HOOKS in React

Hooks

- *Hooks* are a new addition in React 16.8.
- Hooks are functions that let you “hook into” React state and lifecycle features from function components.
- They let you use state without writing a class.
- Before introducing this feature in function component, data from the state had to be passed down as **props from class components to function components** or you had to convert your function component to a class component.
- Now, we can use React hooks, and to **use state** we can use the **useState hook**.
- With Hooks, you can **extract stateful logic** from a component so it can be tested independently and reused. **Hooks allow you to reuse stateful logic without changing your component hierarchy.**

Hooks

```
// State in function components is not defined
import React, { useState } from 'react';
const Hooksdemo=()=> {
    state = {count:0}
    return (
    <>
        <span>{state}</span>
    </>
    );
}
```

Error: 'state' is not defined no-undef

useState()

- **useState** is a *Hook,* call it inside a function component to add some local state to it.
- useState returns a pair: the ***current* state value** and a **function that lets you update** it.
- You can call this function from an **event handler** or somewhere else to modify/update the state value.
- The only argument to useState is the **initial state**.
- The initial state argument is only used **during the first render**.
- Function components can now **use hooks to use state** and you can implement **side-effects** on first render and after every update.

Example: Render Current state Value

```
// state in function component using useState
import React, { useState } from 'react';
const     Hooksdemo=()=> {
   const [state,setState] = useState(0);
   return (
   <>
      <span>{state}</span>
   </>
   );
}

export default Hooksdemo;
```

Example: Update state using Event Handler

```
// state in function component using useState
import React, { useState } from 'react';
const       Hooksdemo=()=> {
   const [state,setState] = useState(0);
   return (
   <>
      <button onClick={()=>(setState(state+1))}>+</button>
      <span>{state}</span>
      <button onClick={()=>(setState(state-1))}>-</button>
   </>
   );
}

export default Hooksdemo;
```

Example 2: Working with List using Hooks

Step1: Lets create a hook for a message and list.

```
import React, {useState } from 'react';                    //imrs
const HooksListDemo = () =>{
   const [message] = useState("List of Letters");
   const [list,setList] = useState(
      [
         {id:1,name:"A"},
         {id:2,name:"B"},
         {id:3,name:"C"},
         {id:4,name:"D"},
      ]
   );
   return (
    <>
                <h1> {message} </h1>
                <ul>  { list.length? list.map(lst => (<li key={lst.id}>{lst.id}-{lst.name}</li>)):null }            </ul>
    </>
   );
}
export default HooksListDemo;
```

Example 2: Working with List using Hooks

Step2: Add event handler to update the list using updateList

```
import React, {useState } from 'react'; //imrs
const HooksListDemo = () =>{
   … //retain previous stuff here

  const onClick = () => {
        updateList(lst => [...list,{id:`${lst.length+1}`,name: “E”}]);
    };

   return (
    <>
                …// retain previous stuff here

                 <button onClick={onClick}>Update</button>
    </>
    );
}
export default HooksListDemo;
```

Example 2: Working with List using Hooks

Step3: update the list by taking the new item from the user

```
import React, {useState } from 'react'; //imrs
const HooksListDemo = () =>{
   … //retain previous stuff here
  const onClick = () => {
        let newname = document.getElementById("newname").value;
        updateList(lst => [...list,{id:`${lst.length+1}`,name: newname}]);
   };
   return (
   <>
              …// retain previous stuff here
               <button onClick={onClick}>Update</button>
              <input type= "text" id="newname"/>
   </>
   );
}
export default HooksListDemo;
```

useEffect()

- The *Effect Hook* lets you **perform side effects** in function components.
- useEffect is executed on **first render and after every update**
- Effects are declared inside the component so they have **access to its props and state**.

```
import React, {useState, useEffect} from 'react';   //imrse
const HooksListDemo = () =>{
   … //retain previous stuff here

   useEffect(()=>{
      // document.title=`You added ${list.length} Items in the List`;
       window.alert(`You added ${list.length} Items in the List`);
    });
  return (
    <>
                …// retain previous stuff here
    </>
    );
}
export default HooksListDemo;
```

Custom Hooks

```
// useFriendStatus.js
import React, { useState, useEffect } from 'react';
function useFriendStatus(props) {
  const [isOnline, setIsOnline] = useState(true);  //null, true, false

  useEffect(() => {
    console.log("User subscribed to Chat!!!");
    return () => {
        console.log(`User unsubscribed from Chat!!!`);

    };
  });
  return isOnline;
}

export default useFriendStatus;
```

Custom Hooks

// FriendStatus.js

```
import React, { useState, useEffect } from 'react';
import useFriendStatus from './useFriendStatus';
function FriendStatus(props) {
    const isOnline = useFriendStatus(1);
    if (isOnline === null) {
      return 'Loading...';
    }
    return isOnline ? 'Online' : 'Offline';
  }
export default FriendStatus;
```

Custom Hooks

// FriendListItem.js

```
import React, { useState, useEffect } from 'react';
import useFriendStatus from './useFriendStatus';
function FriendListItem(props) {
   const isOnline = useFriendStatus();
   return (
     <li  style={{ color: isOnline ? 'green' : 'black' }}>
       {props.id}
     </li>
   );
 }
export default FriendListItem;
```