WHAT IS HOOKS ?

Hooks are the functions to use some react features in functional components.

In other words, Hooks are functions that make Functional components work like Class components.

What is useState?

UseState Hook is a function to add State in Functional Component.

What is state?

State is nothing but just values or variables of your component.


App.js - 06_CODE - Visual Studio Code
App.js
src > App.js > App
1 import React, { useState } from "react";
2
3 const App = () => {
4 const [counter, setCounter] = useState(0);
5
6 function increaseCounter() {
7 setCounter(counter + 1);
8 }
9 return (
10
11 <h1>Counter: {counter}</h1>
12 <button onClick={increaseCounter}>Increase</button>
13
14);
15 };
16
17 export default App;
18
Subscribe


● App.js - 06_CODE - Visual Studio Code
App.js
src > App.js > App
const App = () => {
const [counter, setCounter] = useState(0);
const [name, setName] = useState("")
function increaseCounter() {
setCounter(counter + 1);
}
return (

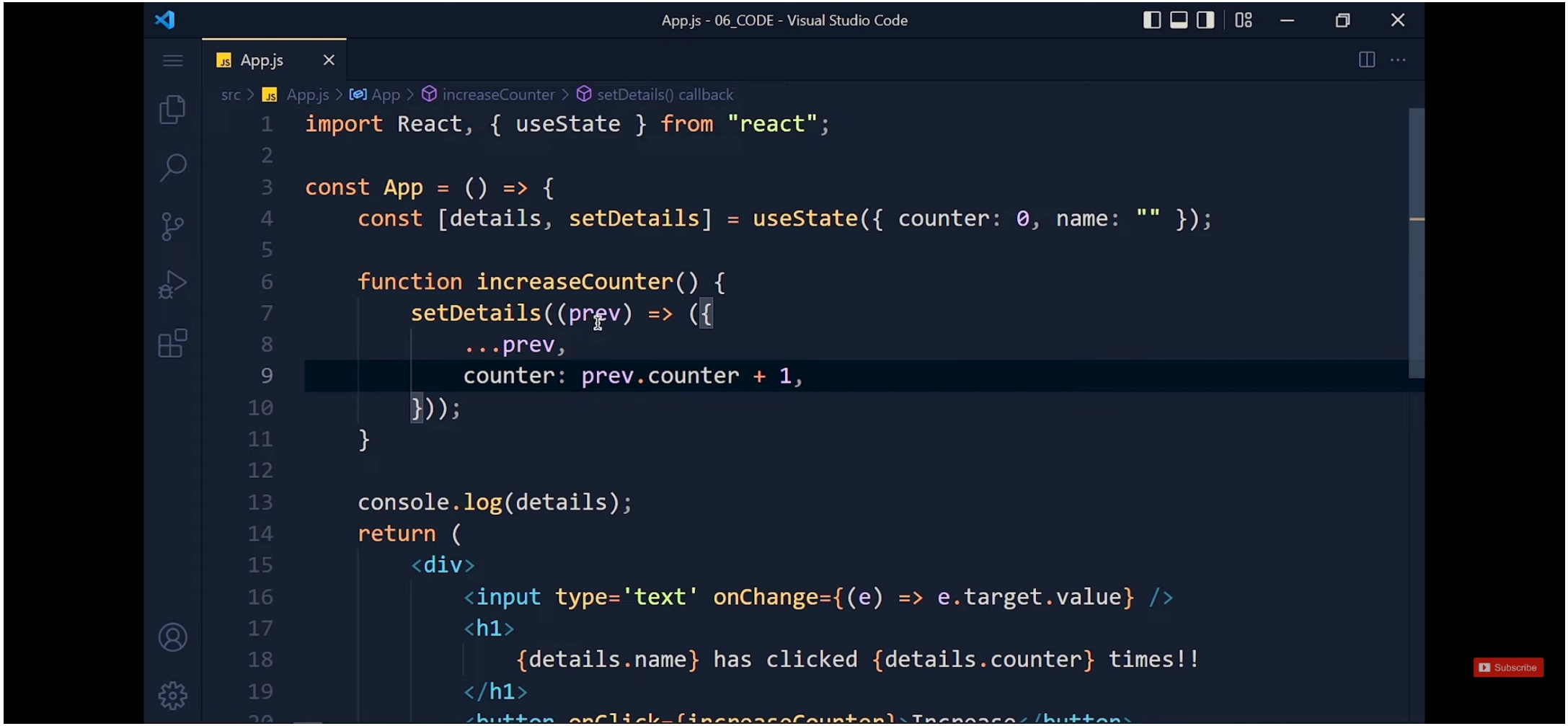
<input type="text" onChange={e => setName(e.target.value)} />
<h1>{name} has clicked {counter} time</h1>
<button onClick={increaseCounter}>Increase</button>

);
};
export default App;
Subscribe

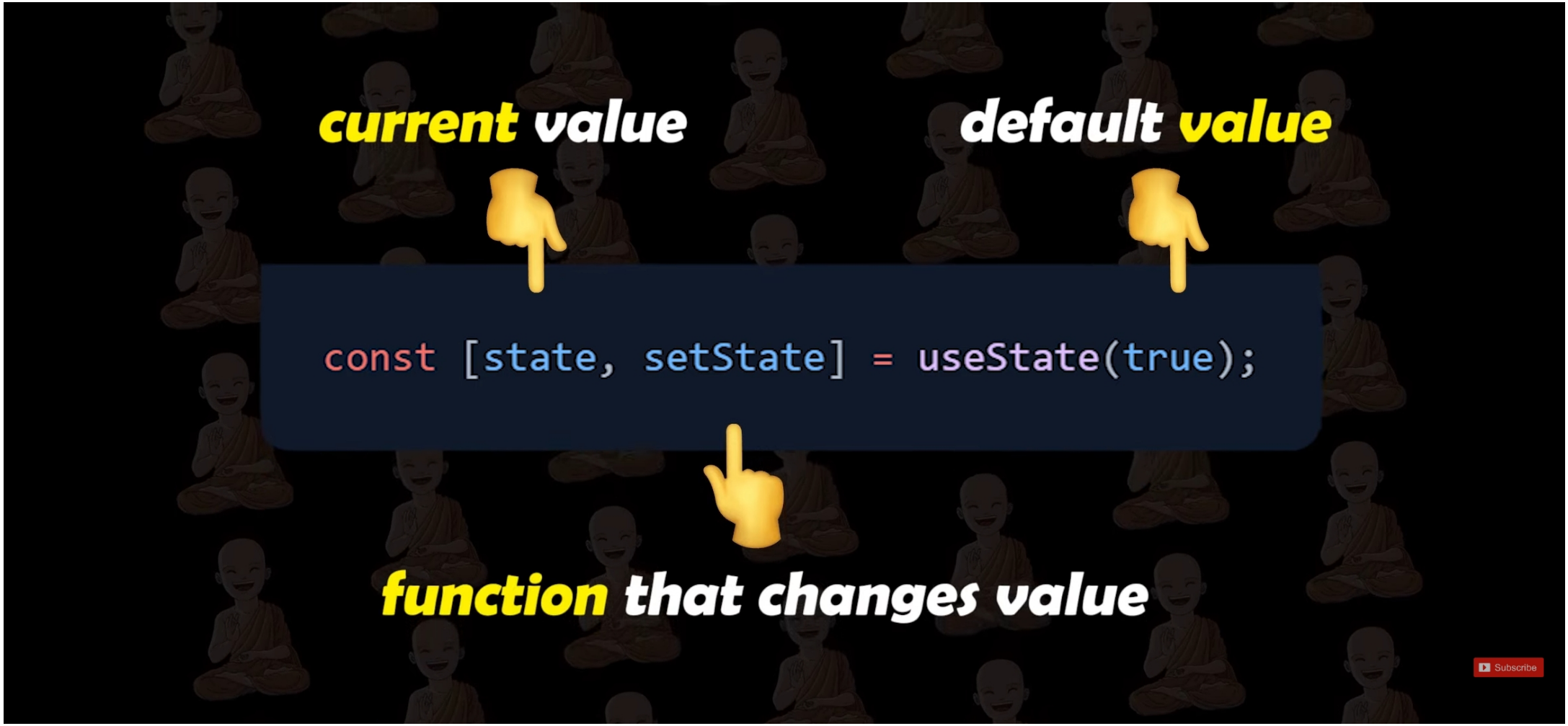

App.js
App.js > App
import React, { useState } from "react";
const App = () => {
const [counter, setCounter] = useState(0);
setCounter(counter + 1);
return (

<h1>Counter: {counter}</h1>
<button onClick={increaseCounter}>Increase</button>

);
};
(property) HTMLInputElement.value:
Returns the value of the data at the cursor's
Subscribe
useState with Object


App.js - 06_CODE - Visual Studio Code
App.js
src > App.js > App > increaseCounter > setDetails() callback
import React, { useState } from "react";
const App = () => {
const [details, setDetails] = useState({ counter: 0, name: "" });
function increaseCounter() {
setDetails((prev) => ({
...prev,
counter: prev.counter + 1,
}));
}
console.log(details);
return (

<input type='text' onChange={(e) => e.target.value} />
<h1>
{details.name} has clicked {details.counter} times!!
</h1>
Subscribe


current value
default value
const [state, setState] = useState(true);
function that changes value
Subscribe

What is side Effect?

Side effects are actions which are performed with the "Outside world".

We perform a side effect when we need to reach outside of our React components to do something.

useEffect

UseEffect is used to perform side effects in our component.

What is side Effect?

Side effects are actions which are performed with the "Outside world".

What is side Effect?

Side effects are actions which are performed with the "Outside world".

We perform a side effect when we need to reach outside of our React components to do something.

Some Common Side Effects

Fetching data from API

updating the DOM document

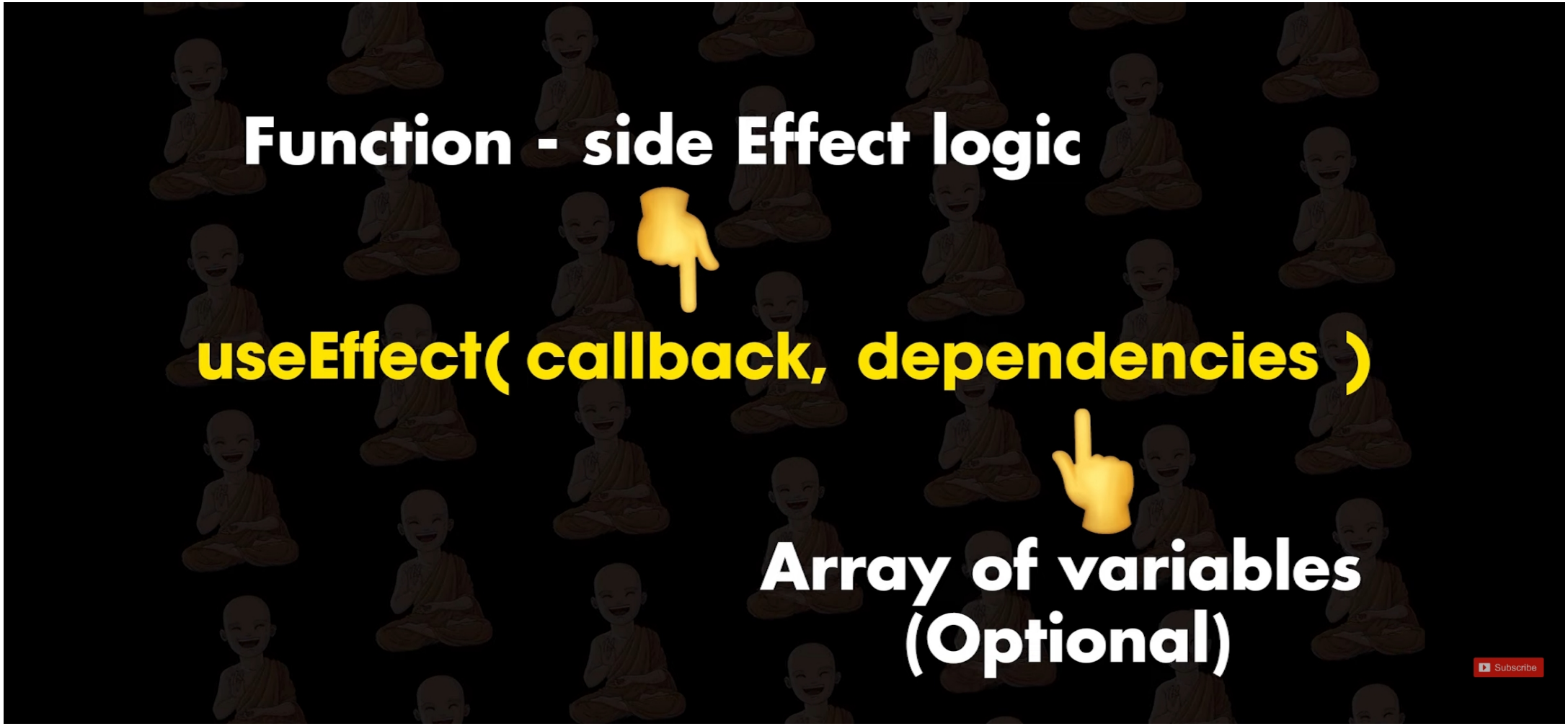
& window

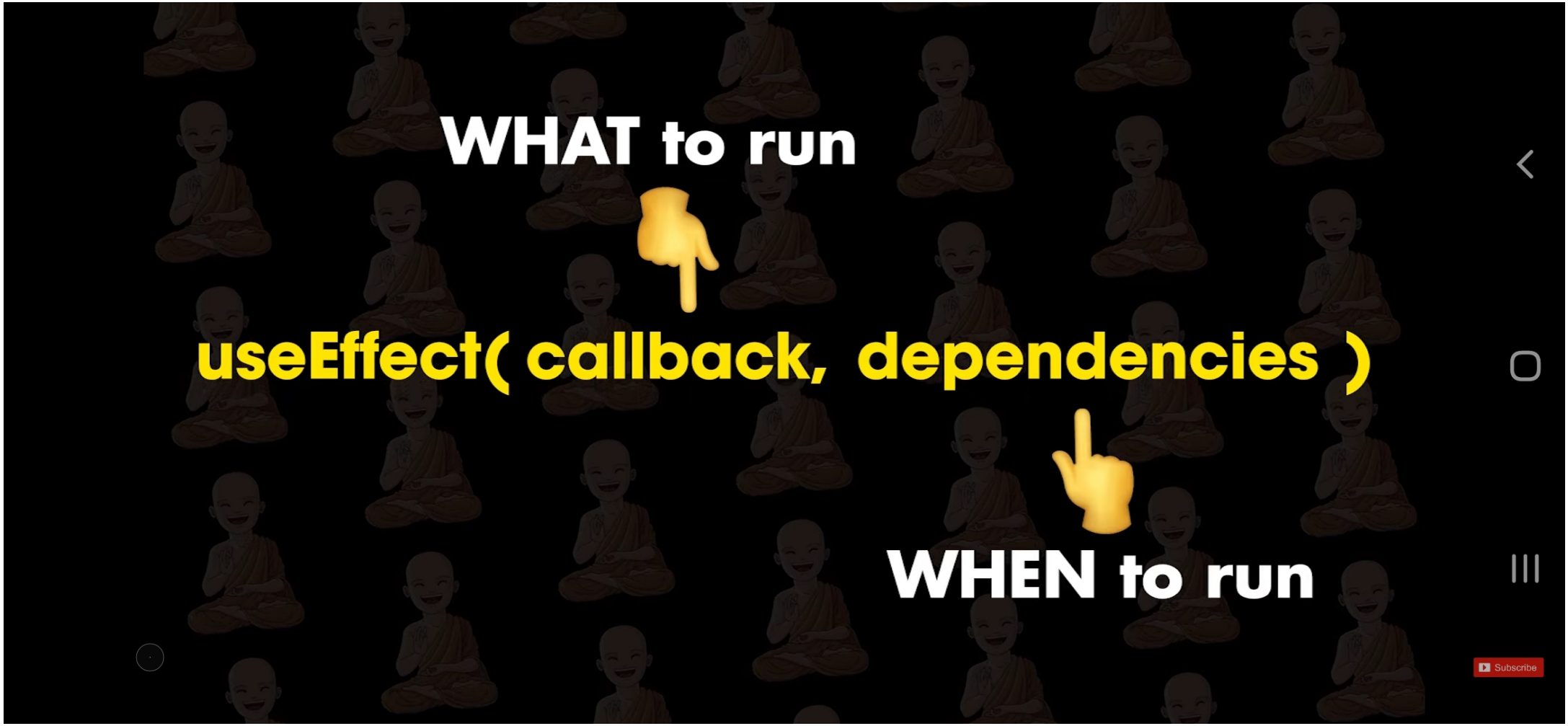
Some Common Side Effects

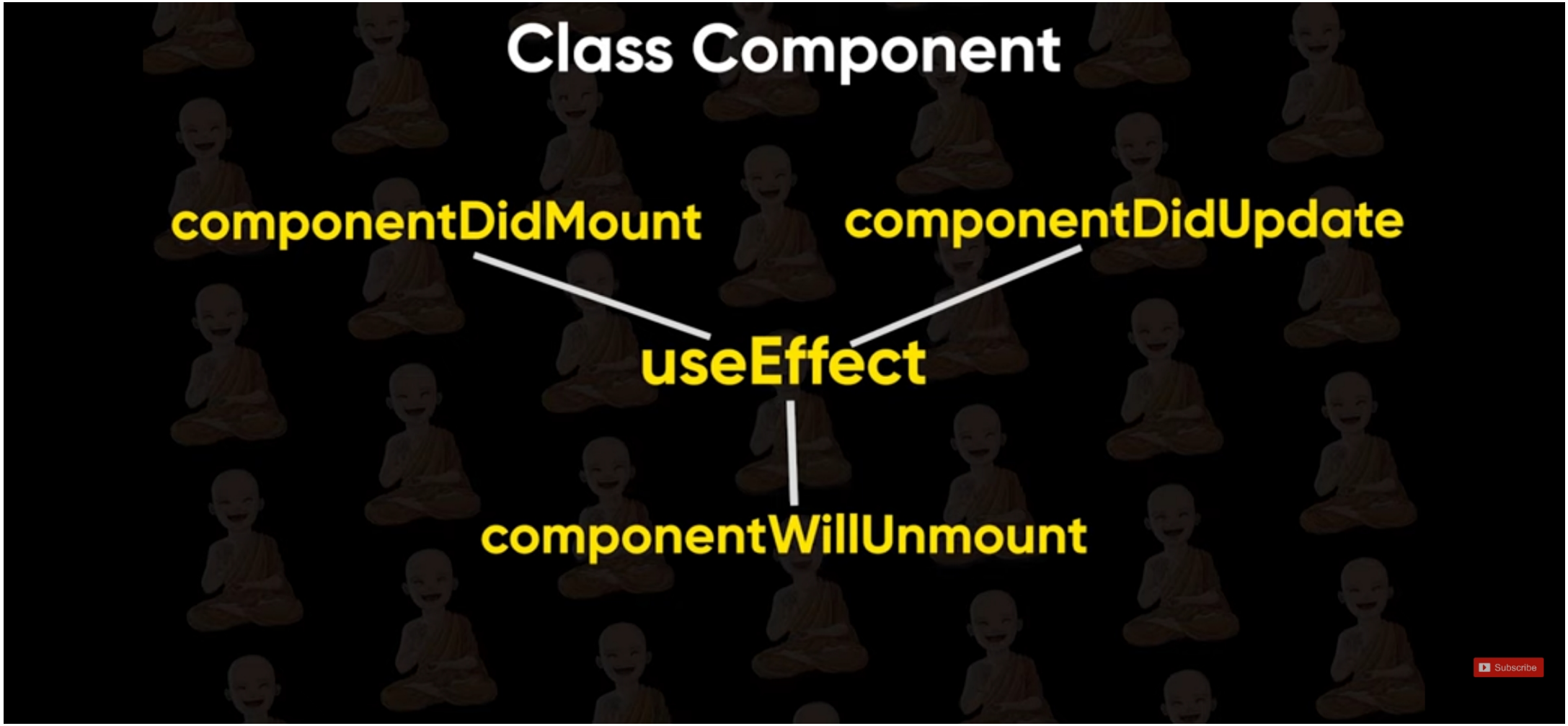
Fetching data from API

updating the DOM document & window

Timer functions setTimeout & setInterval

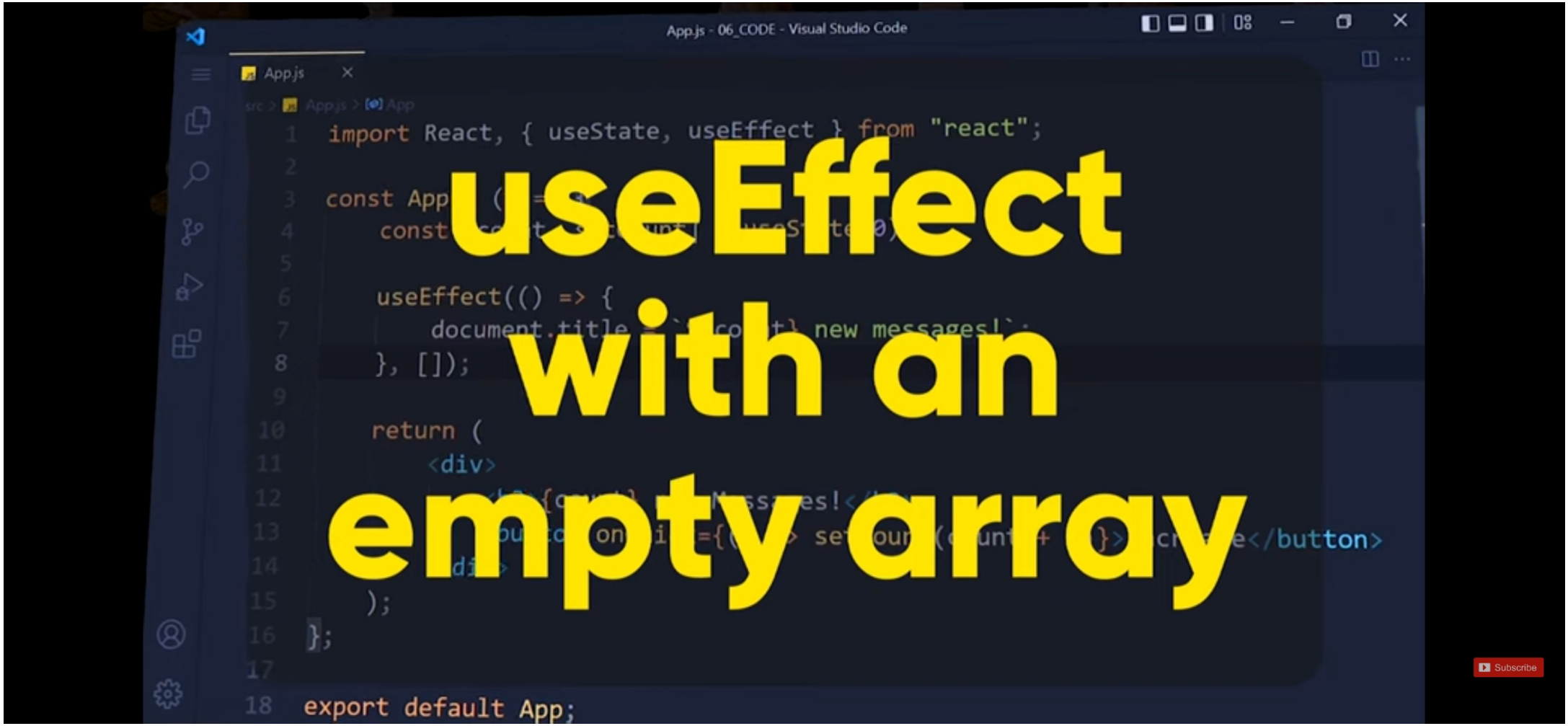

Function - side Effect logic
useEffect(callback, dependencies)
Array of variables
(Optional)
Subscribe


