# How to prevent re-renders on React functional components with React.memo()

How do you not re-render a React functional component?

If you're using a React class component you can use the shouldComponentUpdate method (https://linguinecode.com/post/how-to-avoid-multiple-re-renders-in-react-shouldcomponentupdate) or a React.PureComponent (https://linguinecode.com/post/react-component-vs-purecomponent) class extension.

But, is there an option to do that with functional components?

The answer is yes!

## Ruben Leija

I launched this blog in 2019 and now I write to 65,000 monthly readers about JavaScript. Say hi to me at Twitter, @rleija_ (https://twitter.com/rleija_).

## Do you want more **React articles**?

Hey, here at Linguine Code, we want to teach you everything we know about React. Our only question is, are you in?

```
React

React.memo(YourComponent);
```

Before I jump into the `React.memo()` code example, let's cover a couple basic things about React functional components.

## What is a React functional component and how to create them

React functional components are React components built in a function rather than class.

```
React

// Style 1
function NormalFunctionComponent() {
  return <h1>Greetings earlthing!</h1>
}

// Style 2
const FunctionComponentWithCurlies = () => {
  return <h1>Greetings earlthing!</h1>
}

// Style 3
const FunctionComponentWithParanthesis = () => (
  <h1>Greetings earlthing!</h1>
);
```

As you can see above, there are 3 styles of creating a React functional component.

The one that really stands out the most is style #3. In that style, you can't add any logic into the function.

React functional components are great, because it's really is to read, and really easy to maintain.

But it does lose some of features that the React.Component class provides. Luckily some of those features, if not, near all, have been provided through there recent addition called React hooks (https://linguinecode.com/post/getting-started-with-react-hooks).

## React functional components are always the render lifecycle method

React functional components do not have lifecycle methods.

Let’s study the code below.

```
React

const Greeting = props => {
  console.log('Greeting Comp render');
  return <h1>Hi {props.name}!</h1>
};

function App() {
  const [counter, setCounter] = React.useState(0);

  // Update state variable `counter`
  // every 2 seconds.
  React.useEffect(() => {
    setInterval(() => {
      setCounter(counter + 1);
    }, 2000);
  }, []);

  console.log('App render')
  return <Greeting name="Ruben" />
}
```

I will be using the new React hook api in these examples. If you’re not familiar I recommend you reading these articles:

- Getting started with React hooks (https://linguinecode.com/post/getting-started-with-react-hooks)
- Understanding React useEffect hook (https://linguinecode.com/post/getting-started-with-react-useeffect)

We have 2 React components. One is a greeting component that prints out “Hello [person_name]”, and it also logs in the console when it renders.

The second React component is the App component, and it uses the Greeting component.

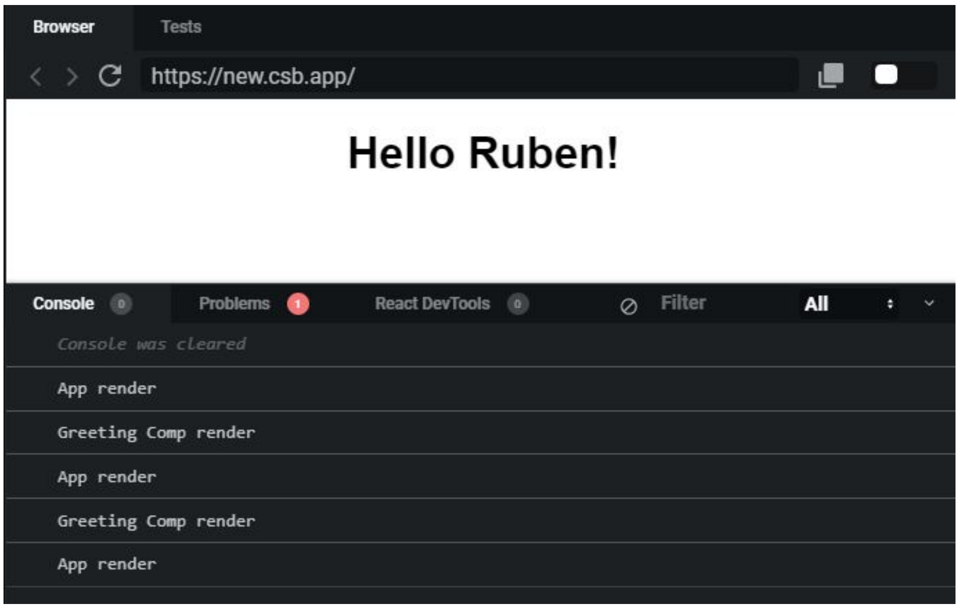
But the App component is doing a couple of other things. One it's creating a new state property called, counter.

And when the App component mounts, it updates the the counter value every 2 seconds.

So what is gonna happen?

