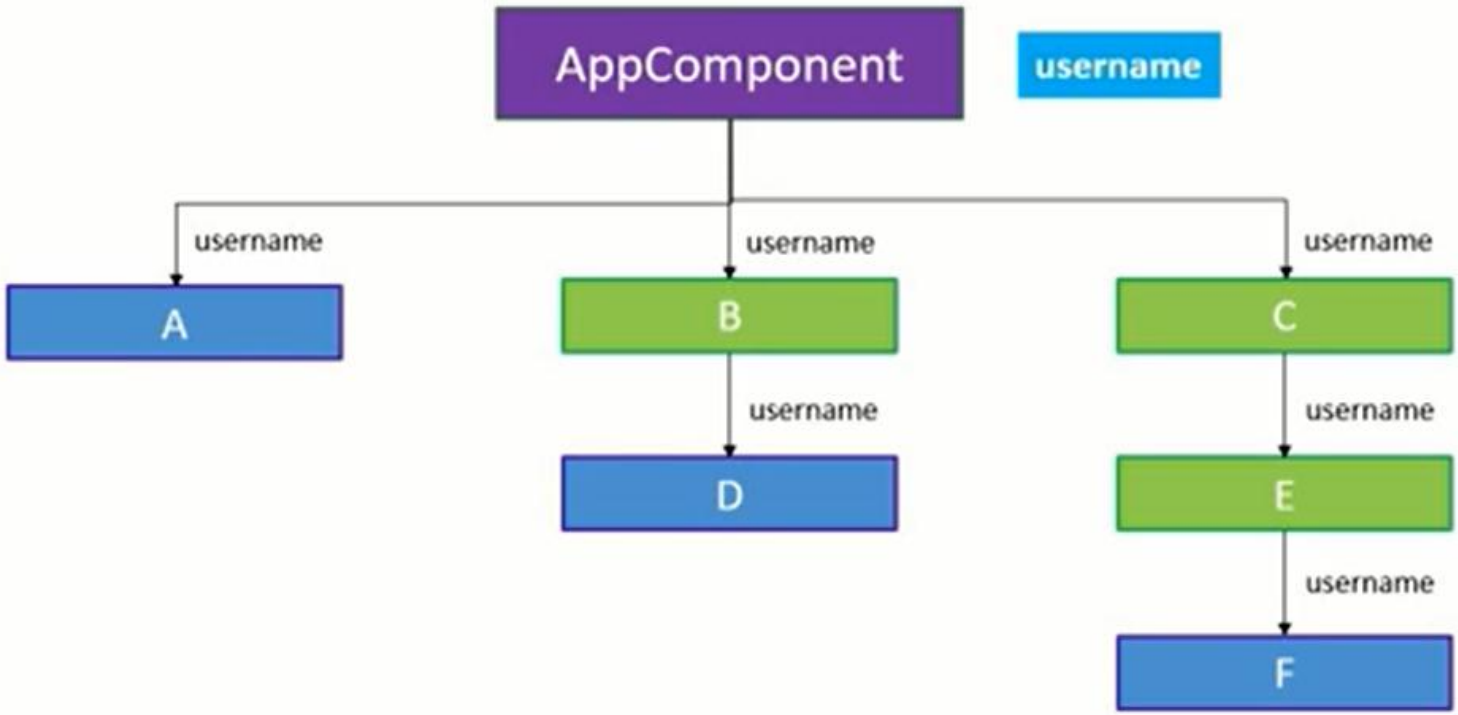
Context


AppComponent
username
username
username
username
A
B
C
username
username
D
E
username
F

Context

Context provides a way to pass data through the component tree without having to pass props down manually at every level.

