React.js

Regards: Hassan Bilal

TABLE OF CONTENTS

01

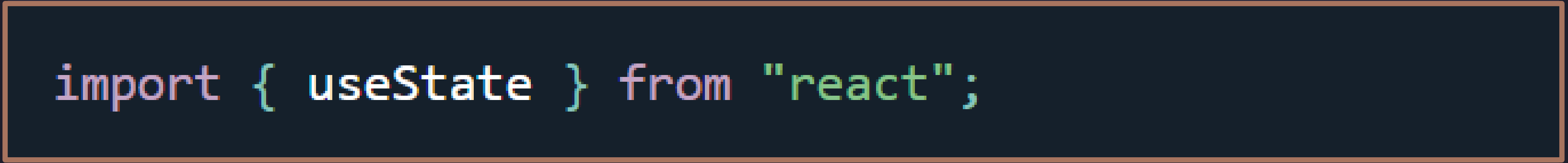
useState hook

useState hook

- State generally means data or properties.
- The React useState Hook allows us to track state in a component.

useState hook – Useage

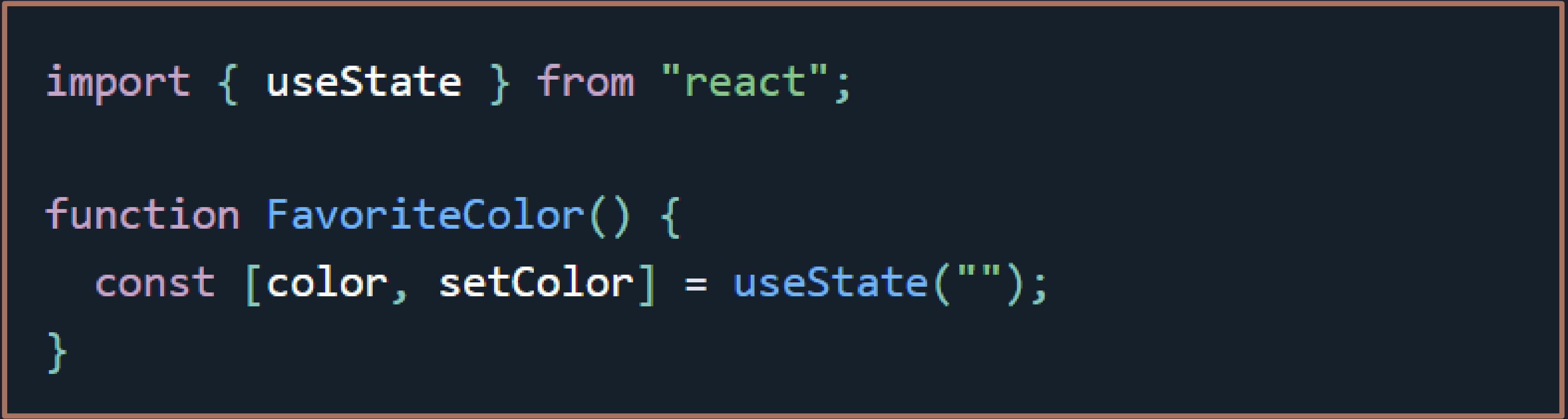
- Import it into our component.
- Notice that we are destructuring useState from react as it is a named export.



```
import { useState } from "react";
```

useState hook – Initialize useState

- We initialize our state by calling useState in our function component.
- useState accepts an initial state and returns two values:
  - The current state.
  - A function that updates the state.

