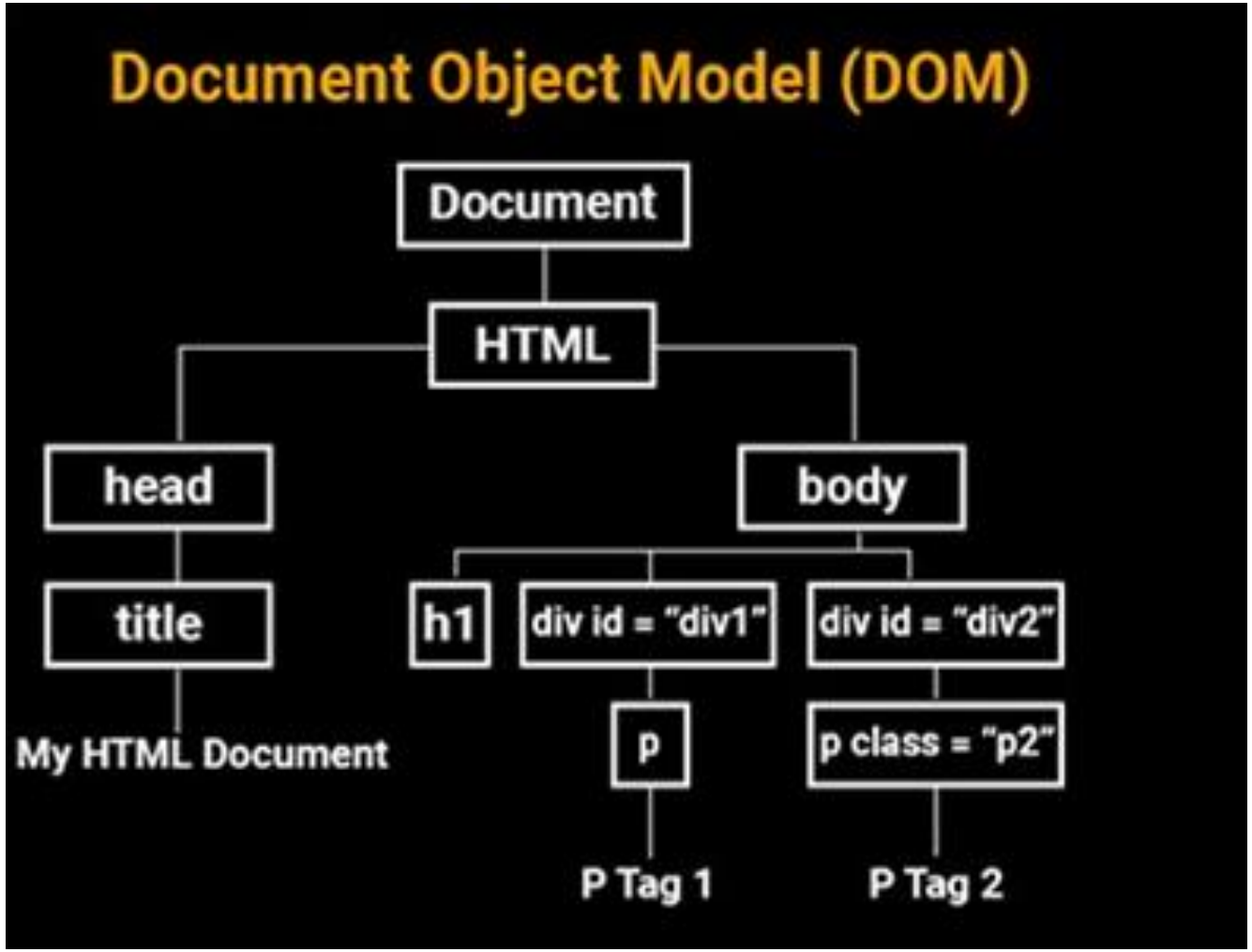
Day – 25

React JS Tutorials

What is Virtual DOM
With Live Example



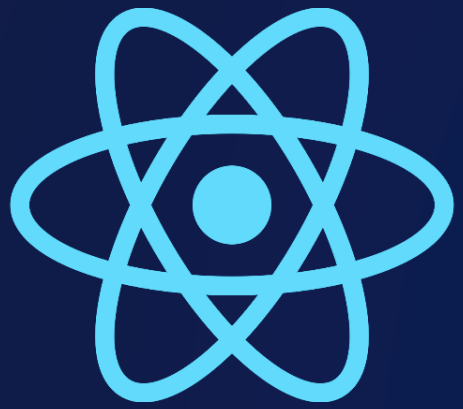
```html
<html
<head>
    <title>My HTML Document</title>
</head>

<body>

    <h1>Heading</h1>
    <div id="div1">
        <p>P Tag 1</p>
    </div>
    <div id="div1">
        <p class="p2">P Tag 2</p>
    </div>
</body>

</html>
```



Day – 27

React JS Tutorials

Use Effect ( ) hook in detail



Syntax of UseEffect ()

```
useEffect(callback,[dependencies(optional)]);
```

Ex:

```
useEffect(() => {
  console.log("use effect")
});
```

```
useEffect(
  () => {
    console.log("effect");
  }, []
)
```

```
useEffect(
  () => {
    console.log("effect");
    getProducts()
  }, [searchInput]
)
```

Day – 30

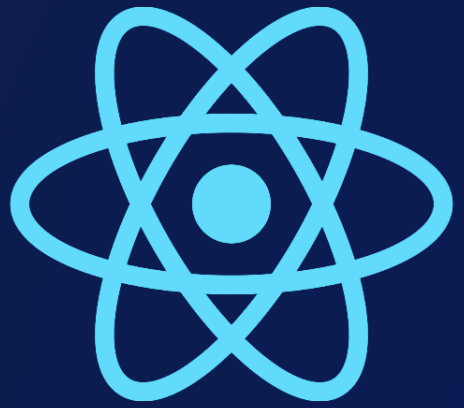
React JS Tutorials

Routing in React -1(React Router)

<BrowserRouter>

<Routes>

<Route>

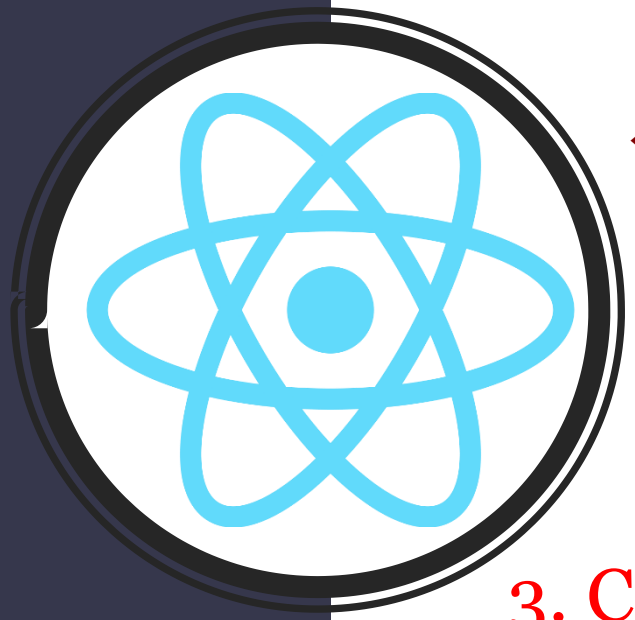


React Routing

1. Install router library

```
npm install react-router-dom
```

2. Configure Routes



```
<BrowserRouter>
      <Routes>
            <Route exact path="/" element={<Home />} />
            <Route path="/about" element={<About />} />
            <Route path="/contact" element={<Contact />} />
      </Routes>
 </BrowserRouter>
```

3. Create Navigation Links

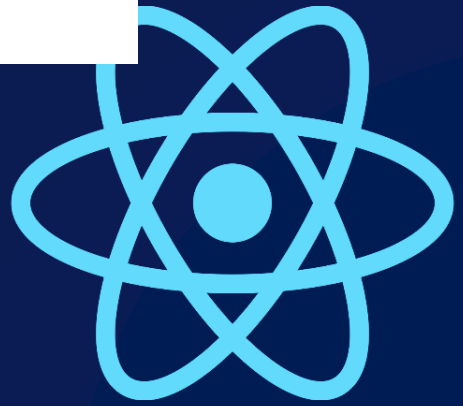
```
<Link to="/">Home</Link>

<Link to="/about">About</Link>

<Link to="/contact">Contact Us</Link>
```

Day – 32

React JS Tutorials

Query Parameters & Route Parameters

Dynamic routing



Query parameters

```
http://amazon.com

http://amazon.com/products?page=5

http://amazon.com/products?category=electronics
```

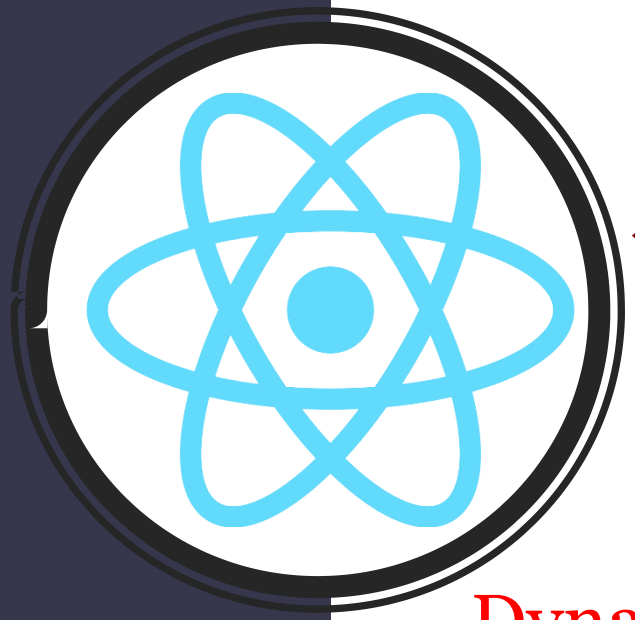
Route parameters

```
http://amazon.com

http://amazon.com/products/10

http://amazon.com/products/100
```

React Routing

Static Routes

```
<BrowserRouter>
   <Routes>
       <Route path="/" element={<Home />} />
       <Route path="about" element={<About />} />
       <Route path="contact" element={<Contact />} />
       <Route path="user" element={<User />}/ >
   </Routes>
</BrowserRouter>
```

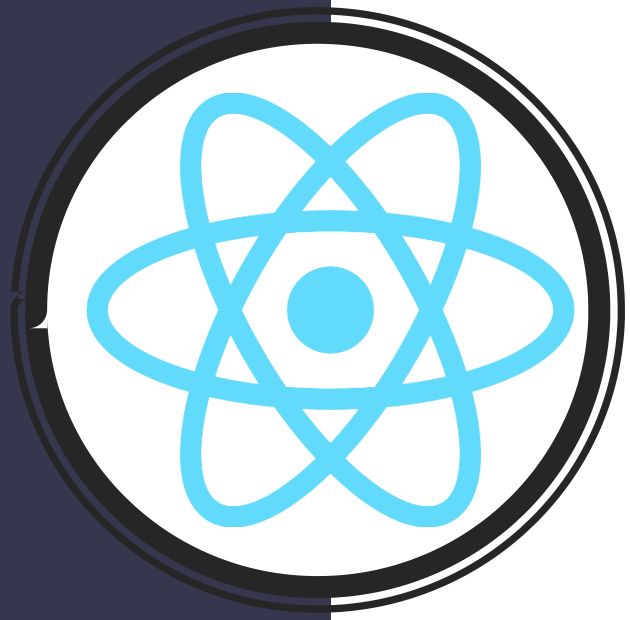


Dynamic Routes

```
<Route path="user/:id" element={<User />} />
```

React Routing

How to use route parameters & Query Parameters



1. `import { useParams, useLocation } from "react-router-dom";`

2 .To access route parameters

```
const { id } = useParams();
```

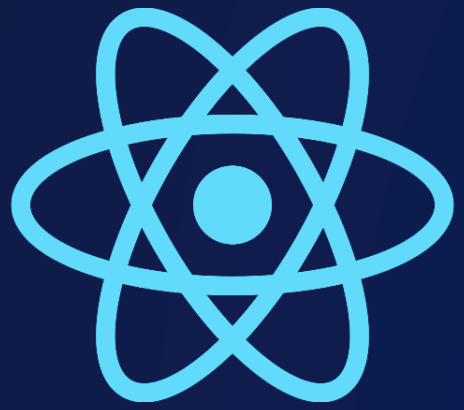
3. To access auqey paremeters

```
const location = useLocation();
const queryParams = new URLSearchParams(location.search);
const page_no = queryParams.get('page');
```

