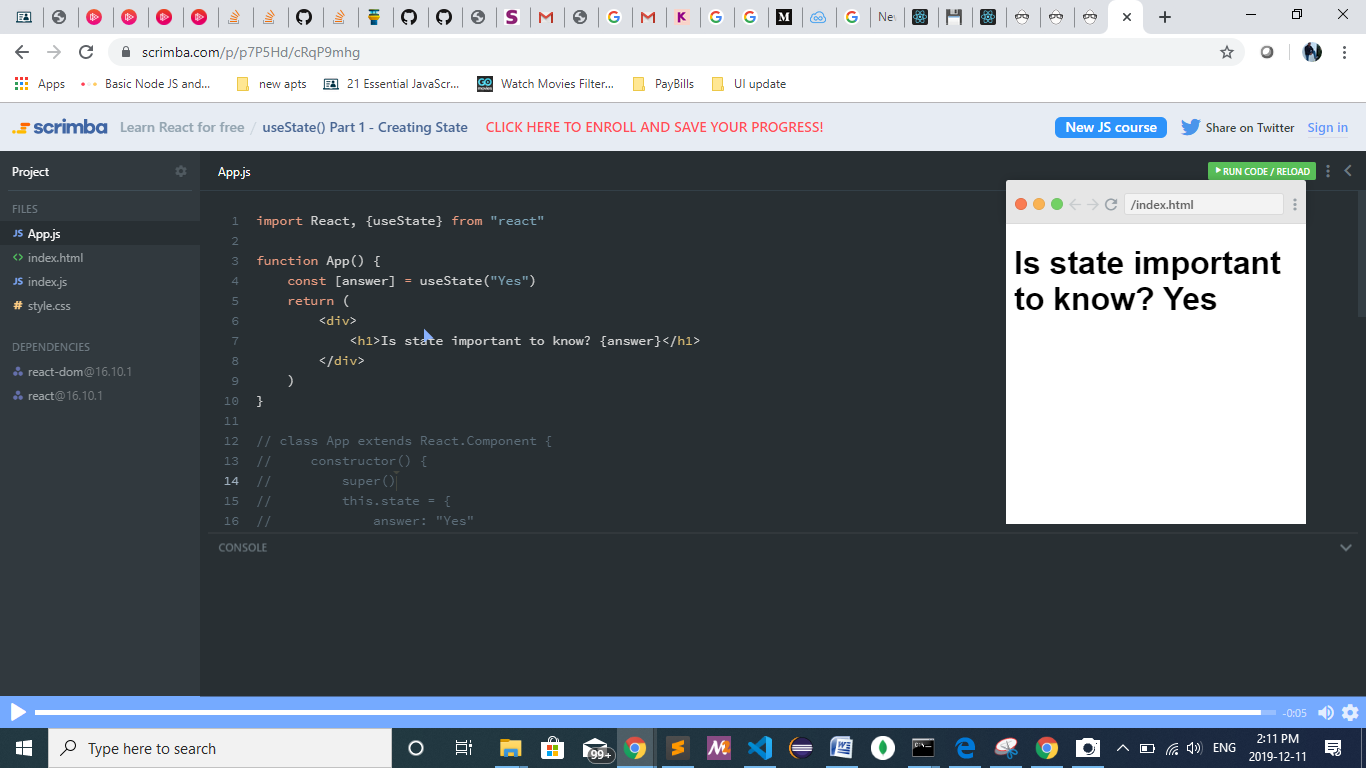
**Hooks**:

**useState()**:

it helps us not use objects for state rather use primitive values.

It returns an array , first element will be the value for the state and second will be a fn for setting the state.