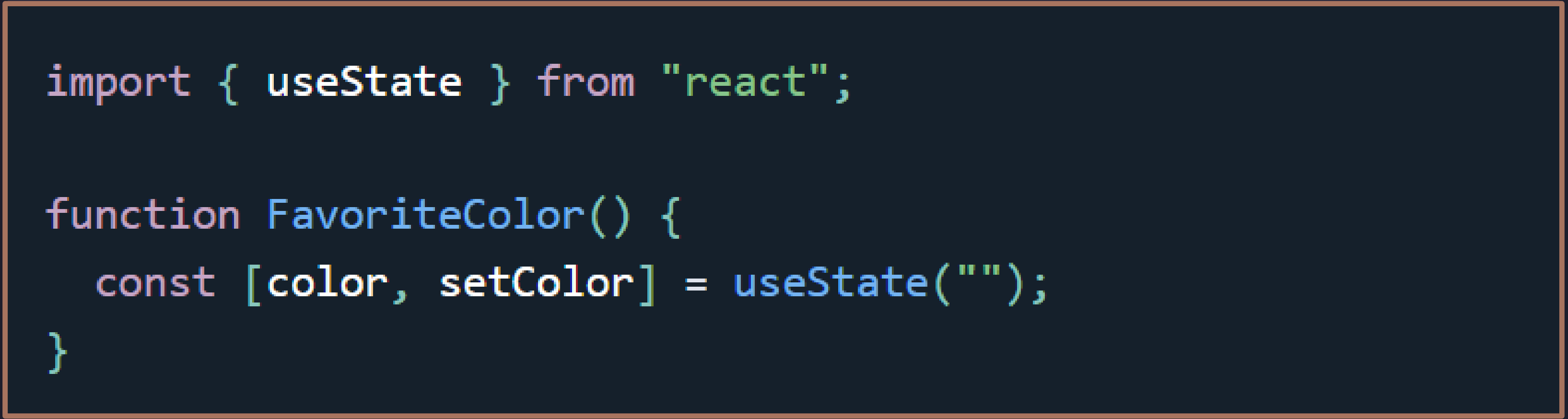


```
import { useState } from "react";

function FavoriteColor() {
  const [color, setColor] = useState("");
}
```

useState hook – Initialize useState

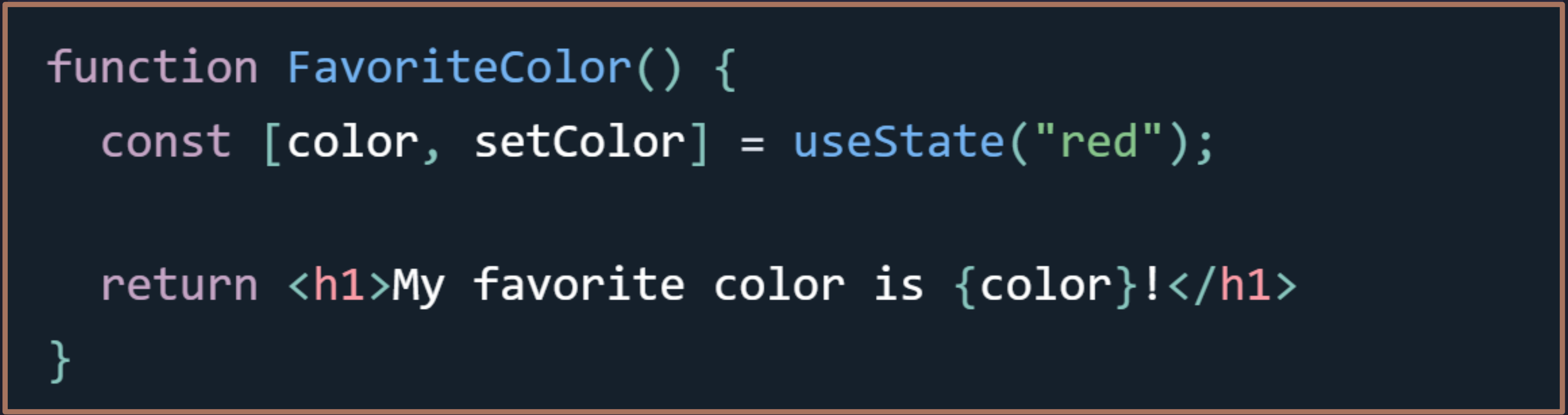
- The first value, color, is our current state.
- The second value, setColor, is the function that is used to update our state.



```
import { useState } from "react";

function FavoriteColor() {
  const [color, setColor] = useState("");
}
```