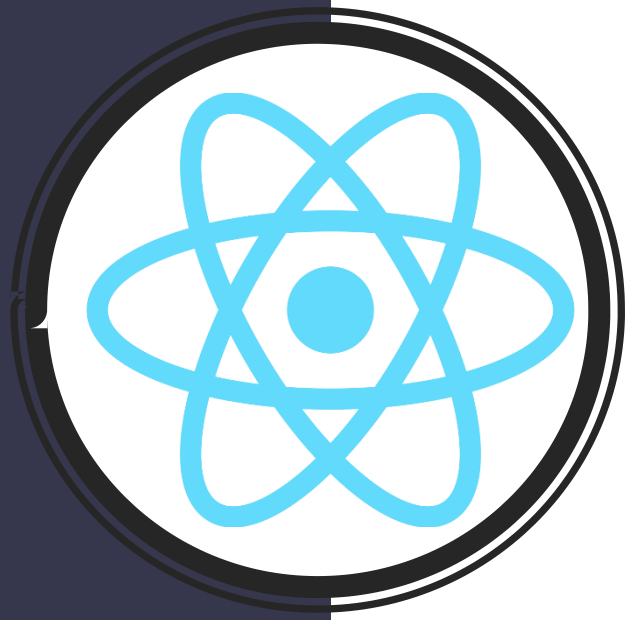
Day – 44

React JS Tutorials

Component Life Cycle methods in React-1



React Component phases(stages)

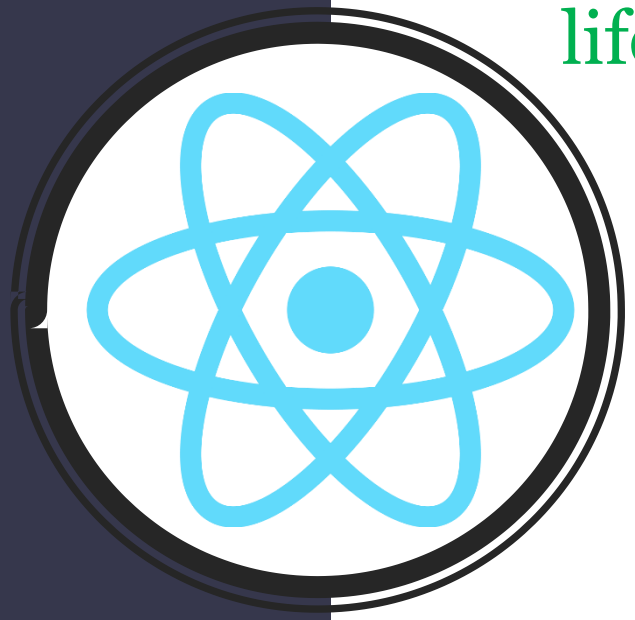


- Mounting
- Updating
- UnMouting

Mounting phase

When a component renders first time then mounting

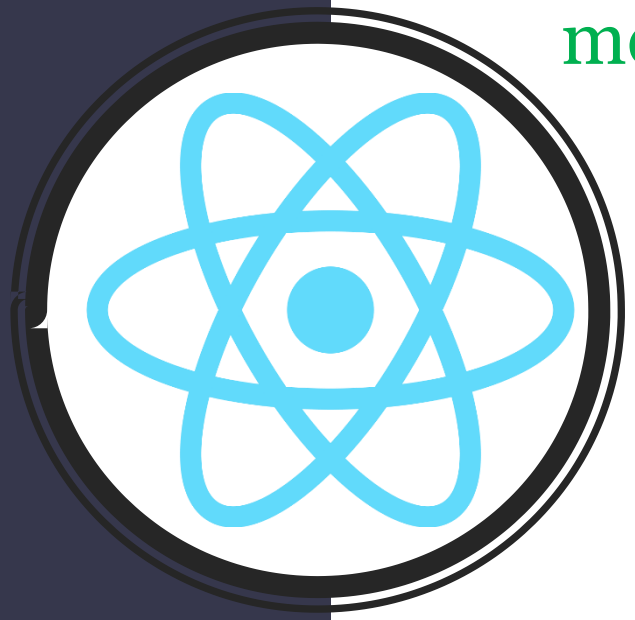
lifecycle methods called as below



- Constructor()
- getDerivedStateFromProps()
- render()
- componentDidMount()

Updating phase

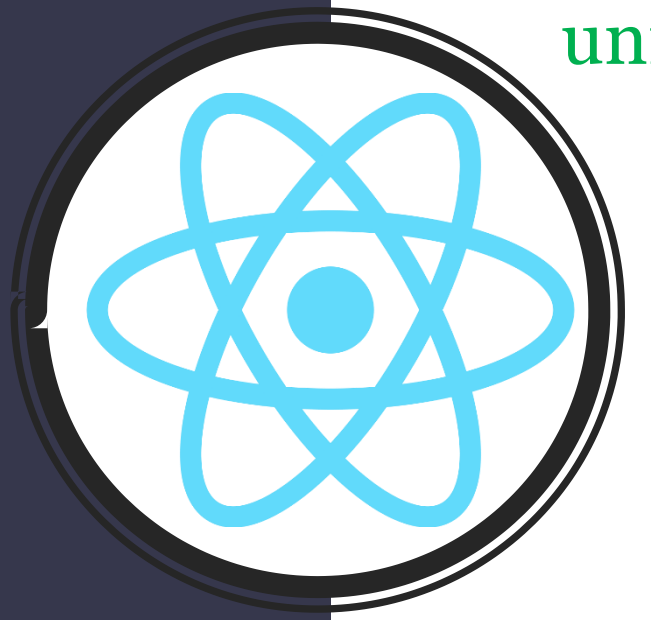
When a component re-renders then updating lifecycle methods called as below



- getDerivedStateFromProps()
- shouldComponentUpdate()
- render()
- getSnapshotBeforeUpdate()
- componentDidUpdate()

Unmounting phase

When a component removed from DOM then unmounting lifecycle methods called .

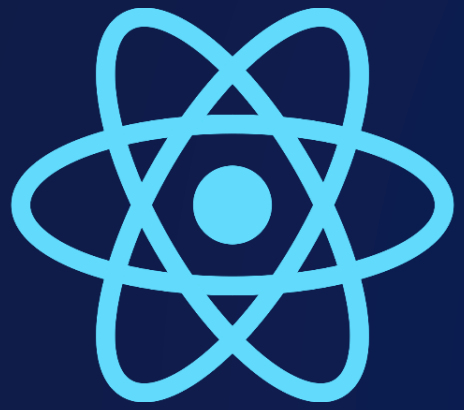


- componentWillUnmount()

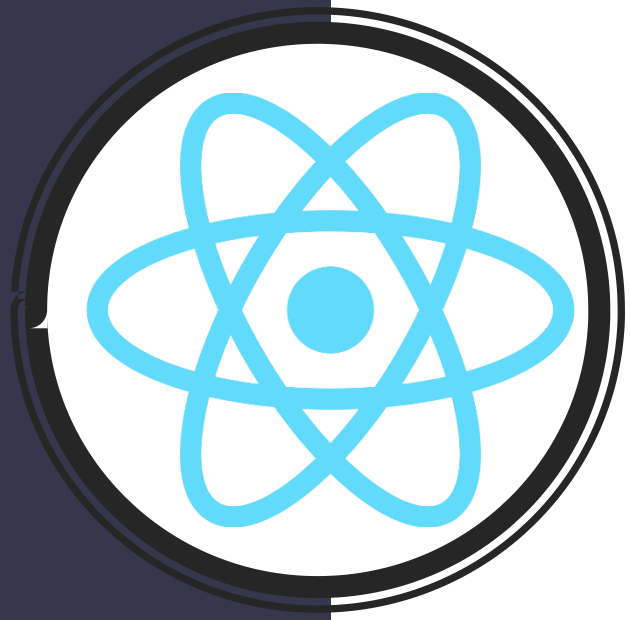
Day – 45

React JS Tutorials

Component Life Cycle methods in React-2



React Component phases(stages)

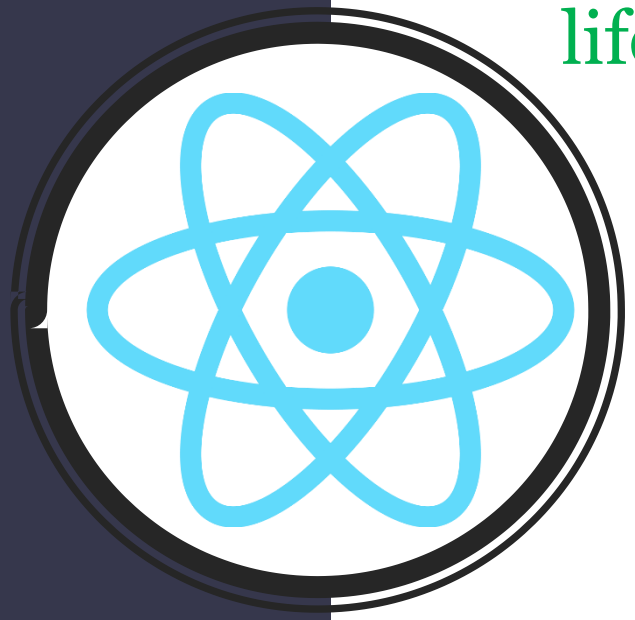


- Mounting
- Updating
- UnMouting

Mounting phase

When a component renders first time then mounting

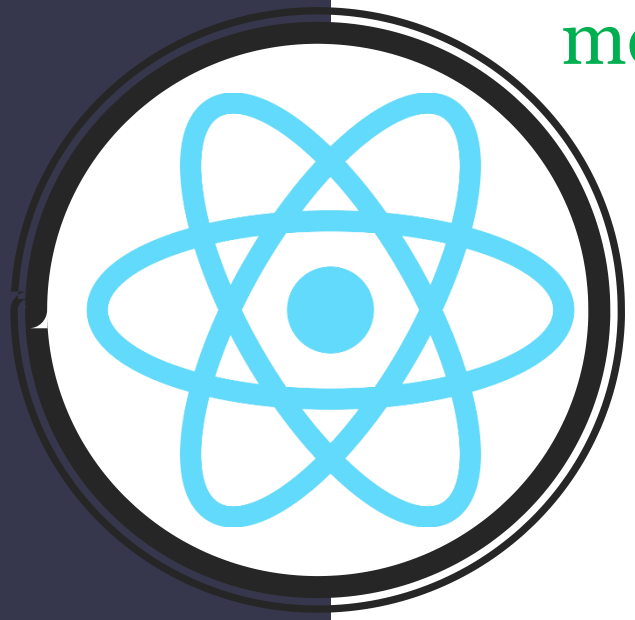
lifecycle methods called as below



- Constructor()
- getDerivedStateFromProps()
- render()
- componentDidMount()