WHAT to run
useEffect(callback, dependencies)
WHEN to run
Subscribe

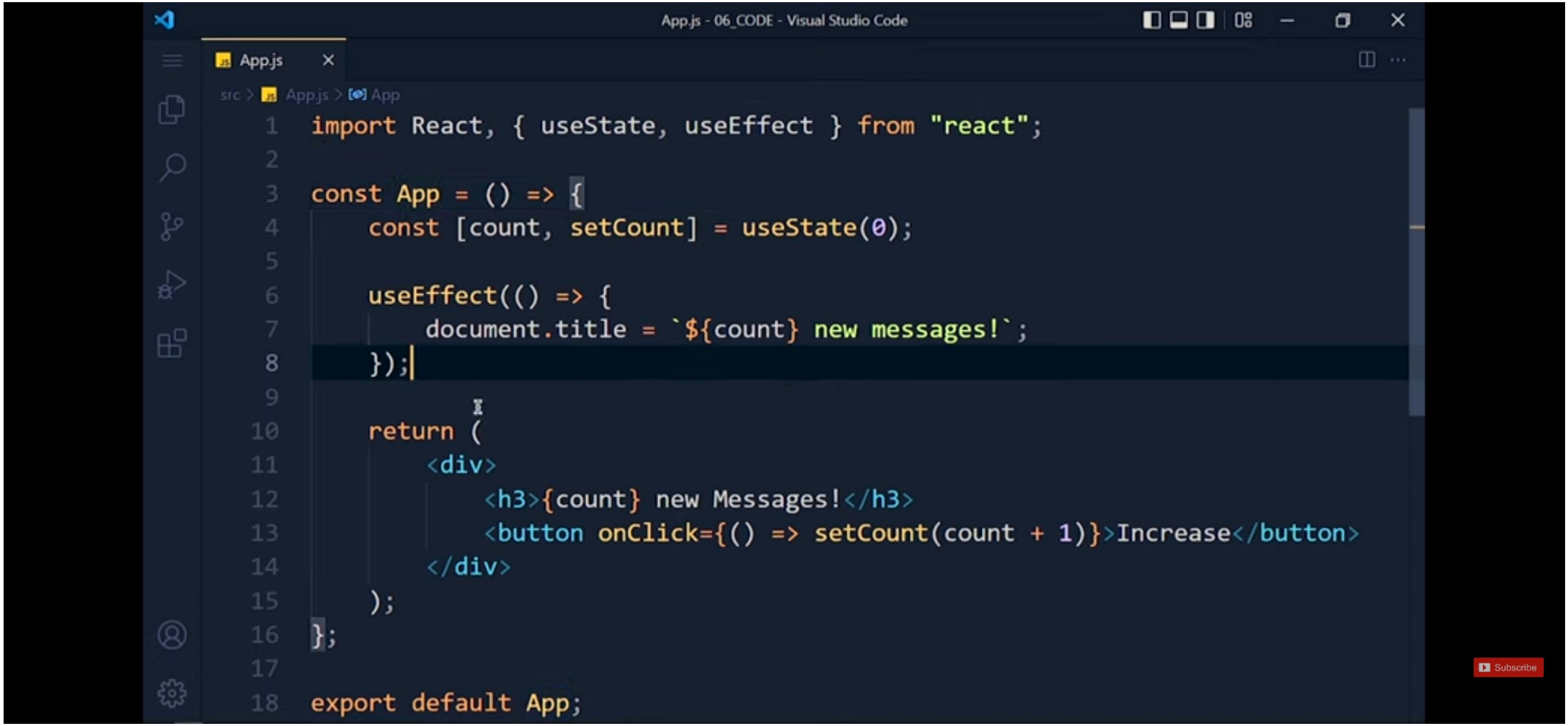

Class Component
componentDidMount
componentDidUpdate
useEffect
componentWillUnmount
Subscribe

VARIATION OF USEEFFECT

- useEffect without dependencies
- useEffect with empty array
- useEffect with variables


App.js - 06_CODE - Visual Studio Code
App.js
import React, { useState, useEffect } from "react";
useEffect(() => {
}, []);
return (

);
};
export default App;
useEffect
with an
empty array
Subscribe


App.js - 06_CODE - Visual Studio Code
App.js
src > App.js > App
import React, { useState, useEffect } from "react";
const App = () => {
const [count, setCount] = useState(0);
useEffect(() => {
document.title = `${count} new messages!`;
});
return (

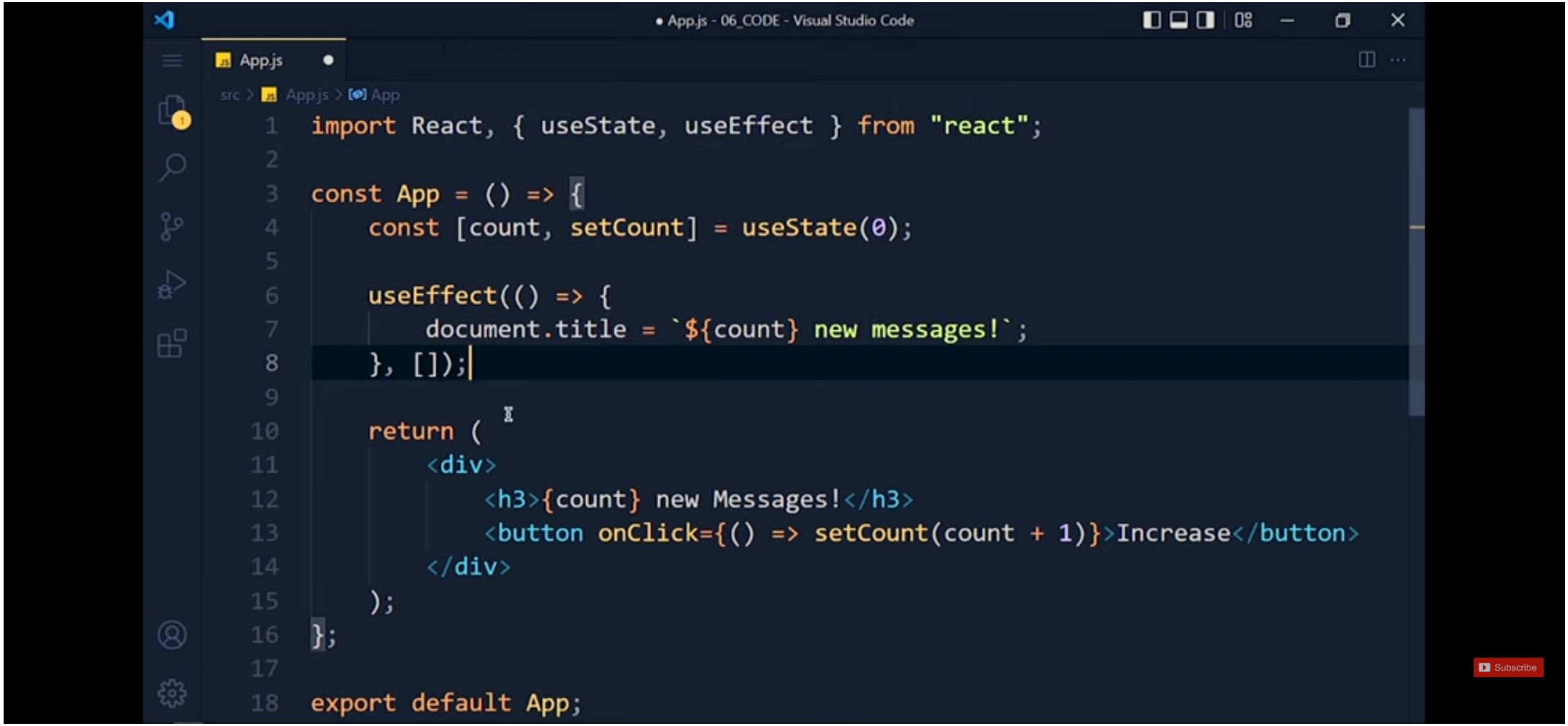
<h3>{count} new Messages!</h3>
<button onClick={() => setCount(count + 1)}>Increase</button>

);
};
export default App;
Subscribe


App.js
src > App.js > App
import React, { useState, useEffect } from "react";
const App = () => {
const [count, setCount] = useState(0);
useEffect(() => {
document.title = `${count} new messages!`;
}, []);
return (

<button onClick={() => setCount(count + 1)}>Increment</button>

);
};
useEffect
with an
empty array
Subscribe


• App.js - 06_CODE - Visual Studio Code
App.js
src > App.js > App
1 import React, { useState, useEffect } from "react";
2
3 const App = () => {
4 const [count, setCount] = useState(0);
5
6 useEffect(() => {
7 document.title = `${count} new messages!`;
8 }, []);
9
10 return (
11
12 <h3>{count} new Messages!</h3>
13 <button onClick={() => setCount(count + 1)}>Increase</button>
14
15);
16 };
17
18 export default App;
Subscribe

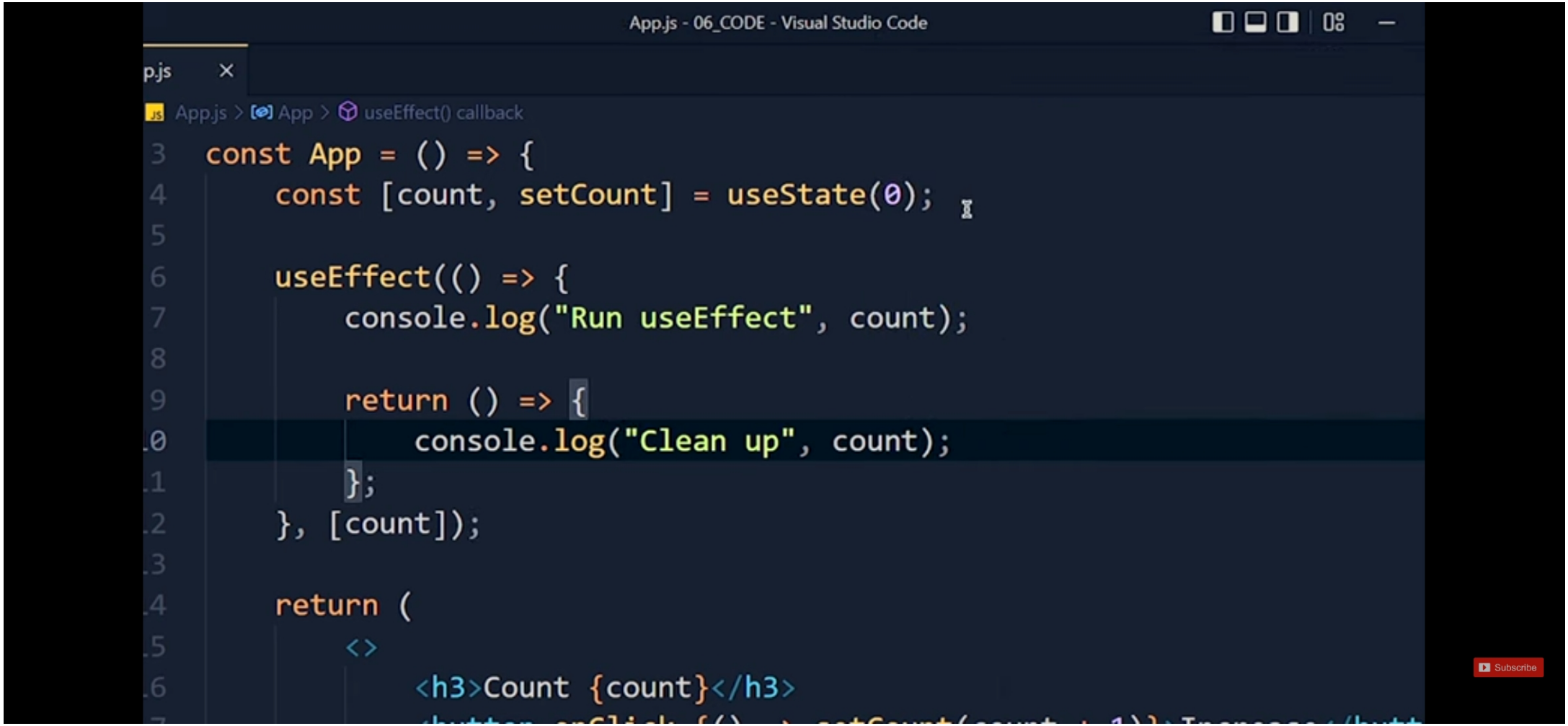

App.js - 06_CODE - Visual Studio Code
App.js
useEffect
with
variables
Subscribe

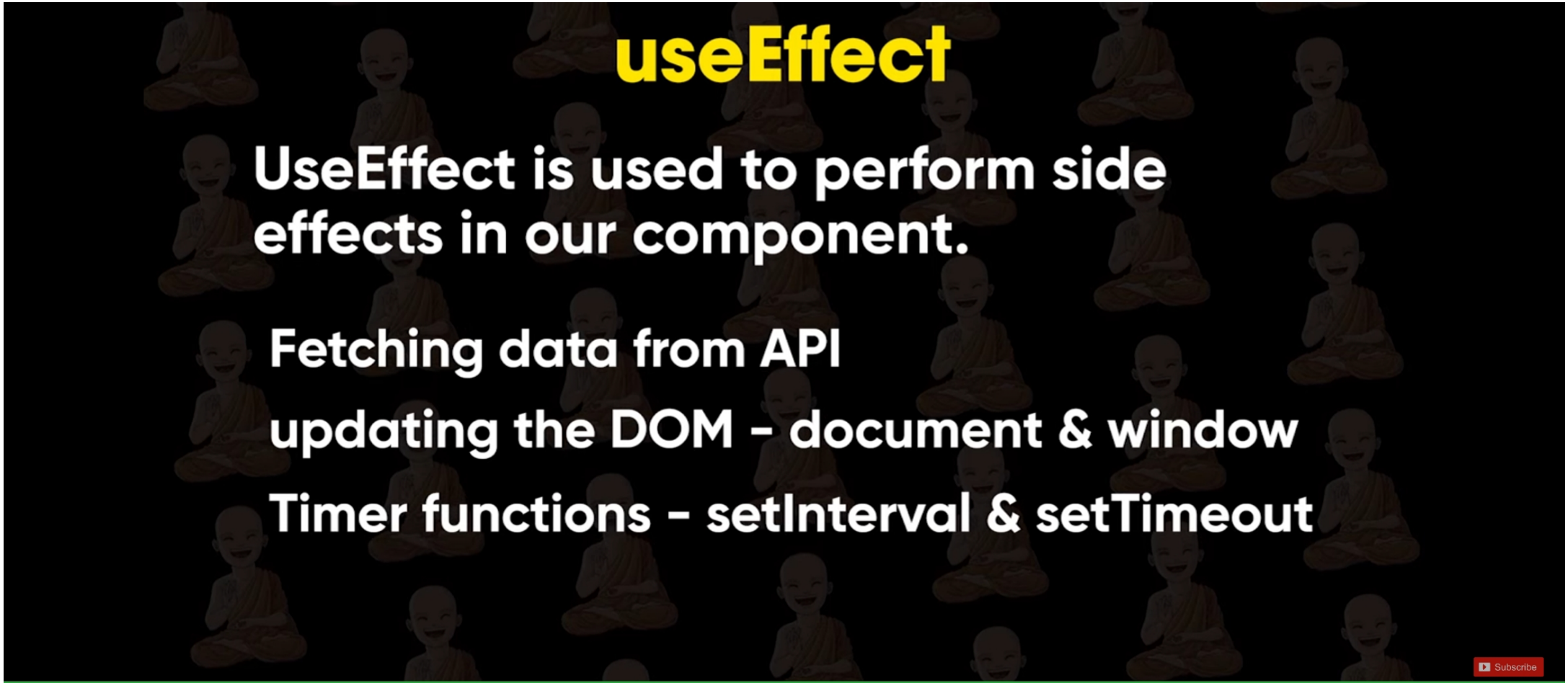

• App.js - 06_CODE - Visual Studio Code
App.js
src > App.js > App
const App = () => {
const [count, setCount] = useState(0);
const [otherCount, setOtherCount] = useState(5);
useEffect(() => {
document.title = `${otherCount} new messages!`;
}, [otherCount, count, ..]);
return (

<h3>{count} new Messages!</h3>
<button onClick={() => setCount(count + 1)}>Increase</button>
<h3>Other Count : {otherCount} </h3>
<button onClick={() => setOtherCount(otherCount + 5)}>
Increase by 5
</button>

);
Subscribe


• App.js - 06_CODE - Visual Studio Code
App.js
Clean-up Function in useEffect
const timer = setInterval(() => {
cleanInterval(timer);
<h3>{time} in seconds</h3>
Subscribe


App.js - 06_CODE - Visual Studio Code
p.js
App.js > App > useEffect() callback
const App = () => {
const [count, setCount] = useState(0);
useEffect(() => {
console.log("Run useEffect", count);
return () => {
console.log("Clean up", count);
};
}, [count]);
return (
<>
<h3>Count {count}</h3>
Subscribe


useEffect
UseEffect is used to perform side effects in our component.
Fetching data from API
updating the DOM - document & window
Timer functions - setInterval & setTimeout
Subscribe

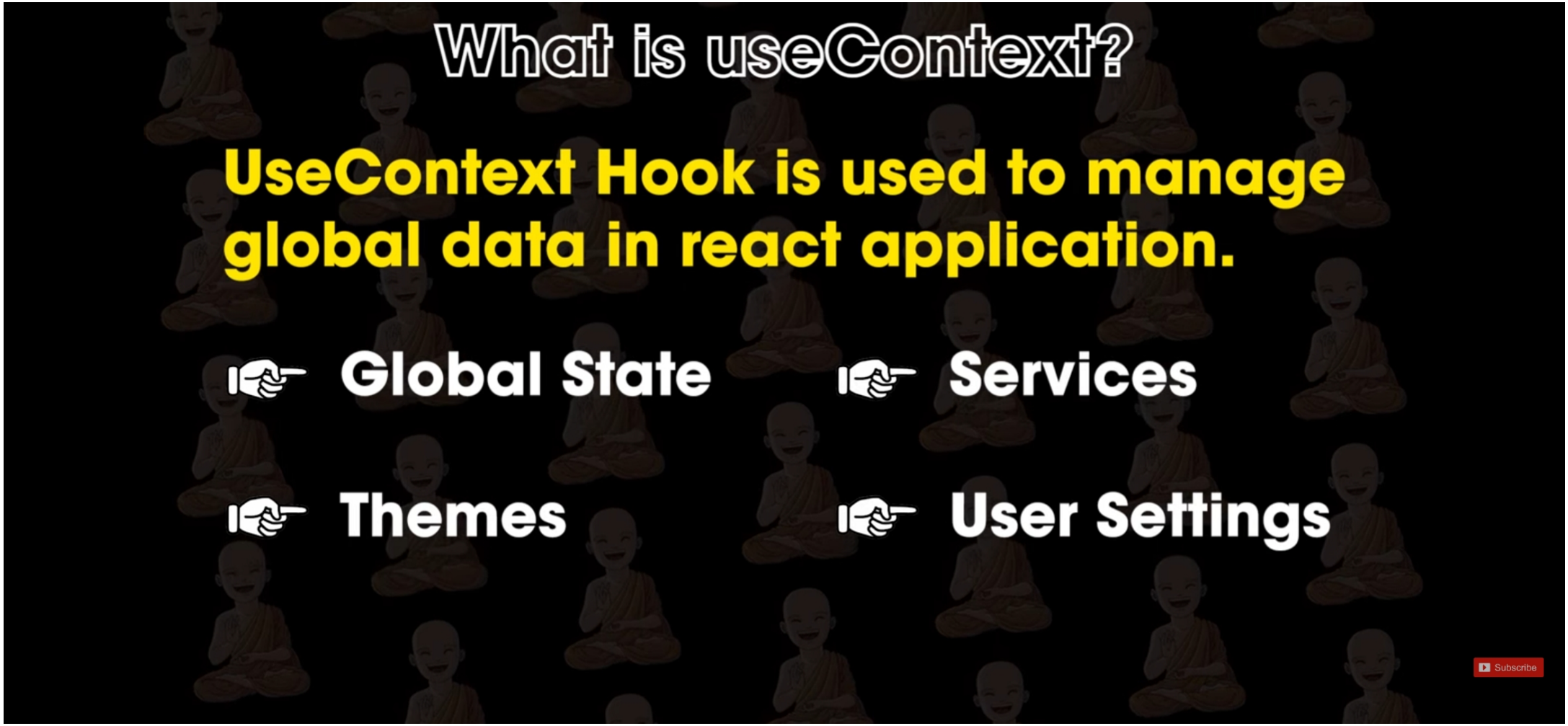
Back online

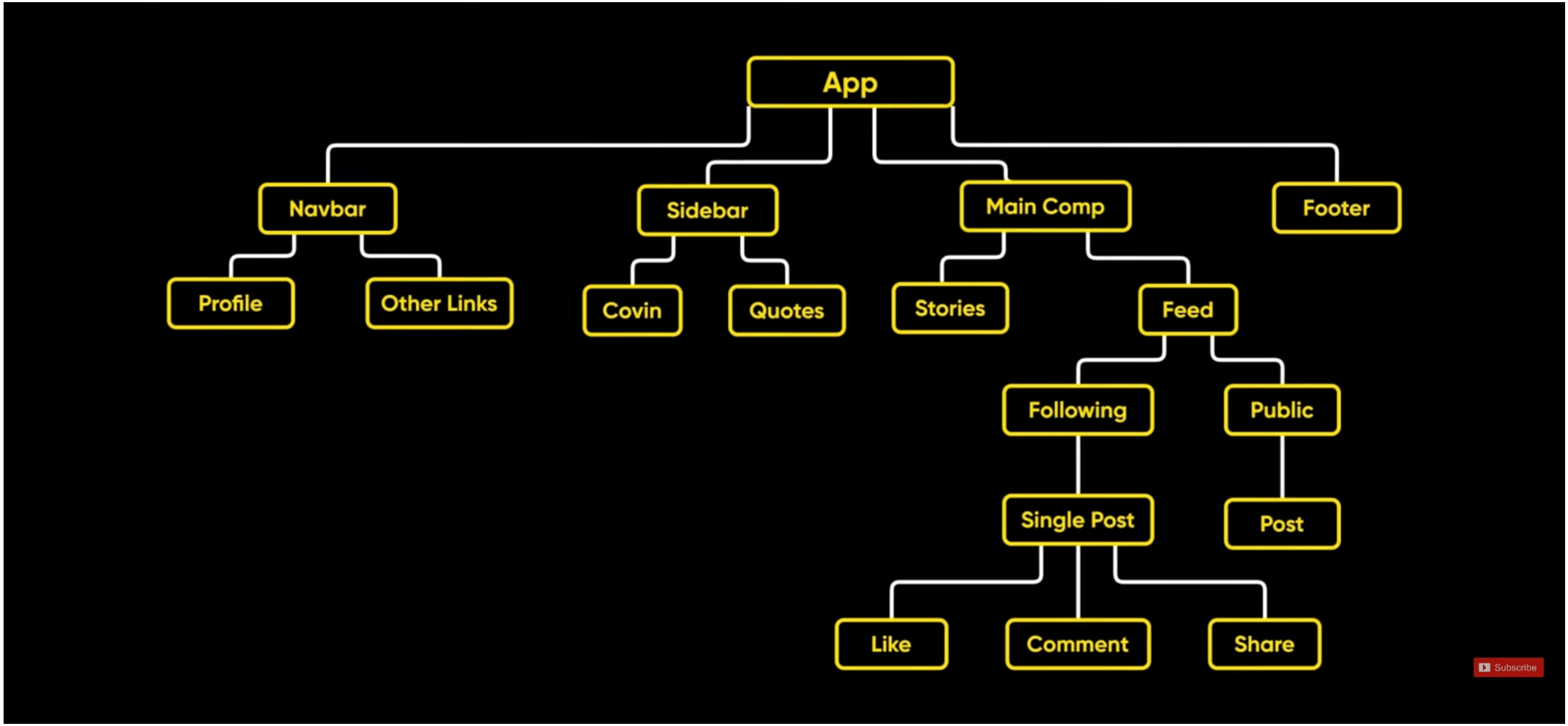
3 VARIATION OF USEEFFECT

- useEffect without dependencies - it runs with first render and also run on any thing changes.
- useEffect with empty array - it runs only on first render.
- useEffect with variables - it runs on first render and runs with that variable change.

React Hooks Number 03

useContext Hook


What is useContext?
UseContext Hook is used to manage global data in react application.
Global State
Services
Themes
User Settings
Subscribe


App
Navbar
Sidebar
Main Comp
Footer
Profile
Other Links
Covin
Quotes
Stories
Feed
Following
Public
Single Post
Post
Like
Comment
Share
Subscribe

HOW TO CREATE CONTEXT?

