Revising React

1. **Initial Steps:**
   1. Setting up the structure for revising the react.
   2. We will use the vite for making the react folder setup, vite is the bundler for making the folder structure for React.
   3. Added tailwind CSS in the project.
2. **Basic Concepts:**
   1. React + React DOM = Web Applications
   2. React + React Native = Mobile Applications
   3. We need react because when we have to update only single element so using JS our whole tree needs to re-render the application but **by using React** when we click on button to change something it **makes the copy of our real DOM called virtual DOM** and it compare the changes made in the virtual DOM and then it merge the changes into the real DOM.
   4. JSX is the combination for HTML and JS.
   5. Our main working file is **App.jsx** and we export our whole code.
   6. In **Main.jsx** file we just import all codes from the while and export it to the **index.html** file.
   7. We can’t directly return the two tags together, to return more than 1 tag we have to use the **empty fragments “<> </>” or** **use a simple empty div tag <div> </div>** to store all the elements.
   8. To declare any variable, functions; declare them outside the return statement inside the function.
3. **Hooks:** Hooks are special type of functions which provide powerful power to react. Hooks are functional components.
   1. **Use State:** It is a special container used for storing and updating the data into the functional component. **State** means the data which changes over time.
      1. **Syntax:**

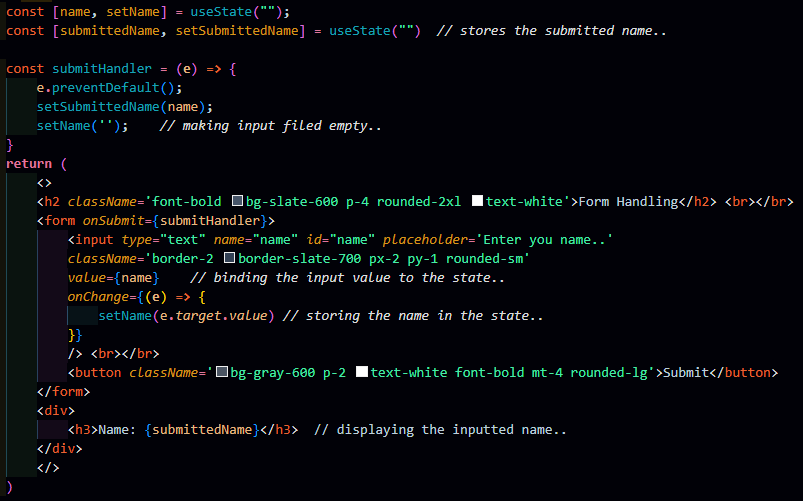


Here, user: It a read only variable, Set user: It is write only variable. Whenever we assign values, we assign to user; in the example you can see that Aman Kumar is assigned to user and set user will be used when we need to change the value of the user.



Like, we are changing the user’s value to new user value using use state.

1. **From Handling:**
   1. **Preventing the default behavior of the form:** After filling the input filed, when we click on submit button it reloads the page and our text in input field is removed to stop this we will prevent the default behavior of our form.
2. **Two way Binding:** 
   1. It is used to input the text in the **input field using Hooks NOT directly** putting the values in the field.
   2. Example for the two-way binding:👇



1. **Components:** 
   1. Components allow you to break down a complex UI into smaller, reusable, and independent pieces of code. You can use these components many times.
   2. Name of component file **must start from capital letter.**
2. **Props/Props Drilling:**
   1. **Props:** When data passed from parent to a child component.
   2. **Props Drilling:** When data passed from parent to a child component, even the component who comes between them don’t need that data but they have to store and pass that data, so that child component can access it.
      1. App (Parent) → ComponentA → ComponentB → ComponentC (Child)
      2. Child needs the data from Parent but A and B need to store and pass the data.
      3. **This leads to memory consumption and made our app slow.**
   3. **Steps for passing properties (props):**
      1. We will create a card component (for example).
      2. To pass the data across the components we will pass a parameter in the Card function component.

****

In this case we have passed the **props as parameter.**

* + 1. In the card component wherever we need that data needs to pass, we will use “**{}**” curly brackets.

****

We have used the {} brackets wherever we need to pass the data.

* + 1. When we call the card component in the **App.jsx** file we will pass all the data with the same variable name.



* + 1. Keep in mind that variable passing while calling the components must match with the variable passed in that component.
  1. We have created a card with Name, Age, Profession, Profile picture and State.
     1. We have created a JSON data in **App.jsx** file.
     2. Then, we have used the **map function on JSON** data and in the return statement we have **returned our card with all the properties available in the JSON data**.

****

* + 1. And in our card component we have passed a parameter and using this parameter we have passed all the properties to their desired place. 

1. **Integrating API**
   1. To use API **install Axios** by using **npm install axios**.