useState hook – Read state



```
function FavoriteColor() {
  const [color, setColor] = useState("red");

  return <h1>My favorite color is {color}!</h1>
}
```

useState hook – Update state

```
function FavoriteColor() {
  const [color, setColor] = useState("red");

  return (
    <>
      <h1>My favorite color is {color}!</h1>
      <button
        type="button"
        onClick={() => setColor("blue")}
      >Blue</button>
    </>
  )
}
```

useState hook – Multiple useState

```
function Car() {
  const [brand, setBrand] = useState("Ford");
  const [model, setModel] = useState("Mustang");
  const [year, setYear] = useState("1964");
  const [color, setColor] = useState("red");

  return (
    <>
      <h1>My {brand}</h1>
      <p>
        It is a {color} {model} from {year}.
      </p>
    </>
  )
}
```

02

useEffect

useEffect hook

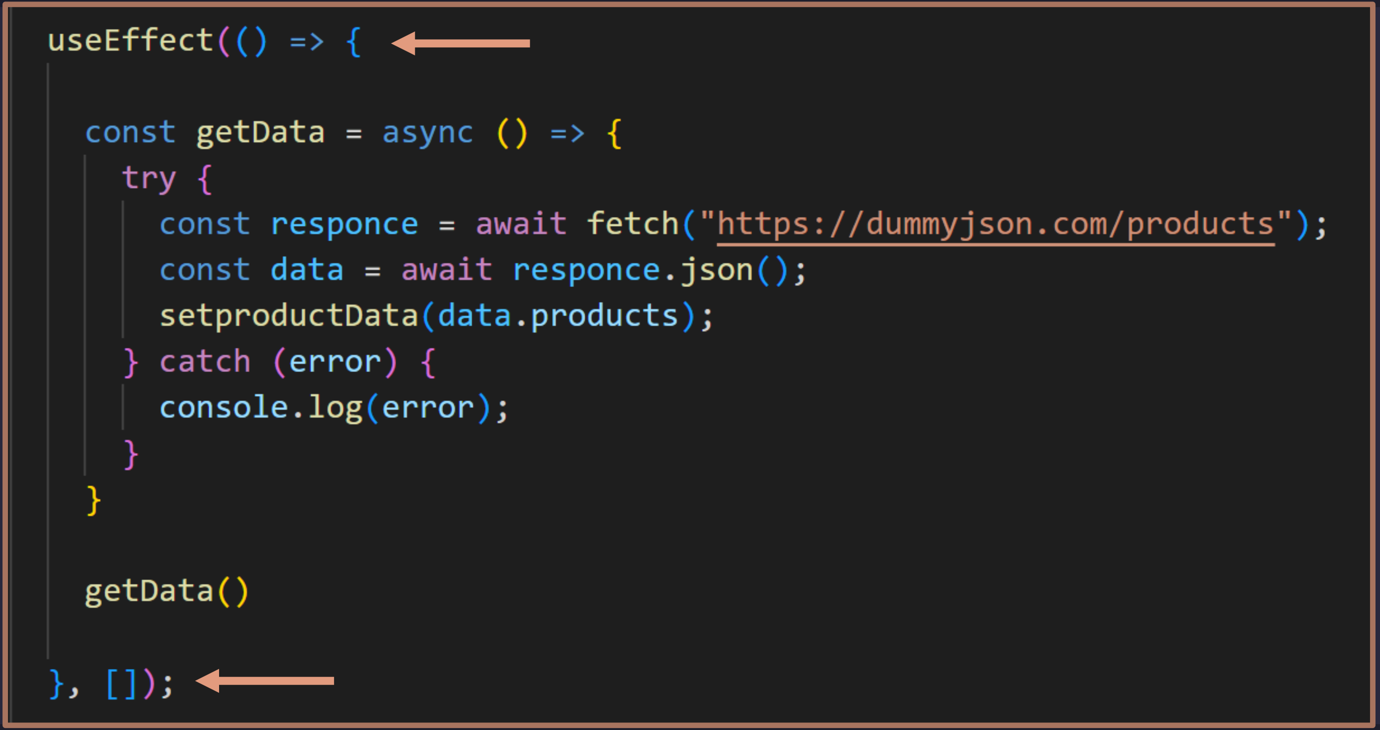
- The useEffect Hook allows you to perform side effects in your components.
- Some examples of side effects are:
  - Fetching data.
  - Directly updating the DOM.
  - Timers.