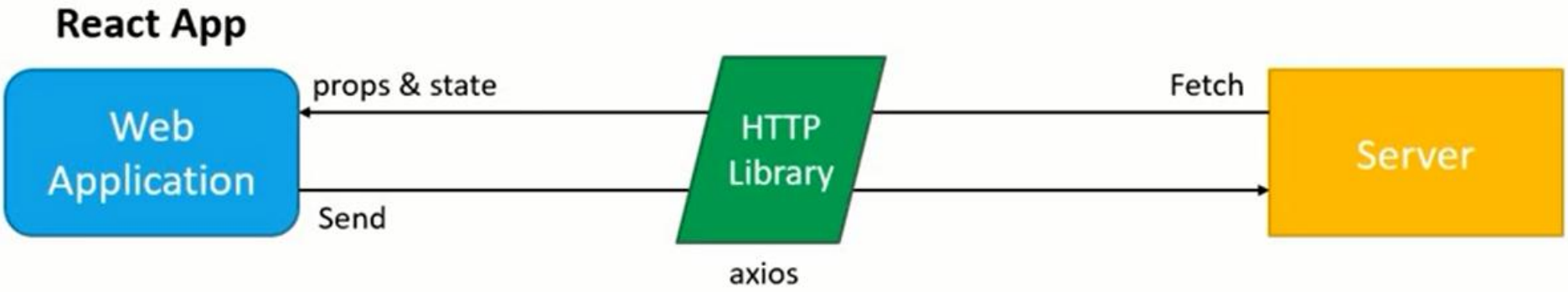
Context

1. Create the context
2. Provide a context value
3. Consume the context value

Consuming Multiple Contexts

```
function Content() {
  return (
    <ThemeContext.Consumer>
      {theme => (
        <UserContext.Consumer>
          {user => (
            <ProfilePage user={user} theme={theme} />
          )}
        </UserContext.Consumer>
      )}
    </ThemeContext.Consumer>
  );
}
```

React and HTTP



Prerequisites

Basics of React

Functional and class components, props, state, etc.

React Tutorial for Beginners on the channel

What are Hooks?

Hooks are a new feature addition in React version 16.8 which allow you to use React features without having to write a class.

Ex: State of a component

Hooks don’t work inside classes

Why Hooks?

Reason Set 1

Understand how ***this*** keyword works in JavaScript

Remember to bind event handlers in class components

Classes don't minify very well and make hot reloading very unreliable

Why Hooks?

Reason Set 2

There is no particular way to reuse stateful component logic

HOC and render props patterns do address this problem

Makes the code harder to follow

There is need a to share stateful logic in a better way

Why Hooks?

Reason Set 3

Create components for complex scenarios such as data fetching and subscribing to events

Related code is not organized in one place

Ex: Data fetching - In componentDidMount and componentDidUpdate

Ex: Event listeners – In componentDidMount and componentWillUnmount

Because of stateful logic – Cannot break components into smaller ones

Noteworthy Points

React version 16.8 or higher

Completely opt in

Hooks don't contain any breaking changes and the release is 100% backwards-compatible.

Classes won't be removed from React

Can't use Hooks inside of a class component

Hooks don't replace your existing knowledge of React concepts

Instead, Hooks provide a more direct API to the React concepts you already know

Summary

Hooks are a new feature addition in React version 16.8

They allow you to use React features without having to write a class

Avoid the whole confusion with 'this' keyword

Allow you to reuse stateful logic

Organize the logic inside a component into reusable isolated units

Rules of Hooks

"Only Call Hooks at the Top Level“

Don’t call Hooks inside loops, conditions, or nested functions

"Only Call Hooks from React Functions“

Call them from within React functional components and not just any regular JavaScript function

Summary - useState

The useState hook lets you add state to functional components

In classes, the state is always an object.

With the useState hook, the state doesn't have to be an object.

The useState hook returns an array with 2 elements.

The first element is the current value of the state, and the second element is a state setter function.

New state value depends on the previous state value? You can pass a function to the setter function.

When dealing with objects or arrays, always make sure to spread your state variable and then call the setter function

useEffect

Updating the document title to the current counter value

```
componentDidMount() {
  document.title = `You clicked ${this.state.count} times`;
}

componentDidUpdate() {
  document.title = `You clicked ${this.state.count} times`;
}
```

useEffect

Timer

```
componentDidMount() {
  this.interval = setInterval(this.tick, 1000)
}
componentWillUnmount() {
  clearInterval(this.interval)
}
```

