

9 Rules of Hooks

Calling Hooks

Hooks should only be called in *React function components* or *custom Hooks*. They cannot be used in class components or regular JavaScript functions.

Hooks can be called at the top level of the following:

- React function components
- Custom Hooks (we are going to learn about creating custom Hooks in the next chapter)

As we can see, Hooks are mostly normal JavaScript functions, except that they rely on being defined in a React function component. Of course, custom Hooks that use other Hooks can be *defined* outside of React function components, but when *using* Hooks, we always need to make sure that we call them inside a React function component. Next, we are going to learn about the rules regarding the order of Hooks.

Exercise 1: Order of Hooks

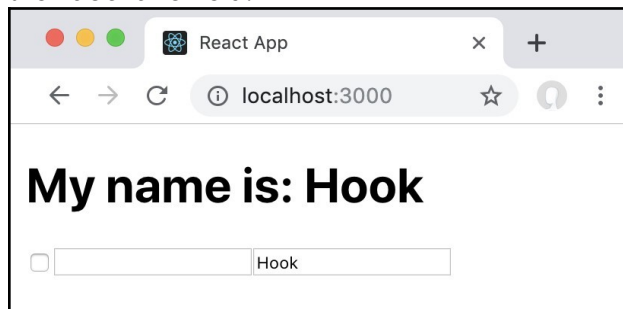
Step 1: Only call Hooks at the top level/beginning of function components or custom Hooks.

Do not call Hooks inside conditions, loops, or nested functions—doing so changes the order of Hooks, which causes bugs. We have already learned that changing the order of Hooks causes the state to get mixed up between multiple Hooks.

In Chapter 2, *Using the State Hook*, we learned that we cannot do the following:

```
const [ enableFirstName, setEnableFirstName ] = useState(false)
const [ name, setName ] = enableFirstName
  ? useState('')
  : [ '', () => {} ]
const [ lastName,
      setLastName ] = useState('')
```

We rendered a checkbox and two input fields for the `firstName` and `lastName`, and then we entered some text in the `lastName` field:



Revisiting our example from Chapter 2, Using the State Hook

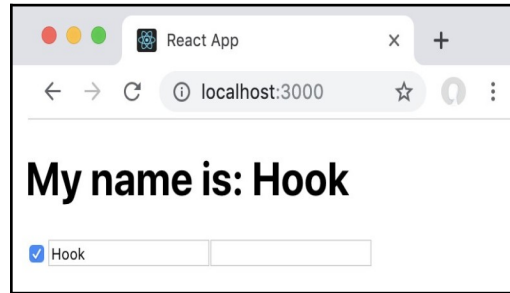
At the moment, the order of Hooks is as follows:

1. `enableFirstName`
2. `lastName`

Next, we clicked on the checkbox to enable the `firstName` field. Doing so changed the order of Hooks, because now our Hook definitions look like this:

1. `enableFirstName`
2. `firstName`
3. `lastName`

Since React solely relies on the order of Hooks to manage their state, the `firstName` field is now the second Hook, so it gets the state from the `lastName` field:



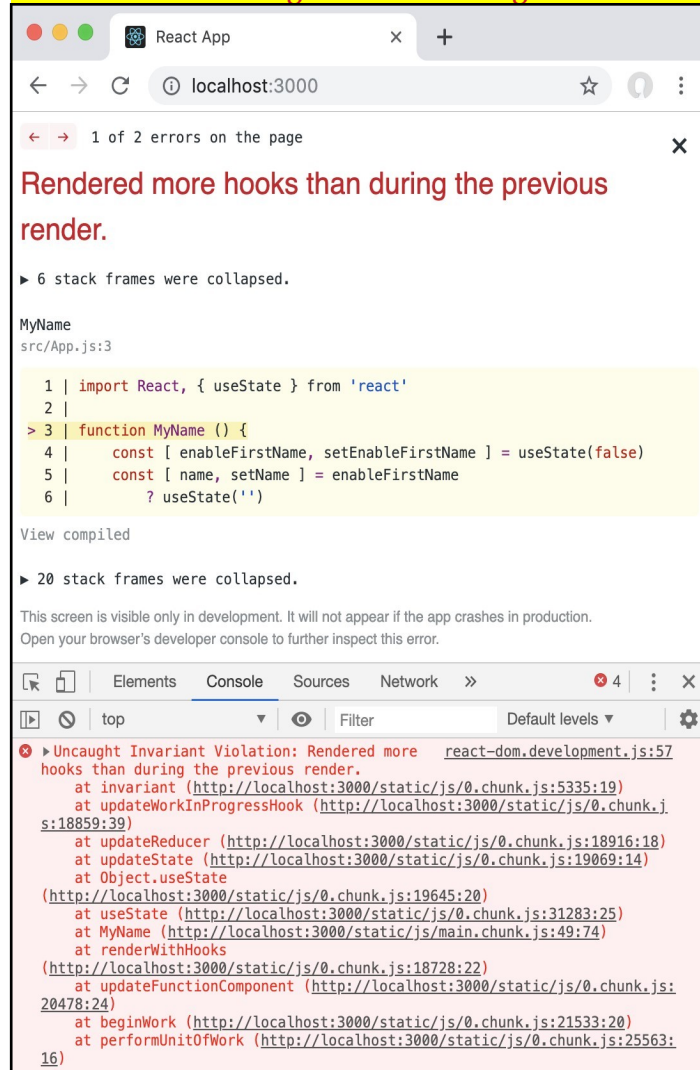
Problem of changing the order of Hooks from Chapter 2, Using the State Hook