useEffect hook

- useEffect accepts two arguments.
- First is a callback function, second is dependency array.
- Runs on every render.
- useEffect renders again only when it’s dependency changes.

```
useEffect(() => { }, [] )
```

useEffect hook – Example



```
useEffect(() => {

  const getData = async () => {
    try {
      const responce = await fetch("https://dummyjson.com/products");
      const data = await responce.json();
      setproductData(data.products);
    } catch (error) {
      console.log(error);
    }
  }

  getData()

}, []);
```

03

useRef

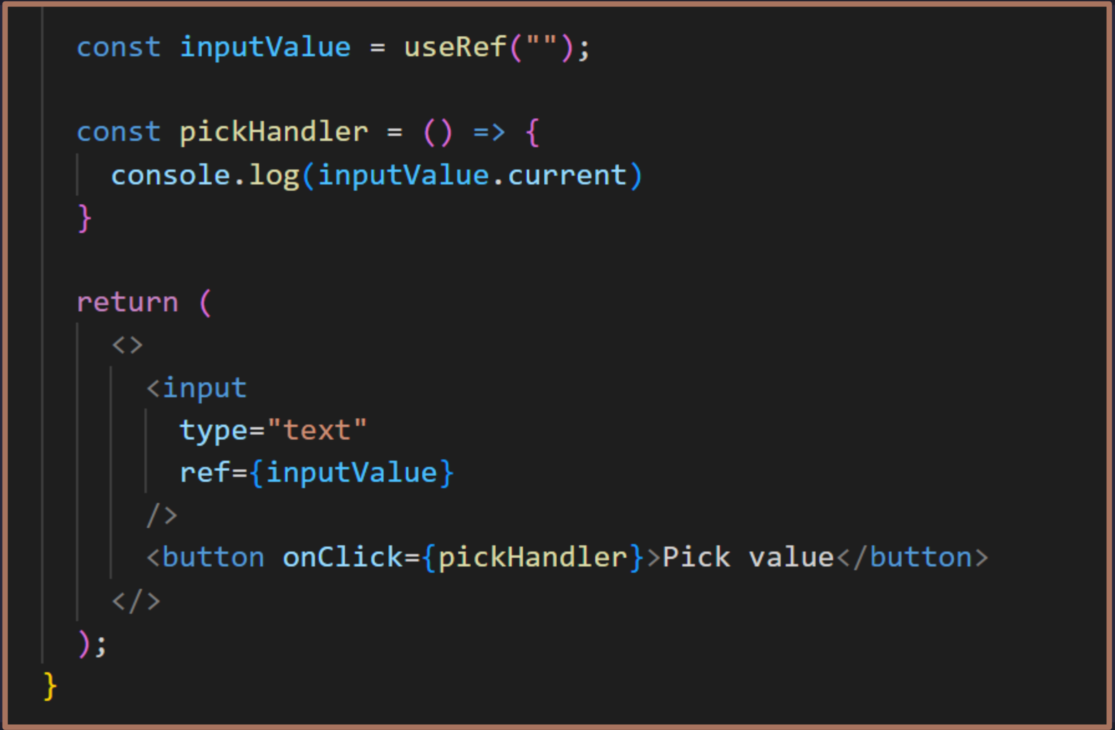
useRef hook

- The useRef Hook allows you to persist values between renders.
- It can be used to store a mutable value that does not cause a re-render when updated.
- It can be used to access a DOM element directly.

useRef hook

- useRef() only returns one item.
- It returns an Object called current.
- When we initialize useRef we set the initial value: useRef(0).

useRef hook



```
  const inputValue = useRef("");

  const pickHandler = () => {
    console.log(inputValue.current)
  }

  return (
    <>
      <input
        type="text"
        ref={inputValue}
      />
      <button onClick={pickHandler}>Pick value</button>
    </>
  );
}
```

03

useMemo

What is memoization?

