REACT

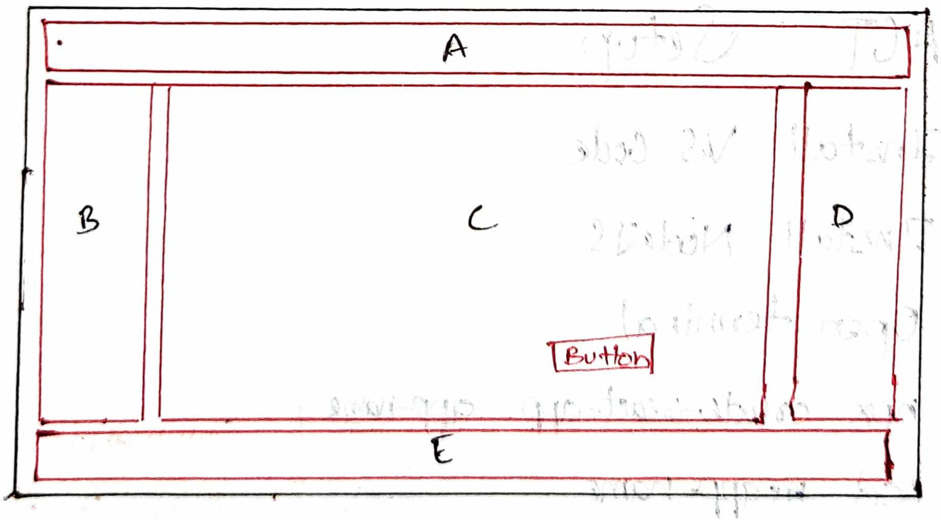
React is a JS liberary used to create UI

React is all about components.

It is a <u>component</u> based architechure.

Components lets you split the UI into independent, reusable pieces, and think about each piece in isolation.

Conceptually, components are like JS functions. They accept arbitrary inputs (called "props") and return React elements describing what should appear on the screen.



A, B, C, D and E are components.

Button is also a component inside a component C.

Why we need React?

→ JS is based on imparative approach but react is based on declarative approach.

→ Declarative approach in Reeact means that we need to tell the end state only and React will work autometically.

Imperative means line by line.

→ SPA approach (Single Page Application)

→ Faster development, etc.

* REACT Alternatives

→ Angular

→ JS

* REACT Setup

→ Install VS Code

→ Install NodeJS

→ Open terminal

→ npx create-react-app app-name

→ cd ~~m~~ app-name

→ npm start

commands to run in terminal

Folder ↴
∨ src
File → | index.js → It is the entry point of JS.
It is the first file to execute.

To use any file we import it first.

| package.json → contains dependencies & scripts
∨ public
| index.html → It is the main HTML file.
(contains root element)
∨ src
| index.js → It fetches the root element from index.html. (root.render(<App />));
(It contains 'App' component)
| App.js → It contains the working of App components.
| App.css → css file for styling

Note: To use any function, first you need to export from where it is created and import where you want to use.

To create a component always create a folder for that component and create JS and css files in it. This folder is created inside src folder.

Create Component

To create a component we create a js file which contains a function, to use that function we export in the same file and import where we want to use it.

The function we created for the component returns the JSX code.

Code (component file)

```
import './item.css';
const name = "Rishabh";
function Item() {
    return <p className='rishabh'> Rishabh </p>
    return <p className='rishabh'> {~~Rish~~ name} </p>
}
export default Item;
```

here we use className because class is a reserved keyword in JS

We use {} when data changes dynamically.

App.js file (code)

```
import './App.css';
import Item from './components/item';
function App() {
    return (
        <div>
            <Item> </Item>
            <ItemDate> </ItemDate>
        </div>
    );}
export default App;
```

It is mandatory to have a parent, when you have multiple child.

Another component

* Use of Props

Props are arguements passed into REACT components.

Props stands for properties.

Code (component file)

```
function Item(props) {
        const itemName = ~~"Rishabh"~~ props.name1;
        return (<p className="Name"> {itemName} </p>);
}
export ______ ;
```

Code (App.js file)

```
import ______ ;
function App() {
    return (
      <div>
         <Item name1 = "Rishabh"></Item>
      </div>
    );
}
```

⇒ Another way (App.js)

```
function App() {
   const response = [        ← array of objects
      {
        itemName : "Kushwaha";
      } — — — —
   ]  } — —
```

```
return (
    <div>
        <Item name1 = {response[0].itemName}></Item>
        - - - ___
    </div>
    );
}
```

Index of object

⇒ But what happens when we do this

```
<Item name1 = {response[0].itemName} I am Rishabh </Item>
```

$\underbrace{\text{I am Rishabh}}_{\text{This will not be visible.}}$

To make this visible, we need to add this in item.js file.

```
<div>
    <p className ="Rishabh"> {itemName} </p>
    {props.children}
</div>
```