Updating phase

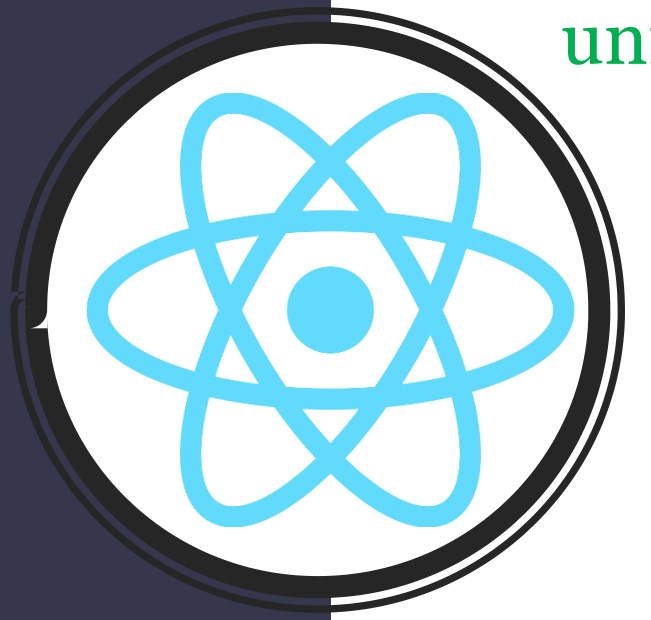
When a component re-renders then updating lifecycle methods called as below



- getDerivedStateFromProps()
- shouldComponentUpdate()
- render()
- getSnapshotBeforeUpdate()
- componentDidUpdate()

Unmounting phase

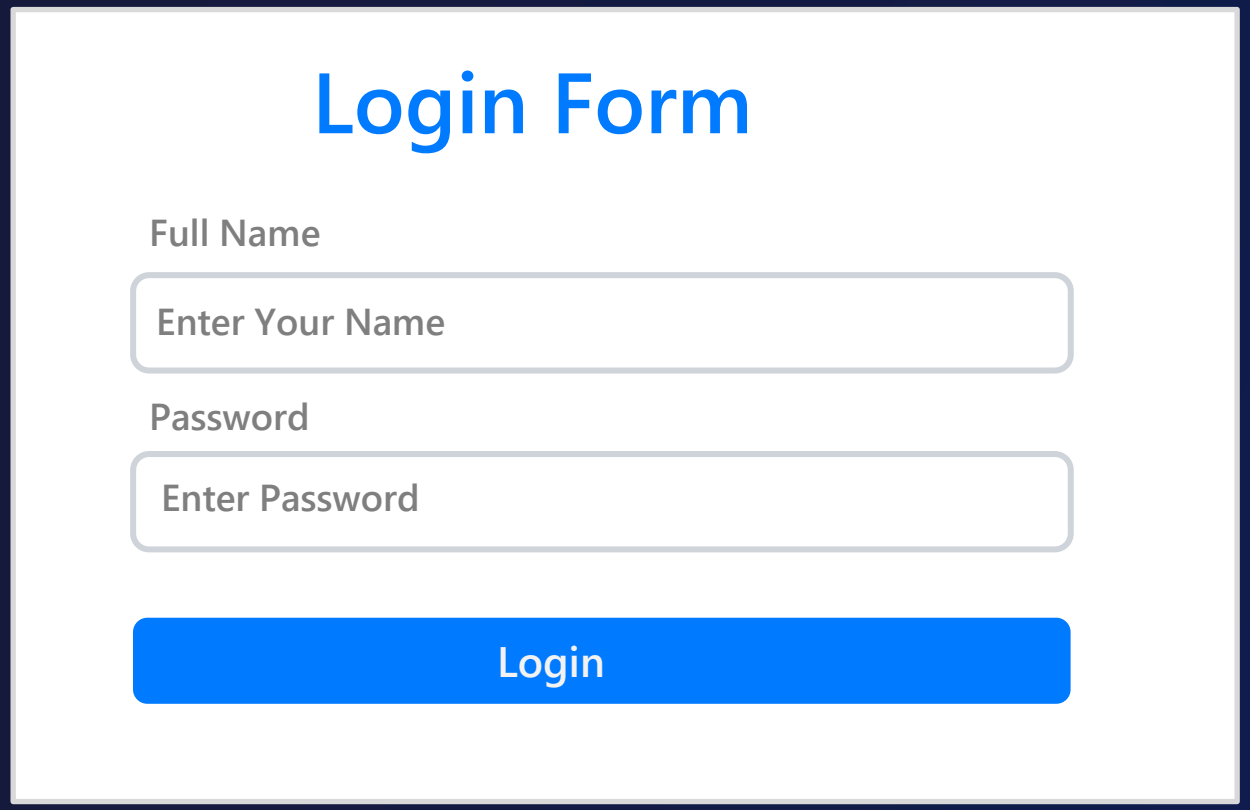
When a component removed from DOM then unmounting lifecycle methods called .



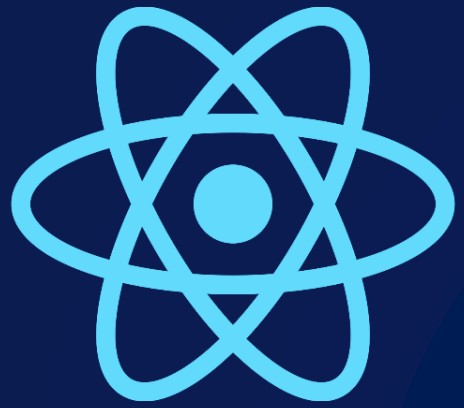
- componentWillUnmount()

