Skills Network

State Management in Function Components

IBM

State Management in Function Components

© IBM Corporation. All rights reserved.

What you will learn

Explain the concept of state management

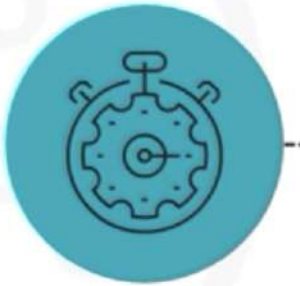
Explain the value of state management when developing function components

Describe the syntax of the useState hook

Manipulate a component's state using the useState hook

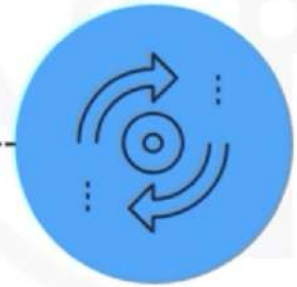
States introduction

Manage data that changes



Condition at a specific time

Render accordingly



State management

Enables local state control

Benefits of hooks

Reuse code logic

No need to

change hierarchy

use nesting

Add state management to function components

Syntax for useState()

General

```
const[stateName, setStateName] = useState(initialState);
```

Example

```
const[color, setColor] = useState("Yellow");
```

Array destructuring

```
const [stateName, setStateName]
```

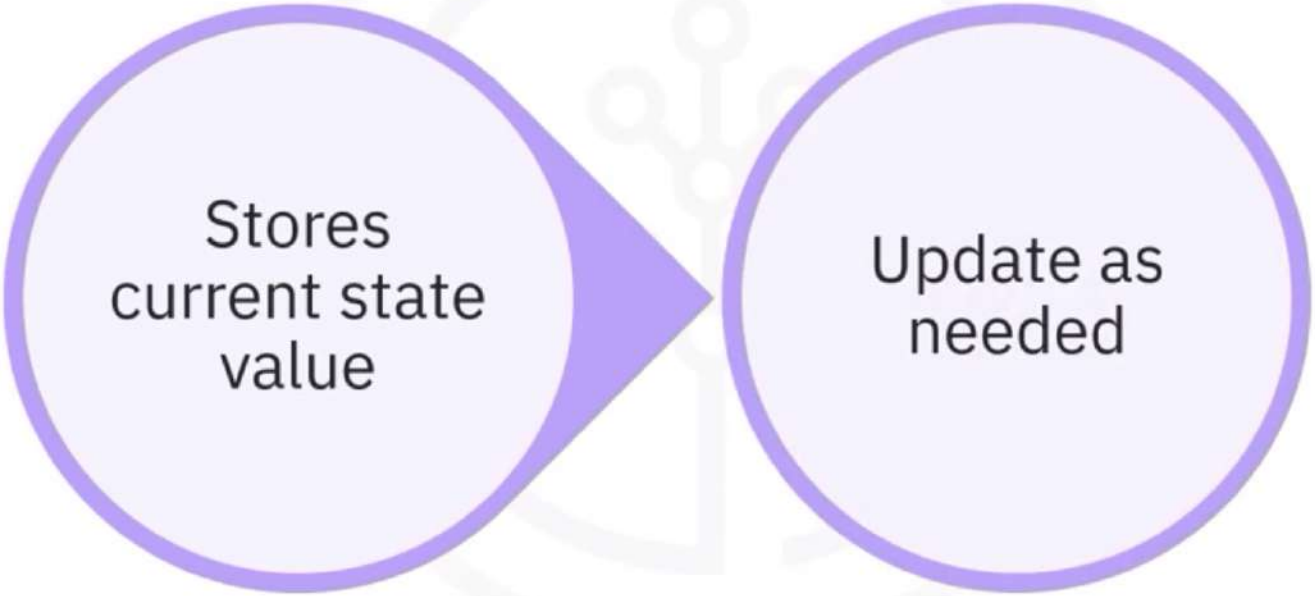
Extract array values

Directly assign

Avoids repetitive indexing

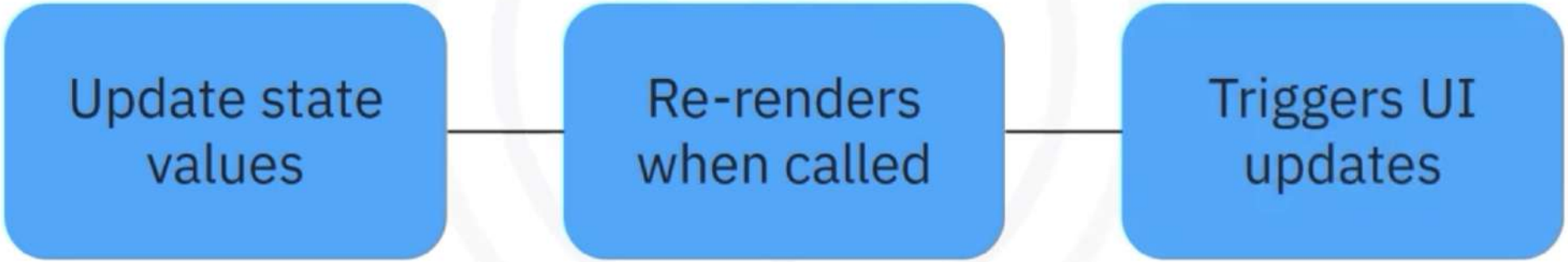
stateName variable

```
const [stateName, setStateName] = useState(initialState);
```



setStateName()

```
const [stateName, setStateName] = useState(initialState);
```



initialState

```
const [stateName, setStateName] = useState(initialState);
```

Initializes the variable

useState example UI

State Management using useState

The name is John

Click to update name

useState example UI

State Management using useState

The name is John Doe

Click to update name

useState example

```
import React, { useState } from 'react'
const StateManagement = () => {
  const[name,setName] = useState('John');
  return (
    <>
      <h1>State Management using useState</h1>
      <p>The name is {name}</p>
    </>
  )
}
export default StateManagement
```

Array destructuring

Button state

```
const StateManagement = () => {
  const [name, setName] = useState('John');
  // State to track button click
  const [buttonClicked, setButtonClicked] = useState(false);
  const updatedName = () => {
    setName('John Doe');
    setButtonClicked(true); // Set to true after updating name
  };
  return (
    <>
      <h1>State Management using useState</h1>
      <p>The name is {name}</p>
      <button onClick={ updatedName } disabled={ buttonClicked } > Click
        to update name</button>
    </>
  );
}
```

Show / Hide example

```
function ToggleMessage() {
  const [isVisible, setIsVisible] = useState(true);
  const toggleVisibility = () => {
    setIsVisible(!isVisible); // Toggle the value of 'isVisible'
  };
  return (
    <div>
      <h2>Toggle Message</h2>
      <button onClick={toggleVisibility}>
        {isVisible ? 'Hide Message' : 'Show Message'}
      </button>
      {isVisible && <p>This is a hidden message.</p>}
    </div>
  );
```

Recap

In this video, you learned that:

- State management allows you to manage data that can change over time within a component
- The useState hook enables function components to manage the component's state locally
- You can use this syntax when using the useState hook:

```
const [stateName, setStateName] = useState(initialState);
```

  - Call the "useState" function and pass in the initial state as a parameter
  - Assign the return value to a const array variable using a destructured array
  - The array has two values: the state name and the state name prefixed with the keyword "set"
- You should use the useState hook when you need to make a component dynamic and interactive