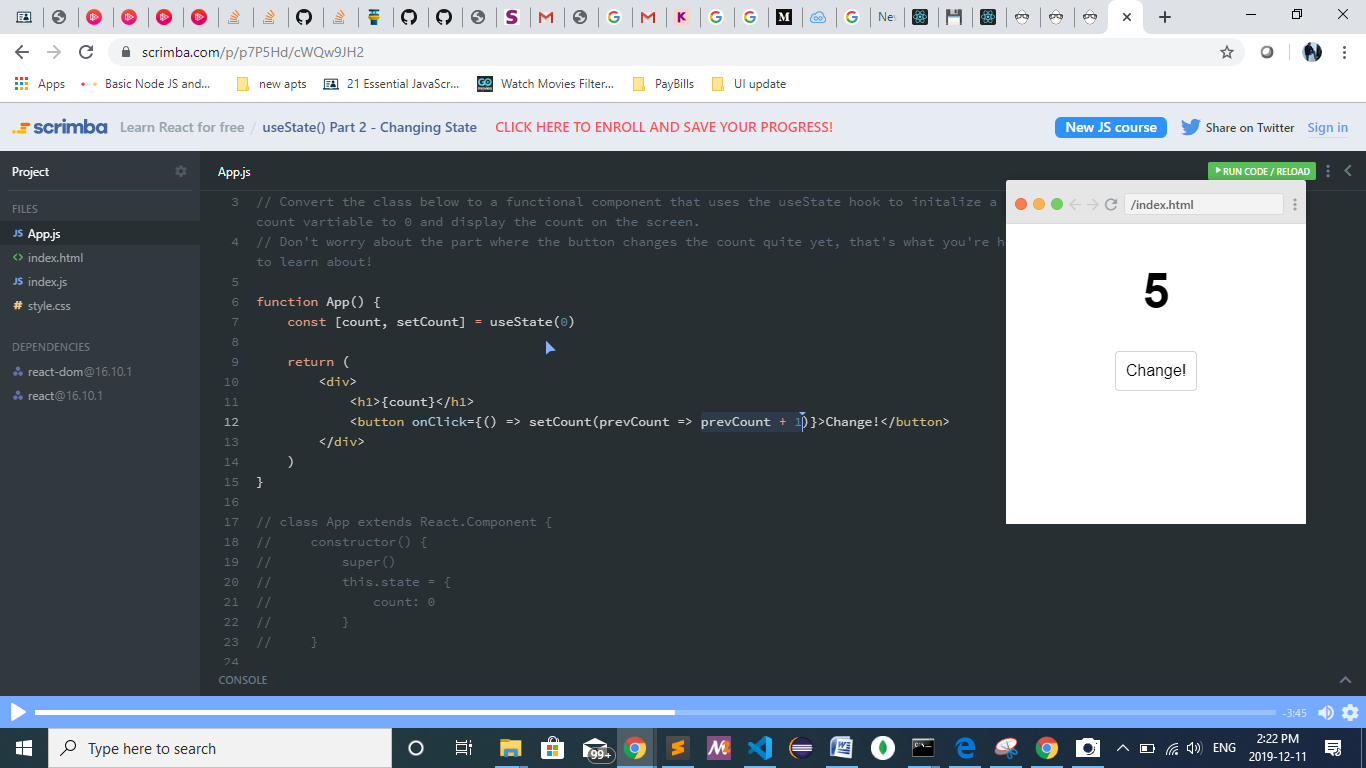


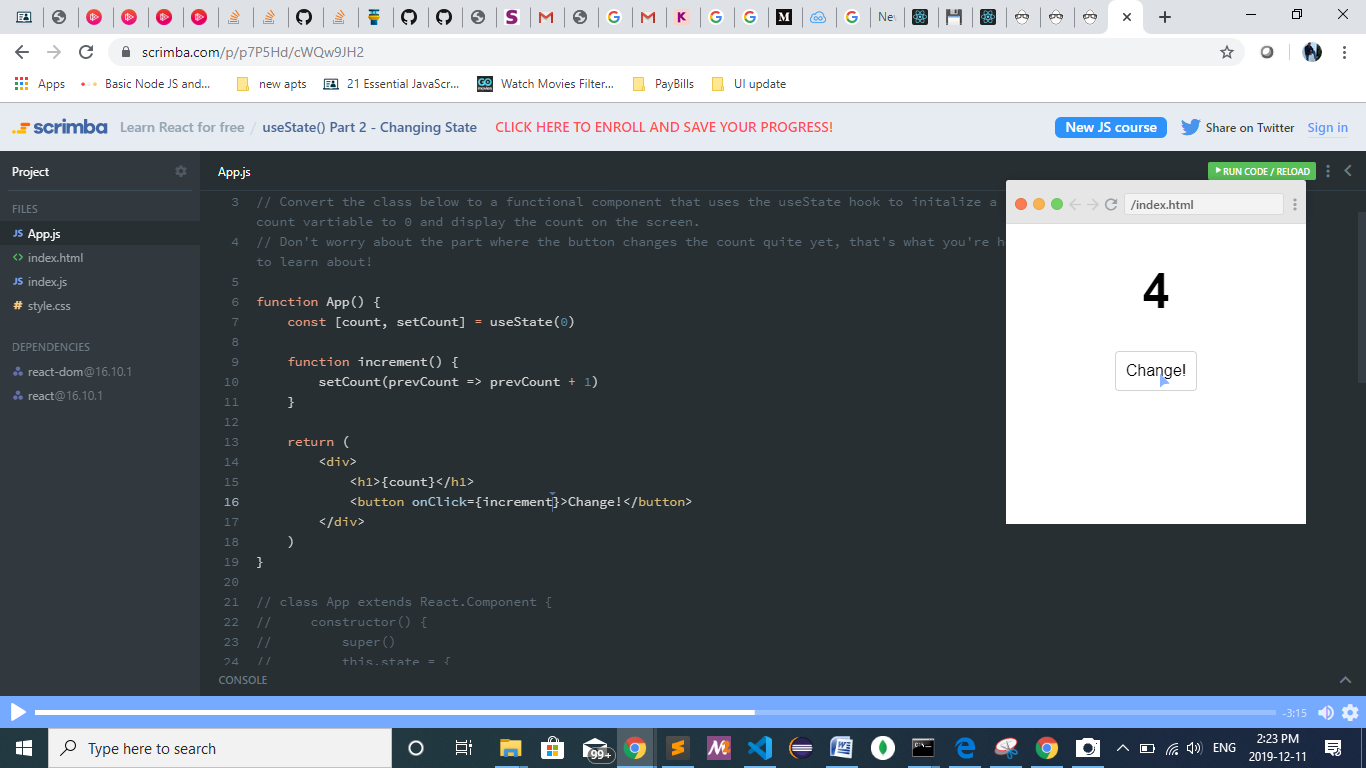
Array destructuring:

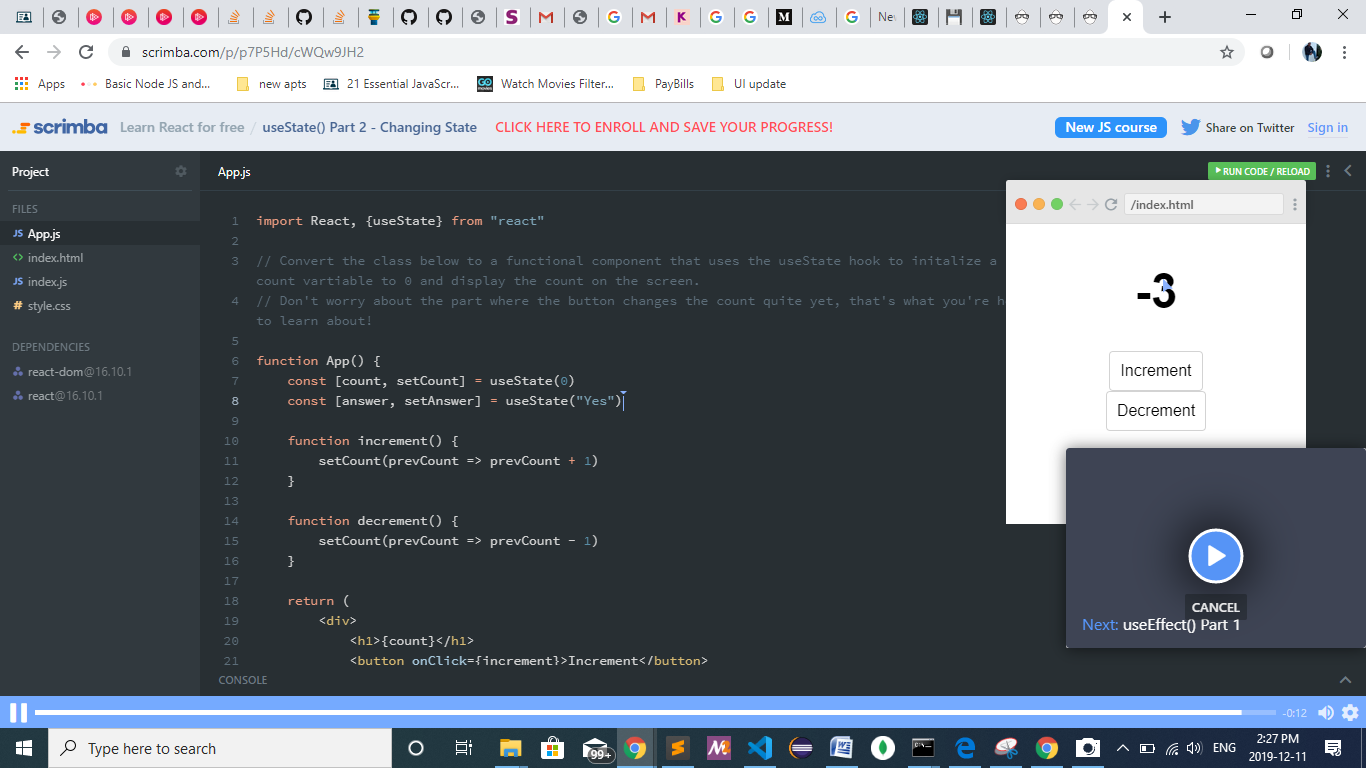
const value=[“yes”,fun()];

const [value]=

Object destructuring:







useEffect():

