19. Input Forms, React Lifecycle and Hooks

Table of Content

1. User Input Forms

▼ **Input and TextArea Inputs**

```
const [inputValue, setInputValue] = useState('');

function handleChange(event) {
  setInputValue(event.target.value);
}

<form onSubmit={this.handleSubmit}>
  <label>
    Name:
    <input type="text" value={inputValue} onChange={handleChange} />
  </label>
  <input type="submit" value="Submit" />
</form>
```

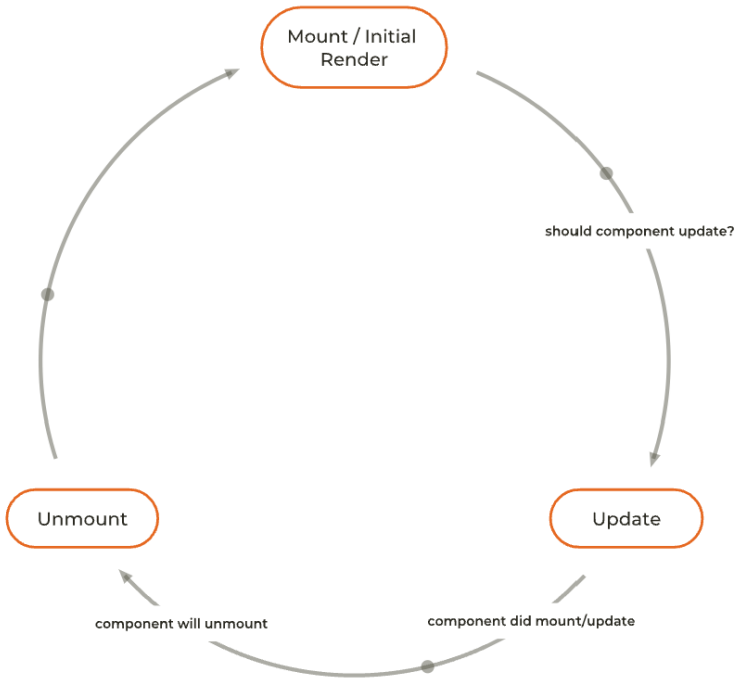
Since the value attribute is set on our form element, the displayed value will always be this.state.value, making the React state the source of truth. Since handleChange runs on every keystroke to update the React state, the displayed value will update as the user types.

▼ **Select and Option**

```
<select value={this.state.value} onChange={this.handleChange}>
  <option value="grapefruit">Grapefruit</option>
  <option value="lime">Lime</option>
  <option value="coconut">Coconut</option>
  <option value="mango">Mango</option>
</select>
```

2. React Lifecycle Events

We have already seen the working of useState hook. Now let's see another react hook called **useEffect** hook



▼ **useEffect Hook**

The Effect Hook lets you perform side effects in function components. • We can use mount and unmount behavior in React hooks to run code at specific lifecycle times in the component.

You can listen to various events by passing different values inside the the second argument - **The dependency Array.**

1. pass nothing:

```
useEffect(() => {
    console.log("useEffect has been called!", button);
  });
```

   Runs when the component mounts and on every re-renders.

2. pass an empty array

```
useEffect(() => {
    console.log("useEffect has been called!", button);
  }, []);
```

   Runs when the component mounts and not on re-renders.

3. passing elements with state changes

```
useEffect(() => {
    console.log("useEffect has been called!", button);
  }, [inputText]);
```

   Runs when the component mounts and the elements inside the array change.

4. Use effect with cleanup

```
useEffect(() => {
    console.log("Use Effect!")
    return () => {
      console.log("Unmount")
    }
  })
```

3. React Hooks

We have seen the useState and useEffect hooks in action, but what actually are hooks?

Hooks are functions that let you “hook into” React state and lifecycle features from function components. Hooks don’t work inside classes — they let you use React without classes.

React provides a few built-in Hooks like `useState`. You can also create your own Hooks to reuse stateful behavior between different components.

Hooks are JavaScript functions, but they impose two additional rules:

- Only call Hooks **at the top level**. Don’t call Hooks inside loops, conditions, or nested functions.
- Only call Hooks **from React function components**. Don’t call Hooks from regular JavaScript functions.

▼ useRef hook

`useRef` is a React Hook that lets you reference a value that’s not needed for rendering.

```
import { useRef } from 'react';

function Stopwatch() {
  const intervalRef = useRef(0);
  // ...
```

`useRef` returns a ref object with a single `current` property initially set to the initial value you provided.

Changing a ref does not trigger a re-render. This means refs are perfect for storing information that doesn’t affect the visual output of your component.

By using a ref, you ensure that:

- You can **store information** between re-renders (unlike regular variables, which reset on every render).

- Changing it **does not trigger a re-render** (unlike state variables, which trigger a re-render).
- The **information is local** to each copy of your component (unlike the variables outside, which are shared).

> 💡 You mostly use the useRef hook to store the reference to the dom elements to read certain properties out of the dom element.

Assignments

1. In the News Application, add a List of Topics for the user to select from and then show the result based on the selected item.
2. Build a Bitcoin exchange converter using this API: https://api.coingecko.com/api/v3/exchange_rates
3. Build a Money exchange using this API: https://open.er-api.com/v6/latest/USD