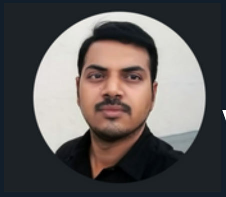
Interviewer Quest :

Why reactjs is so fast?

Generalised ans vs in depth technical answer



Vishwanath.Chiniwar

Generic ans :

React uses Virtual DOM compares the components previous states and updates only the items in the Real DOM that were changed, instead of updating all of the components again, as conventional web applications do.

The above answer can be suitable for freshers but for experienced guys

Experienced Folks ans ---->(next slide)

Vishwanath.Chiniwar

Specific in detail ans :

Concept of Immutability

When React compares the previous state or props with the new state or props, it needs to determine if there has been a change that requires the component to be updated.

If React detects that the reference to the state or props object has changed, it will assume that the object has been changed and that the component needs to be re-rendered. This is because when a new object is created, it is guaranteed to have a different reference than the previous object.

Vishwanath.Chiniwar

Contd.....

On the other hand, if the reference to the state or props object is the same as the previous reference, React assumes that the object has not been changed, and that the component doesn't need to be re-rendered. This is because if the reference is the same, it means that the object has not been replaced with a new one.

Ques: what if just a property of an object is changed without changing reference then will reactjs update the dom or not?

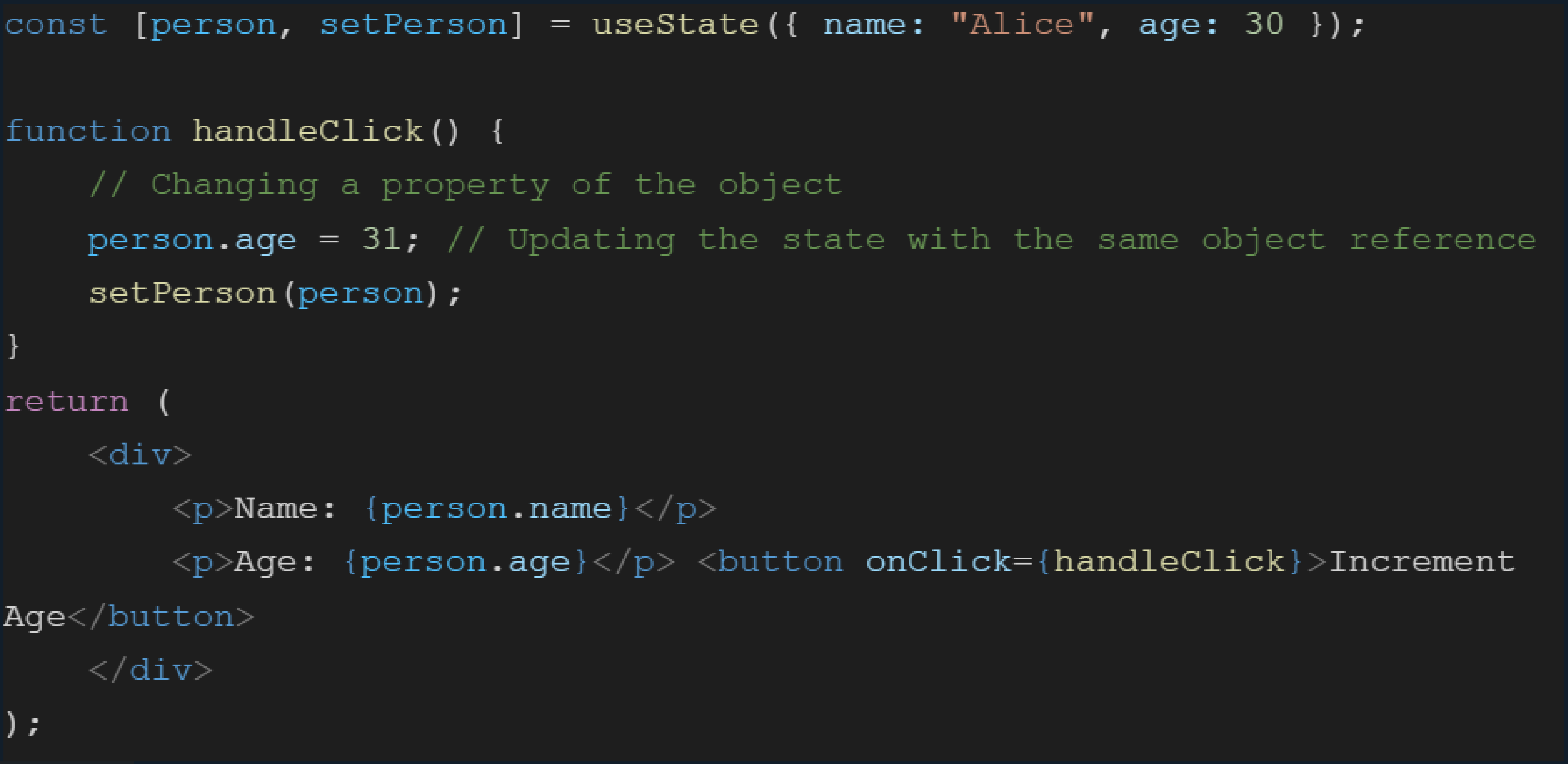
When an object is passed as a prop to a React component, React performs a shallow comparison of the object's properties to determine if the component should re-render.

