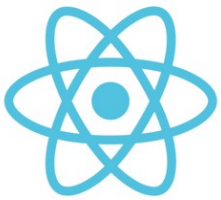

useEffect() React Hooks

- In React, useEffect is one of the most essential hooks **used to manage side effects** in functional components.
- Side effects are operations that occur outside of the component's **render cycle**, such as data fetching, subscriptions, or manually interacting with the DOM.
- The useEffect hook allows you to **perform these side effects after the component has rendered.**
- **Syntax :**
    - useEffect accepts **two arguments**. The second argument is **optional**.

```
useEffect(<function>, <dependency>)
```

    - **Common use cases for useEffect.**
        - Data Fetching:
        - DOM Manipulation:
        - Subscriptions
        - Timers and Intervals:
        - Cleanup Tasks

```jsx
import React, { useEffect } from 'react';

function MyComponent() {
  useEffect(() => {
    /* Side effect code goes here
       This function will run after every
       render of the component */

    // Clean-up function (optional)
    return () => {
      /*  This function will run before the
          component unmounts or before the next
          re-render, depending on the dependencies.
      */
    };
  }, [dependency1, dependency2, ...]);

  // Component JSX
  return <div>...</div>;
}
```

- **Effect Function :**

  - The **first argument of useEffect** is the effect function. This function is called after the **component has rendered**, and it's where you **place the code** for your side effect.
  - It can be an **asynchronous** function, and you can perform any asynchronous **operations** like data fetching inside it.

- **Dependency Array :**

  - The **second argument** of useEffect is an array of dependencies. This array **tells React when to re-run** the effect. and its **optional**.
  - You can include to control when the effect should be executed.

- **Base on Dependency:**
  - If the dependency **array is empty ([])**, the effect will only run once, after the initial render.
  - If the dependency **array is not provided**, the effect will run after every render of the component.
  - If the dependency **array contains variables**, the effect will only be re-executed when any of those variables change.

- **Clean-up Function :**

  - The clean-up function is optional and is used to perform any **necessary clean-up or resource disposal** when the component unmounts or before the next effect runs (depending on the dependencies specified in the dependency array).
  - **Example :** for setinterval( ) function using clearinterval( ) to clear