Adding Interactivity

Session 03

What are Event Handlers?

Attached to Elements:

Associated with specific UI elements through event listeners.

Respond to User Actions:

Execute in response to user interactions like clicks or key presses.

What it State?

Application data, that changes over time.

(commonly after user interactions)

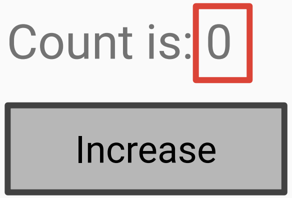
State Handing in React

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

     function handleIncrease(){
          setCount(count + 1)
     }

     return (
          <div>
               <p>Count is: {count}</p>
               <button onClick={handleIncrease}>
                    Increase
               </button>
          </div>
     )
}
```

1.

Component function gets executed

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

     function handleIncrease(){
          setCount(count + 1)
     }

     return (
          <div>
               <p>Count is: {count}</p>
               <button onClick={handleIncrease}>
                    Increase
               </button>
          </div>
     )
}
```

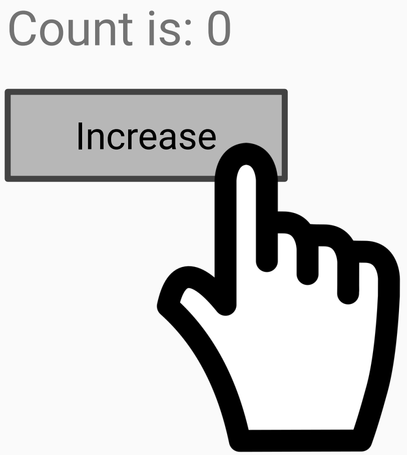

Count is: 0

2.

UI is rendered in the Browser

With initial state value

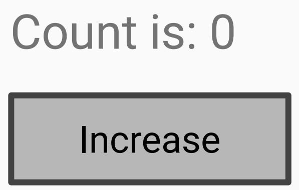
```
function Counter() {
     const [count, setCount] = useState(0);

     function handleIncrease(){
          setCount(count + 1)
     }

     return (
          <div>
               <p>Count is: {count}</p>
               <button onClick={handleIncrease}>
                    Increase
               </button>
          </div>
     )
}
```



3.

User interacts with the UI

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

     function handleIncrease(){
          setCount(count + 1)
     }

     return (
          <div>
               <p>Count is: {count}</p>
               <button onClick={handleIncrease}>
                    Increase
               </button>
          </div>
     )
}
```



4.

Event Handler is triggered

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

    function handleIncrease(){
        setCount(count + 1)
    }

    return (
        <div>
            <p>Count is: {count}</p>
            <button onClick={handleIncrease}>
                Increase
            </button>
        </div>
    )
}
```

5.

State updates triggers re-execution of component function

Count is: 0

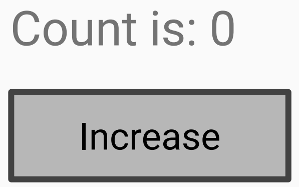
Increase

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

     function handleIncrease(){
          setCount(count + 1)
     }

     return (
          <div>
               <p>Count is: {count}</p>
               <button onClick={handleIncrease}>
                    Increase
               </button>
          </div>
     )
}
```

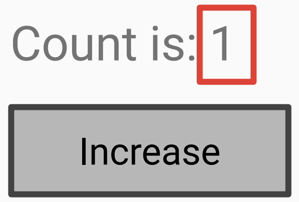
6.

Component function gets re-executed

State variable has a new value (count = 1)



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

     function handleIncrease(){
          setCount(count + 1)
     }

     return (
          <div>
               <p>Count is: {count}</p>
               <button onClick={handleIncrease}>
                    Increase
               </button>
          </div>
     )
}
```



7.

UI is re-rendered in the Browser

With new state value

React Hooks

What are Hooks?

A way to extend the functionality of a React component.

(handles a lot of hidden magic in the background)

Functions, that start with `use`

The Rules of Hooks

Only call Hooks at the top level

- 🔴 Do not call Hooks inside loops
- 🔴 Do not call Hooks after a return statement
- 🔴 Do not call Hooks in event handlers
- 🔴 Do not call Hooks inside functions

No conditionally calling

- 🔴 Do not call Hooks inside conditions (if - else)

How to call Hooks?

Always at the Beginning of a component function.

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

    //… more code
}
```