* replacement for three lifecycle methods :

1. componentDidMount()
2. componentDidUpdate()
3. componentWillUnmount()

* it is the hook to produce side effects in our components(communicating with the server or db , for ex: network request to retrieve data , manual dom manipulation, setting up event listeners or timeouts and intervals manually).
* In componentDidUpdate , we had the option wherein we compare the prevProps and prevState to the currentProp and currentState and manually decide whether to render or not

1. useEffect( () =>

{ setFn(randomcolor() )

} , [var1,var2....] )

The effect will be applied when the second parameter(array of variables ) changes (it can one or many variables to decide on ).

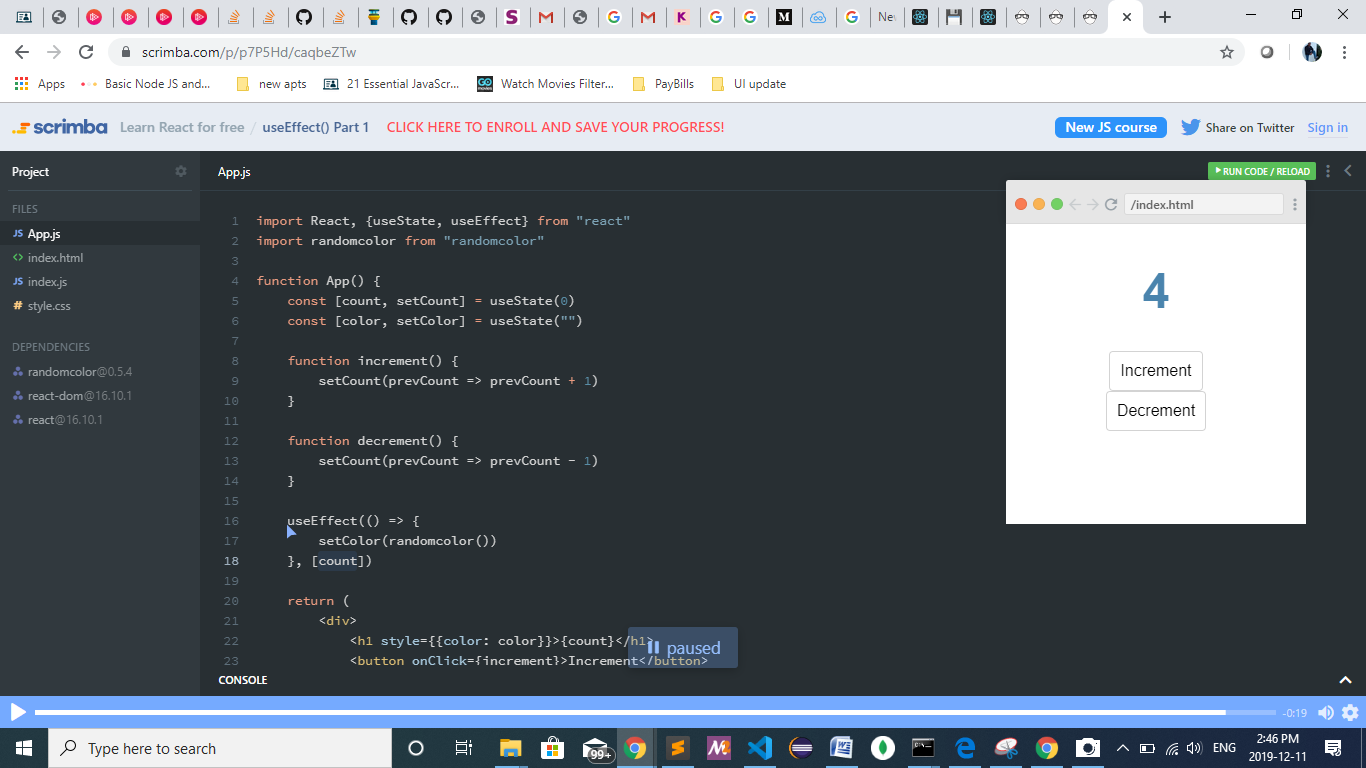
1. To mount for the first time:

useEffect( () =>

{ setFn(randomcolor() )

} , [] )

Empty array says to apply the mount only for the first time or initial mounting and not ever again.



References:

<https://scrimba.com/p/p7P5Hd/cLpdgBAd>