Current Skill Common mistakes

useRef()

The `useRef()` Hook is a function that returns a mutable reference object whose current property is initialized to the passed argument (initialValue). The returned object will persist for the component's entire lifetime.

This example may help you:

```
function App() {
  let [name, setName] = useState("Ned stark");
  // we declare the input inside the variable
  let nameRef = useRef();
  // we are referring to input to change the value
  const submitButton = () => {
    setName(nameRef.current.value);
  };

  return (
    <div className="App">
      <p>{name}</p>
      <h1>Who is your favorite Games of throne character</h1>

      <div>
        <input
          placehoder="enter your preferred GOT character..."
          ref={nameRef}
          type="text"
        />
        <button type="button" onClick={submitButton}>
          Submit
        </button>
      </div>

    </div>

  );

}
```

In this function we are recovering the input value using the useRef method instead of onChange event. That may be a potential bug event if works for now. A ref created with **useRef** will be created only when the component has been mounted. Refs can be used for accessing DOM nodes or React elements, and for keeping mutable variables.

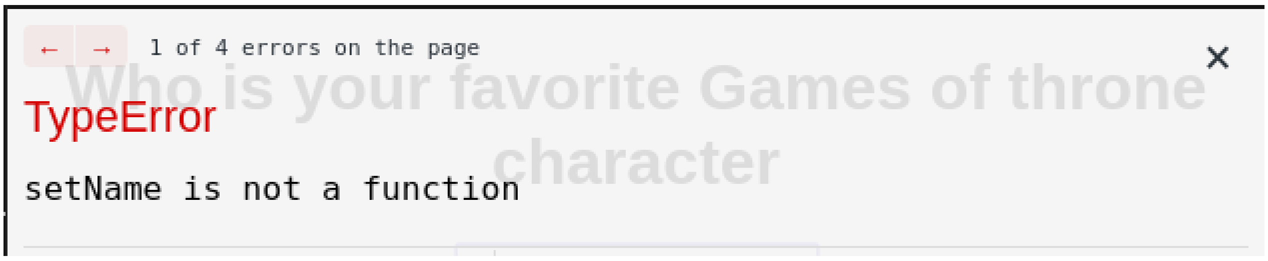
Destructuring using object:

Don't forget: hooks use arrays to store data.

As we have previously mentioned, Hooks are applied in an array:

`const { state, setState} = useState("intial state")`

If we perform destructuring using the curly brackets, we’ll receive this error:



```
← →  1 of 4 errors on the page
```

Who is your favorite Games of throne character

TypeError

```
setName is not a function
```

Reminder that this code is incorrect. We need to use brackets and not curly braces.

`const [state, setState] = useState("intialState)`

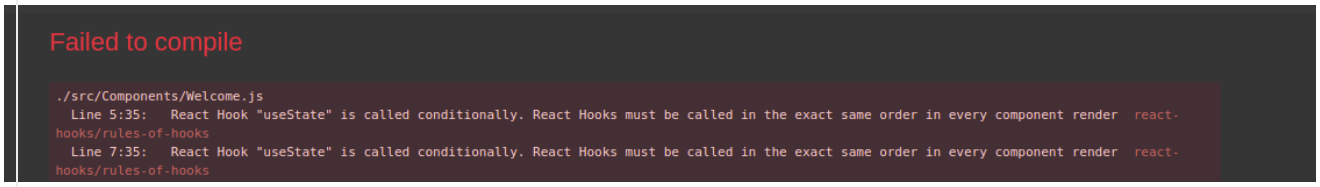
Hooks inside a condition:

We should avoid creating the state hook in a condition because that will violate the hook rules.

Once you break these rules, this error will occur:

```
import React, { useState } from 'react'


const Welcome = props =>{

 if(props.name==='ned stark'){

   const [bgColor,setBgColor ]= useState('white')

 }
 else{
   const [bgColor, setBgColor] = useState('black')
 }

 return (
   <h1 style={{backgroundColor:{bgColor}}} >{props.name}</h1>
 )
}

export default Welcome
```

Output



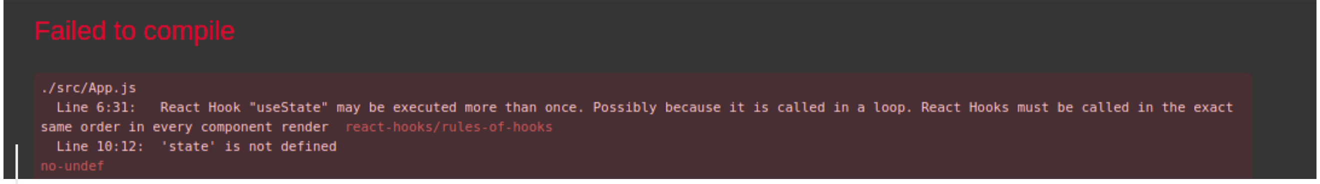
```
Failed to compile

./src/Components/Welcome.js
  Line 5:35:   React Hook "useState" is called conditionally. React Hooks must be called in the exact same order in every component render  react-hooks/rules-of-hooks
  Line 7:35:   React Hook "useState" is called conditionally. React Hooks must be called in the exact same order in every component render  react-hooks/rules-of-hooks
```

Hooks inside a loop:

Another common mistake is to use the hook inside loops. Here is an example to illustrate the error.

```
function App() {
 for (let i = 1; i < 5; i++) {
   const [state, setstate] = useState(i);
 }
 return (
   <div>
     <h1>{state}</h1>
   </div>
 );
}

export default App;
```

Output



```
Failed to compile

./src/App.js
  Line 6:31:   React Hook "useState" may be executed more than once. Possibly because it is called in a loop. React Hooks must be called in the exact
same order in every component render  react-hooks/rules-of-hooks
  Line 10:12:  'state' is not defined
no-undef
```

Nesting before using the state:

We can't nest inside a function before using our state.

```
function App({ date }) {
  function updateCount(byValue) {
    const [currentDate, setCurrentDate] = useState(new Date());
    const [count, setCount] = useState(0);
    setCount(count + byValue);
    setCurrentDate(new Date());
  }

  function formatDate() {
    const hour = currentDate.getHours();
    const minute = currentDate.getMinutes();
    const second = currentDate.getSeconds();

    return `${hour}:${minute}:${second}`;
  }

  const prettyDate = formatDate();

  return (
    <div className="App">
      <h2>
        You clicked {count} times, last time at {prettyDate}!
      </h2>

      <button onClick={() => updateCount(-1)}>Decrement</button>
      <button onClick={() => updateCount(1)}>Increment</button>
    </div>
  );
 }
```

useState inside an effect:

Now, we will go through some points where we invoke the useEffect.

Important notice: The combined use of both the useState and useEffect generates an infinite loop.

So, don’t call a useState inside a useEffect.

Let's Sum Up!

To summarize, we can only call Hooks at the top level.

We should be aware of these rules:

- Don't declare hooks in if statements.
- Don't declare hooks in loops.
- Don't declare hooks in nested function.
- Always make sure to use the brackets instead of the curly brackets.

‹ Previous   next ›