Well, anytime a React component prop or state changes, it's going to get re-rendered. And that React component that has changed, will force any other children React components to re-render as well.

So in the example above, the Greeting component will always get re-rendered.



```
Console was cleared
App render
Greeting Comp render
App render
Greeting Comp render
App render
```

I want to avoid re-rendering the Greeting component, but how?

Since React functional components don't have access to `shouldComponentUpdate()`, and it's not a React pure component class, I'll have to use a new strategy to achieve a pure like function behavior.

We're gonna memorize it! I mean memoize it!

## Memoization a React functional component with React.memo()

To optimize, and prevent multiple React renders, I'm going to use another React tool called `React.memo()`.

### But what's memoization?

In programming, memoization is an optimization technique. It's primarily used to speed up computing by story the result of a function and returning the cached result, when the **same inputs** occur again.

Keyword, **same inputs**.

If we take a look at the `Greeting` component, I think it fits the criteria of the definition.

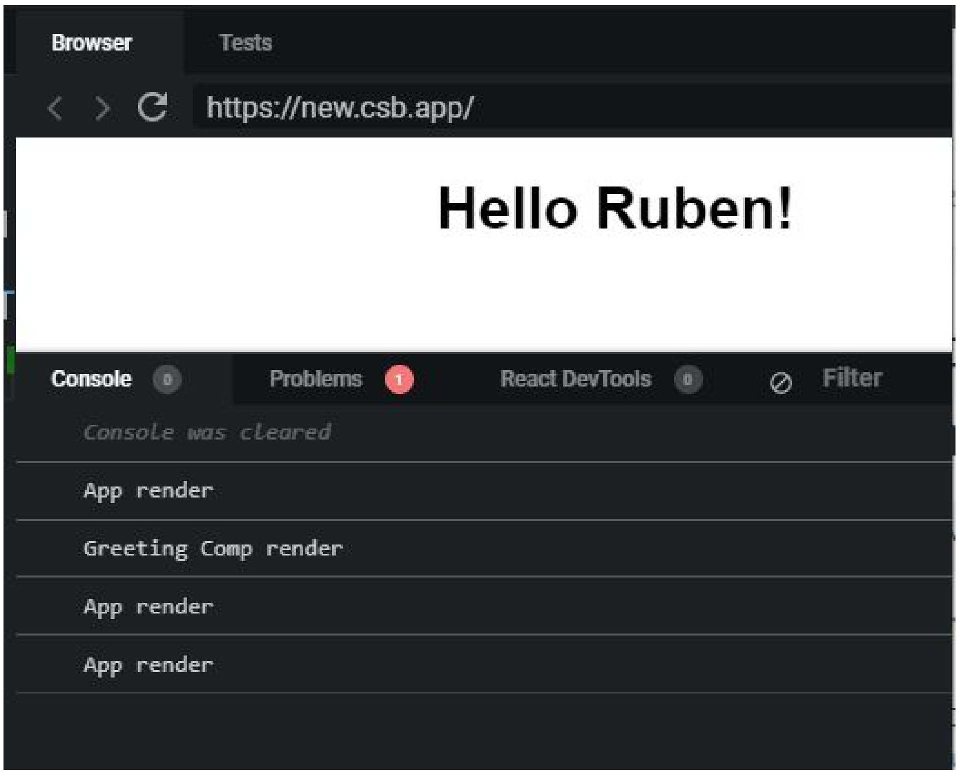
The Greeting component is a function, and it's receiving the same inputs every single time (name equals "Ruben").

So let's put it to use.

```
React

const Greeting = React.memo(props => {
  console.log("Greeting Comp render");
  return <h1>Hi {props.name}!</h1>;
});
```

All I need to do in the Greeting component is wrap it another function called `React.memo()`.



Hello Ruben!

```
Console was cleared
App render
Greeting Comp render
App render
App render
```

In the console, you can see that the App component is still rendering, but the Greeting component only rendered once.

