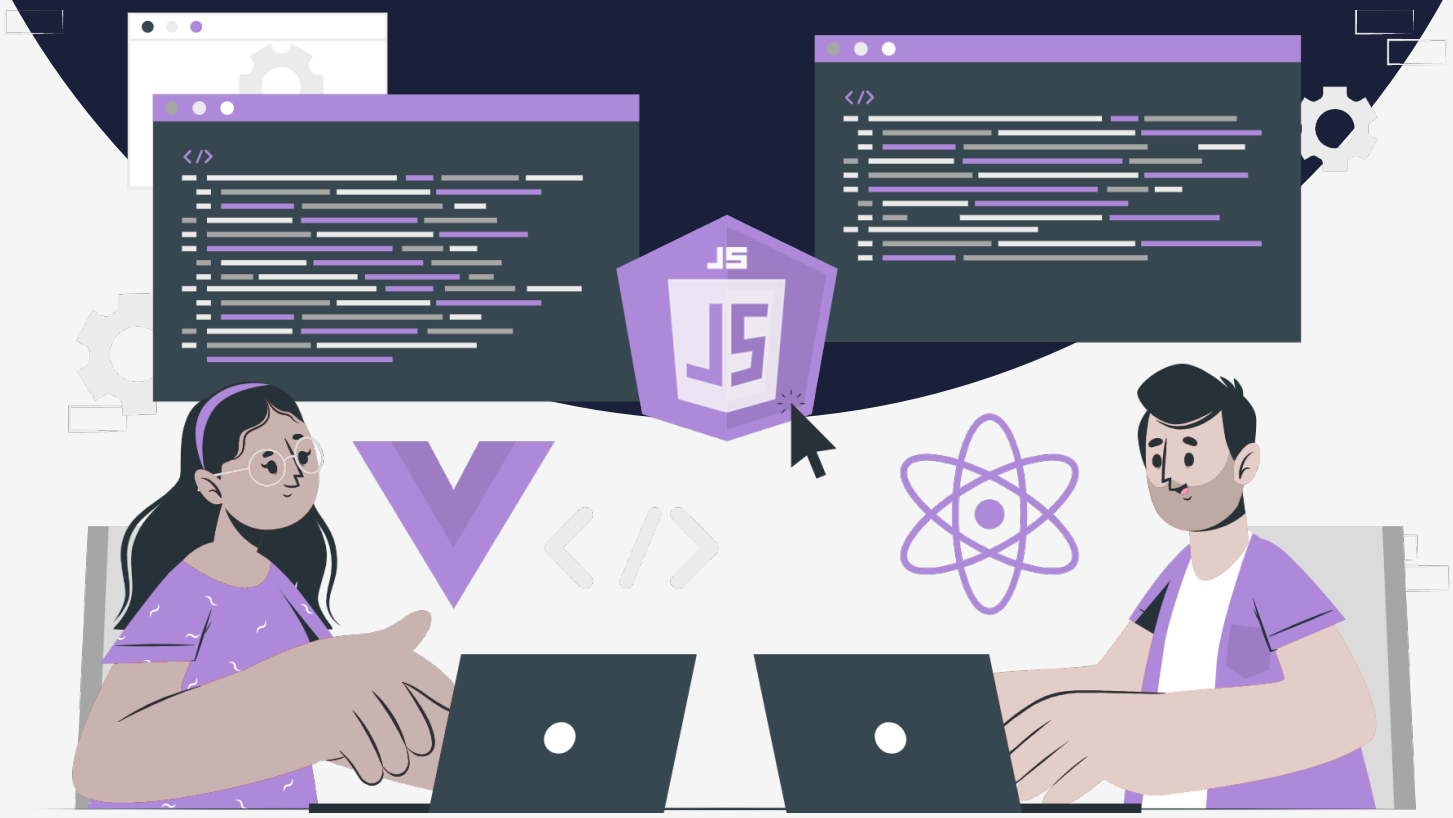
Lesson:

useState()



Topics Covered:

1. What are Hooks ?
2. Features Of Hooks.
3. Rules of Hooks
4. useState()

What are Hooks?
Hooks are in-built functions that allow us to use state and other react features inside functional components. Hooks are functions that start with the word "use" and allow you to "hook into" React features .

Hooks were introduced in React v16.8 as a way to use state and other features in functional components. Before hooks, the only way to use state in a React application was by using class components.Hooks are much more powerful than that.