useEffect

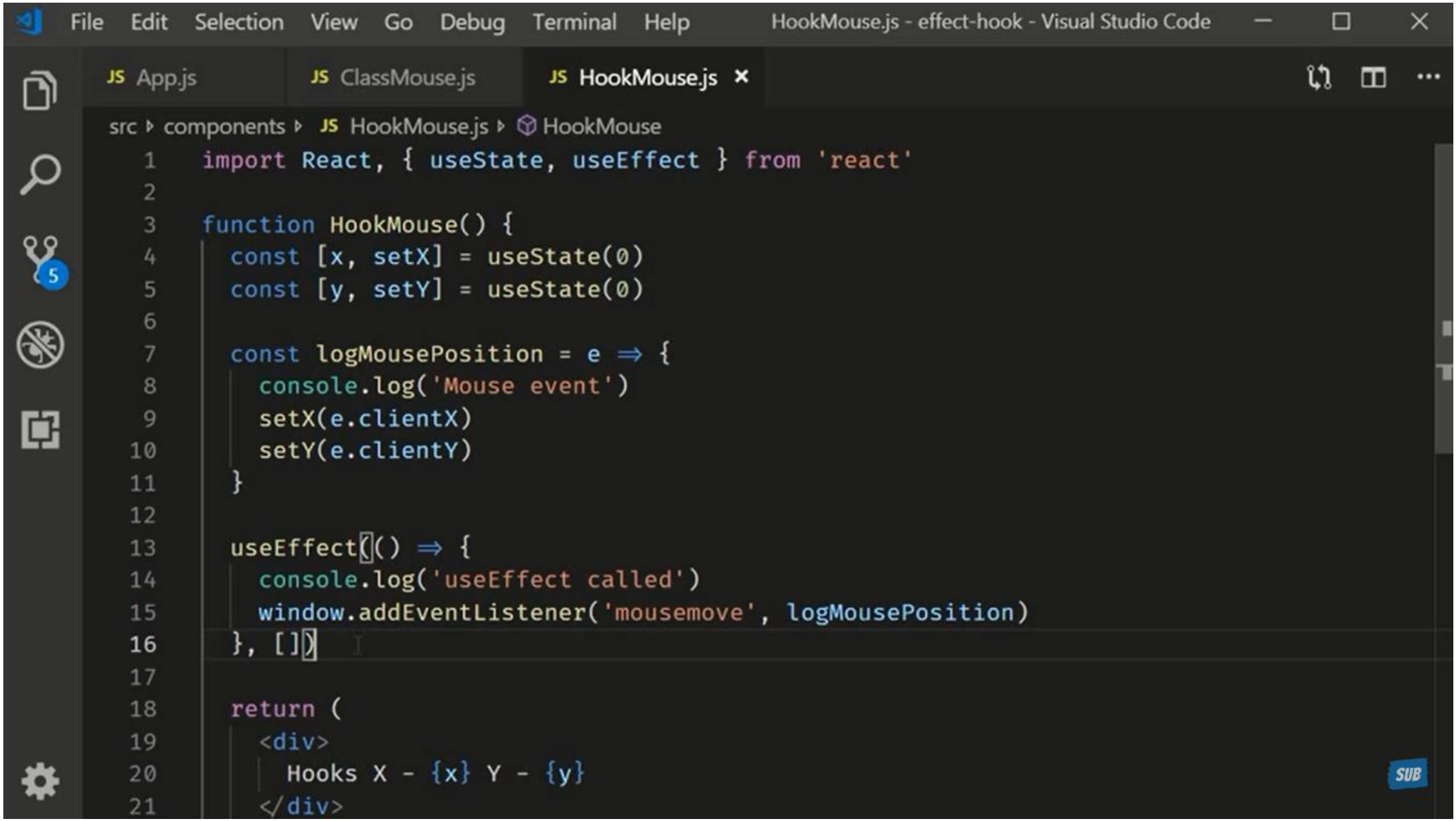
Combine the two side effects

```
componentDidMount() {
  document.title = `You clicked ${this.state.count} times`;
  this.interval = setInterval(this.tick, 1000)
}
componentDidUpdate() {
  document.title = `You clicked ${this.state.count} times`;
}
componentWillUnmount() {
  clearInterval(this.interval)
}
```

useEffect

The Effect Hook lets you perform **side effects** in **functional components**

It is a close replacement for ***componentDidMount***, ***componentDidUpdate*** and ***componentWillUnmount***


File Edit Selection View Go Debug Terminal Help
HookMouse.js - effect-hook - Visual Studio Code
App.js
ClassMouse.js
HookMouse.js
src ▸ components ▸ HookMouse.js ▸ HookMouse
1 import React, { useState, useEffect } from 'react'
2
3 function HookMouse() {
4 const [x, setX] = useState(0)
5 const [y, setY] = useState(0)
6
7 const logMousePosition = e => {
8 console.log('Mouse event')
9 setX(e.clientX)
10 setY(e.clientY)
11 }
12
13 useEffect(() => {
14 console.log('useEffect called')
15 window.addEventListener('mousemove', logMousePosition)
16 }, [])
17
18 return (
19
20 Hooks X - {x} Y - {y}
21
SUB

Multiple useEffects

```
function FriendStatusWithCounter(props) {
  const [count, setCount] = useState(0);
  useEffect(() => {
    document.title = `You clicked ${count} times`;
  });

  const [isOnline, setIsOnline] = useState(null);
  useEffect(() => {
    function handleStatusChange(status) {
      setIsOnline(status.isOnline);
    }

    ChatAPI.subscribeToFriendStatus(props.friend.id, handleStatusChange);
    return () => {
      ChatAPI.unsubscribeFromFriendStatus(props.friend.id, handleStatusChange);
    };
  });
  // ...
}
```