States in React

What is a React Hook ?

In your react codebase, you will find a lot of functions which are serving a specific purpose. The functions can be inbuilt i.e. provided by react or you can make a custom one as well. These are called as React Hooks. Example:

- useState
- useEffect
- useId
- useRef
- and more .....

Generally react hooks have one or more of the following properties:

- Adding state to a component
- Reusable UI logic is mentioned in these
- They might control component lifecycle

Most of the react hooks, starts with the prefix `use` but its not mandatory. You can make your own react hook, without `use` as the prefix.

What is a state ?

States are memory of a component, which is tightly coupled with the component behaviour. To update value of state variable, we cannot just use plain JS code, instead there are special setter function which we need to use to update the state variable. If we don't use setter functions to update the state's value then we will not see any impact on the UI layer.

State variables are very special, any change in the state variable will cause changes in the UI. How ? We will talk the internals later.

To make a state in react, we have a special hook called as `useState`. So, useState is a hook and using this we will be able to get a state variable inside the component. This function takes one argument i.e. initial value of the state variable.

This useState hook returns us a 2 length array which has the following:

- The first element is the state variable we wanted to create

- The second element is the setter function to update the value of the state variable.

```
import { useState } from "react";
function CustomComponent() {
        let [x, setX] = useState(10); // useState hook from react
        return(
                <>
                <button onClick={() => {
                        setX(x + 1);
                        console.log(x);
                }}>Click me</button>
                {x}
                </>

        );

}
```

Presentation - Container Pattern

This design pattern separates logic of a component from the view of the component. To achieve this we divide our component into two parts:

- Presentation Layer : Here the UI part of the component is present. This layer handles how to show the data on the ui.
- Container Layer : Here the logical part of the component is added, and this layer decides what data to show to the user.

For example:

```
// Presentation layer
import TextInput from "../TextInput/TextInput";

import Button from "../Button/Button" ;



function TextInputForm({ handleFormSubmit, handleTextInputChange, value,
inputType, setInputType }) {

return (

        <form className="flex items-end" onSubmit={handleFormSubmit}>
```

```
                <div className="mr-2 flex-1">
                        <TextInput
                                label="Enter a word or phrase"
                                type={inputType}
                                value={value}
                                onChange={handleTextInputChange}
                        />
                </div>

                <div>
                        <Button
                        styleType="warning"
                        text={inputType === 'password' ? 'Show' : 'Hide'}
                        onClickHandler={() => setInputType(inputType ===
'password' ? 'text' : 'password')}
                        />
                </div>

                <div>
                        <Button
                        text="OK"
                        buttonType="submit"
                        />
```

```
                </div>

</form>

);



}



export default TextInputForm;
```

```
// container layer
import { useState } from "react";

import TextInputForm from "./TextInputForm";



// Container component for TextInputForm

function TextInputFormContainer({ onSubmit}) {

        const [value, setValue] = useState('');

        const [inputType, setInputType] = useState('password');



        function handleFormSubmit(event) {

                event.preventDefault();

                console.log('Form submitted', value);

                onSubmit?.(value); // if onSubmit is defined, call it with
the value

        }



        function handleTextInputChange(event) {
```

```
                console.log('Text input changed');

                console.log(event.target.value);

                setValue(event.target.value); // whenever I Type in the
input field, it will update the value

        }



        return (

        // Calling the presentation layer

                <TextInputForm

                        handleFormSubmit={handleFormSubmit}

                        handleTextInputChange={handleTextInputChange}

                        value={value}

                        inputType={inputType}

                        setInputType={setInputType}

                        />

        );



}



export default TextInputFormContainer;
```

Here we can see that presentation layer is able to handle just the UI part and container layer contains the logical part of the component.

children prop

In react we have a special prop associated to every component, that we don't need to pass manually and react automatically passes for us, this is called as children prop.
So till now we were able to pass numbers, strings, object, functions etc as the prop of a component, but what if I want to pass a component as a prop to another component ?
In that case we use children prop.
To pass a component as a prop we just need to wrap the child component inside the parent.

```
function Card({ children }) {

        return (
                <div className="...">
                        {children}
                </div>
        );

}

function Avatar() {
        return (
                <img ... />
        );
}

function display() {

        return (
                <Card>
                        <Avatar />
                </Card>
        )
}
```

Here Card is the parent component and Avatar is the child component, and to render avatar inside the card the card is expecting the children prop which will be having the value of Avatar component and react will automatically pass it for us.