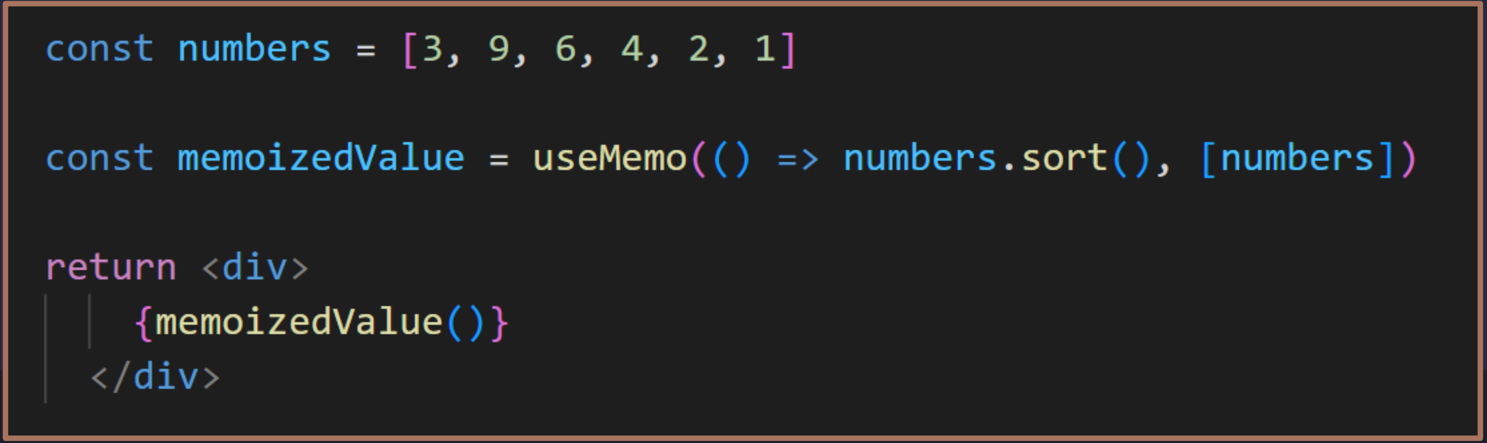
- Memoization is when a complex function stores its output.
- So the next time it is called with the same input.
- It’s similar to caching, but on a more local level.
- It can skip any complex computations and return the output faster as it’s already calculated.
- This can have a significant effect on memory allocation and performance

useMemo hook

- The React useMemo Hook returns a memoized value.
- Think of memoization as caching a value so that it does not need to be recalculated.
- The useMemo Hook only runs when one of its dependencies update.
- This can improve performance.

useMemo hook

- sortedNumbers will become the array [1, 2, 3, 4, 6, 9].
- As long as the numbers variable stays.
- So will sortedNumbers, and it’ll never recompute.



```
const numbers = [3, 9, 6, 4, 2, 1]

const memoizedValue = useMemo(() => numbers.sort(), [numbers])

return <div>
    {memoizedValue()}
  </div>
```

04

useCallback hook

useCallback hook

- The React useCallback Hook returns a memoized callback function.
- This allows us to isolate resource intensive functions.
- So that they will not automatically run on every render.
- The useCallback Hook only runs when one of its dependencies update.
- This can improve performance.

useCallback hook

- The useCallback and useMemo Hooks are similar.
- The main difference is that useMemo returns a memoized value.
- While useCallback returns a memoized function.

useCallback hook

- It will always return the same result unless the numbers is modified.

```
const numbers = [3, 9, 6, 4, 2, 1]

const memoizedFunction = useCallback(() => numbers.sort(), [numbers])

return <div>
    {memoizedFunction()}
  </div>
```

<QnA>

> Thanks! <

Regards: Hassan Bilal