Day – 47

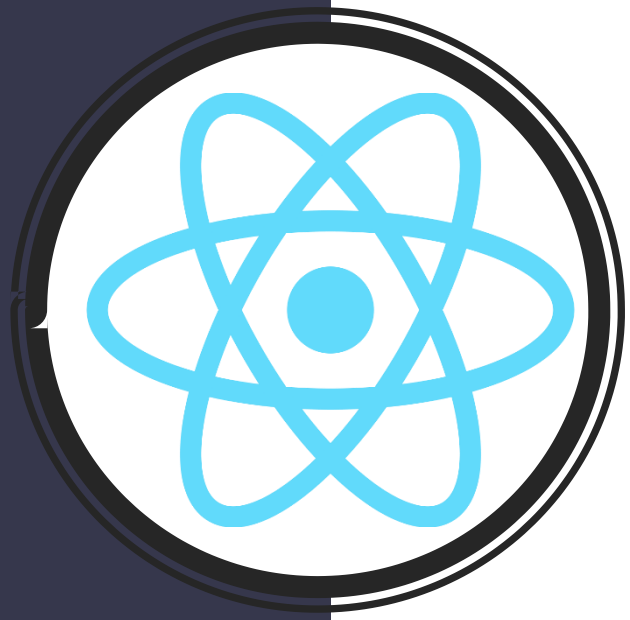
React JS Tutorials



Forms in React-1

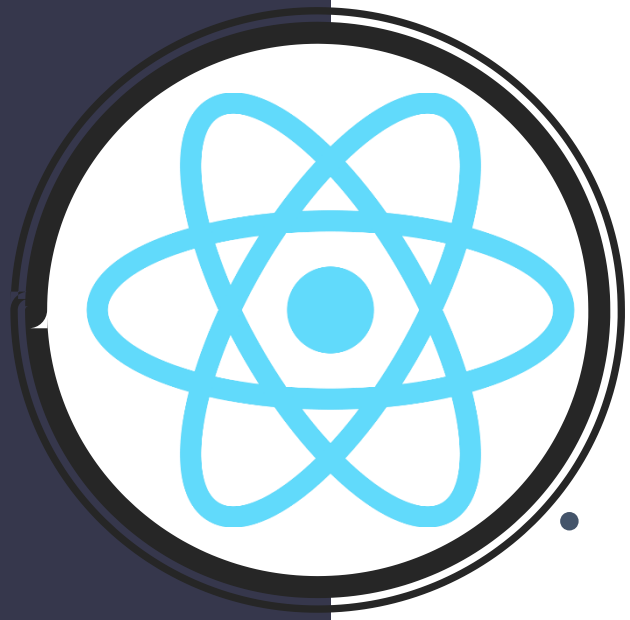


Two ways to handle forms



- Controlled Way
- Un Controlled way

Two ways to handle forms



- Controlled Way

  Using State we will handle form data

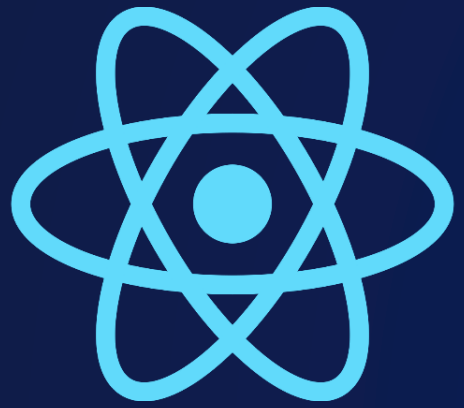
- UnControlled Way

  Using DOM(useRef hook) we will handle form data

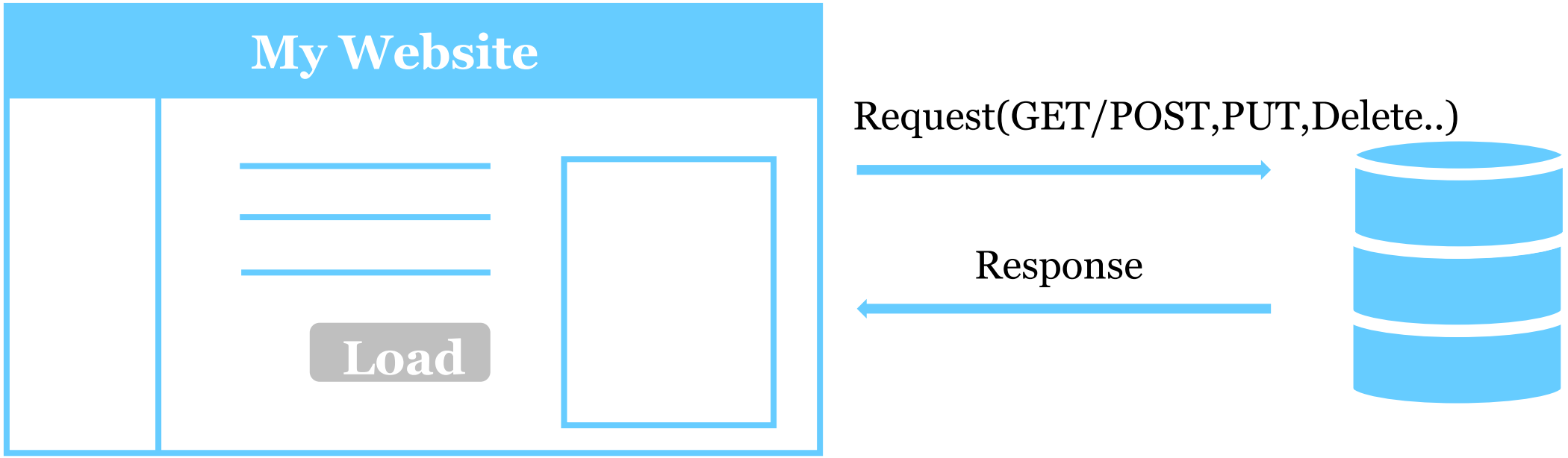
Day – 49

React JS Tutorials

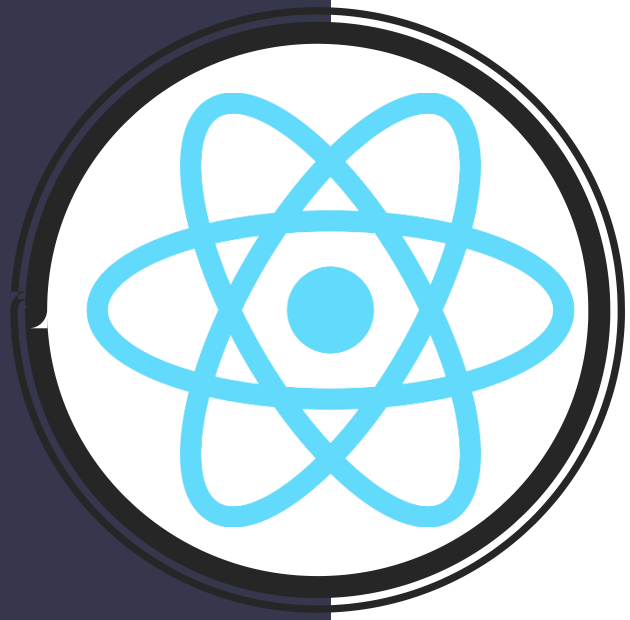
API Integration in React



API Integration

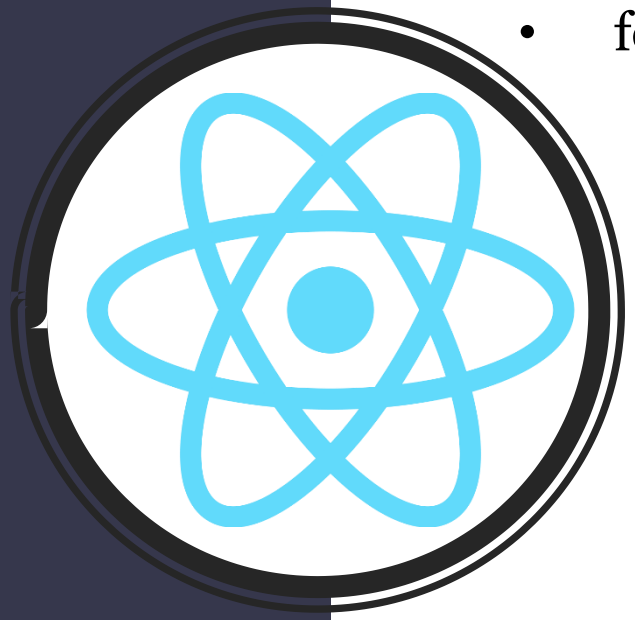


Different ways to do API Call



- fetch() method
- Axios library
- Jquery library

Fetch Method



- fetch() is a method provided by browser which helps to make AJAX call

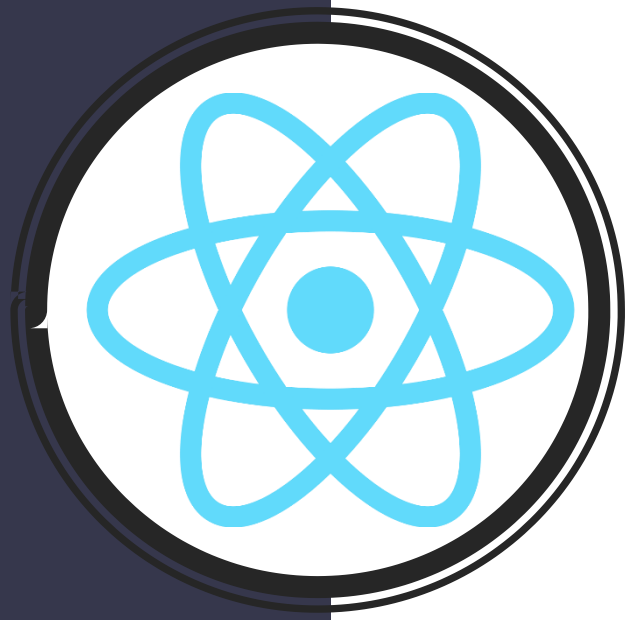
Syntax : fetch (url,[options])

Day – 51

React JS Tutorials

Lazy Loading

Different ways to implement Lazy Loading



- With Conditional Rendering
- With Routing

Day – 52

React JS Tutorials

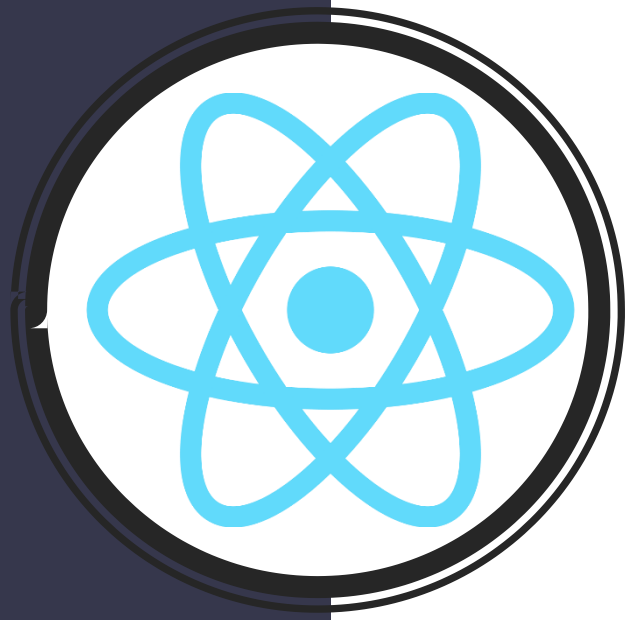
Lazy Loading in Routing

Day – 52

React JS Tutorials

Lazy Loading in Routing

Different ways to implement Lazy Loading



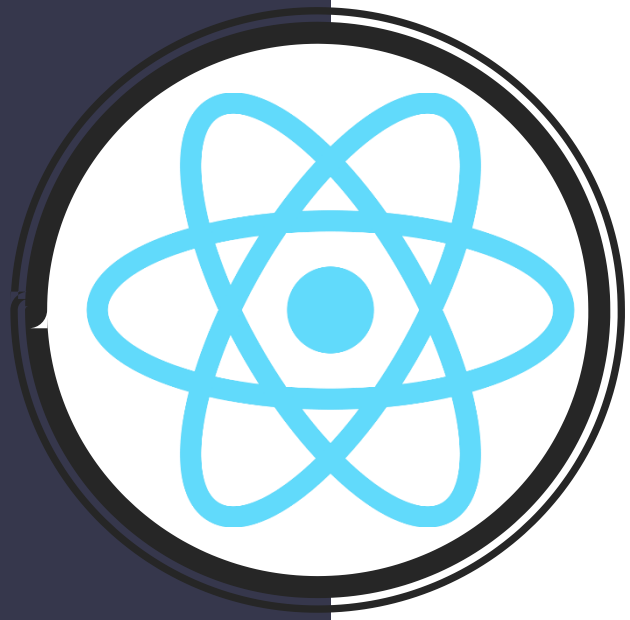
- With Conditional Rendering
- With Routing

Day – 52

React JS Tutorials

Higher Order Components

Higher Order Components



- It is a component that takes another component as input and returns a new component with extra features added to original component

Higher Order Components

```
function Component(props) {
    return (
        <p>Hello {props.name}</p>
    )
}

function App() {
    return <Component name="John" />
}
```

```
function HOC(Component) {
    return (props) => {
      return <Component {...props} styles={styles} />
    }
}
```

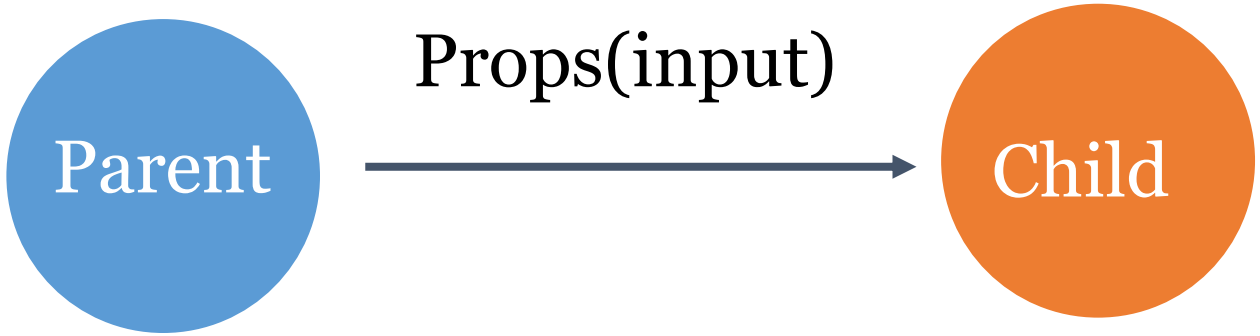
```
let WrappedComponent = HOC(Component)

function App() {
    return <WrappedComponent name="John" />
}
```

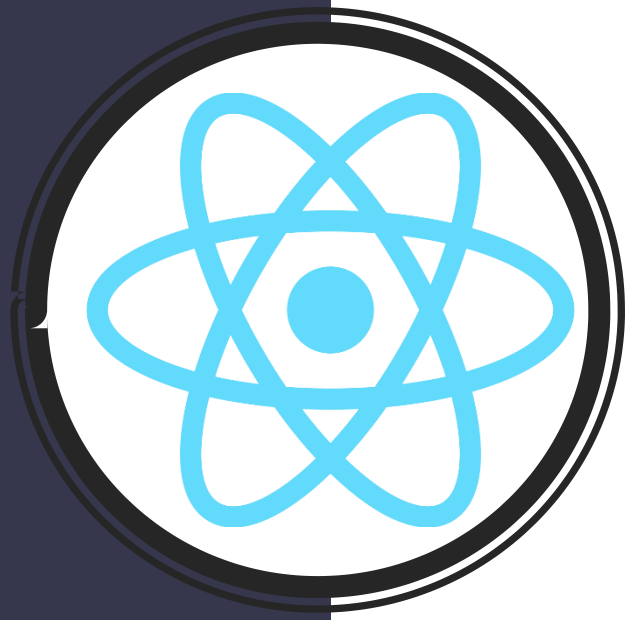
Day – 53

React JS Tutorials

What are Prop Types

Prop Types





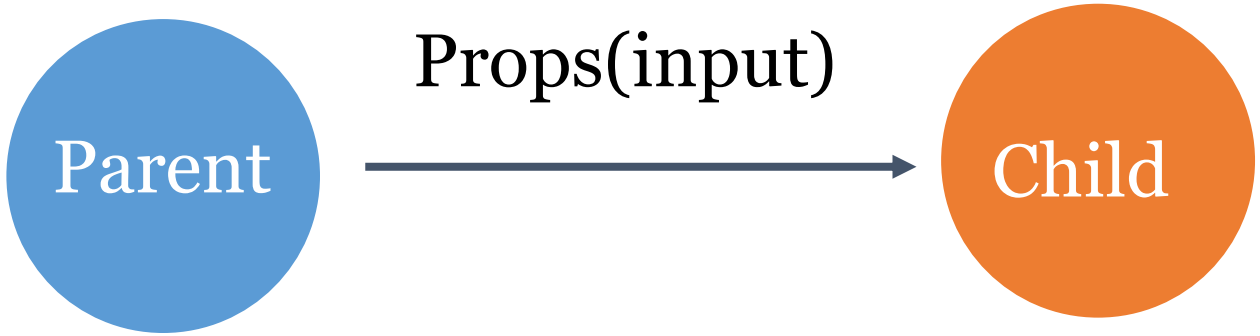
```
function User(props) {
    return (
        <p>Hello {props.name}</p>
    )
}

function App() {
    return <User name="John" />
}
```