Vishwanath.Chiniwar

Cont....

A shallow comparison means that React only checks if the references to the properties are the same, not if the properties themselves have changed. This means that if an object's property changes without the reference to the object itself changing, React may not detect the change and may not trigger a re-render of the component.

For ex:



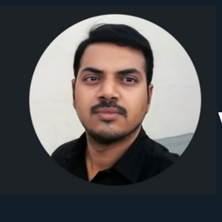
```
const [person, setPerson] = useState({ name: "Alice", age: 30 });

function handleClick() {
    // Changing a property of the object
    person.age = 31; // Updating the state with the same object reference
    setPerson(person);
}
return (
    <div>
        <p>Name: {person.name}</p>
        <p>Age: {person.age}</p> <button onClick={handleClick}>Increment
Age</button>
    </div>
);
```

Vishwanath.Chiniwar

Cont....

when the button is clicked, the age property of the person object is changed from 30 to 31. But the reference to the person object remains the same, React's shallow comparison may not detect the change and may not trigger a re-render of the component.

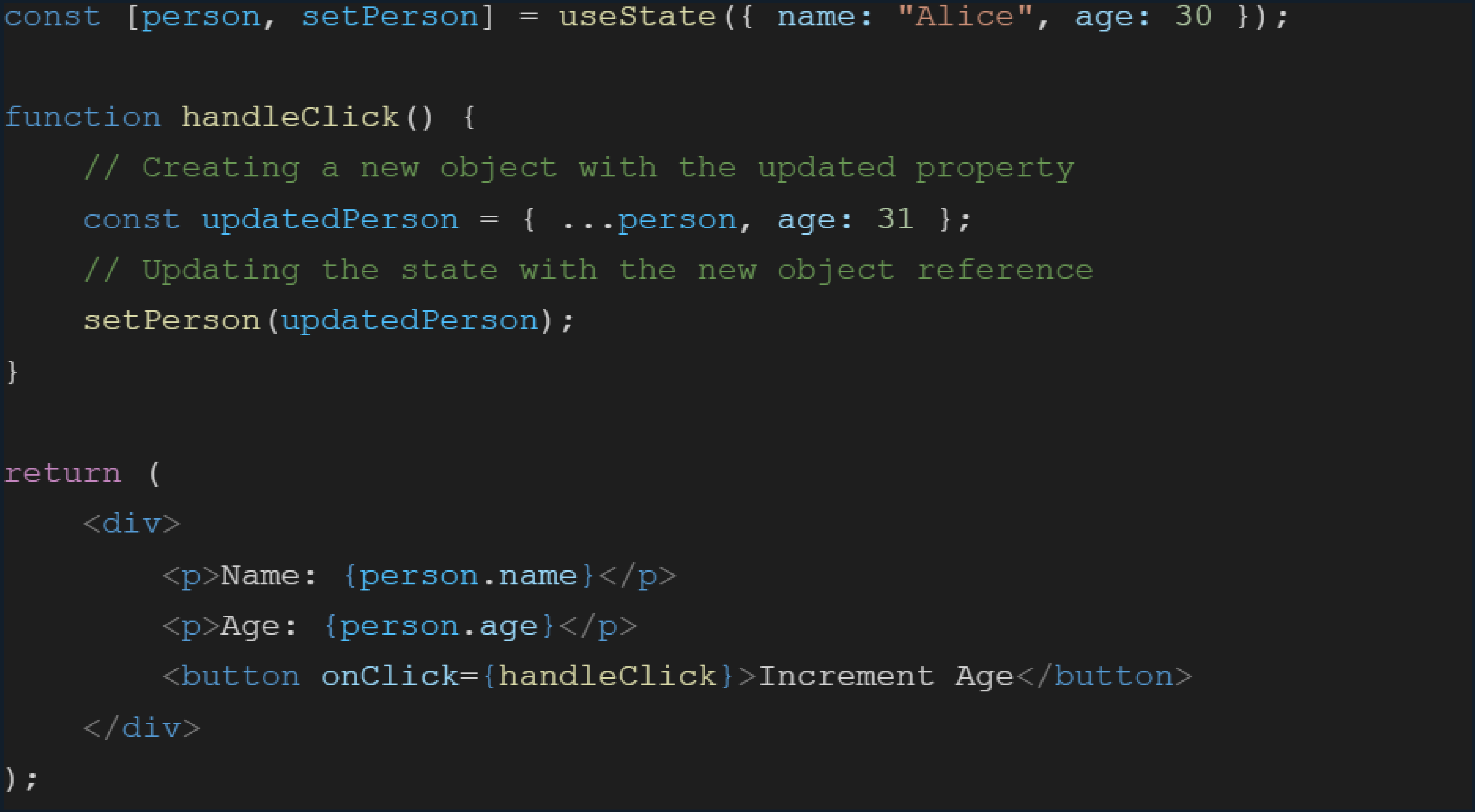