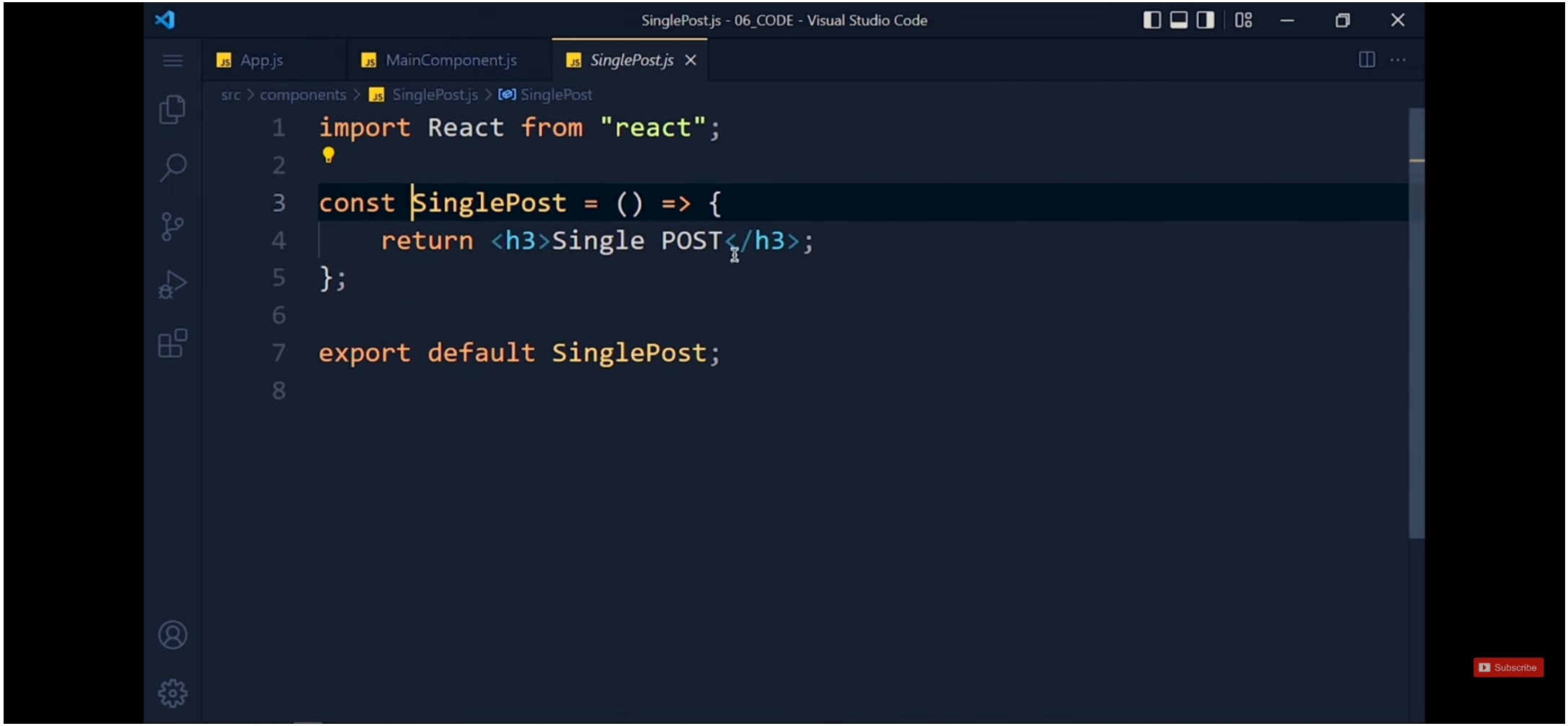
Create context requires 3 Simple Steps:

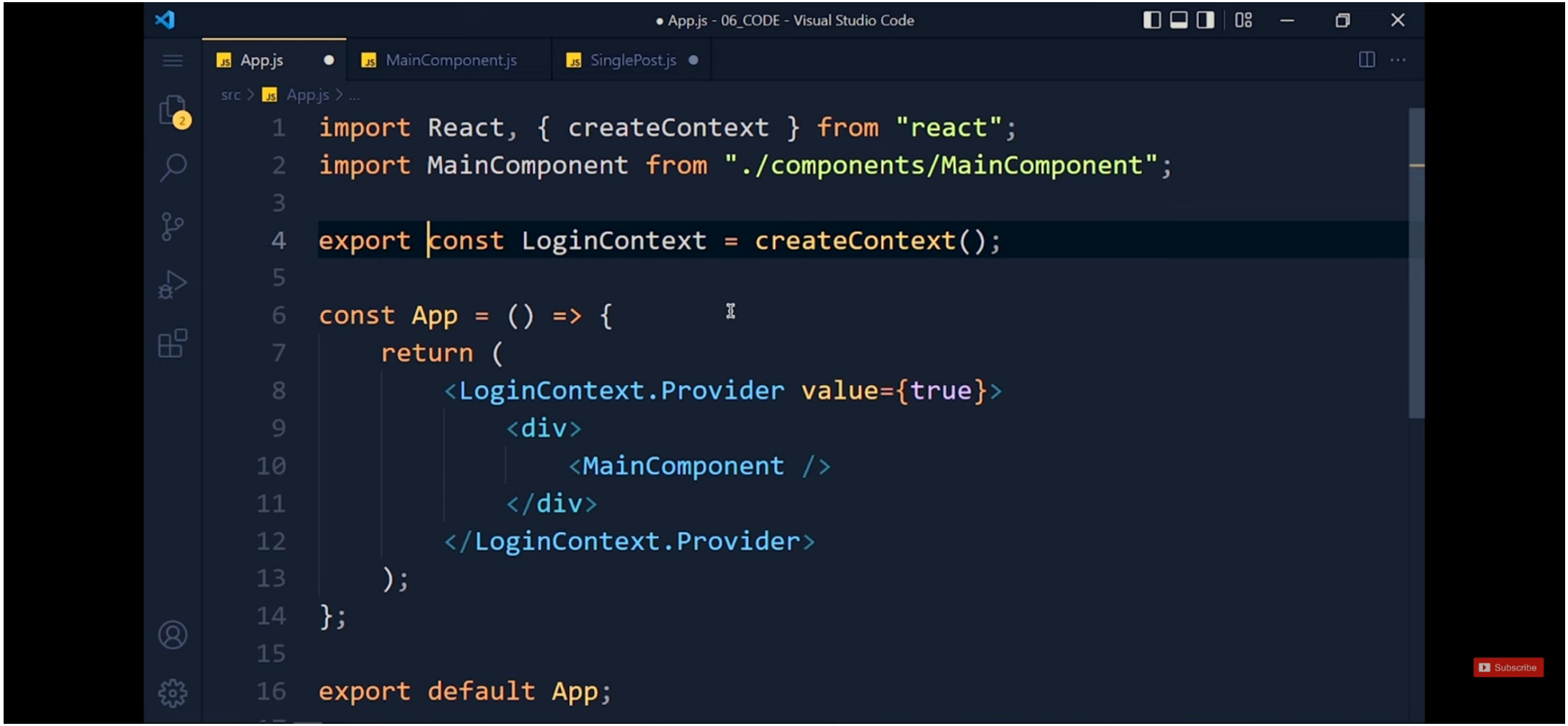
1. Creating the Context
2. Providing the Context
3. Consuming the Context


App.js - 06_CODE - Visual Studio Code
App.js
src > App.js > App
import React, { createContext } from "react";
import MainComponent from "./components/MainComponent";
const LoginContext = createContext();
const App = () => {
return (
<LoginContext.Provider value={true}>

<MainComponent />

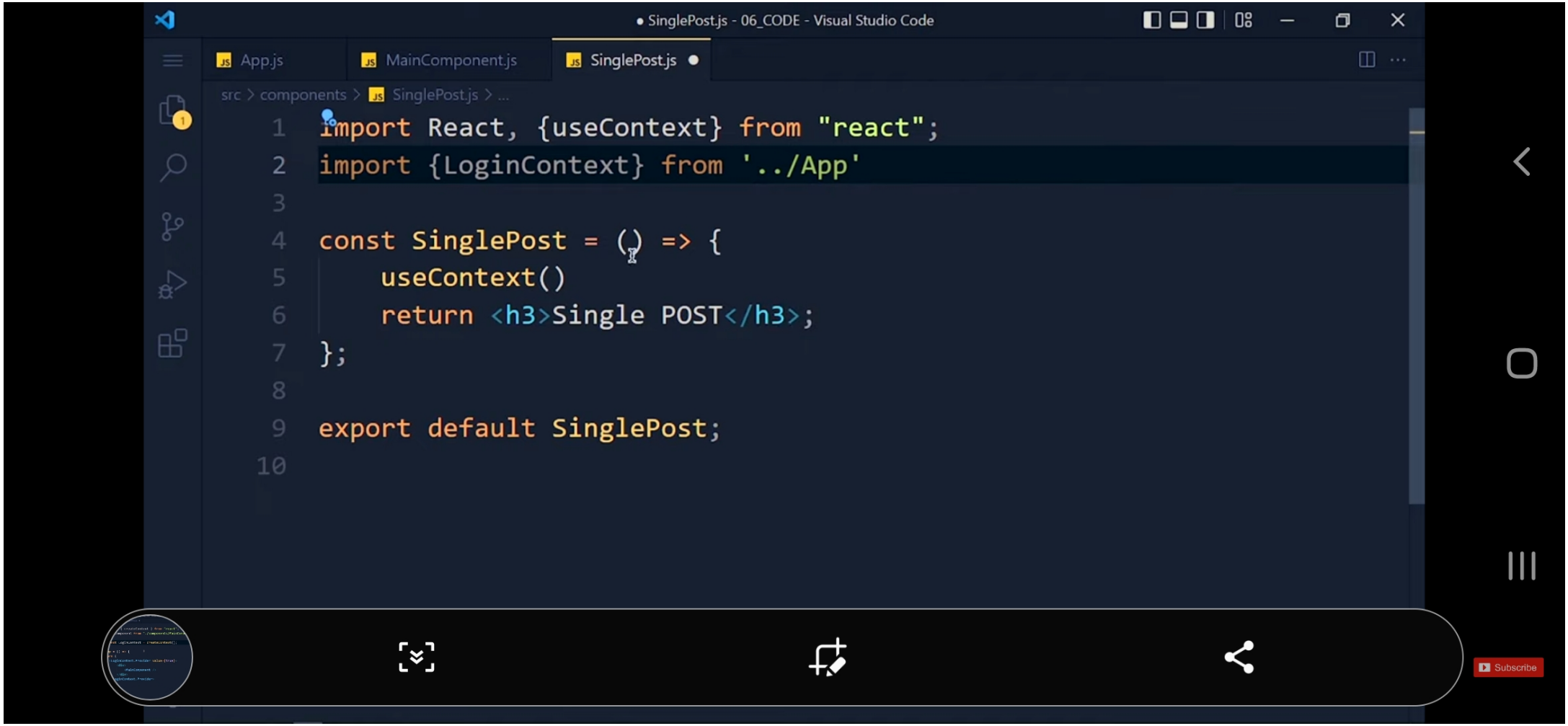
</LoginContext.Provider>
);
};
export default App;
Subscribe

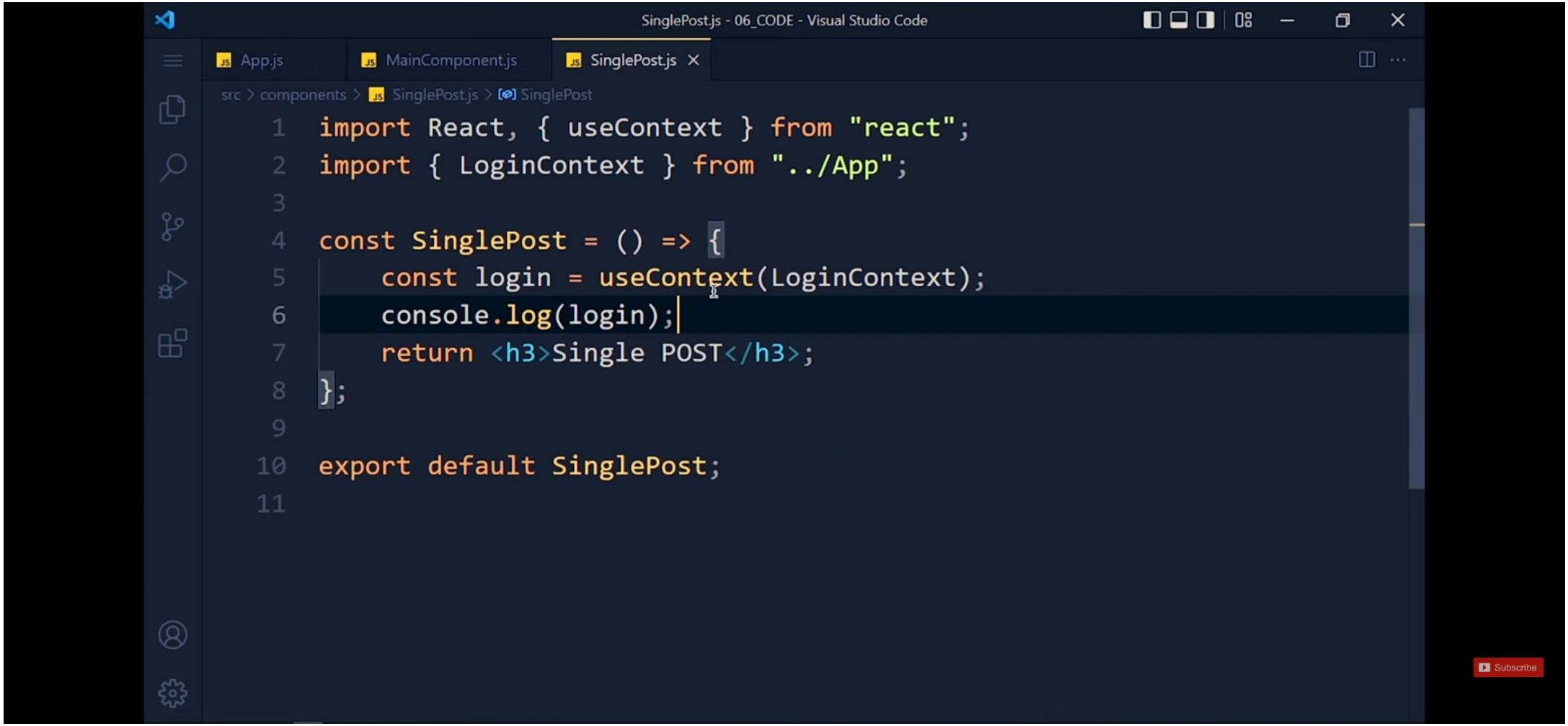

SinglePost.js - 06_CODE - Visual Studio Code
App.js
MainComponent.js
SinglePost.js
src > components > SinglePost.js > SinglePost
1 import React from "react";
2
3 const SinglePost = () => {
4 return <h3>Single POST</h3>;
5 };
6
7 export default SinglePost;
8
Subscribe

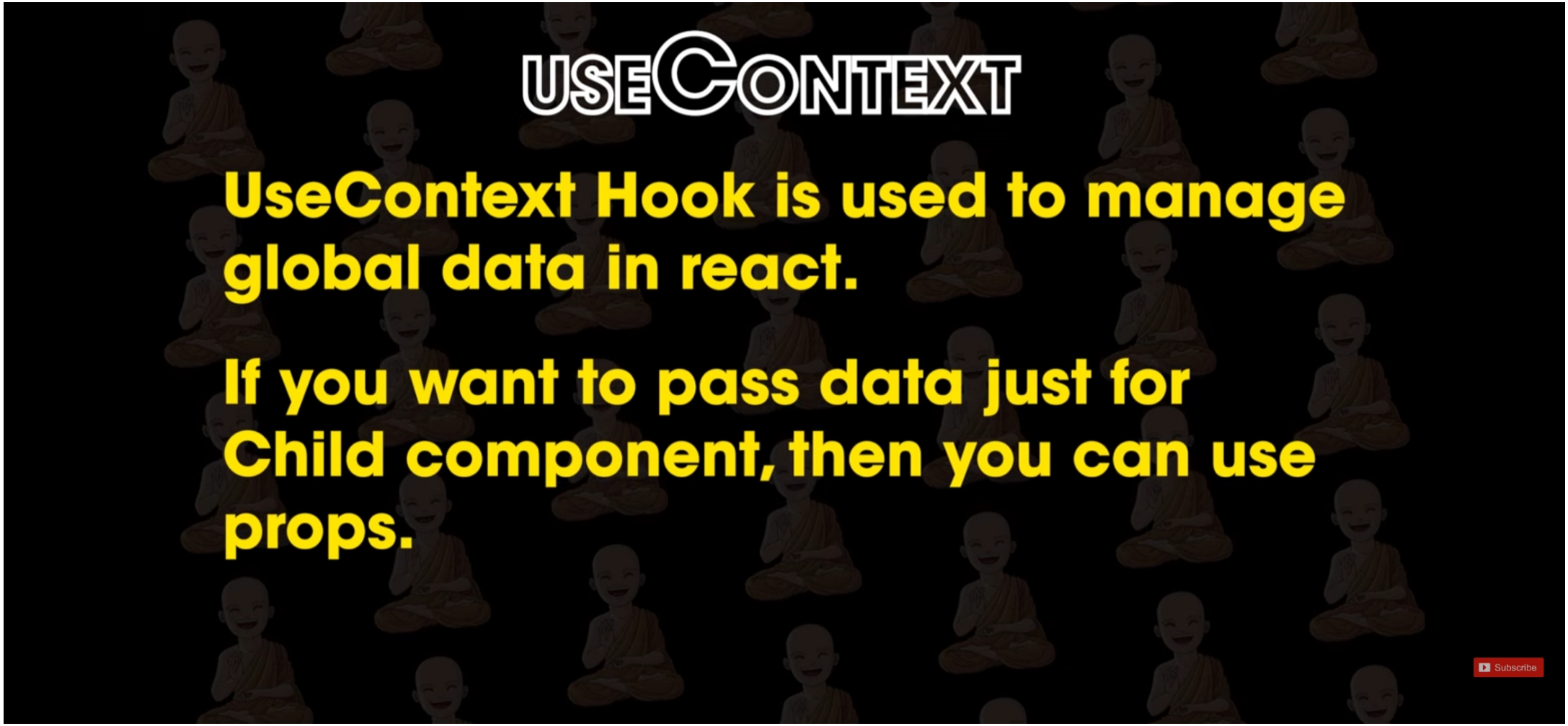

• App.js - 06_CODE - Visual Studio Code
App.js
MainComponent.js
SinglePost.js
src > App.js > ...
import React, { createContext } from "react";
import MainComponent from "./components/MainComponent";
export const LoginContext = createContext();
const App = () => {
return (
<LoginContext.Provider value={true}>

<MainComponent />

</LoginContext.Provider>
);
};
export default App;
Subscribe

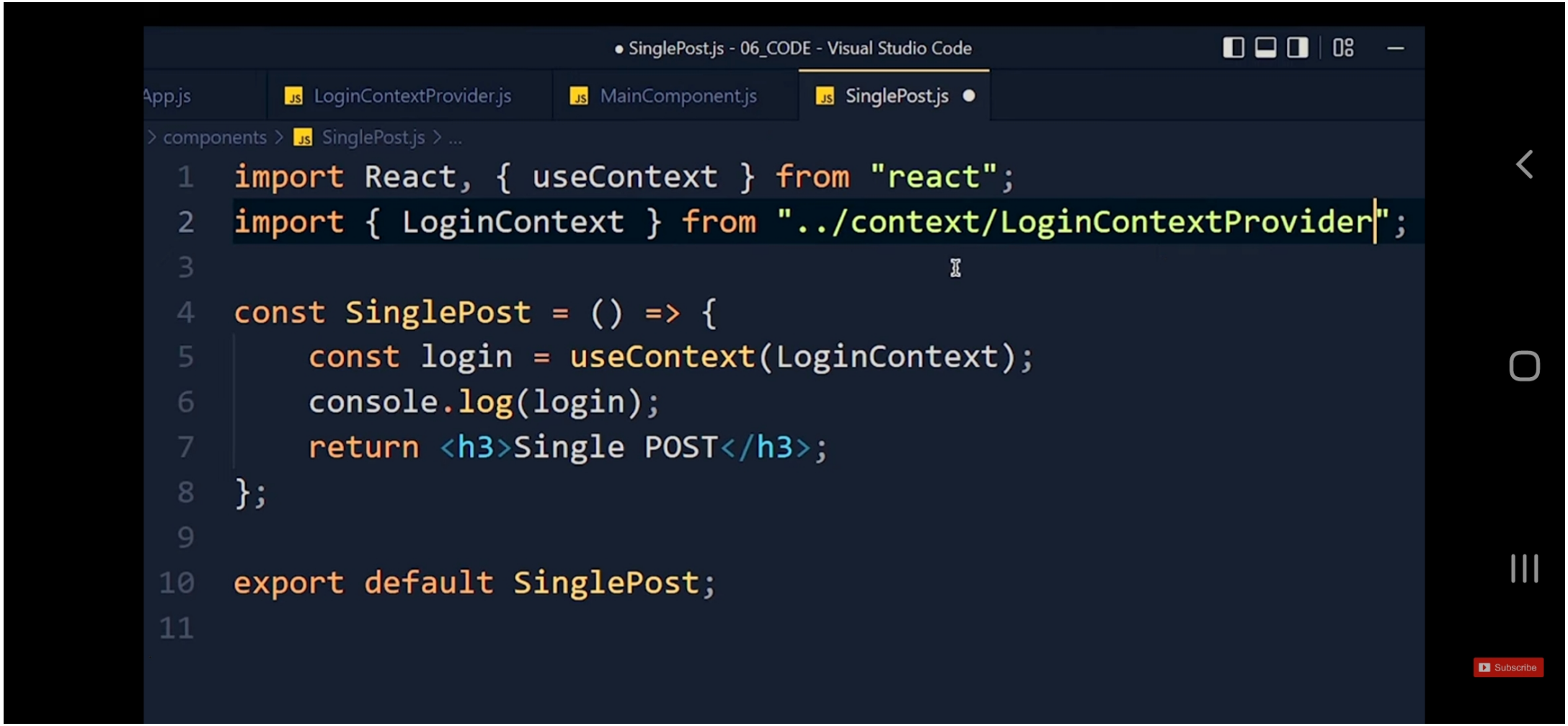

• SinglePost.js - 06_CODE - Visual Studio Code
App.js
MainComponent.js
SinglePost.js
src > components > SinglePost.js > ...
1 import React, {useContext} from "react";
2 import {LoginContext} from '../App'
3
4 const SinglePost = () => {
5 useContext()
6 return <h3>Single POST</h3>;
7 };
8
9 export default SinglePost;
10
Subscribe


SinglePost.js - 06_CODE - Visual Studio Code
App.js
MainComponent.js
SinglePost.js
src > components > SinglePost.js > SinglePost
1 import React, { useContext } from "react";
2 import { LoginContext } from "../App";
3
4 const SinglePost = () => {
5 const login = useContext(LoginContext);
6 console.log(login);
7 return <h3>Single POST</h3>;
8 };
9
10 export default SinglePost;
11
Subscribe


USECONTEXT
UseContext Hook is used to manage global data in react.
If you want to pass data just for Child component, then you can use props.
Subscribe

