Bookstore:**React JS**

Programming language:**JavaScript**

Topic:**States in React JS**

What are states in React JS?

States is a built-in React object used to contain data or information about the component.

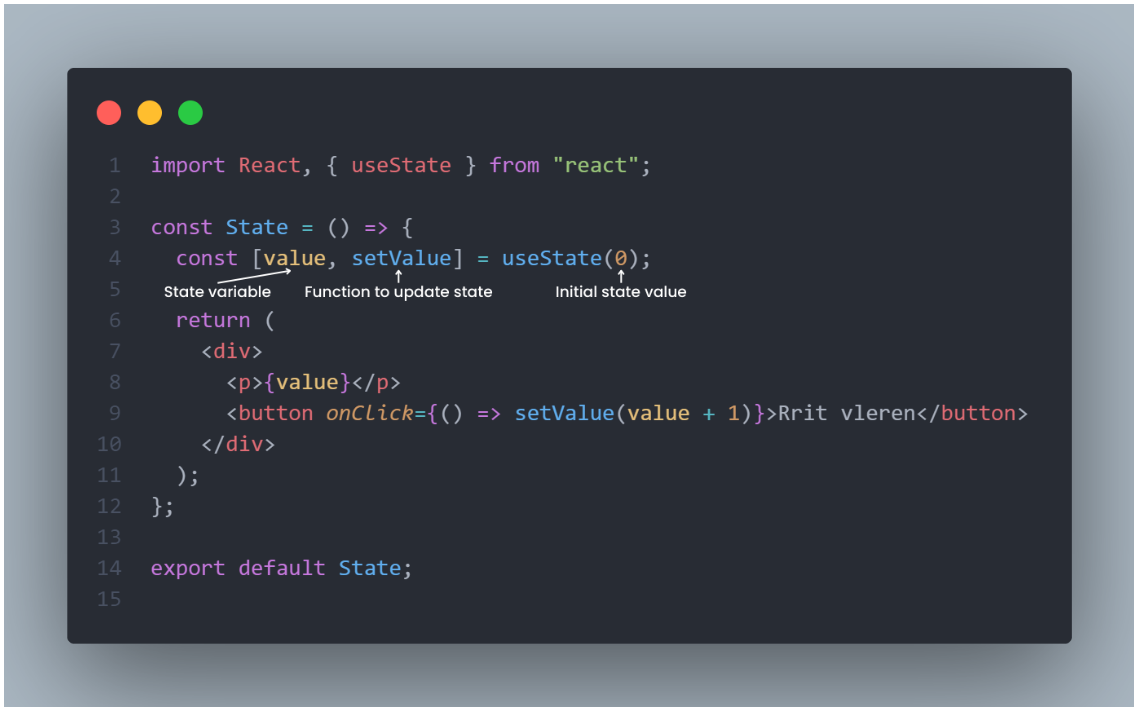
States are data that we need to manage over time within our application.

States are often changed through user input.

Hooks were added in React 16.8.

Prior to this release, there was no mechanism to add state to functional components.

Anatomy of useState() Hook



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

const State = () => {
  const [value, setValue] = useState(0);

  return (
    <div>
      <p>{value}</p>
      <button onClick={() => setValue(value + 1)}>Rrit vleren</button>
    </div>
  );
};

export default State;
```

From the above example we have initialized the state with zero value of integer type, but this value can be any other type like strings, objects, null, text etc.

The useState() hook creates an individual state. It returns an array containing two elements:

- the value of the current state (in the picture it is written "State variable") and
- a function that you can call with a new value to update the state (in the figure it is written "Function to update stateFunction to update state").

Updating the state

The state update function is just a regular function. USE

in the holder***onClick***to replace the current state value. Treatment of

React's internal state values   ensure that your component no

after it reproduces. useState() will return the new value, causing

change of state.

Initializing "lazy" state

The hook itself accepts a parameter which sets the initial value of the variable

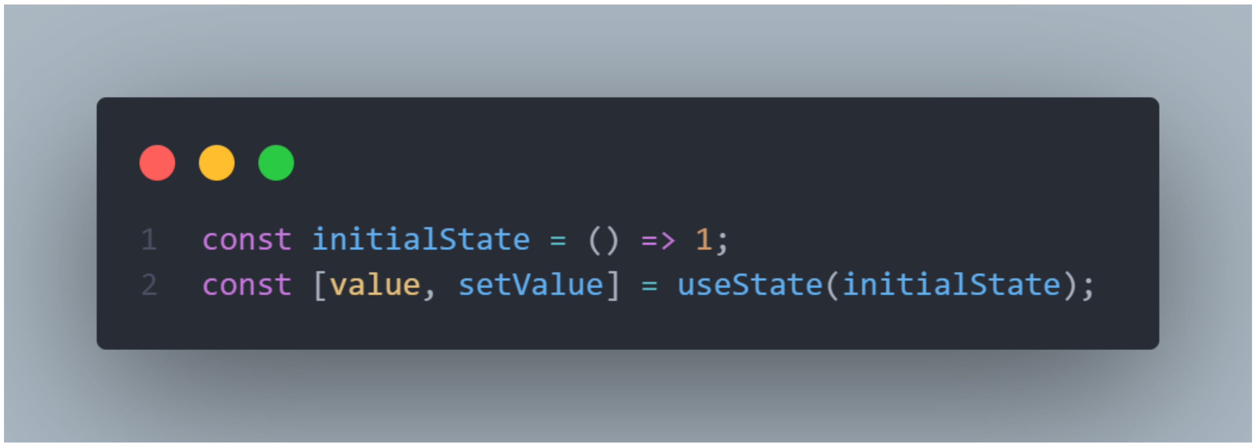
condition. In the example above, the value will be initialized to 1. When not

specify a value, undefined is used. This matches the behavior when

set the state instance property on a component of the class.

If you pass a function to useState(), React will call it and

use its return value as the value of the initial state.



```
const initialState = () => 1;
const [value, setValue] = useState(initialState);
```

This technique enables "lazy" state initialization. The function will not

is called until React is ready to configure state.