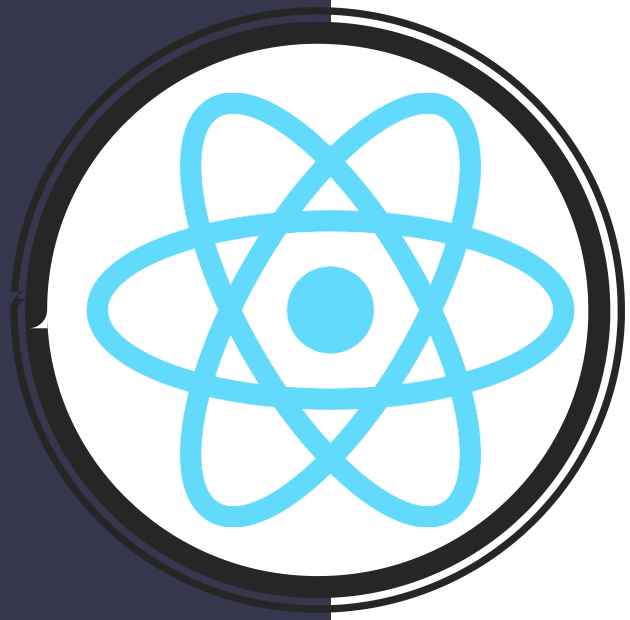
Day – 54

React JS Tutorials

Default Props

Default Props





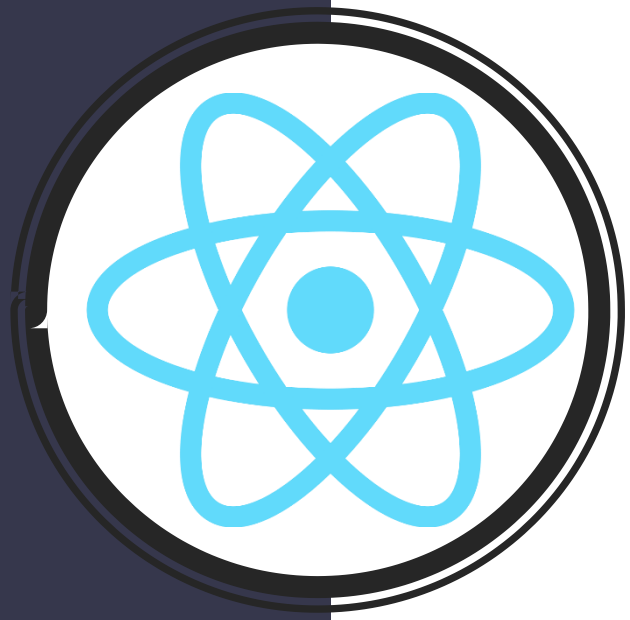
```
function User(props) {
    return (
        <p>Hello {props.name}</p>
    )
}

function App() {
    return <User name="John" />
}
```

Day – 55

React JS Tutorials

JSX & Component Rules

JSX Rules



- Return only single element/component
- Closing tag is must
- camelCase for css properties, events & attributes
- Do not use JavaScript keywords for attributes

Day – 56

React JS Tutorials

React Portals-1

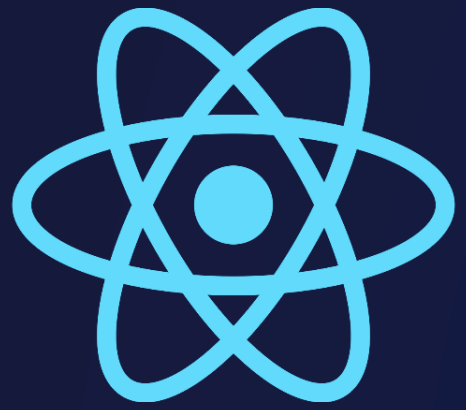
React Portals

```
function App() {
    return (
        <div >
            <div>This  is app component</div>
            <Popup></Popup>
        </div>

    )
}


function Popup() {
    return (
        <>
            <h1>This is popup component</h1>
            <p>Popup content</p>
        </>

    )
}
```

```
function App() {
    return (
        <div >
            <div>This  is app component</div>
            <h1>This is popup component</h1>
            <p>Popup content</p>
        </div>

    )
}
```

Day – 58

React JS Tutorials

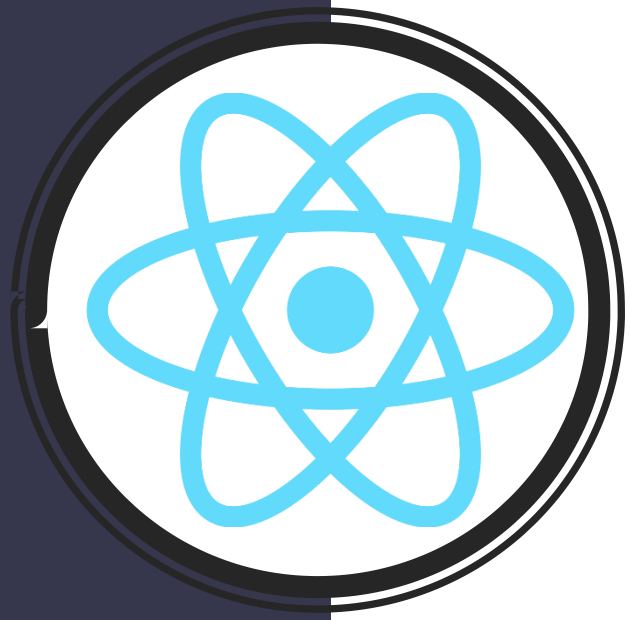


CRUD Operations-1

Add new

ID	Name	Price	Category	
1	redme note 11	25000	mobiles	Edit Delete
2	vivo v20	20000	mobiles	Edit Delete
3	Mac book Air	100000	mobiles	Edit Delete
4	Hp i7	500000	laptops	Edit Delete

CRUD Operations



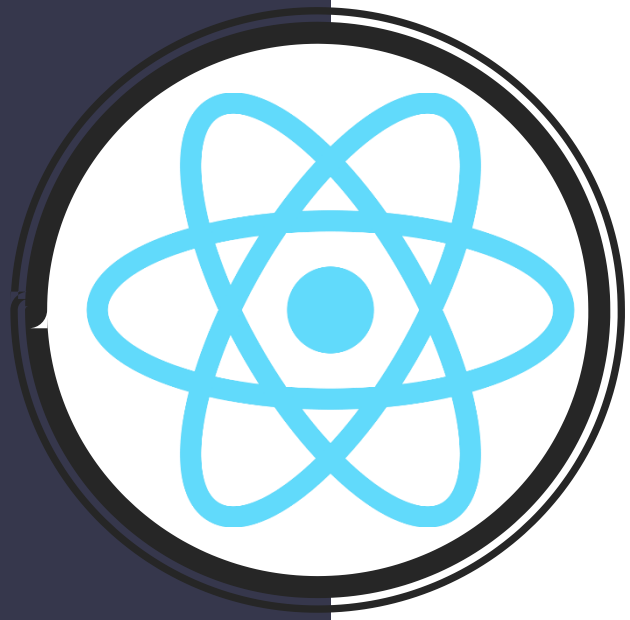
C - CREATE

R - READ

U - UPDATE

D - DELETE

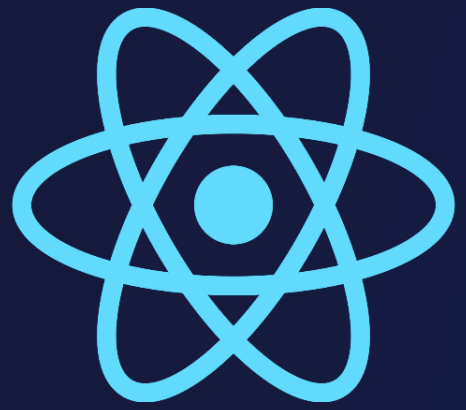
CRUD Operations



- npx create-react-app crud
- npm install bootstrap
- npm install axios

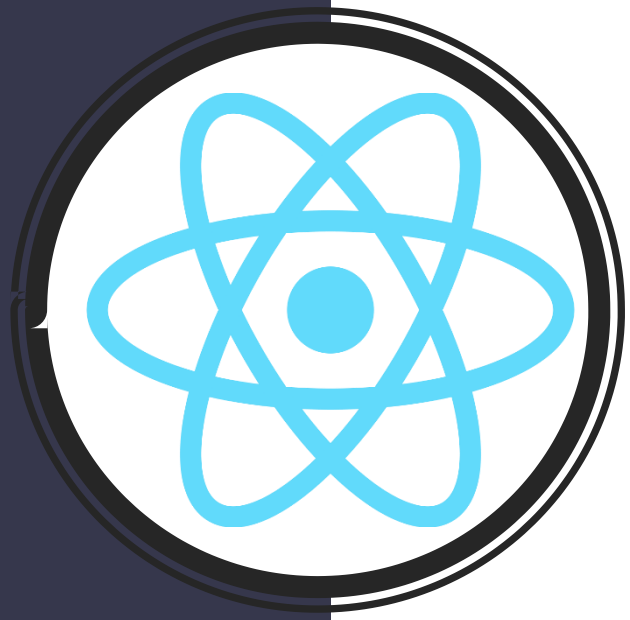
Day – 60

React JS Tutorials



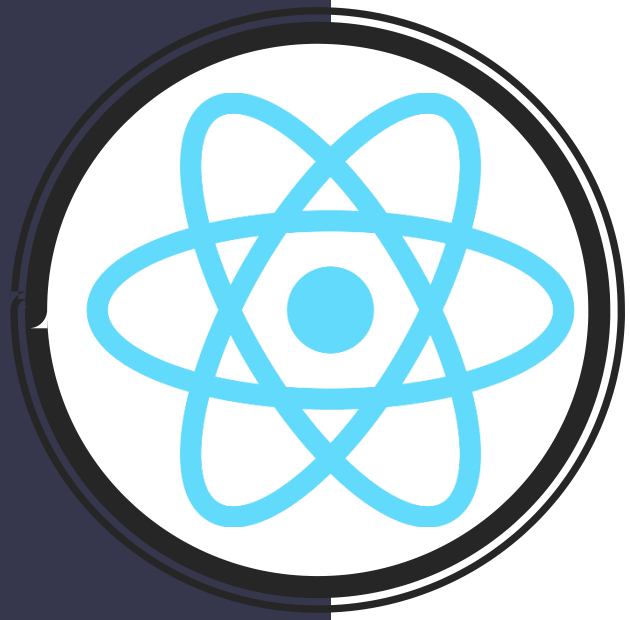
Use Reducer() hook-1

Use Reducer()



- Used to handle state of component like useState()
- Handling complex state useReducer() is good

Syntax :



Use State()

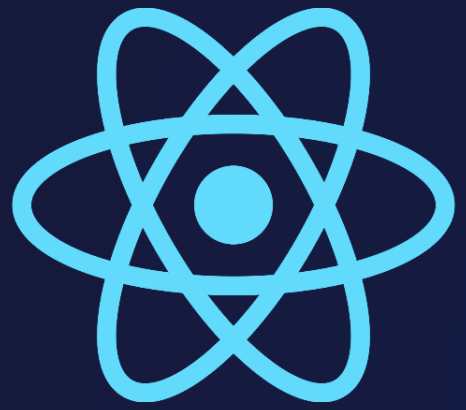
```
useState(initialstate);

const[count,setCount] = useState(initialstate);
```

Use Reducer()

```
useReducer(function,inititalstate);

const[count,dispatch] = useReducer(function,inititalstate);
```

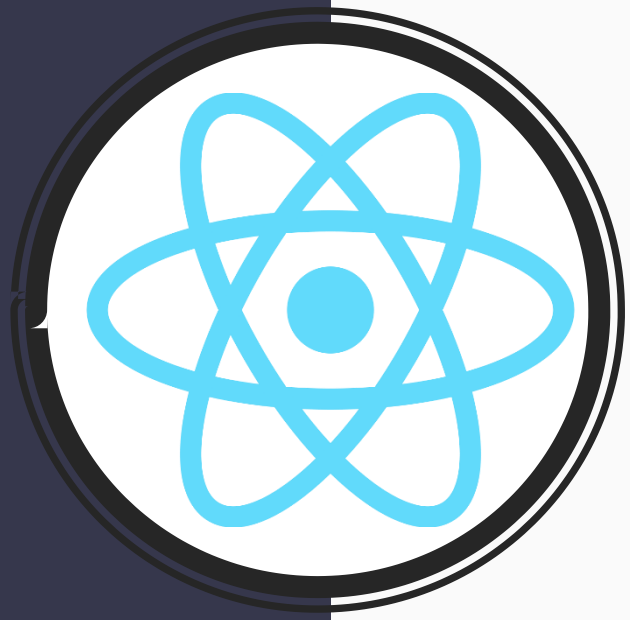
Day – 61

React JS Tutorials



Use Reducer() hook-2

Syntax :

