What is 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. Examples:

- useState
- useEffect
- useId
- useRef
- and more.....
  Generally react hooks have one or more of the following properties (which can control the....)
- 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 prefix.

What is 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 of there are special setter function which we need to use 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.

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.
  State Variables are very special, any change in the state variable will cause change in the UI.

```
function CustomComponent() {
```

```
    let x = 10;
     return(
        <>
        <button onClick={() =>{
            x = x + 1;
            console.log(x);
        }}>ClickMe</button>
        {x}
        </>
     );
}
export default CustomComponent;
```

when you run above code the number will increase in console not in UI so we are using useState hook. If we use below code the number will change in UI.

if you want to import the press `Cntrl + space` it will show us the recommondation click on first one it will import automatically.

```
import { useState } from "react";
function CustomComponent() {
     const [x,setX] = useState(10);
     return(
        <>
        <button onClick={() =>{

            setX(x + 1);

            console.log(x);

        }}>ClickMe</button>
        {x}
        </>
     );
}
export default CustomComponent;
```

HANGMAN Project :

Short circuiting :

Short-circuiting is a behavior exhibited by logical operators(&&,||) where the evaluation of the second operand is skipped if the outcome can be determined by evaluating the first operand alone
Example :

```
{label && <span className = "text-gray"-700>{label} </span}
```

if label is true then only label text will display on UI means span tag will execute . if label is false span tag also not execute and label also wont execute .

Onchange:

Just like the button have `onClick` there are events associated to input tag also `onChange` when you type of input we need some change that's why we are using this one.

- Every form has its own event called as `onSubmit` . what happen when we submit button for that we need prop.

```
function TextInputForm({onSubmit}) {
    return (
        <form className="flex" onSubmit={onSubmit}>
            <div className="mr-2 flex-1">
               <TextInput
               label="Enter a word or phrase"
                type = "password"  // it will not visible to other because of password type
               />

            </div>
            <div className="flex">
                <Button
                text="OK"
                type="submit"
                 />
               </div>
        </form>
    );

}
```

```
export default TextInputForm;
```

When we write like this `onSubmit` form is submitted automatically also it is refreshing the page. so here we have to use `event.preventDefault` method to avoid refreshing the page automatically.

```
function TextInputForm({}) {
    function handleFormSubmit(event) {
        event.preventDefault();
        console.log("form submitted");
    }
    return (
        <form className="flex" onSubmit={handleFormSubmit}>
            <div className="mr-2 flex-1">
           <TextInput
                label="Enter a word or phrase"
                type = "password"  // it will not visible to other because of
password type

              />
            </div>
            <div className="flex">
                <Button
                text="OK"
                type="submit"
                 />
              </div>
        </form>
    );
}
```

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.

- `Custom Hook` : If you want to use code again and again (code means not UI part and logic part) then create another file and write that code in that file. If you think it is reusable code then only. (ex: check crypto-currency project you will understand check useFetchCoinHistory file that is the code i think reusable that's why i separate it)
  When we need this ?
  take flipkart as an example if you want to create the premium version of that app we need same code (same code means without premium version code) but with extra functionalities at that time we need that code at that point of time we can reuse those files again and again

For example:

```
// this is your presentation component
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} // it will not visible to other because of
password type

                value = {value}

                onChange={handleTextInputChange}   // Calls the handler on
input change

               />

            </div>
```

```
            <div>
                <Button
                    styleType="warning"
                    text={inputType === 'password' ? 'Show' : 'Hide'}
                    onClickHandler={() => setInputType(inputType === 'password' ? 'text' : 'password')} //input type is password set to text if not set to password
                />
            </div>
            <div className="flex">
                <Button
                text="OK"
                buttonType="submit"
                 />
               </div>
        </form>
    );
}

export default TextInputForm;
```

```
// this is container component for TextInputForm
import React,{ useState } from 'react'
import TextInputForm from './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 it is present then i will call the onsubmit callback and  pass the value property there  means if onsubmit is defined, call it will the value

    }

    function handleTextInputChange(event) {
        console.log('text input changed');
        console.log(event.target.value); //Accesses the current value of the input field whenever a change occurs

        setValue(event.target.value);  //whenever i type something 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 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.