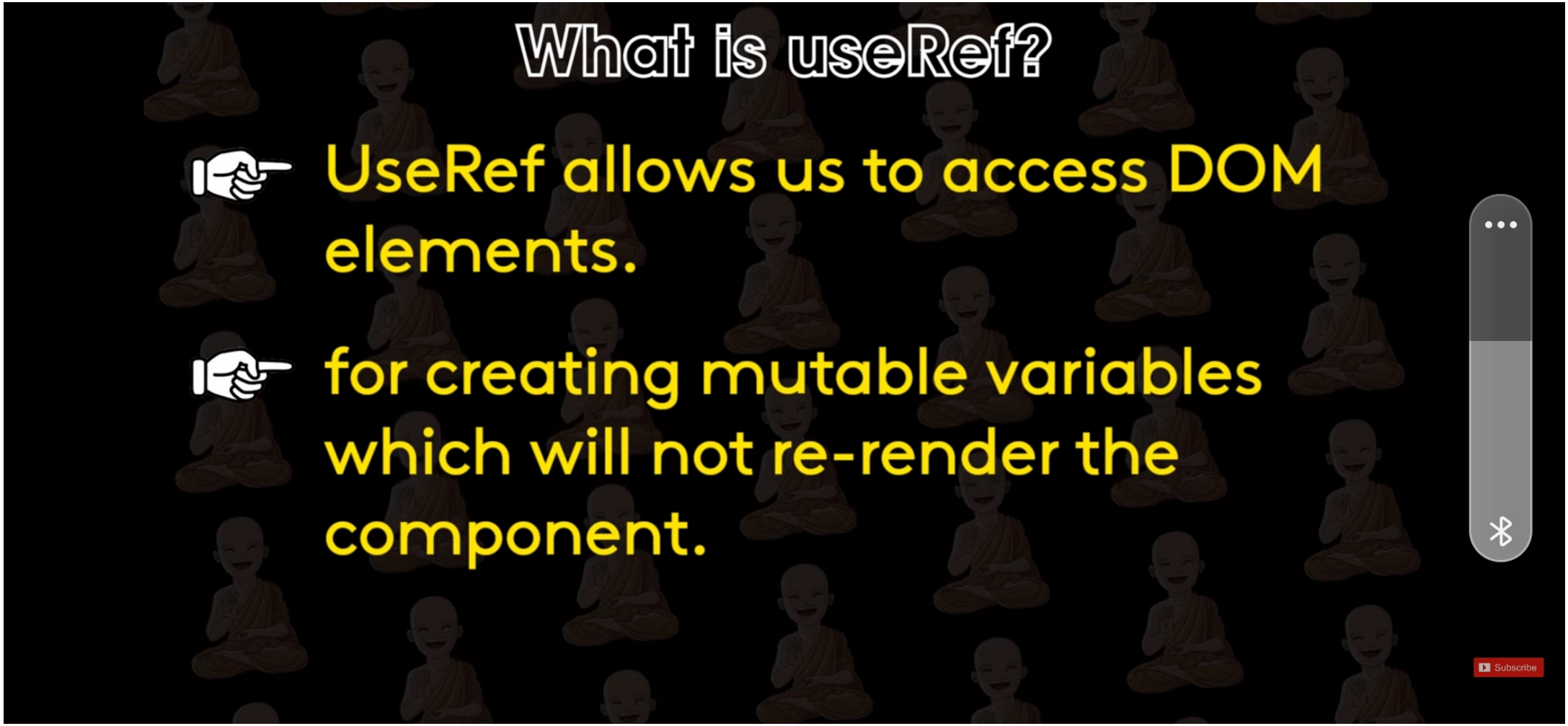

• App.js - 06_CODE - Visual Studio Code
App.js
LoginContextProvider.js
MainComponent.js
SinglePost.js
src > App.js > ...
1 import React, { createContext } from "react";
2 import MainComponent from "./components/MainComponent";
3
4 export const LoginContext = createContext();
5
6 const App = () => {
7 return (
8 <LoginContext.Provider value={ture}>
9
10 <MainComponent />
11
12 </LoginContext.Provider>
13);
14 };
15
16 export default App;
Subscribe


LoginContextProvider.js - 06_CODE - Visual Studio Code
App.js
LoginContextProvider.js
MainComponent.js
SinglePost.js
src > context > LoginContextProvider.js > LoginContextProvider
1 import React, { createContext, useState } from "react";
2
3 export const LoginContext = createContext();
4
5 const LoginContextProvider = ({ children }) => {
6 const [userDetails, setUserDetails] = useState(true);
7 return (
8 <LoginContext.Provider value={userDetails}>
9 {children}
10 </LoginContext.Provider>
11);
12 };
13
14 export default LoginContextProvider;
15
Subscribe


• SinglePost.js - 06_CODE - Visual Studio Code
App.js
LoginContextProvider.js
MainComponent.js
SinglePost.js
components > SinglePost.js > ...
1 import React, { useContext } from "react";
2 import { LoginContext } from "../context/LoginContextProvider";
3
4 const SinglePost = () => {
5 const login = useContext(LoginContext);
6 console.log(login);
7 return <h3>Single POST</h3>;
8 };
9
10 export default SinglePost;
11
Subscribe

React Hooks Number 04

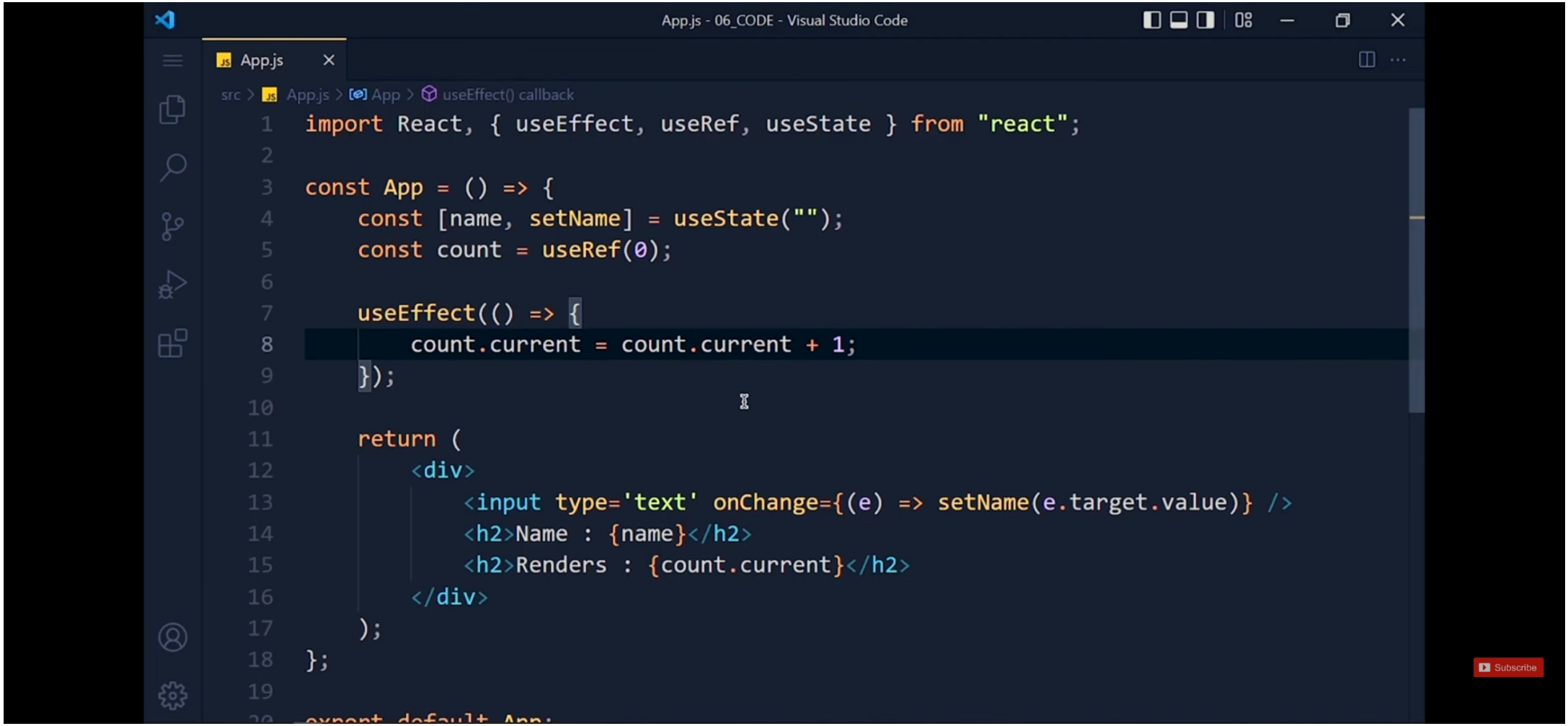
useRef Hook


What is useRef?
UseRef allows us to access DOM elements.
for creating mutable variables which will not re-render the component.
Subscribe

App.js

App.js > App

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

const App = () => {
    const [name, setName] = useState("");
    const [count, setCount] = useState(0);

    useEffect(() => {
        setCount((prev) => prev + 1);
    });

    return (
        <div>
            <input type='text' onChange={(e) => setName(e.target.value)} />
            <h2>Name : {name}</h2>
            <h2>Renders : {count}</h2>
        </div>
    );
```


App.js - 06_CODE - Visual Studio Code
App.js
src > App.js > App > useEffect() callback
import React, { useEffect, useRef, useState } from "react";
const App = () => {
const [name, setName] = useState("");
const count = useRef(0);
useEffect(() => {
count.current = count.current + 1;
});
return (

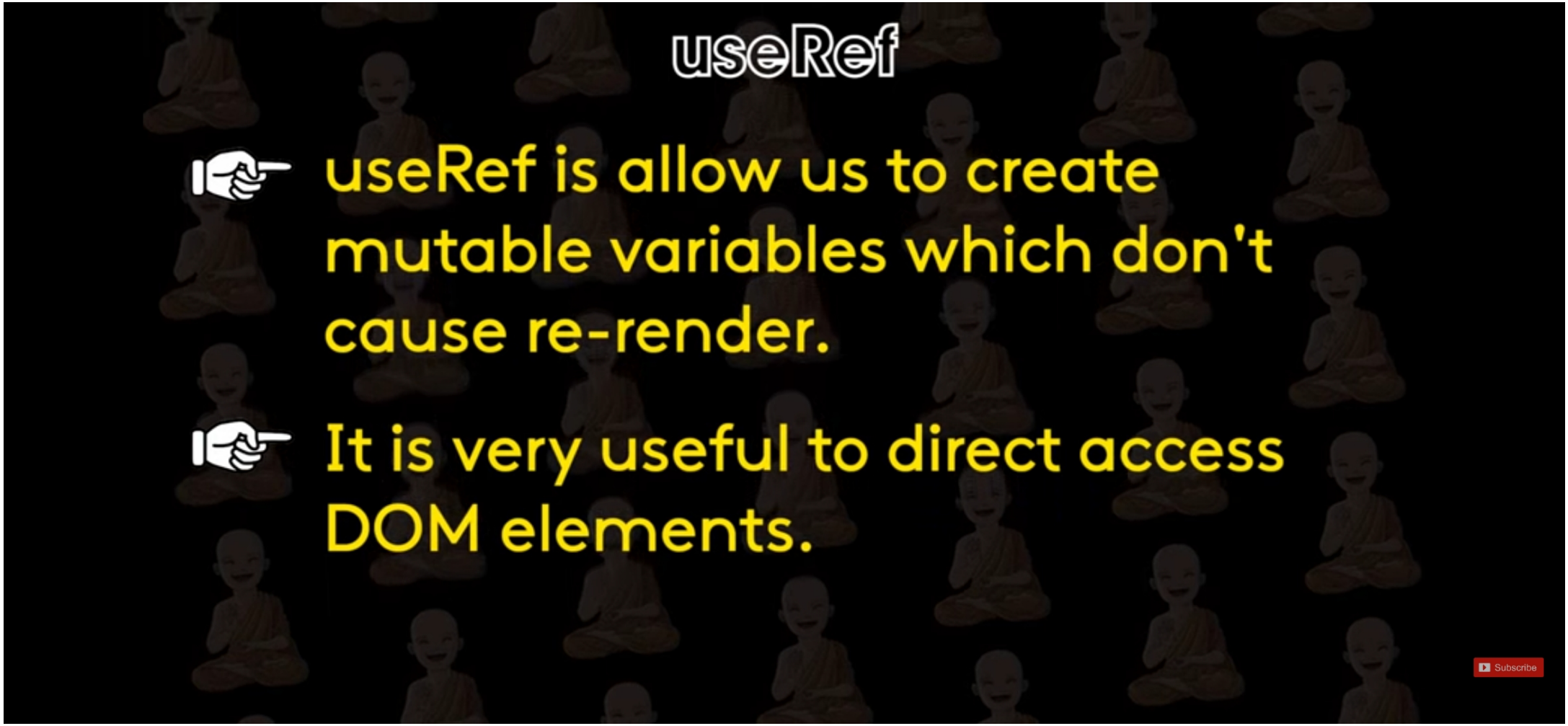
<input type='text' onChange={(e) => setName(e.target.value)} />
<h2>Name : {name}</h2>
<h2>Renders : {count.current}</h2>

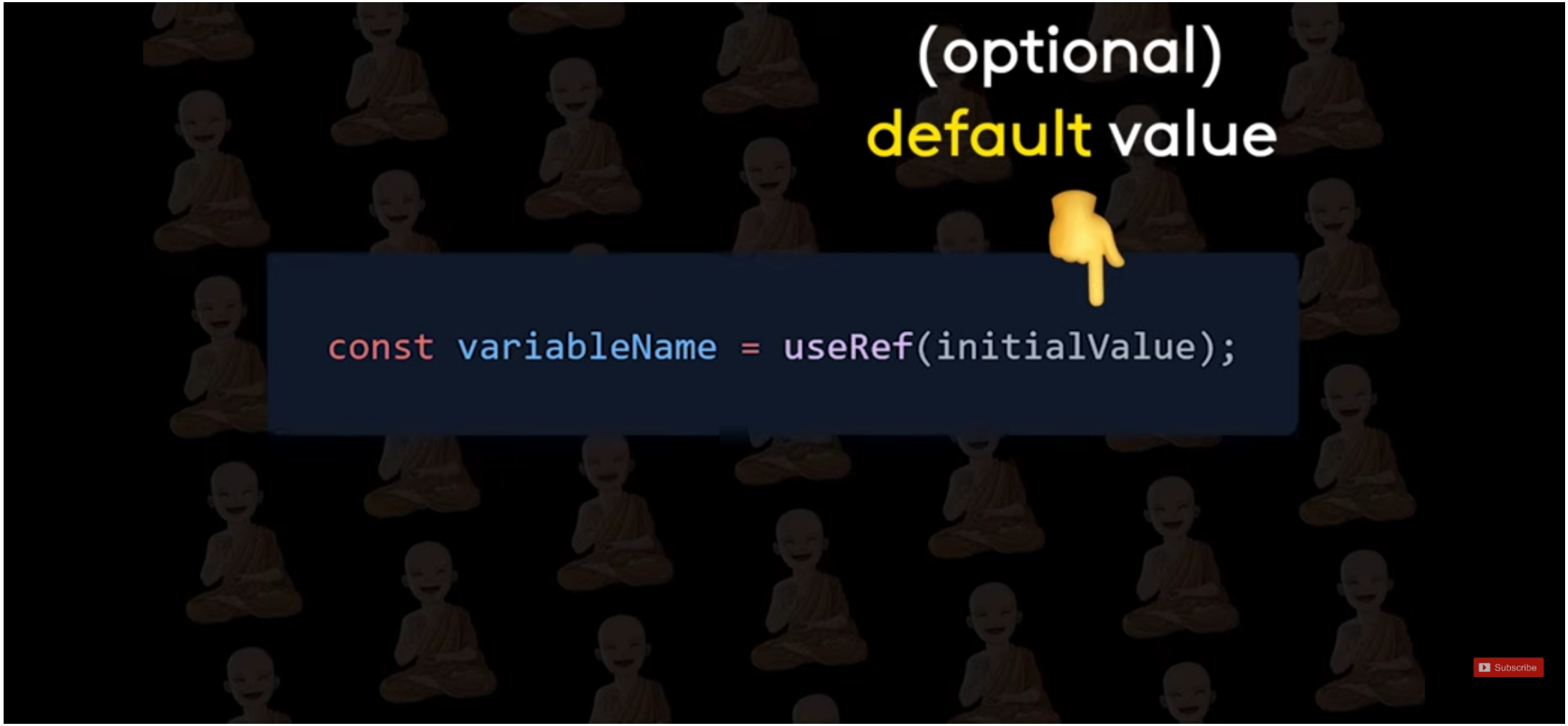
);
};
Subscribe

App.js

src > App.js > App > handleClick

```
1  import React, { useRef } from "react";
2
3  const App = () => {
4      const inputEle = useRef();
5      const handleClick = () => {
6          console.log(inputEle.current);
7          inputEle.current.style.width = "300px";
8          inputEle.current.focus();
9      };
10
11     return (
12         <div>
13             <input type='text' ref={inputEle} />
14             <button onClick={handleClick}>Click Here</button>
15         </div>
16     );
17 };
18
```


useRef
useRef is allow us to create mutable variables which don't cause re-render.
It is very useful to direct access DOM elements.
Subscribe


(optional)
default value
const variableName = useRef(initialValue);
Subscribe

React Hooks Number 05

useReducer Hook

What is useReducer?

useReducer is used to manage state in our react application.

In other words, useReducer works like a state management tool.

What is State Management?

State Management is used to manage all states of application in a simple way.

Always use the useReducer hook when you have a lot of states and methods to handle.

src > App.js > reducer

```
1  import React, { useReducer } from "react";
2
3  const initialState = { count: 0 };
4
5  const reducer = (state, action) => {
6      switch(action.type) {
7          case "increase":
8              return { count: state.count + 1 };
9
10         case "decrease":
11             return { count: state.count - 1 };
12
13         default:
14             return state;
15     }
16 };
17
```

```
13          default:
14              return state;
15      }
16  };
17
18  const App = () => {
19      const [state, dispatch] = useReducer(reducer, initialState);
20
21      const increaseCount = () => {
22          dispatch({ type: "increase" });
23      };
24
25      const decreaseCount = () => {
26          dispatch({ type: "decrease" });
27      };
28
29      return (
30          <div>
31              <h2>Count : {state.count}</h2>
32              <button onClick={increaseCount}>Increase</button>
```

App.js

src > App.js > reducer

```
1  import React, { useReducer } from "react";
2
3  const ACTION = {
4      INCREASE : "increase",
5      DECREASE : "decrease"
6  }
7
8  const initialState = { count: 0 };
9
10 const reducer = (state, action) => {
11     switch (action.type) {
12         case ACTION.INCREASE:
13             return { count: state.count + 1 };
14
15         case ACTION.DECREASE:
16             return { count: state.count - 1 };
17
```

App.js

src > App.js > ...

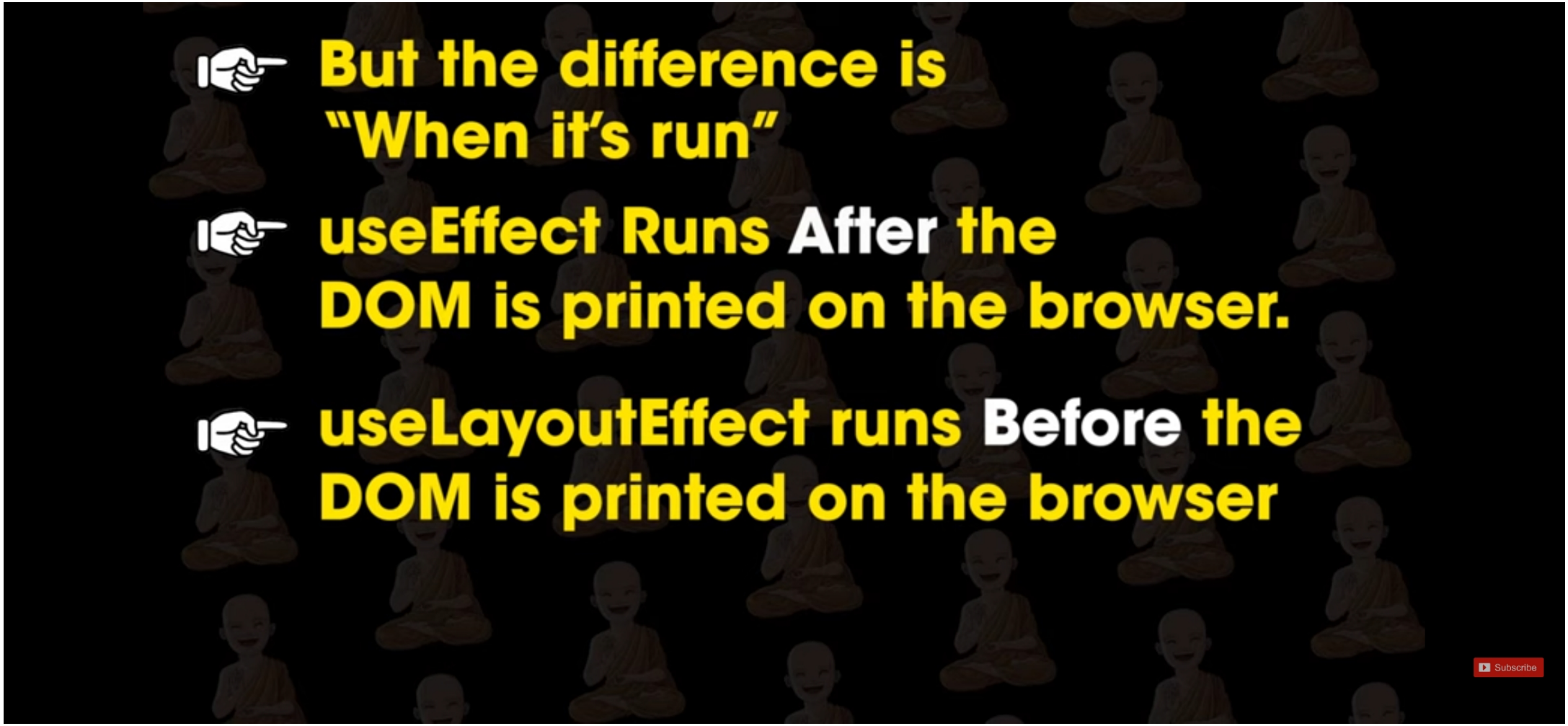
```jsx
import React, { useReducer } from "react";

const App = () => {
    const [state, dispatch] = useReducer(reducer, initialState);

    const increaseCount = () => {
        dispatch({ type: ACTION.INCREASE });
    };

    const decreaseCount = () => {
        dispatch({ type: ACTION.DECREASE });
    };