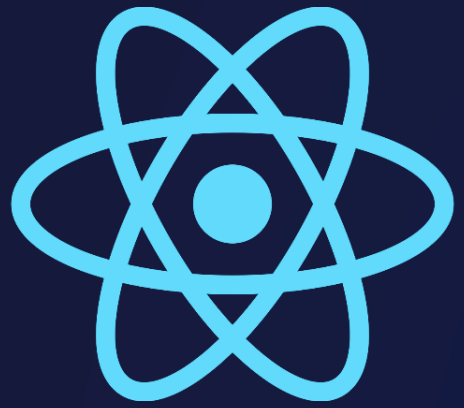


Use Reducer()

```
const[count,dispatch] = useReducer(function,inititalstate);
```

Day – 62

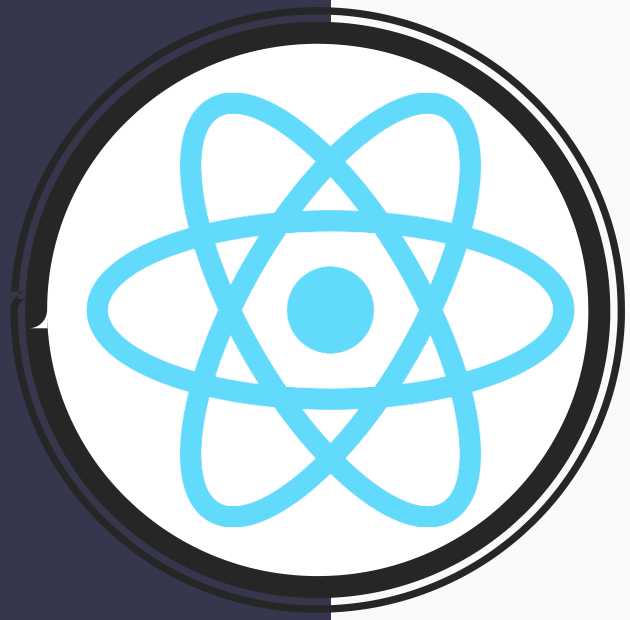
React JS Tutorials



REDUX -1

Introduction

NGRX :

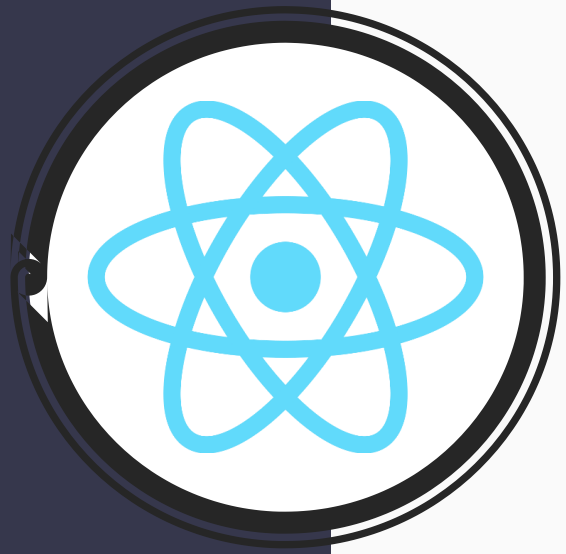


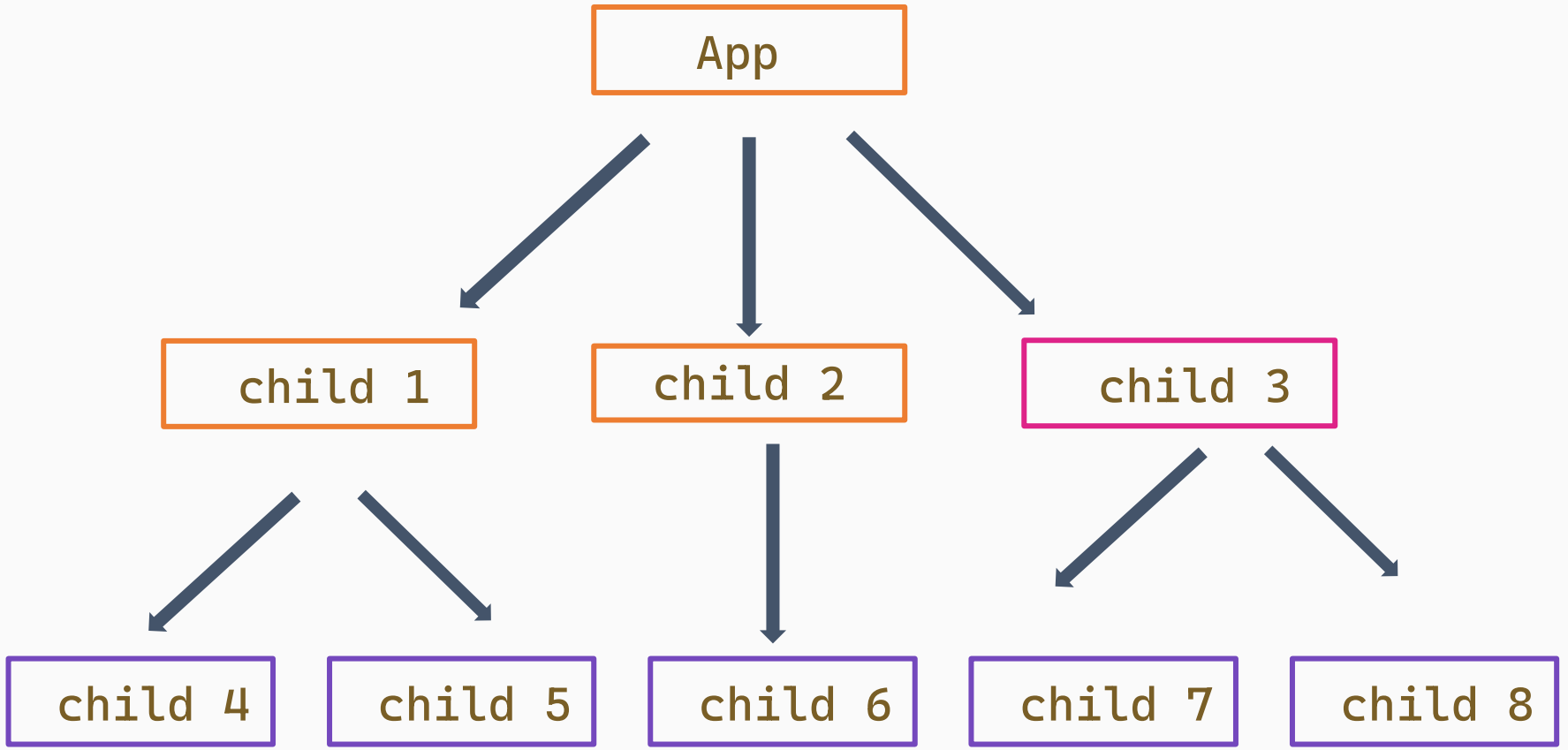
It is a library based on redux pattern

Used for State management for whole application

- Store
- Reducer
- Dispatch
- Action

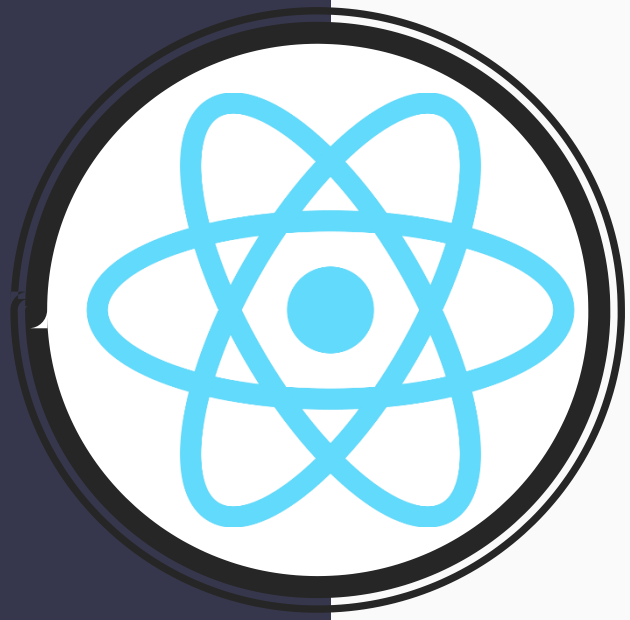
Redux :

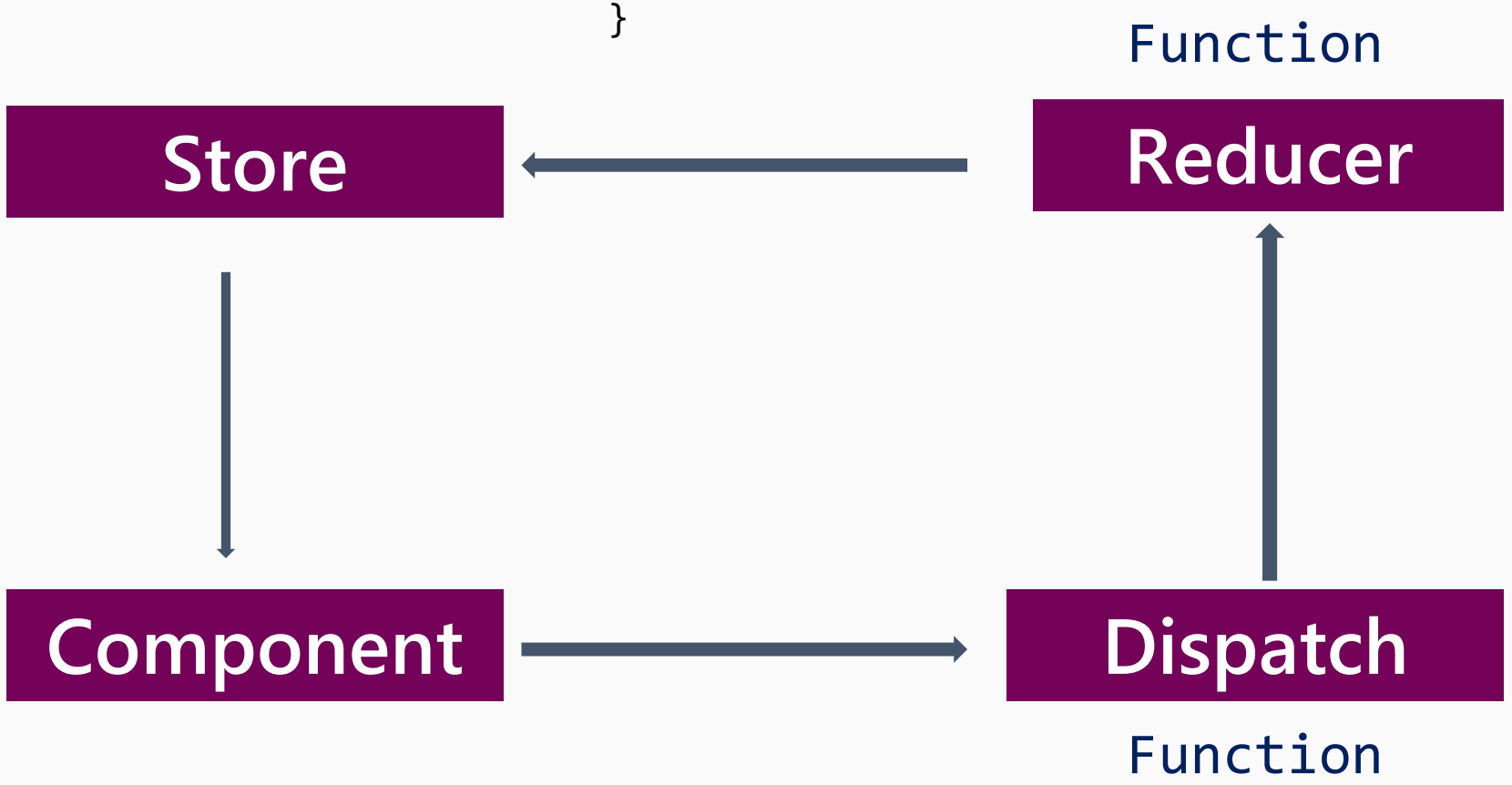



App
child 1
child 2
child 3
child 4
child 5
child 6
child 7
child 8

Redux :

```
let data ={
    products:[],
    orders:[],
    cart:[],
    userDetails:{}


}
```

