Understanding `Object.is` in React

Last updated by | Subramanya Dixit | May 6, 2025 at 9:37 AM GMT+5:30

📘 Why Learn About `Object.is`?

React relies on **shallow comparison** to determine whether it should trigger a re-render. One of the methods it uses internally for these comparisons is `Object.is()`. Knowing how it works helps you understand **why state updates may or may not cause re-renders**.

🧠 What is `Object.is`?

`Object.is()` is a built-in JavaScript method that determines whether two values are **the same value**. It is **similar to the `===` operator**, but with a few differences in edge cases.

```
### ✅ Syntax

Object.is(value1, value2);
```

✅ Key Differences from `===`:

Comparison	`===`	`Object.is()`
`Object.is(NaN, NaN)`	`false`	✅ `true`
`Object.is(+0, -0)`	✅ `true`	`false`

🔄 How React Uses `Object.is`

React internally uses `Object.is()` for **state comparisons** in functions like `useState` and `React.memo()`.

✅ In `useState`:

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

// This will NOT re-render, because 0 === 0
setCount(0);
```

Under the hood, React uses:

```
if (!Object.is(prevState, newState)) {
  // trigger re-render
}
```

✅ In `React.memo()`:

React compares **props** using shallow comparison, typically via `Object.is` to avoid unnecessary re-renders.

```
const MyComponent = React.memo(function MyComponent({ name }) {
  return <p>Hello, {name}</p>;
});
```

If `name` hasn't changed (`Object.is(prevProps.name, nextProps.name)` is true), React skips re-rendering.

🔍 Visual Example

```
console.log(Object.is(100, 100)); // true
console.log(Object.is('a', 'a')); // true
console.log(Object.is({}, {}));   // false (different references)
console.log(Object.is(+0, -0));   // false
console.log(Object.is(NaN, NaN)); // true
```

💡 Why It Matters in React

- `Object.is` is **faster and more precise** than deep equality checks.
- It enables **performance optimizations** in hooks like `useState` , `useMemo` , `React.memo` , and context updates.
- Helps you avoid unnecessary renders by ensuring changes are **actual changes**.

❓ Quiz Questions

1. What does `Object.is(NaN, NaN)` return?

a) `false`
✅ **b) `true`**
c) `undefined`
d) `NaN`

2. Which value pair will `Object.is()` consider different, but `===` will consider equal?

a) `null` and `null`
✅ **b) `+0` and `-0`**
c) `[]` and `[]`
d) `'a'` and `'a'`

3. Why does React use `Object.is()` ?

a) For deep equality
b) To stringify objects
✅ **c) To perform shallow comparison efficiently**
d) For sorting elements

🛠️ Practice Tasks

1. Test `Object.is()` with different values (e.g. objects, primitives).
2. Create a component using `React.memo` and observe re-render behavior.
3. Simulate `useState` behavior by building a mini version that uses `Object.is()` for comparison.