    return (
        <div>
            <h2>Count : {state.count}</h2>
            <button onClick={increaseCount}>Increase</button>
            <button onClick={decreaseCount}>Decrease</button>
        </div>
    );
};
```

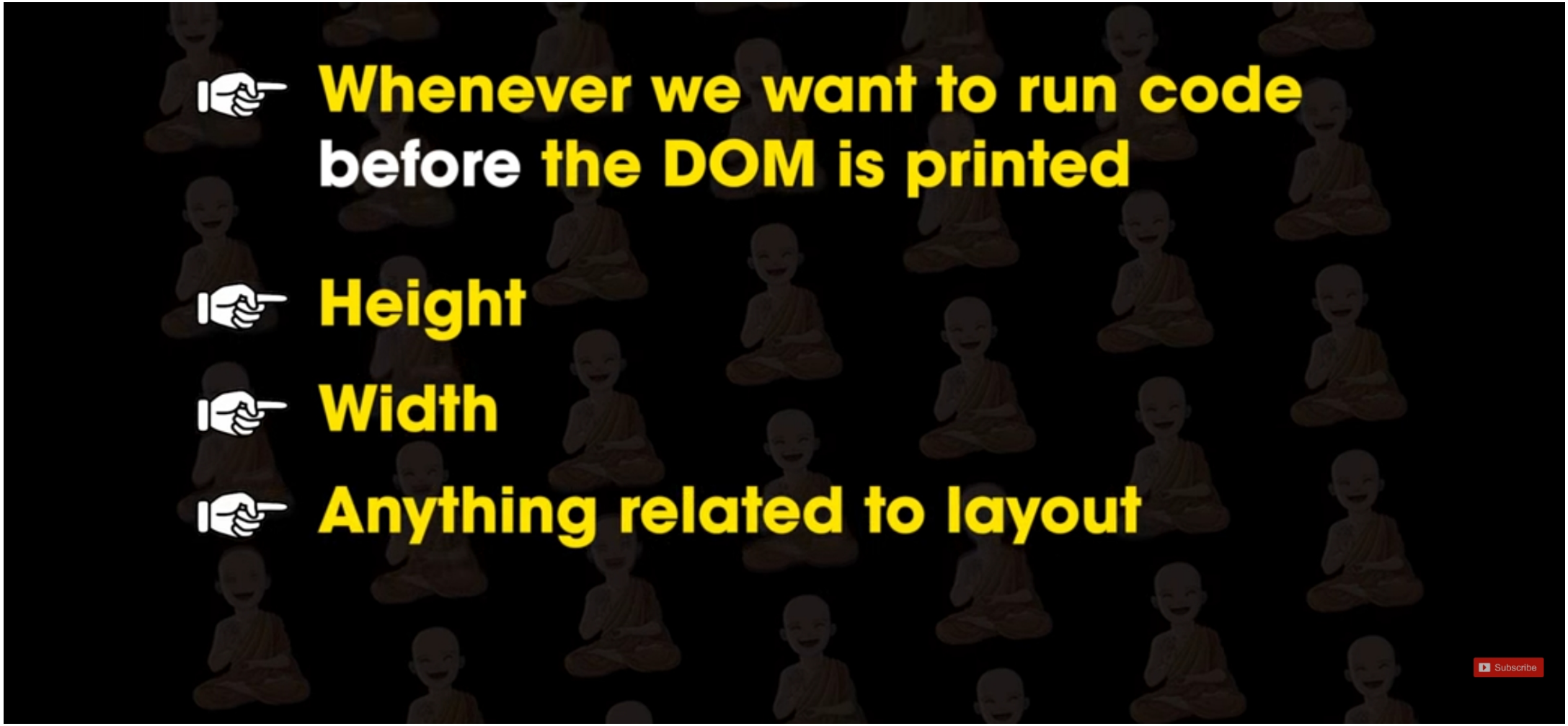
React Hooks Number 06

useLayoutEffect Hook

What is useLayoutEffect?

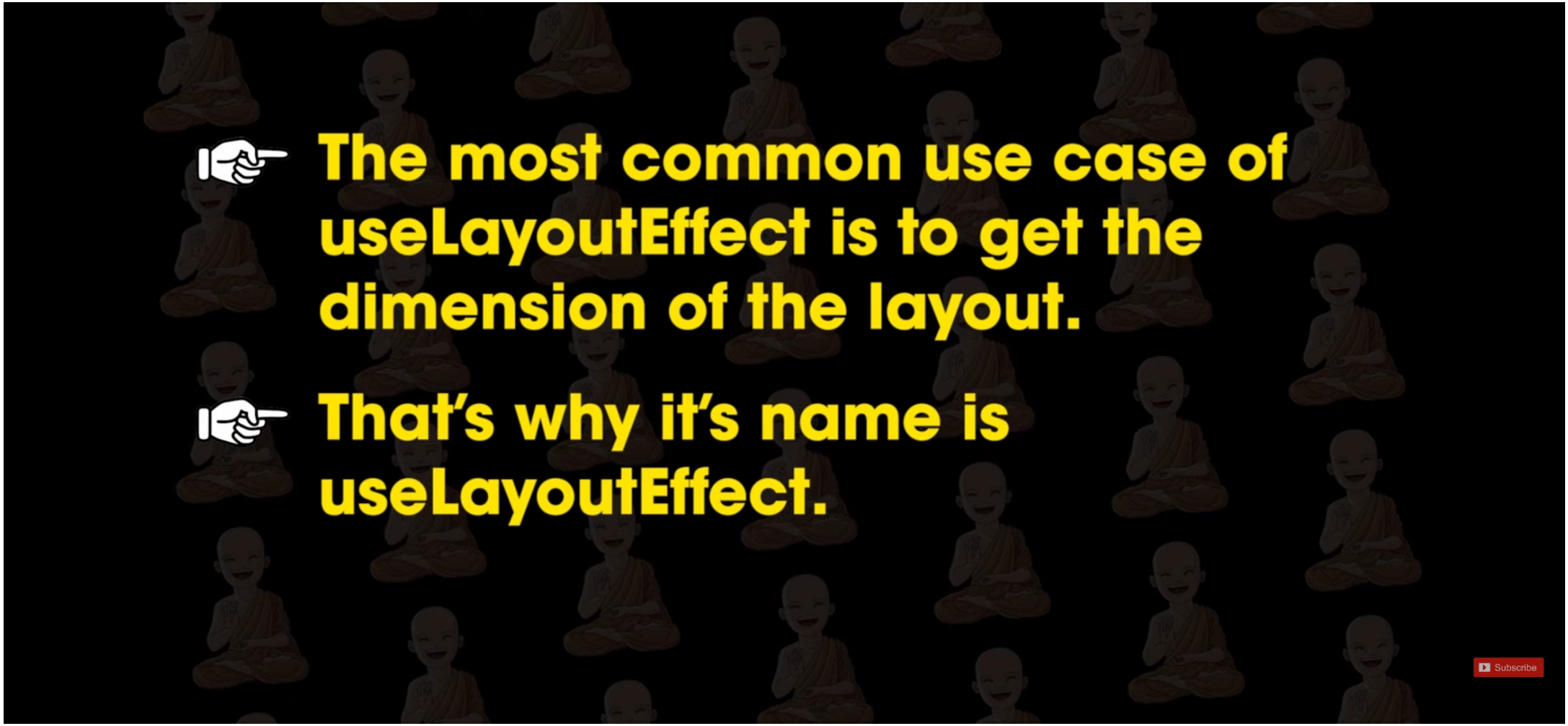
- useLayoutEffect works exactly the same as useEffect (Also the same syntax).
- But the difference is "When it's run"


But the difference is
"When it's run"
useEffect Runs After the
DOM is printed on the browser.
useLayoutEffect runs Before the
DOM is printed on the browser
Subscribe

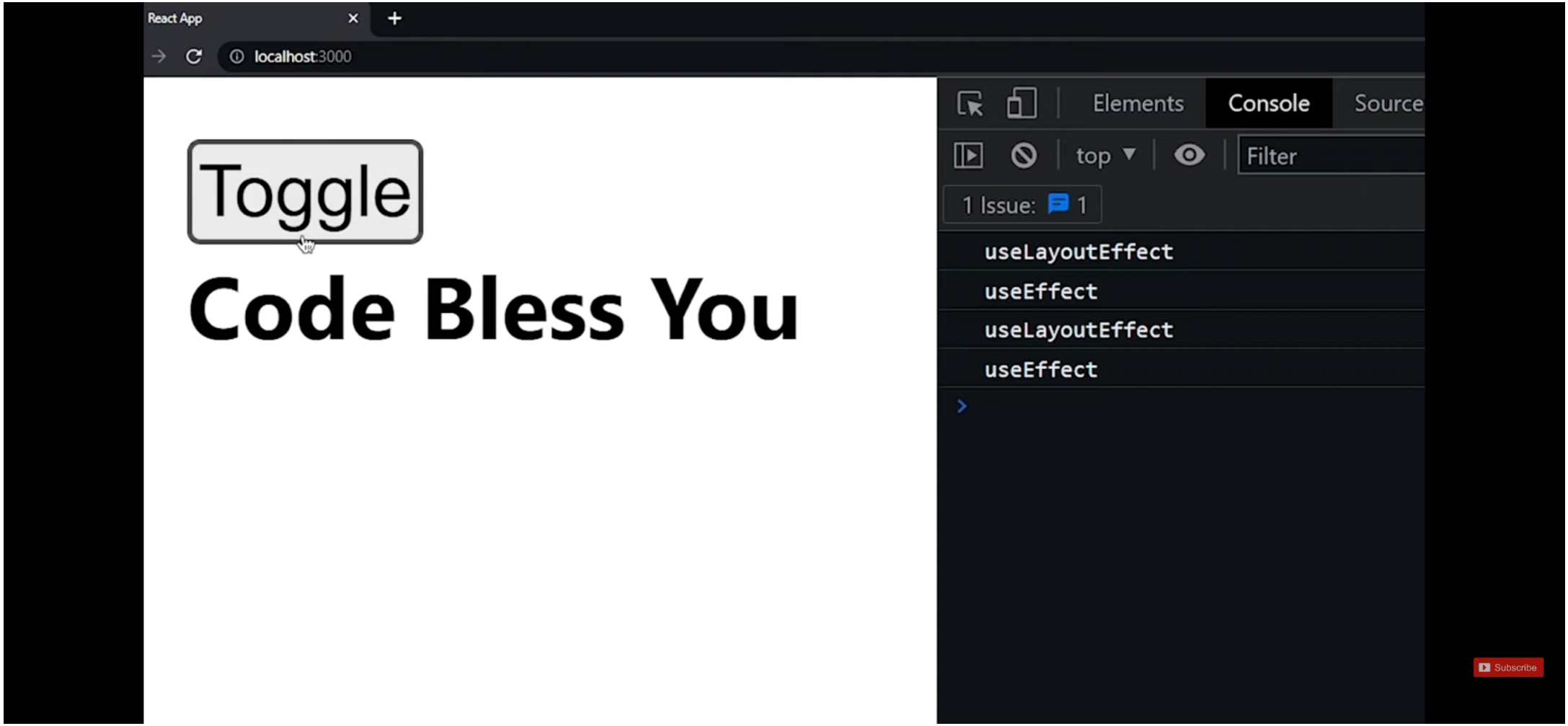

Whenever we want to run code before the DOM is printed
Height
Width
Anything related to layout
Subscribe

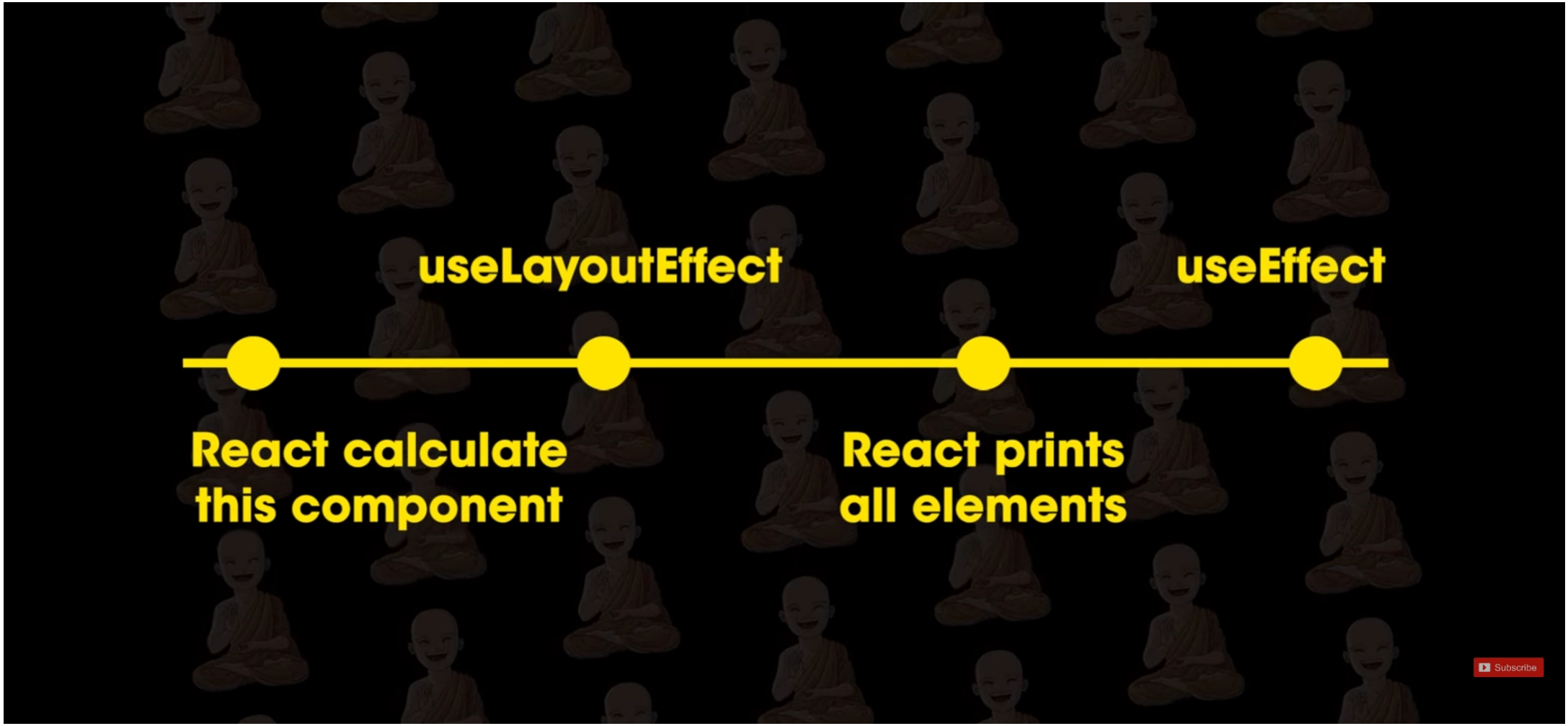
useLayoutEffect runs Synchronously