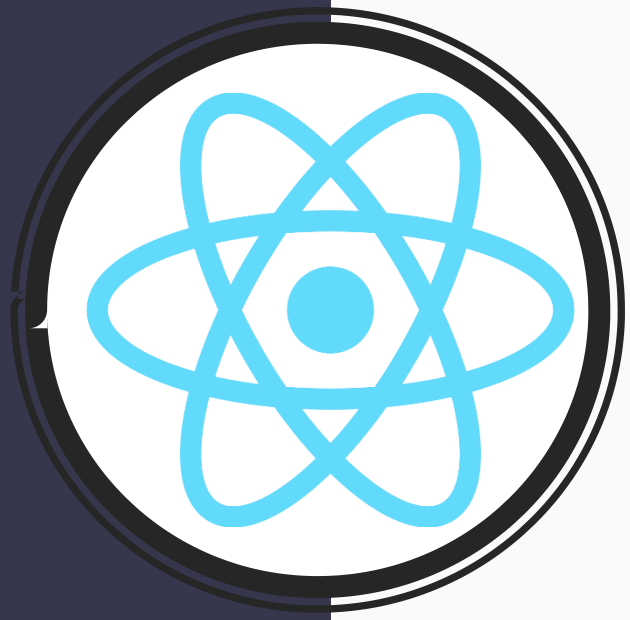




Redux :

JavaScript library

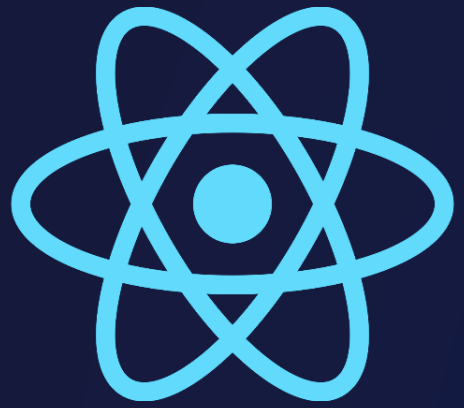
Used for State management for whole application



- Store
- Reducer
- Dispatch
- Action

Day – 63

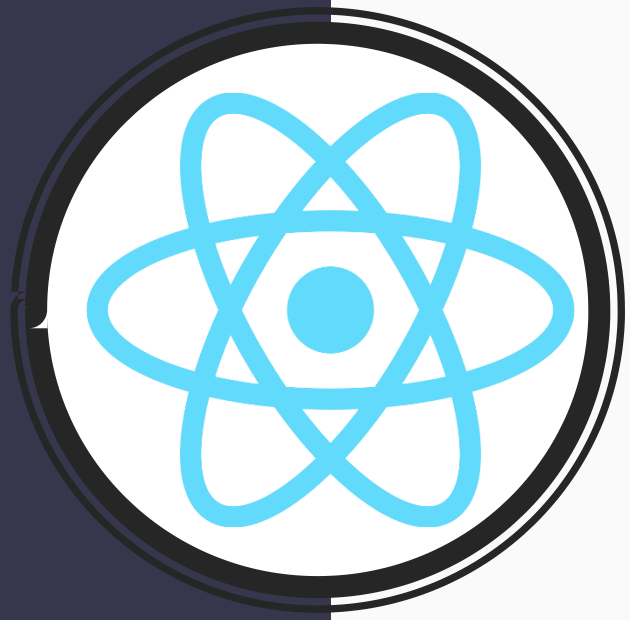
React JS Tutorials



REDUX -2

Redux Implementation

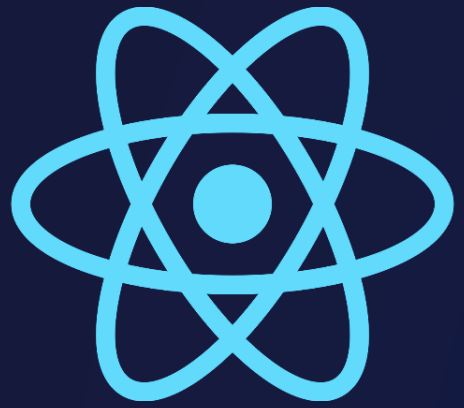
Redux :



```
npm install redux

npm install react-redux
```

Day – 64

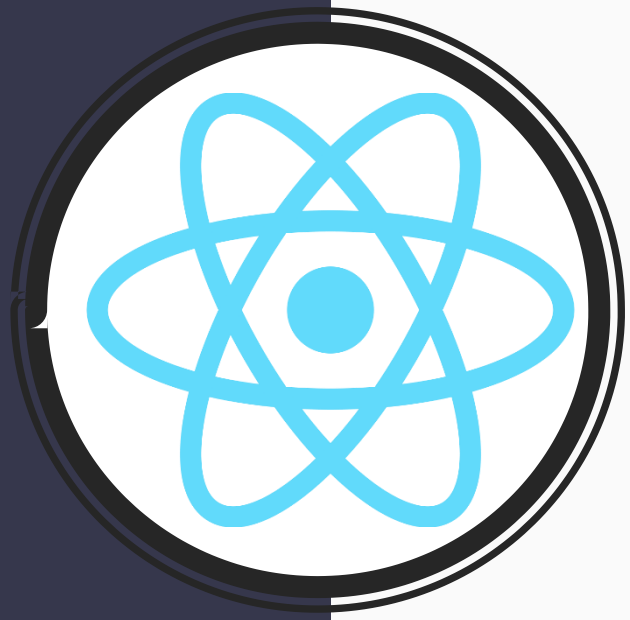
React JS Tutorials



REDUX -3

Use Redux in React Application

Redux :



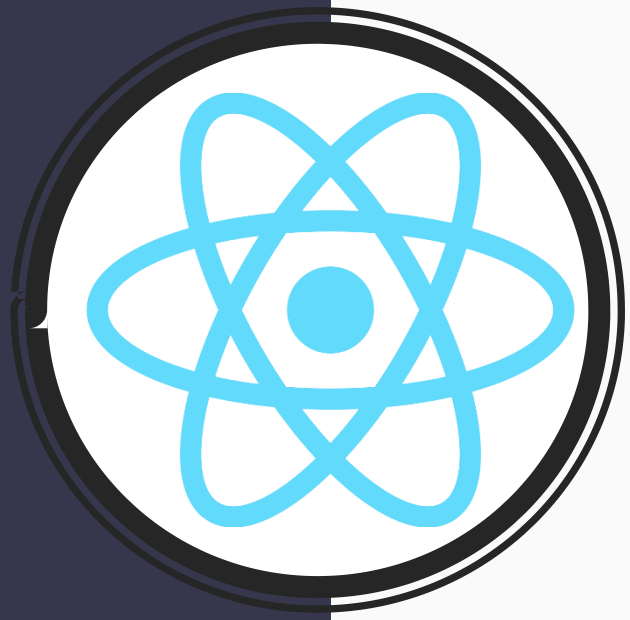
```
npm install redux

npm install react-redux
```

