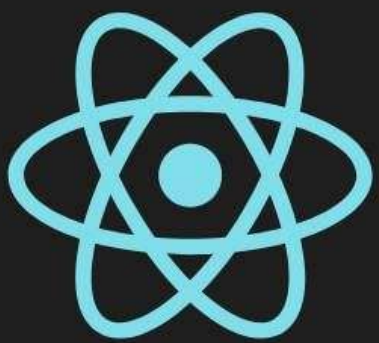

4 useRef use cases

@richwebdeveloper

1. Accessing DOM Elements

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

function FocusInput() {
  const inputRef = useRef(null);

  useEffect(() => {
    inputRef.current.focus();
  }, []);

  return <input type="text" ref={inputRef} />;
}
```

- useRef is commonly used to access and manipulate DOM elements directly.
- Example: Focusing an input field when a component mounts or interacting with third-party libraries that require **DOM manipulation.**

2. Storing Mutable Values

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

function Timer() {
  const [count, setCount] = useState(0);
  const intervalRef = useRef(null);

  useEffect(() => {
    intervalRef.current = setInterval(() => {
      setCount((prevCount) => prevCount + 1);
    }, 1000);

    return () => clearInterval(intervalRef.current);
  }, []);

  return <div>Count: {count}</div>;
}
```

- useRef can store and update mutable values without triggering a re-render.
- Example: Keeping track of **previous state values**, implementing instances of a class, or managing timers and intervals.

3. Referencing Parent Components

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

function Parent() {
  const childRef = useRef(null);

  const handleChildMethod = () => {
    childRef.current.childMethod();
  };

  return (
    <div>
      <Child ref={childRef} />
      <button onClick={handleChildMethod}>Call Child Method</button>
    </div>
  );
}

const Child = React.forwardRef((props, ref) => {
  const childMethod = () => {
    console.log('Child method called');
  };

  React.useImperativeHandle(ref, () => ({
    childMethod,
  }));

  return <div>Child Component</div>;
});
```

- useRef can be used to **create a reference to a parent component from a child component.**
- Example: Accessing methods or properties of a parent component from a child component.

4. Implementing Scroll Behavior

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

function InfiniteScroll() {
  const containerRef = useRef(null);

  useEffect(() => {
    const handleScroll = () => {
      const container = containerRef.current;
      if (
        container.scrollTop + container.offsetHeight >=
        container.scrollHeight
      ) {
        // Fetch more data or perform infinite scroll logic
      }
    };

    const container = containerRef.current;
    container.addEventListener('scroll', handleScroll);

    return () => {
      container.removeEventListener('scroll', handleScroll);
    };
  }, []);

  return <div ref={containerRef} style={{ height: '500px', overflowY: 'scroll' }}>
    {/* Content to be scrolled */}
  </div>;
}
```

- useRef can be used to track scroll positions or implement scroll-related functionality.
- Example: Implementing infinite scrolling, scroll-based animations, or **persisting scroll positions across component re-renders.**

Did you find it Useful?

Leave a comment!

Alamin CodePapa

@CodePapa360

FOLLOW FOR MORE

Like Comment Repost