To work around this issue, we should always use immutable data structures(use spread operator), such as creating a new object with the updated property and passing it to the setState function instead of updating the existing object directly.



Vishwanath.Chiniwar

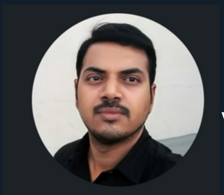
Cont....

For ex:



```
const [person, setPerson] = useState({ name: "Alice", age: 30 });

function handleClick() {
    // Creating a new object with the updated property
    const updatedPerson = { ...person, age: 31 };
    // Updating the state with the new object reference
    setPerson(updatedPerson);
}

return (
    <div>
        <p>Name: {person.name}</p>
        <p>Age: {person.age}</p>
        <button onClick={handleClick}>Increment Age</button>
    </div>
);
```

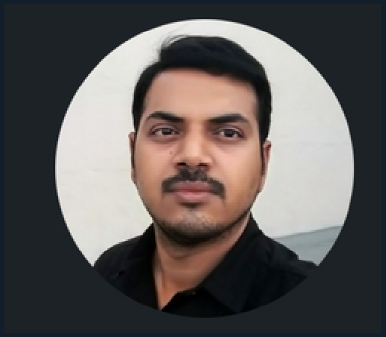
By using immutable data structures and always creating new objects with the updated properties, React can perform a reliable shallow comparison and detect changes even when only a single property of an object has changed.



Vishwanath.Chiniwar

AI generated image

Thanks for checking these slides. Consider following me on LinkedIn with below link



Vishwanath Chiniwar