```
useLayoutEffect(() => {
  const dimension = textRef.current.getBoundingClientRect();
  textRef.current.style.paddingTop = `${dimension.height}px`;
}, [toggle]);
```

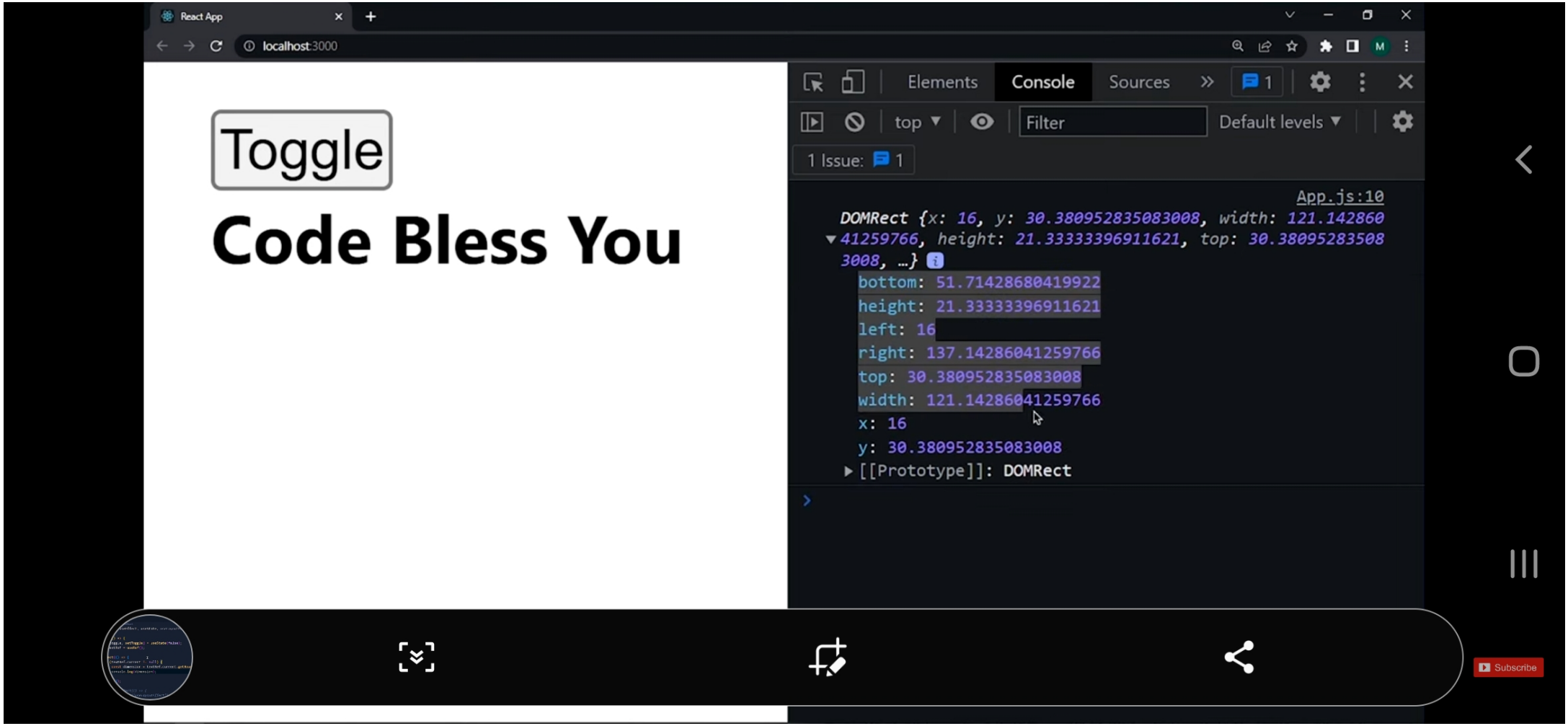

The most common use case of useLayoutEffect is to get the dimension of the layout.
That's why it's name is useLayoutEffect.
Subscribe

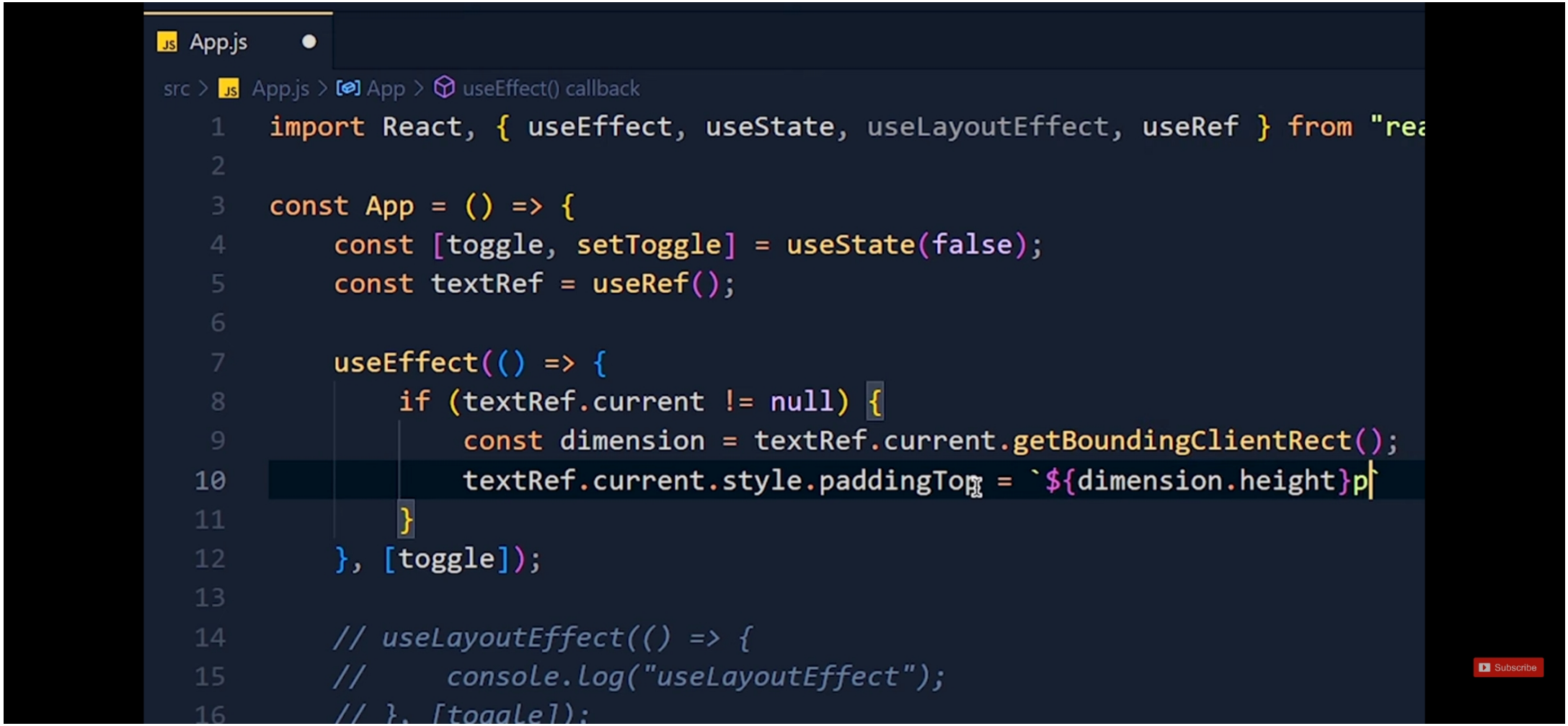

App.js
src > App.js > App
import React, { useEffect, useState, useLayoutEffect } from "react";
const App = () => {
const [toggle, setToggle] = useState(false);
useEffect(() => {
console.log("useEffect");
}, [toggle]);
useLayoutEffect(() => {
console.log("useLayoutEffect");
}, [toggle]);
return (
<>
<button onClick={() => setToggle(!toggle)}>Toggle</button>
{toggle && <h4>Code Bless You</h4>}
Subscribe

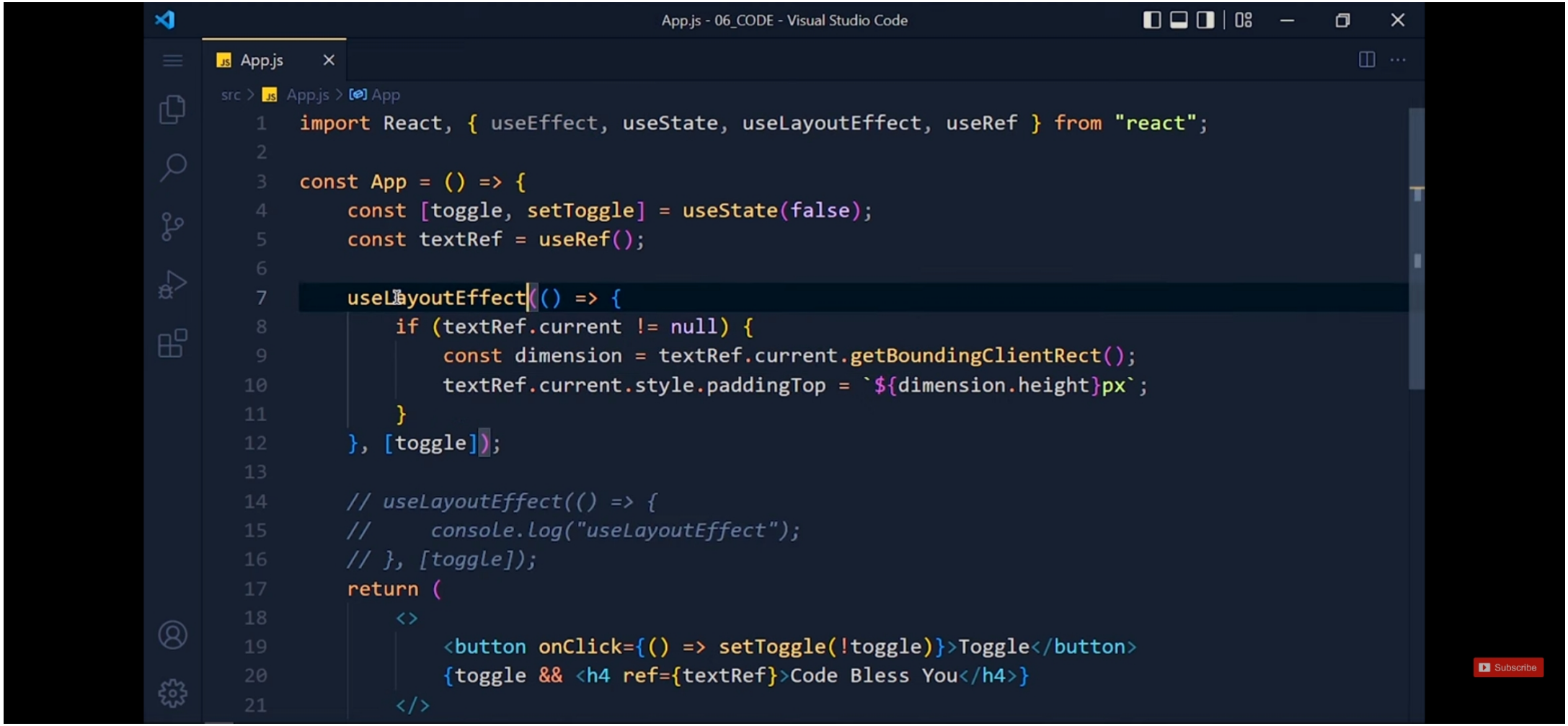

React App
localhost:3000
Toggle
Code Bless You
Elements
Console
Source
top
Filter
1 Issue: 1
useLayoutEffect
useEffect
useLayoutEffect
useEffect
Subscribe

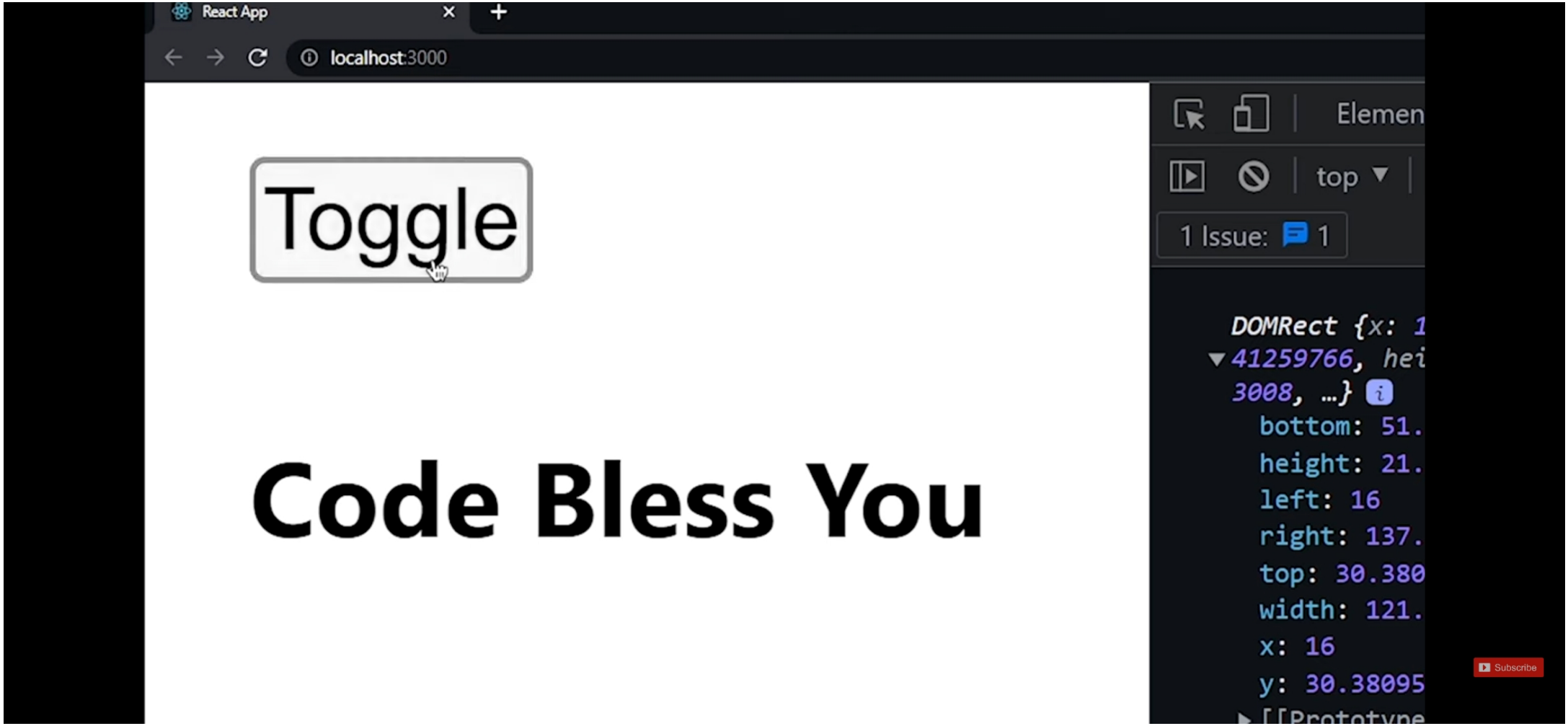

useLayoutEffect
useEffect
React calculate this component
React prints all elements
Subscribe

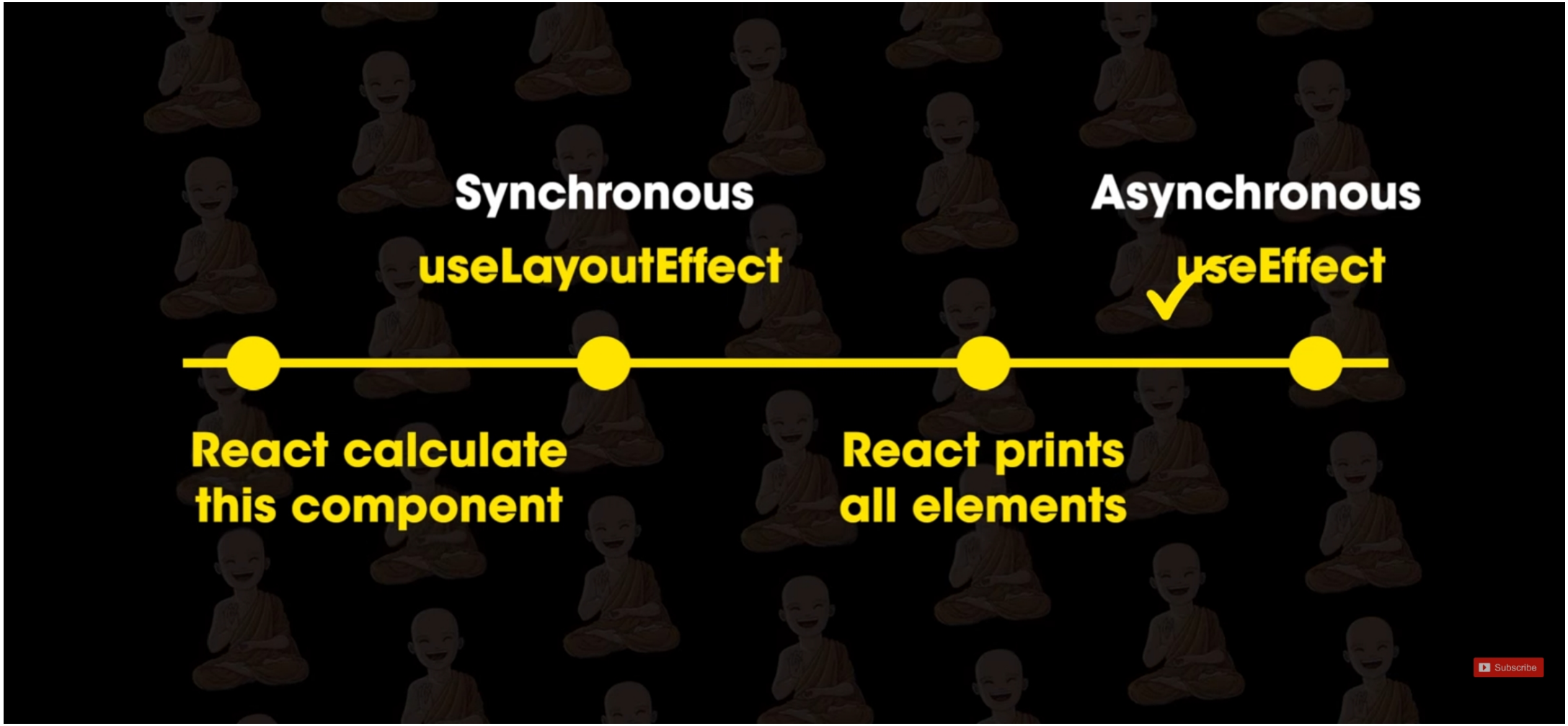

App.js
src > App.js > App > useEffect() callback
import React, { useEffect, useState, useLayoutEffect, useRef } from "rea
const App = () => {
const [toggle, setToggle] = useState(false);
const textRef = useRef();
useEffect(() => {
if (textRef.current != null) {
const dimension = textRef.current.getBoundingClientRect();
console.log(dimension);
}
}, [toggle]);
// useLayoutEffect(() => {
// console.log("useLayoutEffect");
// }, [toggle]);
Subscribe


React App
localhost:3000
Toggle
Code Bless You
Elements
Console
Sources
top
Filter
Default levels
1 Issue: 1
App.js:10
DOMRect {x: 16, y: 30.380952835083008, width: 121.142860
41259766, height: 21.33333396911621, top: 30.38095283508
3008, …}
bottom: 51.71428680419922
height: 21.33333396911621
left: 16
right: 137.14286041259766
top: 30.380952835083008
width: 121.14286041259766
x: 16
y: 30.380952835083008
[[Prototype]]: DOMRect
Subscribe


App.js
src > App.js > App > useEffect() callback
import React, { useEffect, useState, useLayoutEffect, useRef } from "rea
const App = () => {
const [toggle, setToggle] = useState(false);
const textRef = useRef();
useEffect(() => {
if (textRef.current != null) {
const dimension = textRef.current.getBoundingClientRect();
textRef.current.style.paddingTop = `${dimension.height}p
}
}, [toggle]);
// useLayoutEffect(() => {
// console.log("useLayoutEffect");
// }, [toggle]);
Subscribe


App.js - 06_CODE - Visual Studio Code
App.js
src > App.js > App
import React, { useEffect, useState, useLayoutEffect, useRef } from "react";
const App = () => {
const [toggle, setToggle] = useState(false);
const textRef = useRef();
useLayoutEffect(() => {
if (textRef.current != null) {
const dimension = textRef.current.getBoundingClientRect();
textRef.current.style.paddingTop = `${dimension.height}px`;
}
}, [toggle]);
// useLayoutEffect(() => {
// console.log("useLayoutEffect");
// }, [toggle]);
return (
<>
<button onClick={() => setToggle(!toggle)}>Toggle</button>
{toggle && <h4 ref={textRef}>Code Bless You</h4>}
</>
Subscribe


React App
localhost:3000
Toggle
Code Bless You
Elemen
top
1 Issue: 1
DOMRect {x: 1
41259766, hei
3008, …}
bottom: 51.
height: 21.
left: 16
right: 137.
top: 30.380
width: 121.
x: 16
y: 30.38095
Subscribe


Synchronous
useLayoutEffect
Asynchronous
useEffect
React calculate this component
React prints all elements
Subscribe

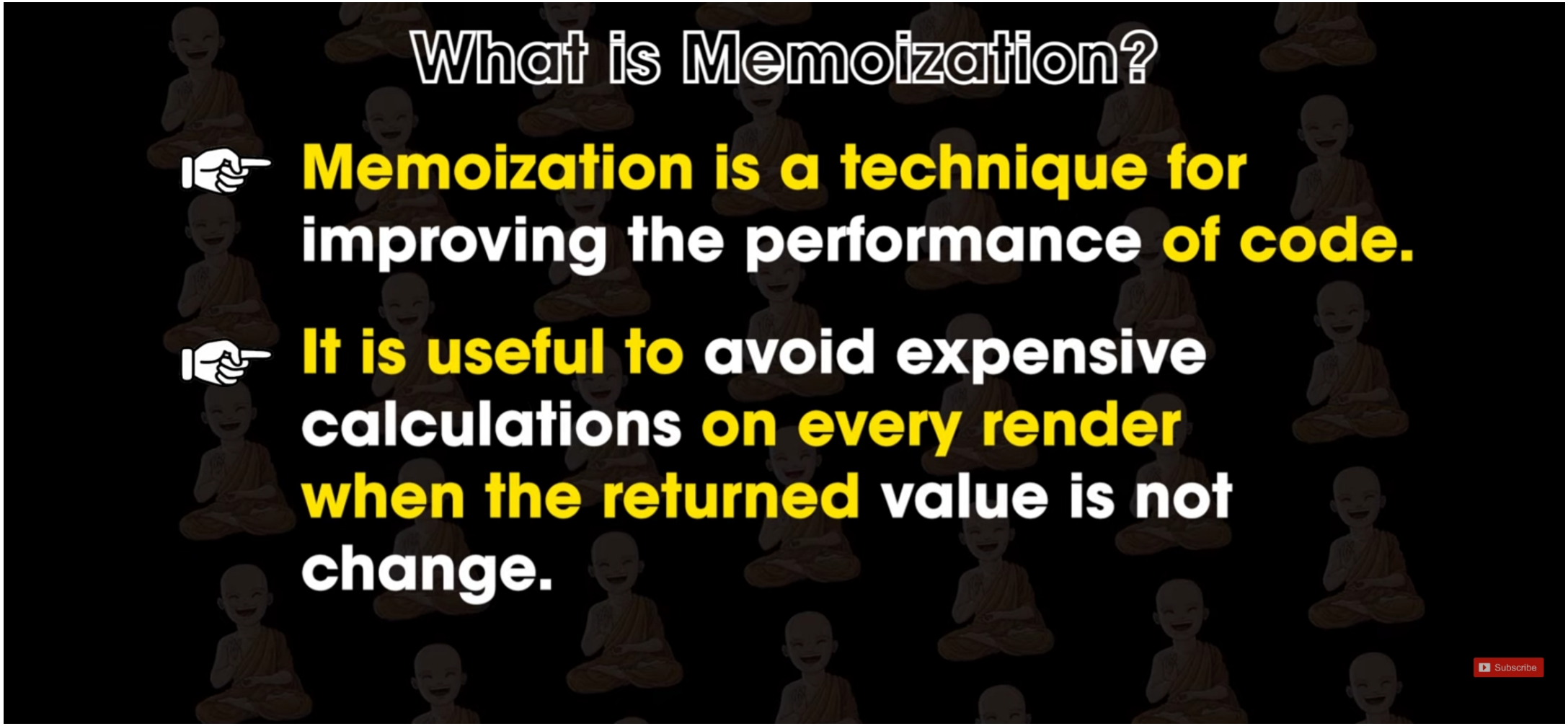
React Hooks Number 07

useMemo Hook

Subscribe

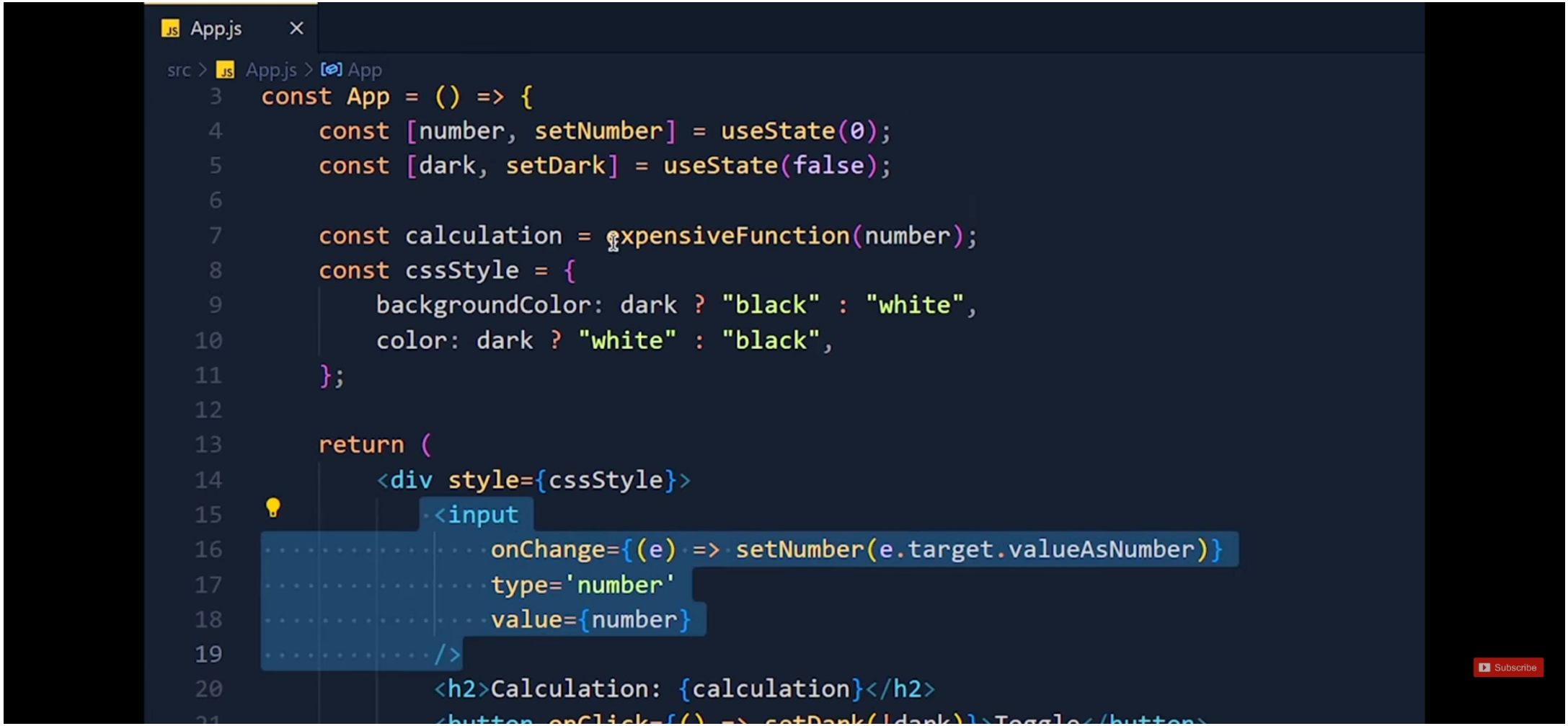
What is useMemo?

useMemo hook is used to apply Memoization in React.


What is Memoization?
Memoization is a technique for improving the performance of code.
It is useful to avoid expensive calculations on every render when the returned value is not change.
Subscribe

App.js > App

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

const App = () => {
    const [number, setNumber] = useState(0);
    const [dark, setDark] = useState(false);

    const calculation = expensiveFunction(number);
    const cssStyle = {
        backgroundColor: dark ? "black" : "white",
        color: dark ? "white" : "black",
    };

    return (
        <div style={cssStyle}>
            <input
                onChange={(e) => setNumber(e.target.valueAsNumber)}
```


App.js
src > App.js > App
const App = () => {
const [number, setNumber] = useState(0);
const [dark, setDark] = useState(false);
const calculation = expensiveFunction(number);
const cssStyle = {
backgroundColor: dark ? "black" : "white",
color: dark ? "white" : "black",
};
return (

<input
onChange={(e) => setNumber(e.target.valueAsNumber)}
type='number'
value={number}
/>
<h2>Calculation: {calculation}</h2>
Subscribe


App.js
src > App.js > expensiveFunction
/>
<h2>Calculation: {calculation}</h2>
<button onClick={() => setDark(!dark)}>Toggle</button>

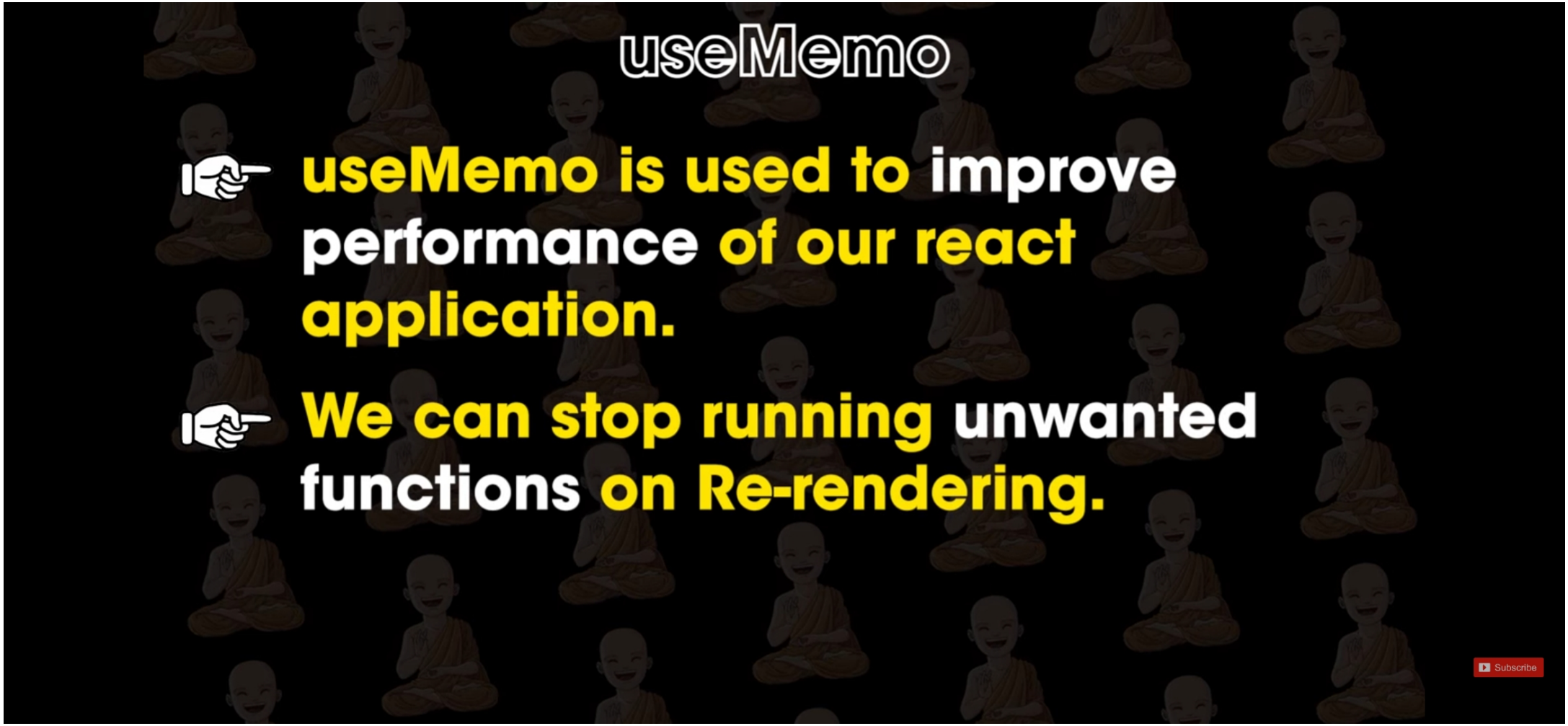
);
};
function expensiveFunction(num) {
console.log("Loop Started");
for (let i = 0; i < 1000000000; i++) {}
return num;
}
export default App;
Subscribe

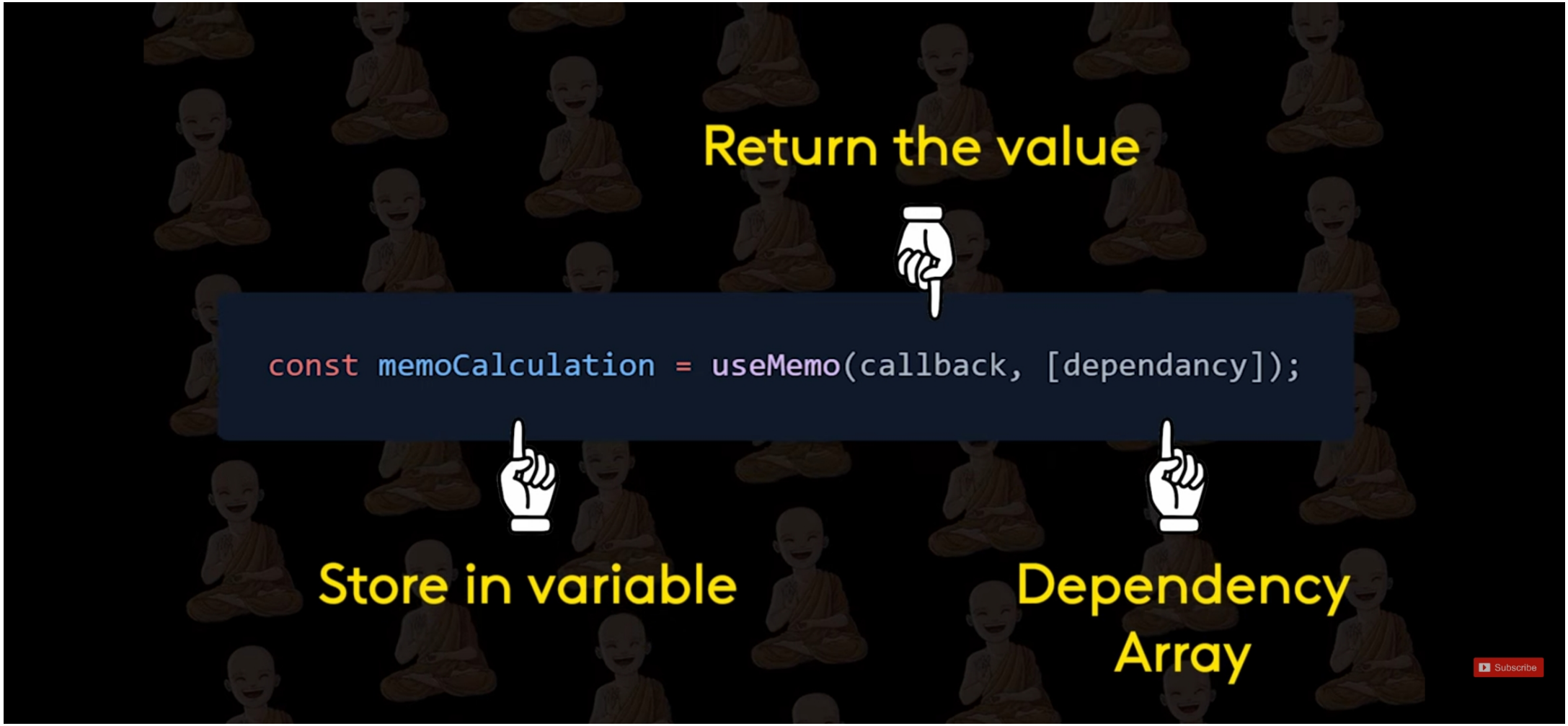
App.js

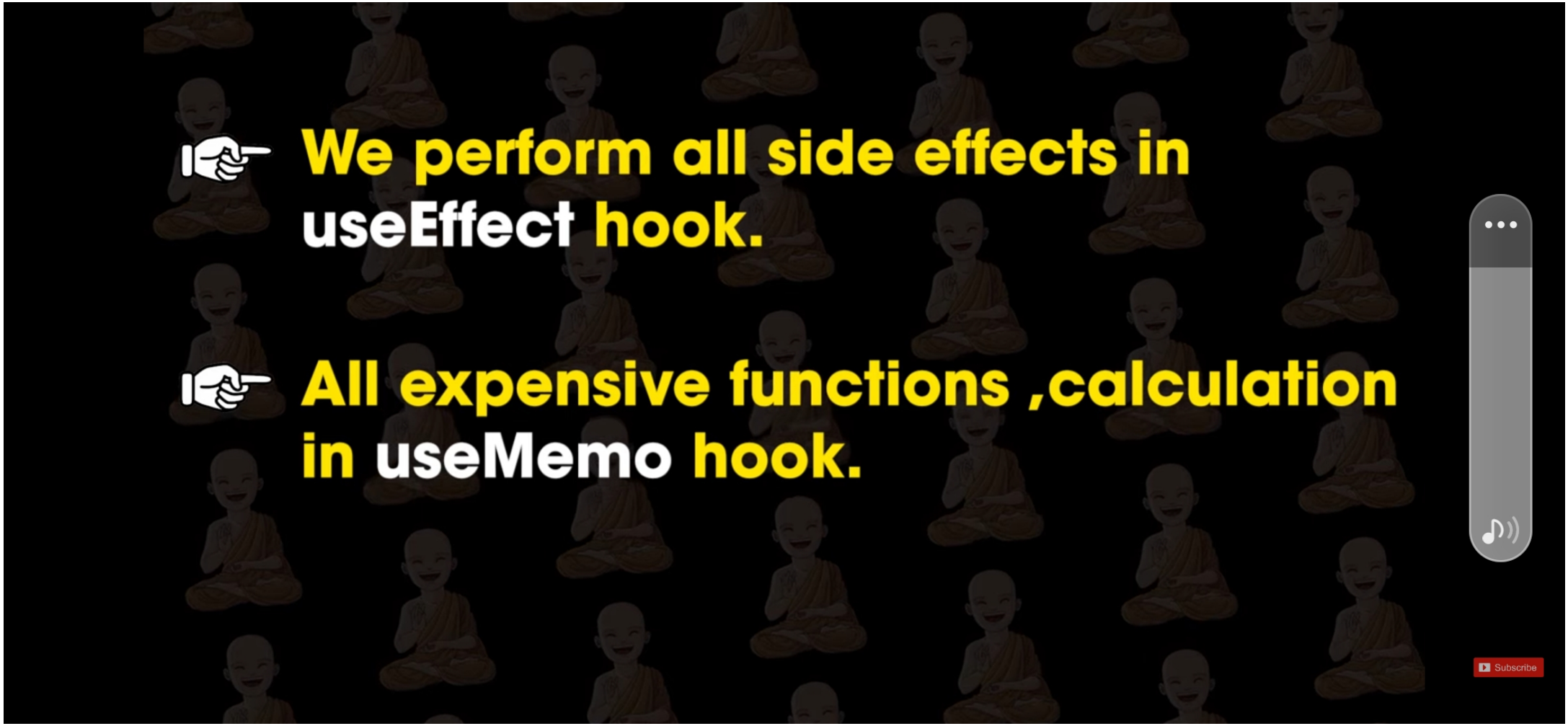
src > App.js > App > memoCalculation

```
1  import React, { useState, useMemo } from "react";
2
3  const App = () => {
4      const [number, setNumber] = useState(0);
5      const [dark, setDark] = useState(false);
6
7      const memoCalculation = useMemo(() => {
8          return expensiveFunction(number)
9      }, [number])
10
11     const calculation = expensiveFunction(number);
12     const cssStyle = {
13         backgroundColor: dark ? "black" : "white",
14         color: dark ? "white" : "black",
15     };
16
17     return (
18         <div style={cssStyle}>
```


