React.memo() explained

in simple terms in
very detailed way

srikanth tekmudi
@srikanth tekmudi

React.memo

React.memo is a higher order component

If your component renders the same result given the same props, you can wrap it in a call to React.memo for a performance boost in some cases by memoizing the result. This means that React will skip rendering the component, and reuse the last rendered result.

It means if a componenet wrapped with React.memo it only re-renders only if propos from it parent changed and it will helps in performance boost.

Let’s understand with example .

App.js

```
1  import {useState} from "react";
2  export default function App() {
3    const [count, setCount] = useState(0);
4
5    console.log("Parent Render");
6
7    return (
8      <div className="App">
9        <h1>Hi this is parent count:{count}</h1>
10       <button onClick={() => setCount(count + 1)}>Click</button>
11       <Child isDivisibleby5={count % 5 === 0 ? "True" : false} />
12     </div>
13   );
14 }
```

srikanth tekmudi
@srikanth tekmudi

Child.js

```
import React from "react";
const Child = ({isDivisibleby5}) => {
  console.log("child render");
  return (
    <div>
      <h1>Hi Child </h1>
      <p>Yes isDivisible By 5 {isDivisibleby5}</p>
    </div>
  );
};

export default Child;
```

```
Parent Render
child render
Parent Render
child render
Parent Render
child render
```

So here Parent and child re-renders every time button clicked.
But there is no need for child to re-render as value not changing everytime.
For child value changed only if count divisible by 5.

In this situation we can optimise code ..such that
child need to re-render only it's prop changed ...we can do this by Wrapping this component in React.memo() hook as shown below

```
import React from "react";

const Child = ({ isDivisibleby5 }) => {
  console.log("child render");
  return (
    <div className="box">
      <h1>Hi Child </h1>
      <p>isDivisible By 5 {isDivisibleby5}</p>
    </div>
  );
};

export default React.memo(Child);
```

srikanth tekmudi
@srikanth tekmudi

React.Memo()

By default React.memo() does a shallow comparison of props and objects of props.

To customize the props comparison you can use the second argument to indicate an equality check function:

```
1 React.memo(Component, [areEqual(prevProps, nextProps)]);
```

areEqual(prevProps, nextProps) function must return true if prevProps and nextProps are equal.

After wrapping child.js with React.memo the output looks like this

```
Console was cleared
4 Parent Render
child render
```

srikanth tekmudi
@srikanth tekmudi

MockInterview

If you are a student or experienced and trying to switch to new company but loosing opportunites because of fear or anxiety or lack of confıdence.

Yes I am here to help you.I am providing MOCK INTERVIEW service with very minimal cost .

What you will get?

It will help you in testing your knowledge,simulate real interview and helps reduce your anxiety and stress in real interview.

The interview will be 1:00hr-1:35 minutes covering all JS concepts and common HTML,CSS,React questions.

Feedback where you need to improve.

If you are intrested:
Connect with me in linkedin or sechudle intervew at topmate.io/srikanthsri in the link given in comment.

Srikanth tekmudi

Frontend Dev

Endorse me if you like my content

Like this post if this helpful