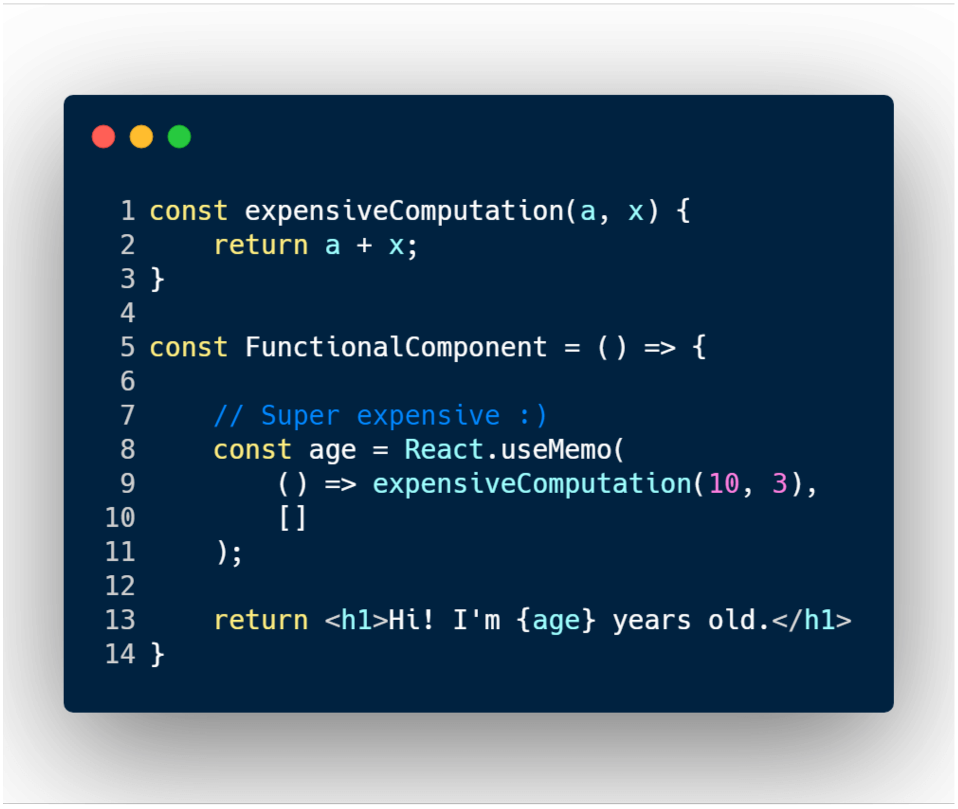
Thus avoiding multiple renders, hoorraayyy!

## Don't confuse React.memo() vs React.useMemo()

You might run into `React.useMemo()` and think that it's the same as `React.memo()`. In a way, yes but no.

`React.memo()` is a HOC (higher order component) meant to optimize a React functional component.

`React.useMemo()` is more generic, and optimizes a value.



```
const expensiveComputation(a, x) {
    return a + x;
}

const FunctionalComponent = () => {

    // Super expensive :)
    const age = React.useMemo(
        () => expensiveComputation(10, 3),
        []
    );

    return <h1>Hi! I'm {age} years old.</h1>
}
```

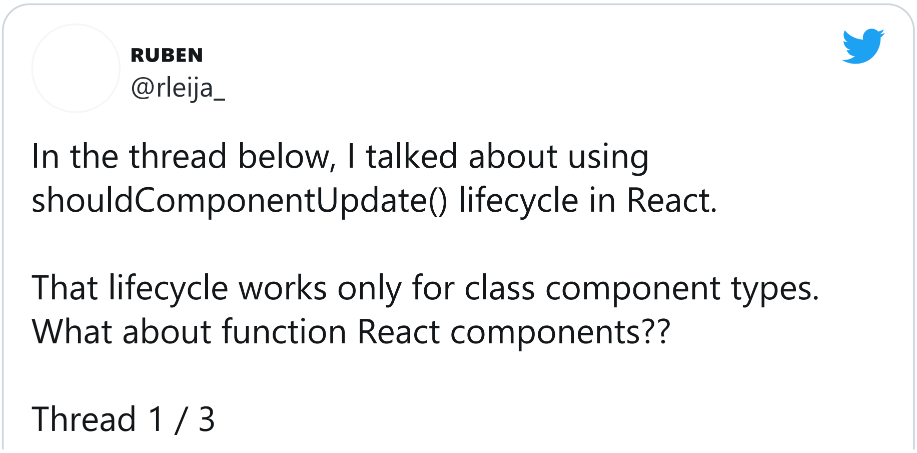
## Conclusion

`React.memo()` is a great React tool that lets you optimize performance, and replicate a `shouldComponentUpdate` and `React.PureComponent` (https://linguinecode.com/post/react-component-vs-purecomponent) for functional components.

Don’t confuse it with `React.useMemo()` (https://linguinecode.com/post/react-usememo-usecallback-hook)!

Happy coding!

Oh wow, you’ve made it this far! If you enjoyed this article perhaps like or retweet the thread on Twitter:



**RUBEN**
@rleija_

In the thread below, I talked about using shouldComponentUpdate() lifecycle in React.

That lifecycle works only for class component types. What about function React components??

Thread 1 / 3

**RUBEN** @rleija_

Imagine if you have a lot of sub components. An update to state or props would render each individual sub component.

Depending on that sub component, that can be pretty bad news.

The shouldComponentUpdate() lifecycle is great to avoid multiple renders.

Thread 1/4

4:50 PM · May 28, 2020

♡ 1 See RUBEN's other Tweets

I like to tweet about React and post helpful code snippets. Follow me there (https://twitter.com/rleija_) if you would like some too!