Features Of Hooks :

- It provides a simpler way for developers to use state and other features in functional components.
- They allow for more flexibility and reusability in your code, making it easier to share stateful logic across different components in your application.
- They also allow for better code organisation and separation of concerns, making it easier to understand and maintain large applications.
- Classes are a barrier to learning React thoroughly; We would need to understand how `this` works in javascript which is differ from other languages. React Hooks solves this issue by allowing us to use the best react features without using class components.
- When components become larger and carry many operations, It becomes difficult to understand in the long run.Hooks solve

Rules of Hooks

- Only call hooks at the top level of your component , not inside loops, conditions, or nested functions.
- Only call hooks from React functional components or from other hooks.
- Do not call hooks from regular JavaScript functions.
- Do not call hooks inside of conditions or loops.

There are some basic Hooks that are commonly used in React

1. useState()
2. useEffect()
3. useContext()
4. useCallback()
5. useReducer()
6. useMemo()

1.useState():
The useState() hook allows us to update, handle and manipulate state inside functional components

The convention is to name state variables like below:

```
Unset
const [state, setState] = useState(intialState);
```

The useState hook in React allows you to add state to functional components. It takes an initial state as its first argument and returns an array with two elements: the current state and a function that can be used to update the state.

To use the useState() hook, you must first import it from React.
It allows you to keep track of a piece of data that can change over time and can be updated in response to user interactions or other events.

useState` hook can hold any type of data.

1.object type

```
JavaScript
import React, { useState } from "react";

const Course = () =>{
  const [coursePrice, setCoursePrice] = useState({
    WEBDEV:4000,
    DATASCIENCE:3000,
    JAVA:2500
  })

  const updateCoursePrice = (courseName) => {
      setCoursePrice({...coursePrice, courseName: 4000});
  }

  return (
    <h1>Datascience course price is
${coursePrice.DATASCIENCE}</h1>

   <input onChange={(e)=> setCoursePrice(e.target.value)}/>
  )
}
export default Course;
```

2. Array type

```javascript
import React, {useState} from "react";

const Course = () => {
  const [topics, setTopics] = useState(["WEB
DEV","DATASCIENCE", "JAVA"])

  const updateTopics = (topicName) => {
      setTopics([...topics, topicName]);
  }

  return (
    <h1>Welcome to PW Skill</h1>
    <ul>
       {topics.map((topic) => (<li> {topic} </li>))}
    </ul>

     <input onChange={(e)=> updateTopics(e.target.value)}/>
     </button>
  );
}
```

3.String type:

```javascript
import React, { useState } from "react";

const Course = () =>{
  const [course, setCourse] = useState("Web Dev")

  const updateCourse = (courseName) => {
      setCourse(courseName);
  }

  return (
    <h1>My course name is {course}</h1>  //My course name is
Web Dev


    <input onChange={(e)=> updateCourse(e.target.value)}/>
  )
}

export default Course;
```

4.boolean type

```javascript
import { useState } from 'react';

function Toggle() {
  const [isToggled, setIsToggled] = useState(false);

  return (
    <>
      <button onClick={() => setIsToggled(!isToggled)}>
        {isToggled ? 'Turn Off' : 'Turn On'}
      </button>
      {isToggled && <p>The toggle is on</p>}
    </>
  );
}

export default Toggle;
```

In this example, we use the `useState` hook to create a state variable called isToggled which initially is set to false, and a function setIsToggled that we can call to update the state.

We also use the state isToggled in the JSX to toggle the text of the button and displaying a text when the toggle is on.

We pass the opposite value of the current state with the setIsToggled function, this way we can toggle the state of the component, which is a boolean value in this case.

5. Number type

```javascript
import { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count +
1)}>Increment</button>
      <button onClick={() => setCount(count -
1)}>Decrement</button>
    </>
  );
}
```

In this example, we first import the `useState` hook from the react library.In the Counter component, we call `useState` and pass in an initial value of 0 for the count.

`useState` returns an array with two elements: the current state (in this case, the count) and a function to update the state (in this case, setCount).

Updating objects in the state

```javascript
import { useState } from 'react';

function MyComponent() {
  const [user, setUser] = useState({ name: 'John Doe', age: 30
});

  const handleChange = (event) => {
    setUser({ ...user, [event.target.name]: event.target.value
});
  }

  return (
    <>
      <form>
        <label>
          Name:
          <input type="text" name="name" value={user.name}
onChange={handleChange} />
        </label>
        <br />
        <label>
          Age:
          <input type="number" name="age" value={user.age}
onChange={handleChange} />
        </label>
      </form>
      <p>Name: {user.name}, Age: {user.age}</p>
    </>
  );
}
```

We use the useState hook to create a state variable called user which initially is set to an object, and a function setUser that we can call to update the state.

We also use the state user to set the value of the form inputs and to display the user's name and age.
We use the handleChange function to update the user state variable when the input values change, we use the spread operator to keep the previous properties of the object and change the one that we want.