Day – 69

React JS Tutorials

React Application with Vite

React with Vite

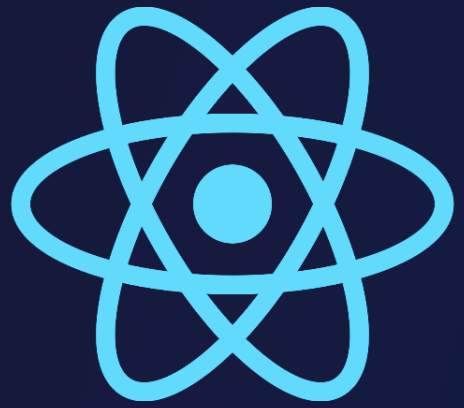


1. `npx create-vite app-name`

2. `npm run dev`

Day – 70

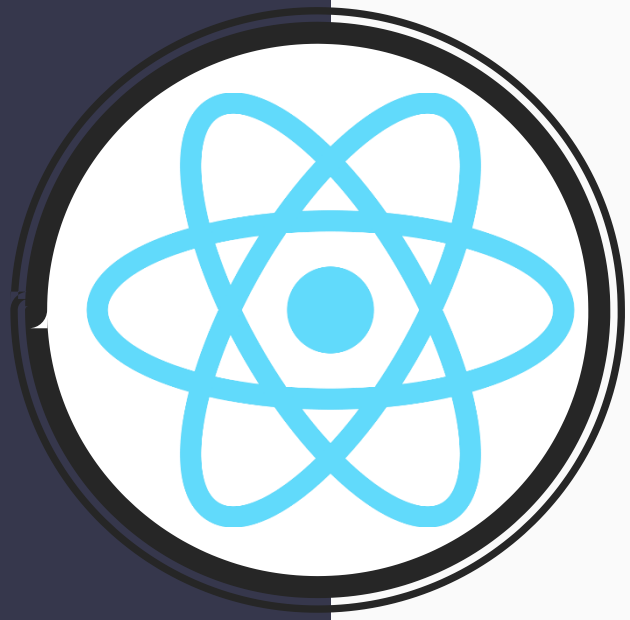
React JS Tutorials



REDUX Toolkit-1

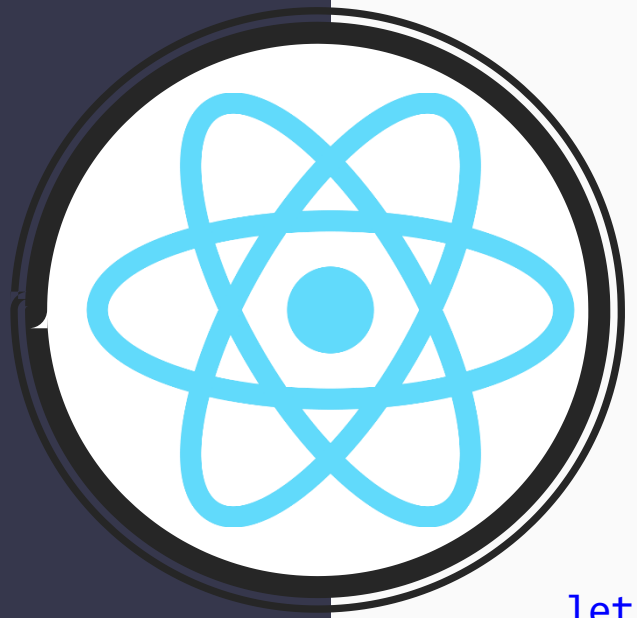
Introduction

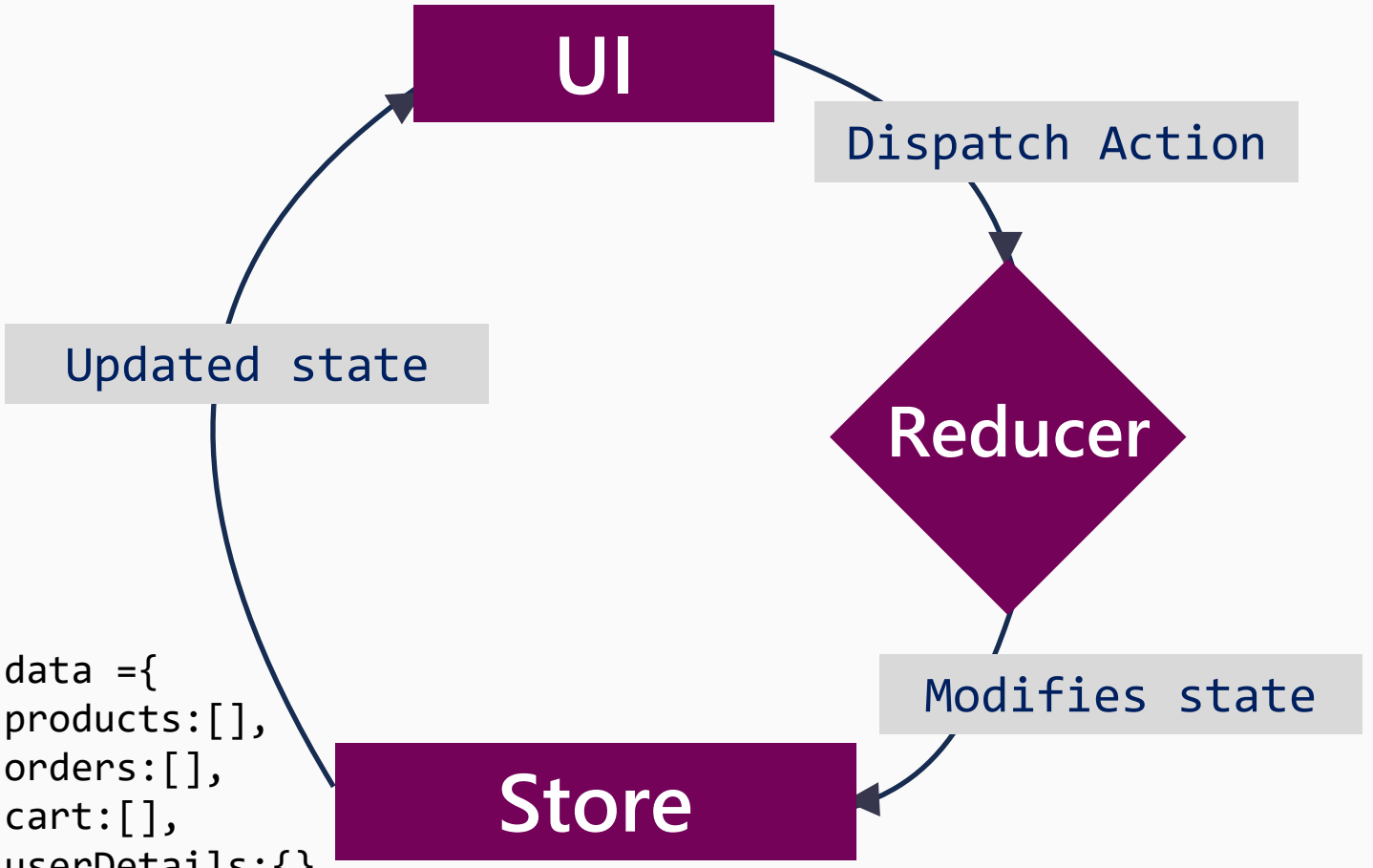
Redux-Toolkit :



- Store
- Reducer
- Dispatch
- Action

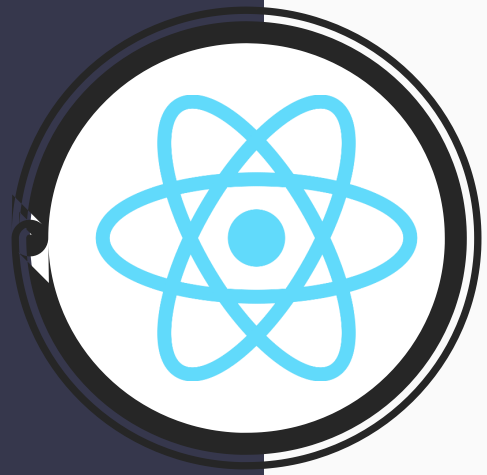
Redux-toolkit :






```
let data ={
    products:[],
    orders:[],
    cart:[],
    userDetails:{}


}
```

Redux-toolkit :



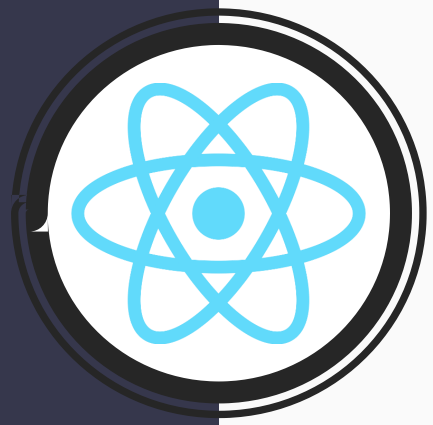
1. It is a global state management library like redux.

2. It simplifies state management setup.

3. createSlice function is used to create reducer & will also generates action creators automatically.

Redux-toolkit :

4.Provides feature to modify state directly.



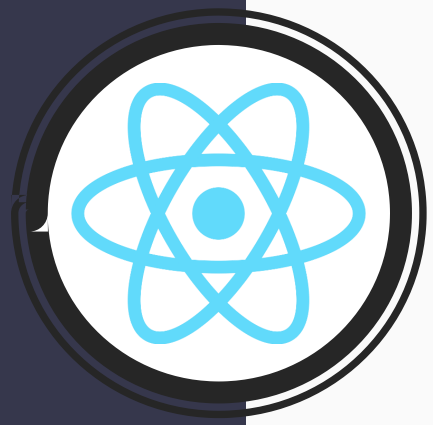
```
function accountReducer(state = initialState, action)
{
  switch (action.type) {
    case “updateMobile":
      return { ...state, mobile: action.payload };
  }
}
```

redux

```
const userSlice = createSlice({
  name: "user",
  initialState,
  reducers: {
    updateMobile: (state, action) => {
      state.mobile = action.payload;
    },
  },
});
```

redux-toolkit

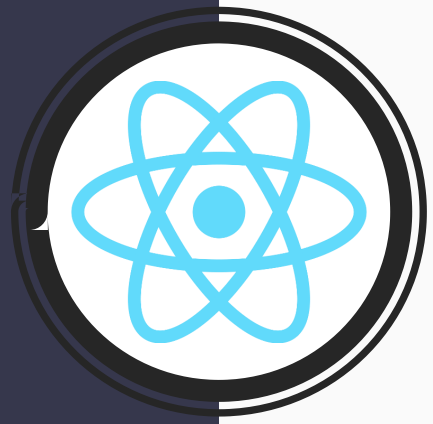
Redux-toolkit :



5. Supports asynchronous state changes, no need of any middleware libraries.

6. Redux Devtools library integrated

Redux-Toolkit Setup :



- npm install @reduxjs/toolkit
- npm install react-redux
- Use createSlice for state update logic

```
const userSlice = createSlice({
  name: "user",
  initialState:data,
  reducers: {
    updateMobile: (state, action) => {
      state.mobile = action.payload;
    },
  },
});
```

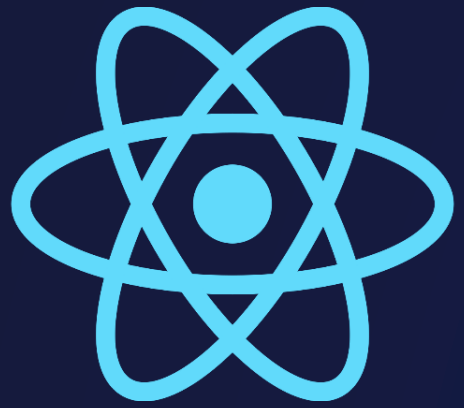
```
const data = {
  balance: 0,
  fullName: "",
  mobile: null,
};
```

- Configure slice using configureStore method

```
const store = configureStore({
  reducer: {
    user: userSlice.reducer,
  },
});
```

Day – 71

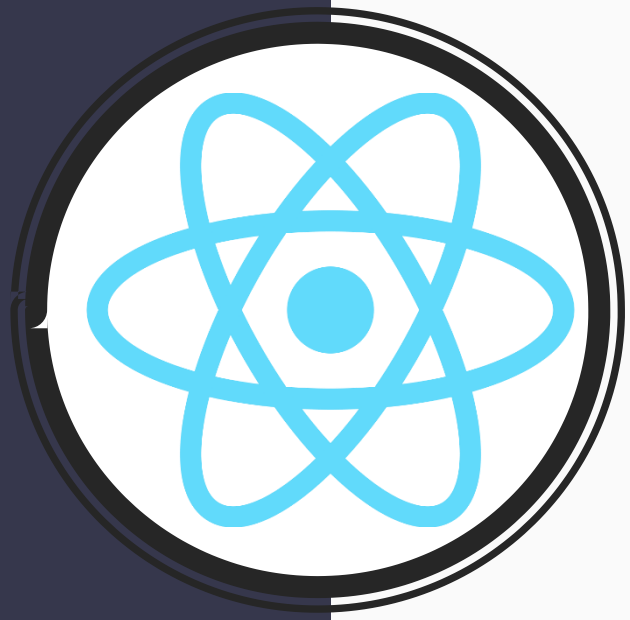
React JS Tutorials



REDUX Toolkit-2

Implementation Example

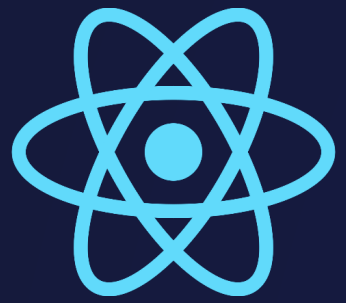
Redux-Toolkit



```
1. npx create-vite app-name --template react

2.npm install @reduxjs/toolkit

3.npm install react-redux
```

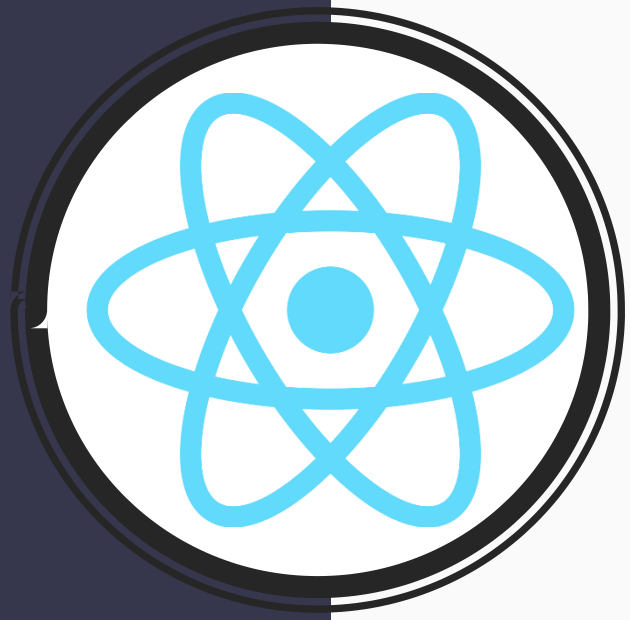
Day – 75

React JS Tutorials



Deploy/Host React Application

Hosting providers



- Netlify
- Heroku
- Vercel
- aws

Build Command

```
npm run build
```