Concepts in Focus

1. Using Scheduler methods

1.1 Displaying the current time

We use

`Date` Built-in Constructor function to display the current time

JSX

```jsx
import { useState } from "react"
...
const Clock = () => {
  const [date, setDate] = useState(new Date())

  return (
    <ClockContainer>
```

```
        <Heading>Clock</Heading>
        <ClockImage src="IMAGE_URL" alt="clock" />
        <Time> {date.toLocaleTimeString()} </Time>
      </ClockContainer>
    )
}

export default Clock
```

Collapse

1.2 Updating the time for every second

In general, it's a best practice to update the time after the component render

We schedule an interval inside the

`useEffect` for updating time after the component render

File: src/components/Clock/index.js

JSX

```jsx
import { useState, useEffect } from "react"
...
const Clock = () => {
  const [date, setDate] = useState(new Date())

  useEffect(() => {
    const timerId = setInterval(() => {
      setDate(new Date())
    }, 1000)
  })

  return (<ClockContainer>...</ClockContainer>)
}

export default Clock
```

Collapse

1.3 Avoiding unnecessary scheduling of intervals

In the code given in ***section 1.2***, Effect executes multiple times, and many intervals are scheduled, but one Interval is enough to update the time for every second

The effect should execute and an interval should be scheduled only once, so pass **empty dependency array** as a second argument to the useEffect

File: src/components/Clock/index.js

JSX

```jsx
...
const Clock = () => {
  const [date, setDate] = useState(new Date())

  useEffect(() => {
    const timerId = setInterval(() => {
      setDate(new Date())
    }, 1000)
    console.log("Effect executed and Interval scheduled")
  }, [])

  return (<ClockContainer>...</ClockContainer>)
}

export default Clock
```

Collapse

2. Effect Hook

2.1 Cleanup function

The effect which we pass to the

`useEffect` can return a function called Cleanup function

Syntax:

```jsx
useEffect(() => {
  // function body
  return () => {
    // cleanup
  }
}
```

In the Cleanup function we can perform actions like Clearing timers, Cancelling Network calls, etc

The Cleanup function will execute

- When the component unmounts
- Before the execution of next effect *(will be covered in next sessions)*

2.2 Handling Unmounting case

When a component is unmounted, the scheduled interval should be cleared to avoid unnecessary problems

To clear the scheduled interval when the component unmounts, call

`clearInterval()` in the Cleanup function

File: src/components/Clock/index.js

```jsx
...
const Clock = () => {
  const [date, setDate] = useState(new Date())

  useEffect(() => {
    const timerId = setInterval(() => {
      setDate(new Date())
    }, 1000)
    ...
```

```
    return () => {
      clearInterval(timerId)
      console.log("Cleanup Function executed")
    }
  }, [])
  return <ClockContainer>...</ClockContainer>
}

export default Clock
```

Collapse

3. Clock Application Code

Use the below command to get the Final Code for Clock Application

SHELL

```
ccbp start RHSIVMF74E
```

MARKED AS COMPLETE

Submit Feedback