App.js
src > App.js > App
10
11 const cssStyle = {
12 backgroundColor: dark ? "black" : "white",
13 color: dark ? "white" : "black",
14 };
15
16 return (
17
18 <input
19 onChange={(e) => setNumber(e.target.valueAsNumber)}
20 type='number'
21 value={number}
22 />
23 <h2>Calculation: {calculation}</h2>
24 <button onClick={() => setDark(!dark)}>Toggle</button>
25
26);
27 };
Subscribe


useMemo
useMemo is used to improve performance of our react application.
We can stop running unwanted functions on Re-rendering.
Subscribe


Return the value
const memoCalculation = useMemo(callback, [dependancy]);
Store in variable
Dependency Array
Subscribe


We perform all side effects in useEffect hook.
All expensive functions ,calculation in useMemo hook.
Subscribe

React Hooks Number 08

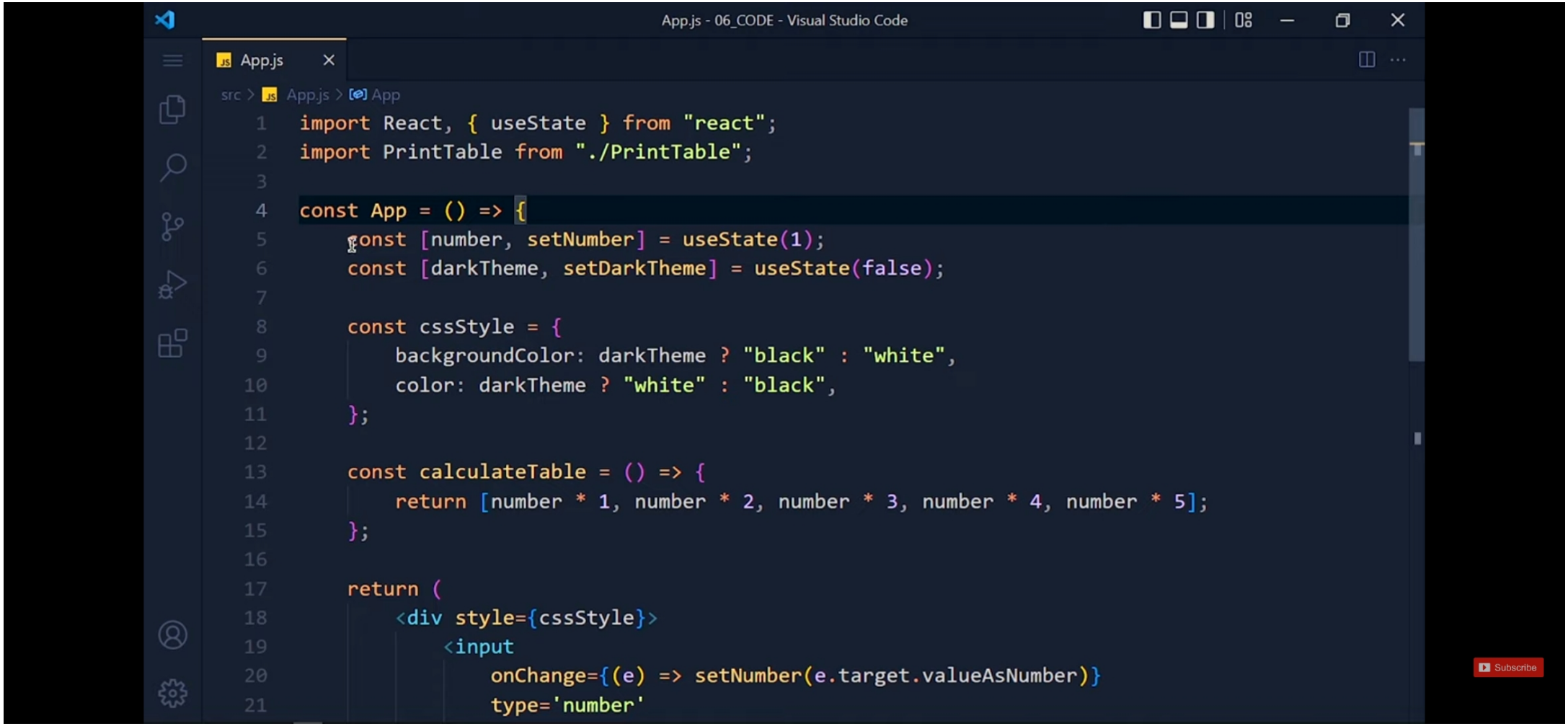
useCallback Hook

Subscribe

What is useCallback?

useCallback is used to return Memoize function.

It's also useful for preventing functions from being re-created on re-rendering.


App.js - 06_CODE - Visual Studio Code
App.js
src > App.js > App
import React, { useState } from "react";
import PrintTable from "./PrintTable";
const App = () => {
const [number, setNumber] = useState(1);
const [darkTheme, setDarkTheme] = useState(false);
const cssStyle = {
backgroundColor: darkTheme ? "black" : "white",
color: darkTheme ? "white" : "black",
};
const calculateTable = () => {
return [number * 1, number * 2, number * 3, number * 4, number * 5];
};
return (

<input
onChange={(e) => setNumber(e.target.valueAsNumber)}
type='number'
Subscribe


App.js
src > App.js > App > darkTheme
1 import React, { useState } from "react";
2 import PrintTable from "./PrintTable";
3
4 const App = () => {
5 const [number, setNumber] = useState(1);
6 const [darkTheme, setDarkTheme] = useState(false);
8 const cssStyle = {
9 backgroundColor: darkTheme ? "black" : "white",
10 color: darkTheme ? "white" : "black",
11 };
12
13 const calculateTable = () => {
14 return [number * 1, number * 2, number * 3, number * 4, number * 5];
15 };
16
17 return (
Subscribe

App.js

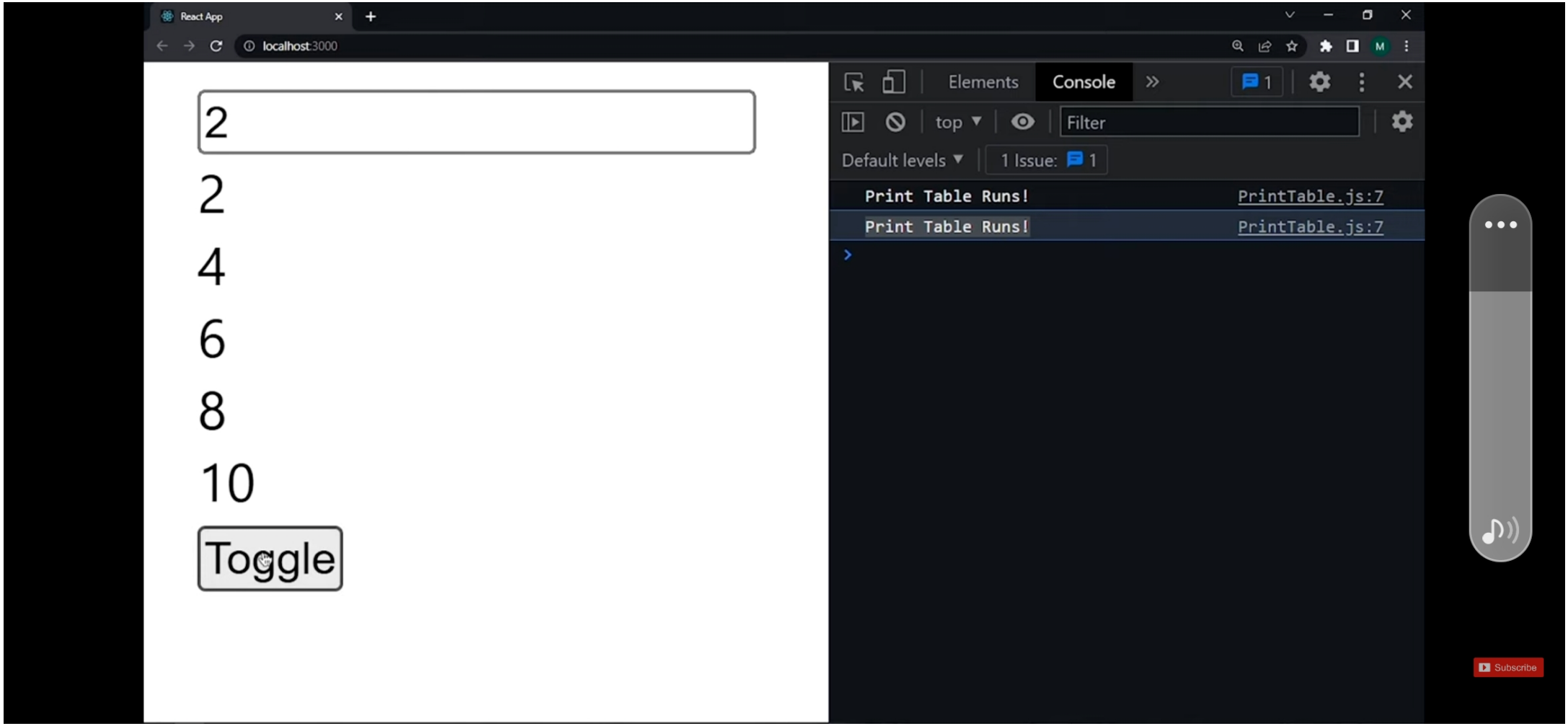
src > App.js > App

```
10          color: darkTheme ? "white" : "black",
11      };
12
13      const calculateTable = () => {
14          return [number * 1, number * 2, number * 3, number * 4, number * 5];
15      };
16
17      return (
18          <div style={cssStyle}>
19              <input
20                  onChange={(e) => setNumber(e.target.valueAsNumber)}
21                  type='number'
22                  value={number}
23              />
24              <PrintTable calculateTable={calculateTable} />
25              <button onClick={() => setDarkTheme(!darkTheme)}>Toggle</button>
26          </div>
```

App.js | PrintTable.js

src > PrintTable.js > PrintTable

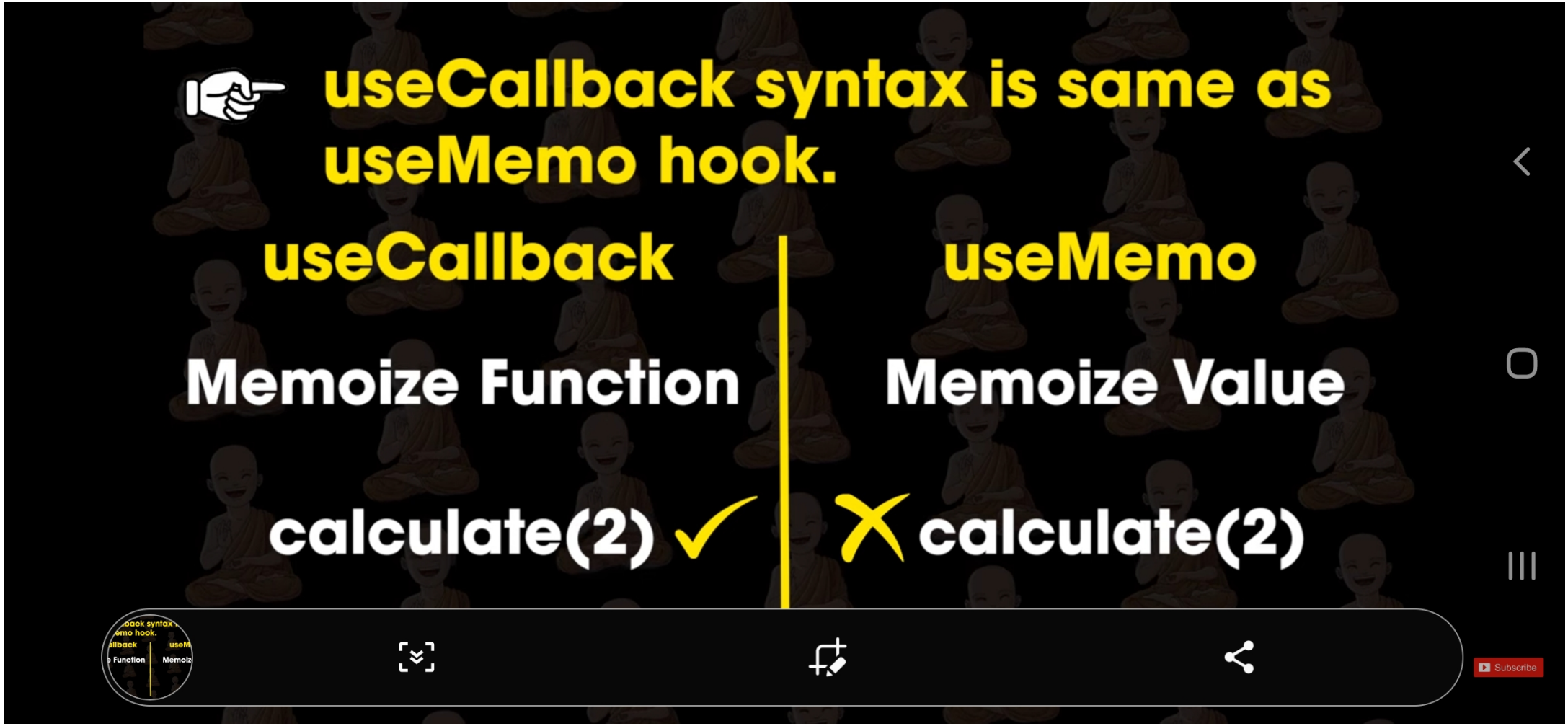
```jsx
import React, { useEffect, useState } from "react";

const PrintTable = ({ calculateTable }) => {
    const [rows, setRows] = useState([]);

    useEffect(() => {
        console.log("Print Table Runs!");
        setRows(calculateTable());
    }, [calculateTable]);

    return rows.map((row, index) => {
        return <p key={index}>{row}</p>;
    });
};

export default PrintTable;
```


React App
localhost:3000
2
2
4
6
8
10
Toggle
Elements
Console
1
top
Filter
Default levels
1 Issue: 1
Print Table Runs!
PrintTable.js:7
Print Table Runs!
PrintTable.js:7
Subscribe

App.js ● PrintTable.js

src > App.js > App > calculateTable > useCallback() callback

```
1  import React, { useState, useCallback } from "react";
2  import PrintTable from "./PrintTable";
3
4  const App = () => {
5      const [number, setNumber] = useState(1);
6      const [darkTheme, setDarkTheme] = useState(false);
7
8      const calculateTable = useCallback(() => {
9          return [number * 1, number * 2, number * 3, number * 4, number * 5];
10     }, [number]);
11
12     const cssStyle = {
13         backgroundColor: darkTheme ? "black" : "white",
14         color: darkTheme ? "white" : "black",
15     };
16
17     return (
```

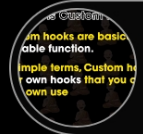

useCallback syntax is same as useMemo hook.
useCallback
useMemo
Memoize Function
Memoize Value
calculate(2)
calculate(2)
Subscribe

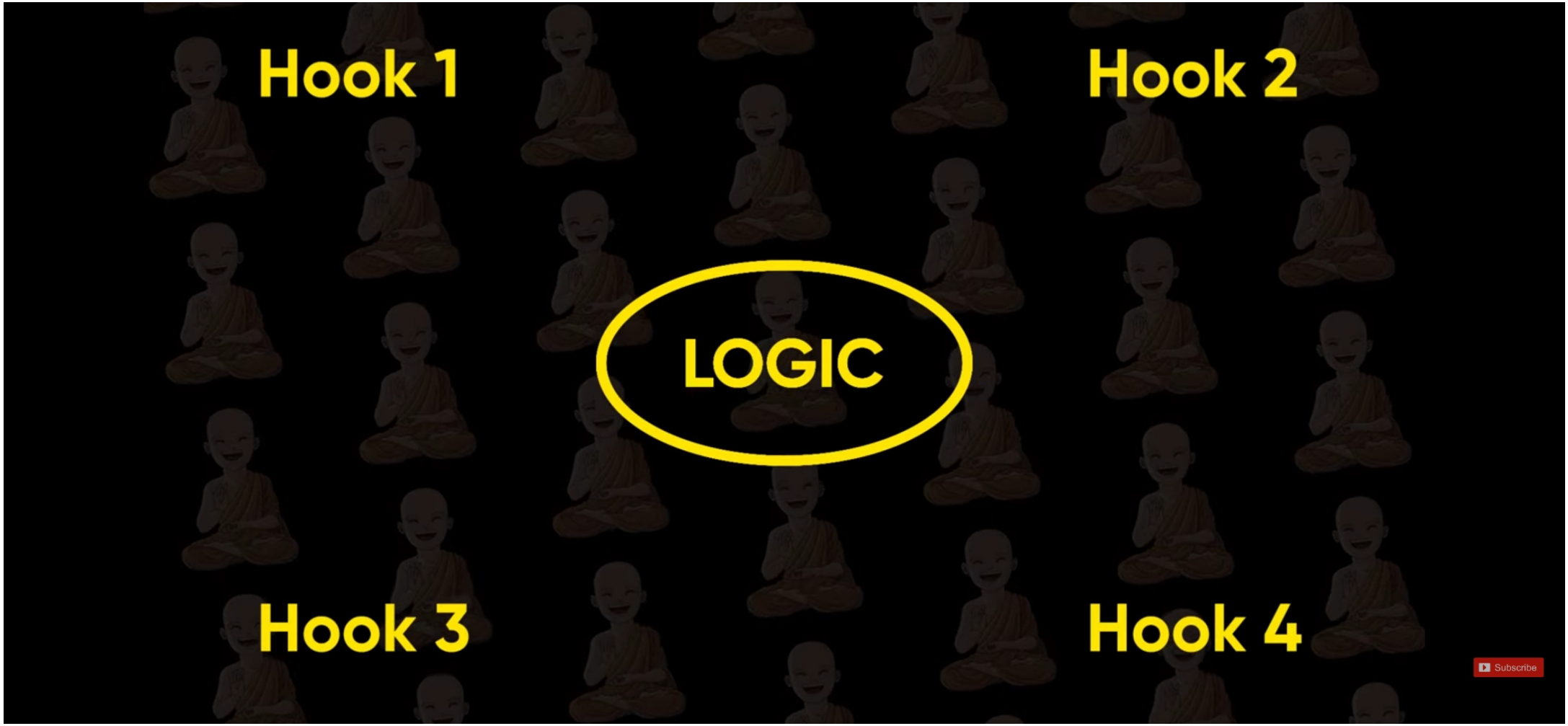

And The Last One
Custom Hooks
Subscribe

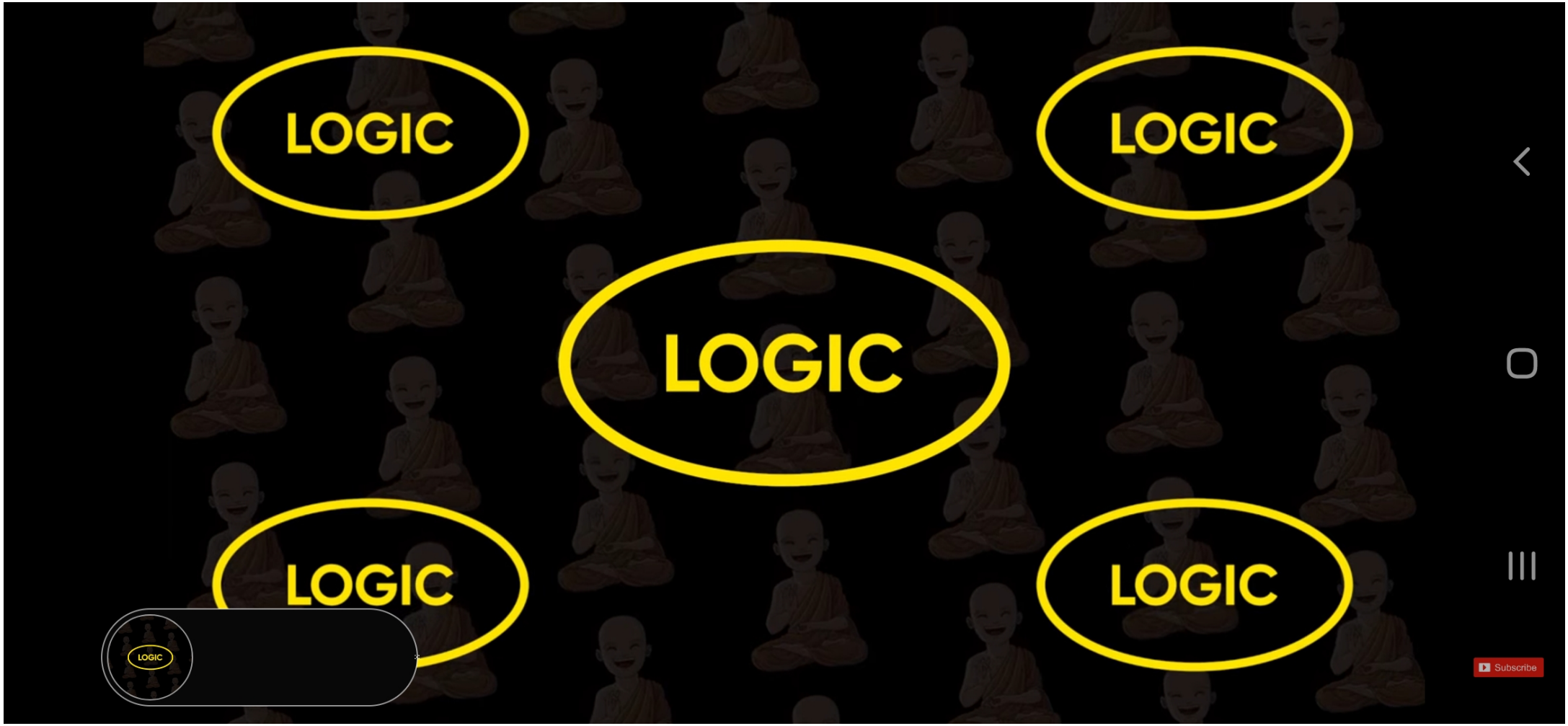
What is Custom hook?

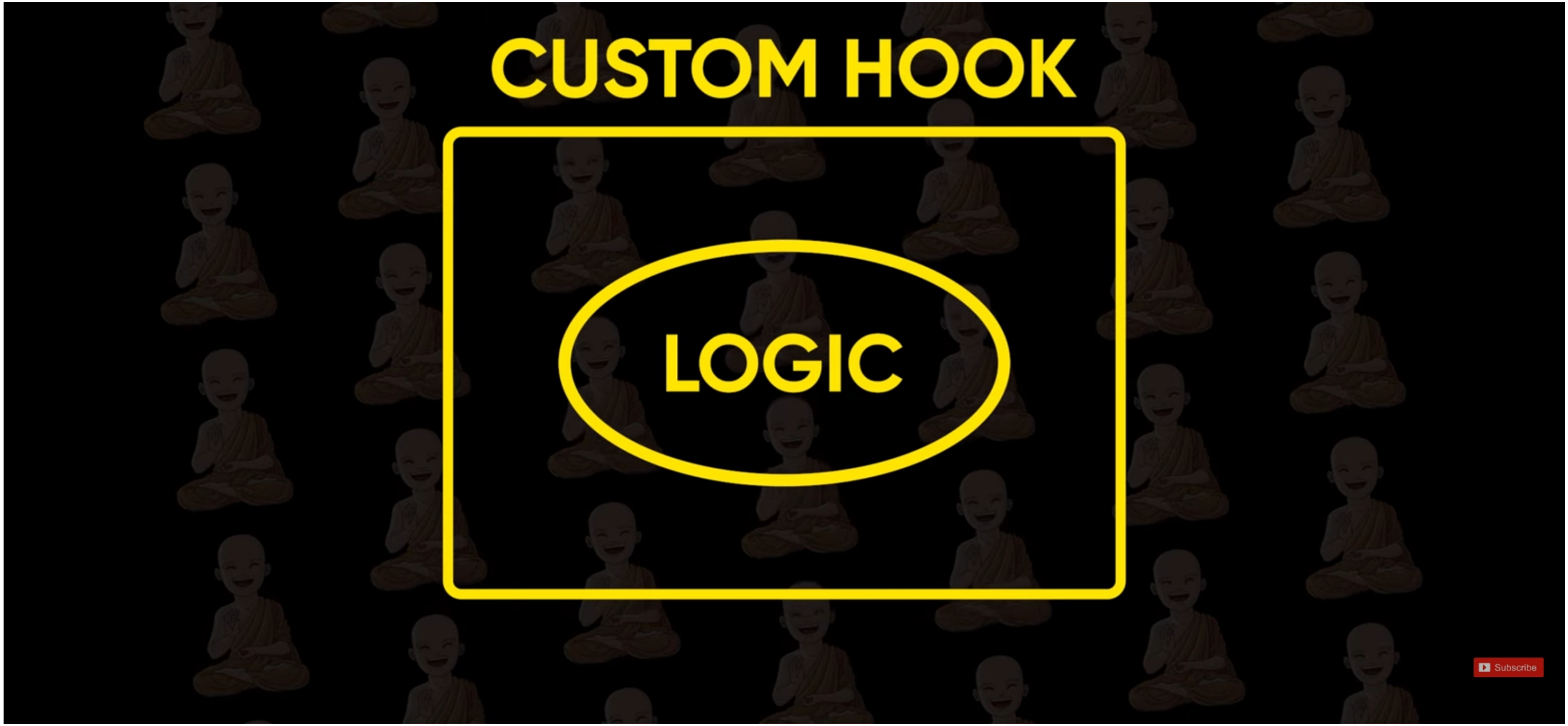
Custom hooks are basically a reusable function.

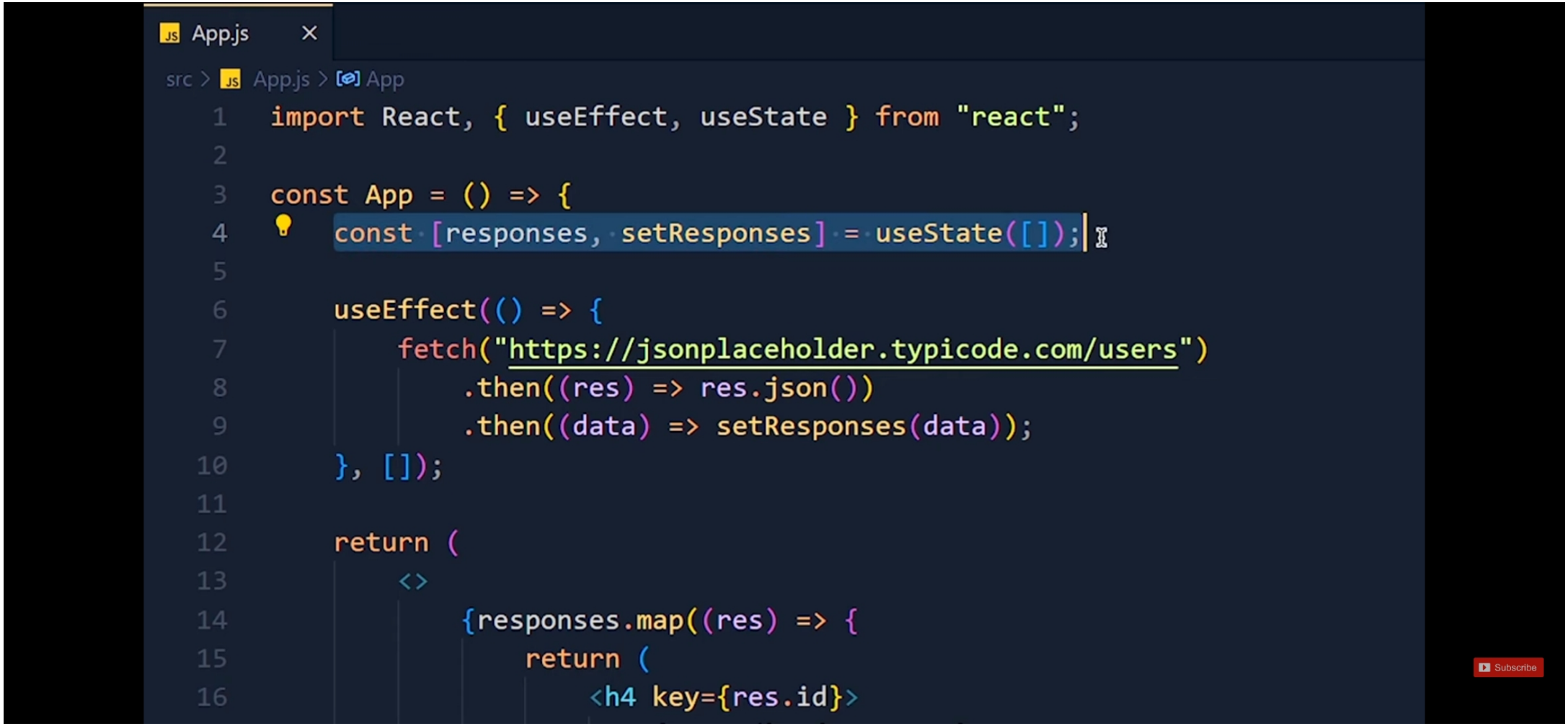
In simple terms, Custom hooks are your own hooks that you create for your own use and you can use them multiple times in your project.

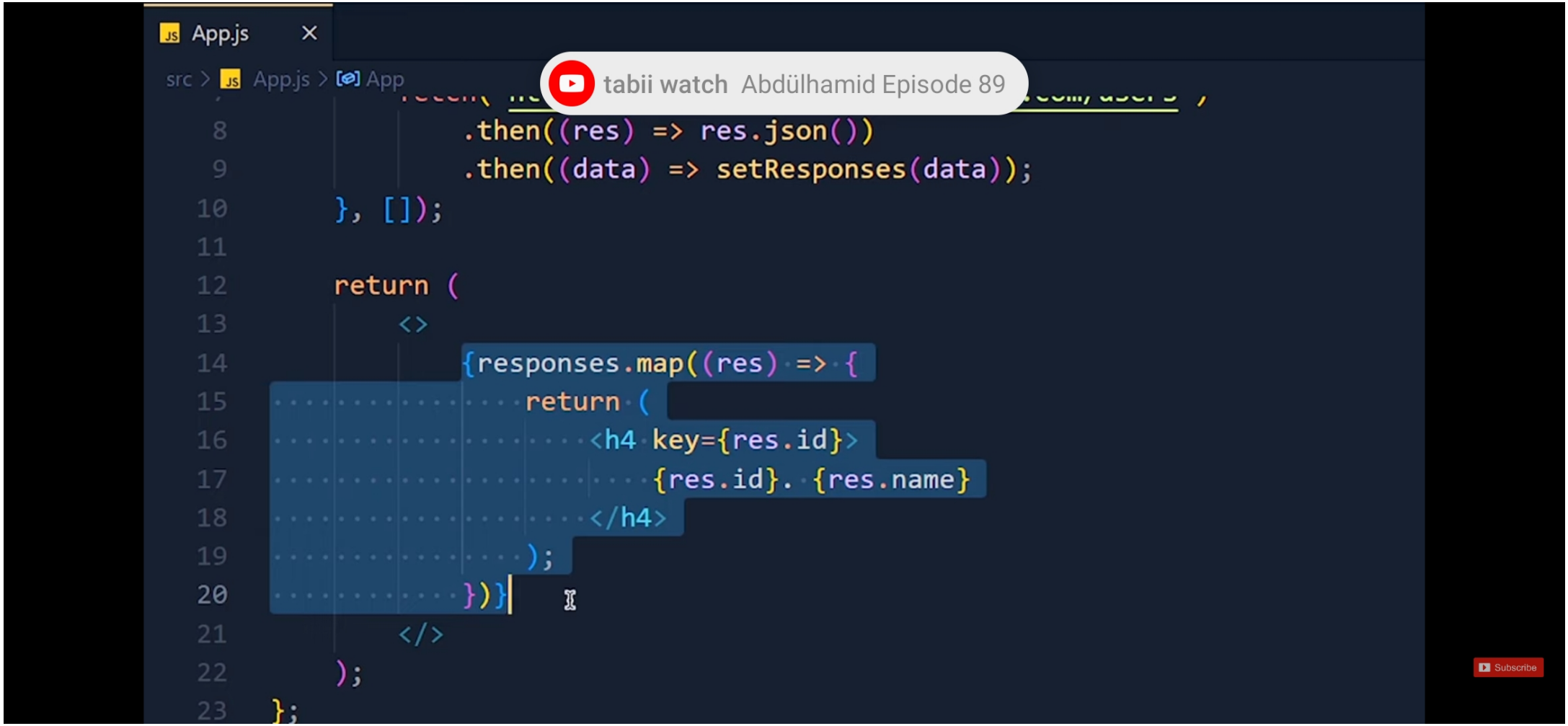


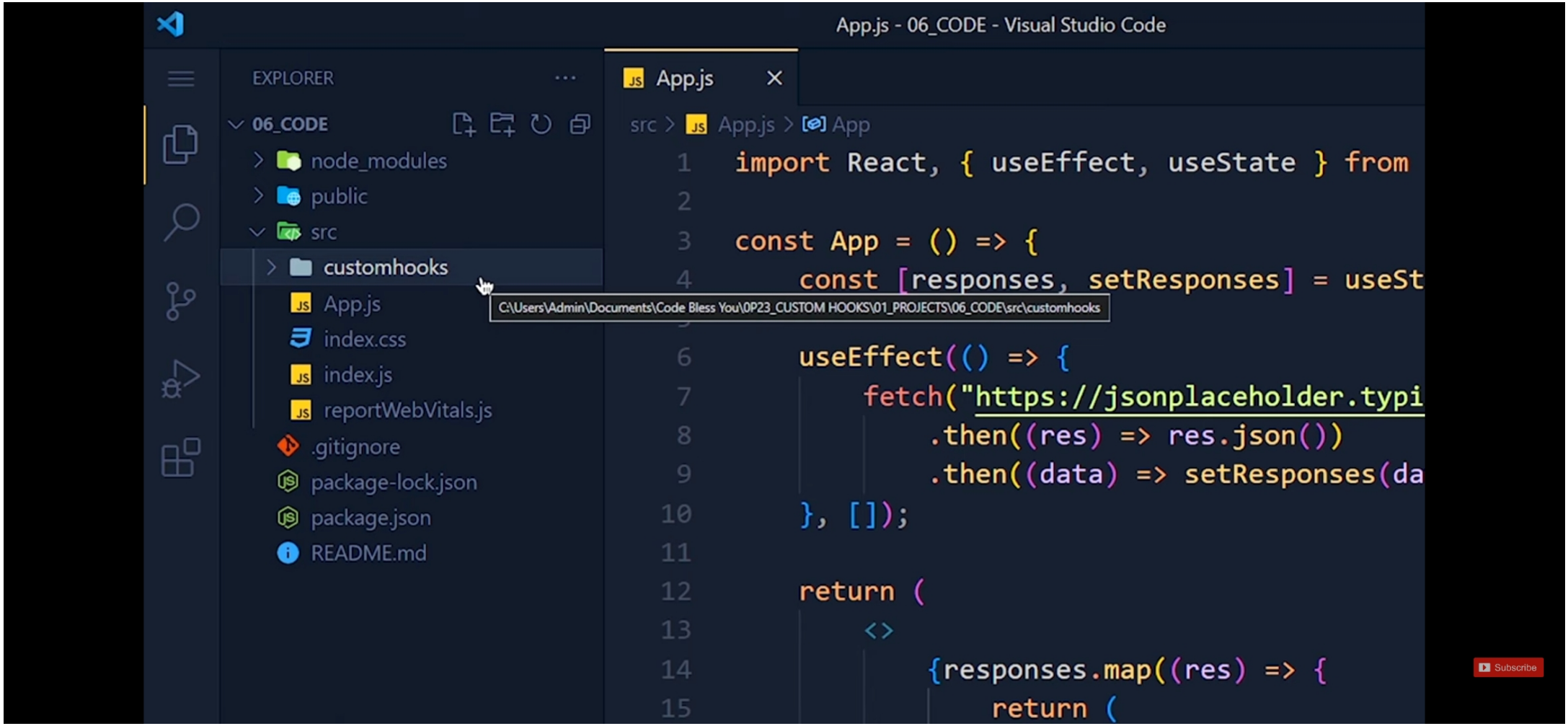

Hook 1
Hook 2
LOGIC
Hook 3
Hook 4
Subscribe


LOGIC
LOGIC
LOGIC
LOGIC
LOGIC
LOGIC
Subscribe


CUSTOM HOOK
LOGIC
Subscribe


App.js
src > App.js > App
import React, { useEffect, useState } from "react";
const App = () => {
const [responses, setResponses] = useState([]);
useEffect(() => {
fetch("https://jsonplaceholder.typicode.com/users")
.then((res) => res.json())
.then((data) => setResponses(data));
}, []);
return (
<>
{responses.map((res) => {
return (
<h4 key={res.id}>
Subscribe

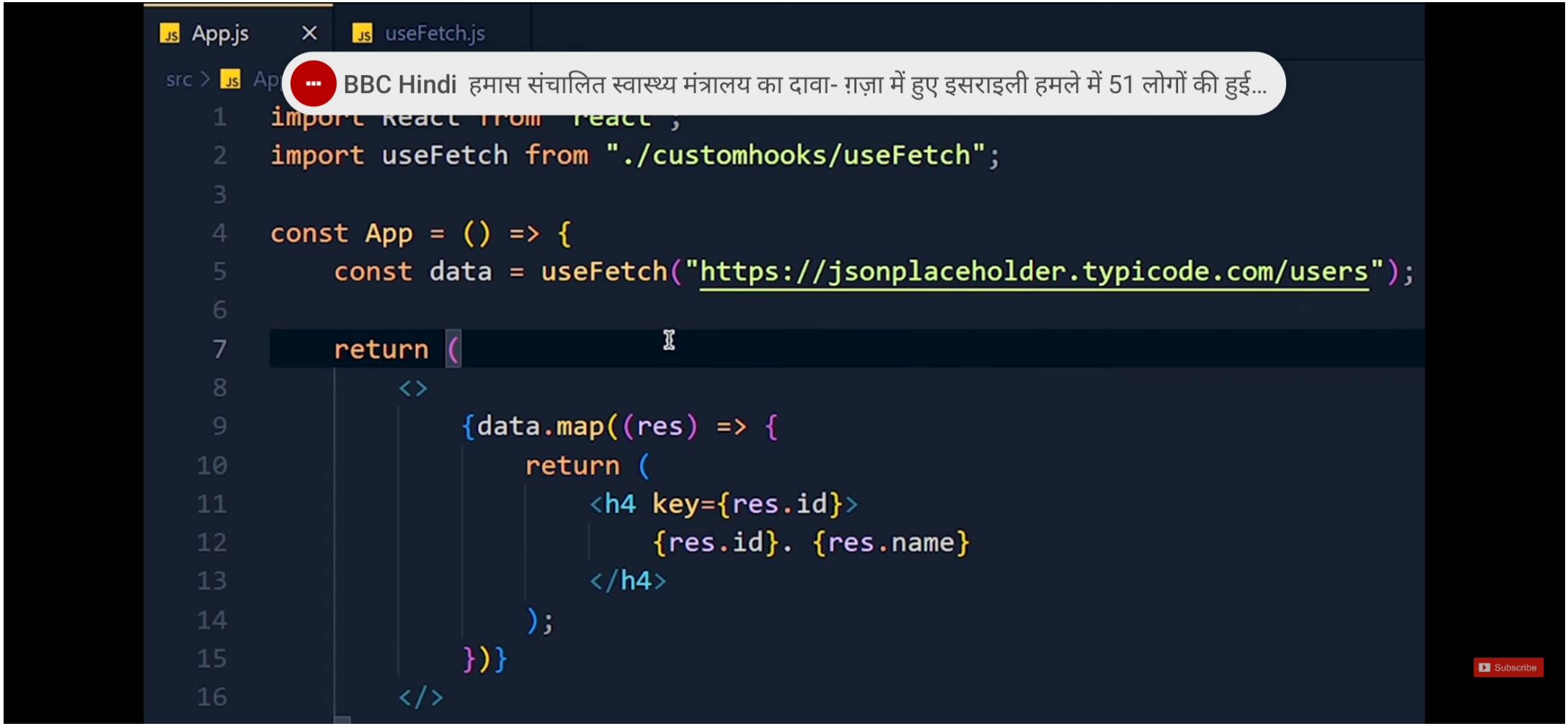

App.js
src > App.js > App
tabii watch Abdülhamid Episode 89
.then((res) => res.json())
.then((data) => setResponses(data));
}, []);
return (
<>
{responses.map((res) => {
return (
<h4 key={res.id}>
{res.id}. {res.name}
</h4>
);
})}
</>
);
};
Subscribe

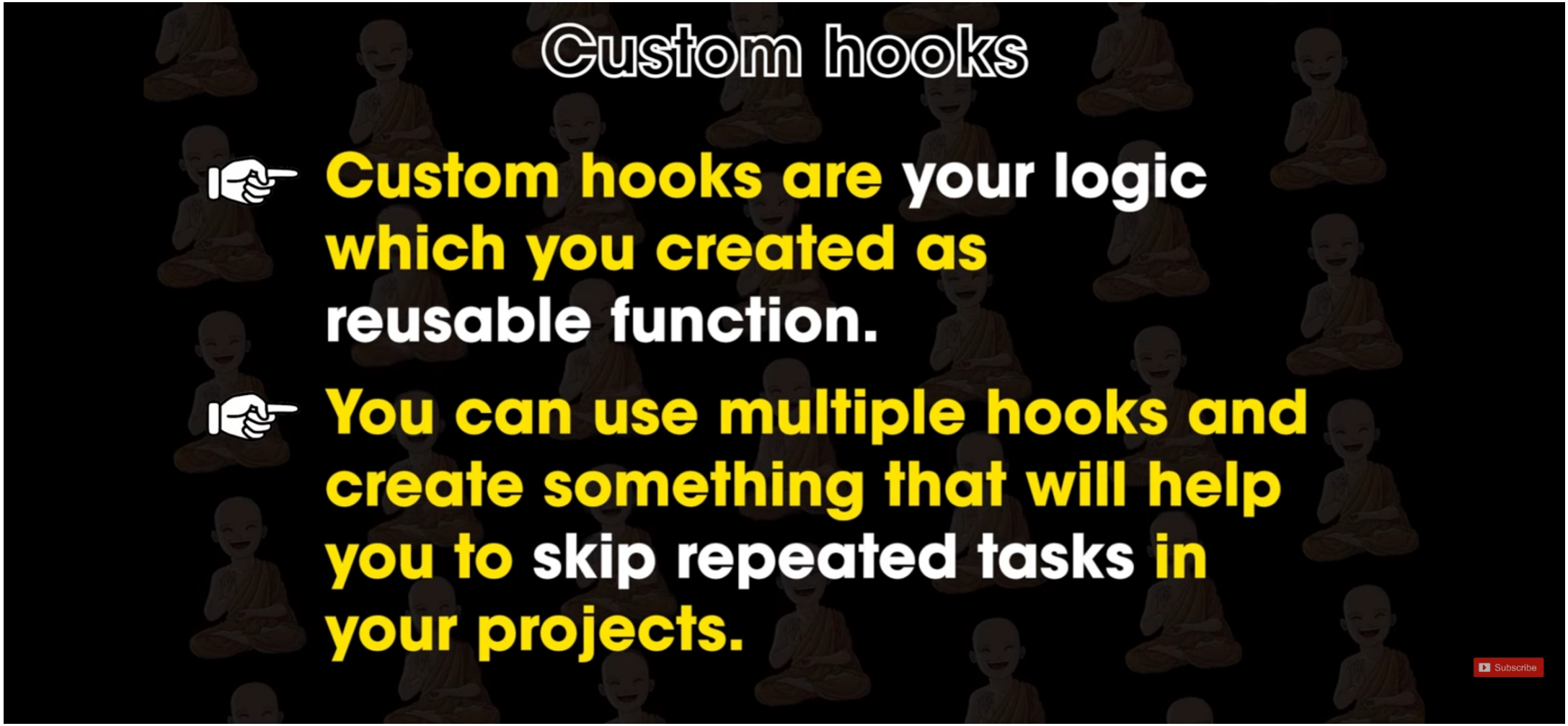

App.js - 06_CODE - Visual Studio Code
EXPLORER
App.js
06_CODE
node_modules
public
src
customhooks
App.js
index.css
index.js
reportWebVitals.js
.gitignore
package-lock.json
package.json
README.md
C:\Users\Admin\Documents\Code Bless You\0P23_CUSTOM HOOKS\01_PROJECTS\06_CODE\src\customhooks
src > App.js > App
import React, { useEffect, useState } from
const App = () => {
const [responses, setResponses] = useSt
useEffect(() => {
fetch("https://jsonplaceholder.typi
.then((res) => res.json())
.then((data) => setResponses(da
}, []);
return (
<>
{responses.map((res) => {
return (
Subscribe

App.js ● useFetch.js ×

src > customhooks > useFetch.js > useFetch

```
1  import React, { useEffect, useState } from "react";
2
3  const useFetch = (url) => {
4      const [responses, setResponses] = useState([]);
5
6      useEffect(() => {
7          fetch(url)
8              .then((res) => res.json())
9              .then((data) => setResponses(data));
10     }, []);
11     return responses;
12 };
13
14 export default useFetch;
15
```

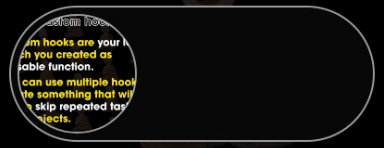

App.js
useFetch.js
src > App
BBC Hindi हमास संचालित स्वास्थ्य मंत्रालय का दावा- ग़ज़ा में हुए इसराइली हमले में 51 लोगों की हुई...
import React from "react";
import useFetch from "./customhooks/useFetch";
const App = () => {
const data = useFetch("https://jsonplaceholder.typicode.com/users");
return (
<>
{data.map((res) => {
return (
<h4 key={res.id}>
{res.id}. {res.name}
</h4>
);
})}
</>
Subscribe


Custom hooks
Custom hooks are your logic which you created as reusable function.
You can use multiple hooks and create something that will help you to skip repeated tasks in your projects.
Subscribe

reusable function.

You can use multiple hooks and create something that will help you to skip repeated tasks in your projects.

You can simply use that custom hooks in your different projects.



Subscribe