If we use the real `useState` Hook from React in example 2 *Can we define conditional Hooks?* from Chapter 2, *Using the State Hook*, we can see that React automatically detects when the order of Hooks has changed, and it will show a warning:



React printing a warning when detecting that the order of Hooks has changed

When running React in development mode, it will additionally crash with an Uncaught Invariant Violation error message when rendering more Hooks than in the previous render:



React crashing in development mode when the number of Hooks changed

As we can see, changing the order of Hooks or conditionally enabling Hooks is not possible, as React internally uses the order of Hooks to keep track of which data belongs to which Hook.

Exercise 2: Enforcing the rules of Hooks

If we stick to the convention of prefixing Hook functions with `use`, we can automatically enforce the other two rules:

- Only call Hooks from React function components or custom Hooks
- Only call Hooks at the top level (not inside loops, conditions, or nested functions)

In order to enforce the rules automatically, React provides an `eslint` plugin called `eslint-plugin-react-hooks`, which will automatically detect when Hooks are used, and will ensure that the rules are not broken. ESLint is a linter, which is a tool that analyzes source code and finds problems such as stylistic mistakes, potential bugs, and programming errors.

Step 1: Setting up `eslint-plugin-react-hooks`

We are now going to set up the React Hooks `eslint` plugin to automatically enforce the rules of Hooks.

Let's start installing and enabling the `eslint` plugin:

1. First, we have to install the plugin via `npm`:

```
> npm install --save-dev eslint-plugin-react-hooks
```

We use the `--save-dev` flag here, because `eslint` and its plugins are not required to be installed when deploying the app. We only need them during the development of our app.

2. Then, we create a new `.eslintrc.json` file in the root of our project folder, with the following contents. We start by extending from the `react-app` ESLint configuration:

```
{
  "extends": "react-app",
```

3. Next, we include the `react-hooks` plugin that we installed earlier:

```
  "plugins": [
    "react-hooks"
  ],
```

4. Now we enable two rules. First, we tell `eslint` to show an error when we violate the `rules-of-hooks` rule. Additionally, we enable the `exhaustive-deps` rule as a warning:

```
  "rules": {
    "react-hooks/rules-of-hooks": "error",
    "react-hooks/exhaustive-deps": "warn"
  }
}
```

5. Finally, we adjust `package.json` to define a new `lint` script, which is going to call `eslint`:

```
  "scripts": {
    "lint": "npx eslint src/",
```

Now, we can execute `npm run lint`, and we are going to see that there are **5 warnings** and **0 errors**:

```
1. fish /Users/dan/Development/Hands-On-Web-Development-with-Hooks/Chapter09/chapter...
dan@galaxy ~/D/H/C/chapter9_1 (master)> npm run lint

> chapter8_5@1.0.0 lint /Users/dan/Development/Hands-On-Web-Development-with-Hoo
ks/Chapter09/chapter9_1
> npx eslint src/

/Users/dan/Development/Hands-On-Web-Development-with-Hooks/Chapter09/chapter9_1/
src/pages/HomePage.js
  22:8  warning  React Hook useEffect has a missing dependency: 'dispatch'. Eith
er include it or remove the dependency array  react-hooks/exhaustive-deps

/Users/dan/Development/Hands-On-Web-Development-with-Hooks/Chapter09/chapter9_1/
src/post/CreatePost.js
  32:6  warning  React Hook useEffect has a missing dependency: 'cancelDebounce'
. Either include it or remove the dependency array  react-hooks/exhaustive-deps
  46:6  warning  React Hook useEffect has a missing dependency: 'navigation'. Ei
ther include it or remove the dependency array      react-hooks/exhaustive-deps

/Users/dan/Development/Hands-On-Web-Development-with-Hooks/Chapter09/chapter9_1/
src/user/Login.js
  30:6  warning  React Hook useEffect has a missing dependency: 'dispatch'. Eith
er include it or remove the dependency array  react-hooks/exhaustive-deps

/Users/dan/Development/Hands-On-Web-Development-with-Hooks/Chapter09/chapter9_1/
src/user/Register.js
  23:8  warning  React Hook useEffect has a missing dependency: 'dispatch'. Eith
er include it or remove the dependency array  react-hooks/exhaustive-deps

✖ 5 problems (0 errors, 5 warnings)
  0 errors and 5 warnings potentially fixable with the `--fix` option.
```

Executing ESLint with the react-hooks plugin

We will now try to break the rules of Hooks; for example, by editing `src/user/Login.js` and making the second Input Hook conditional:

```
const { value: password, bindToInput: bindPassword } = loginFailed ?
useInput('') : [ '', () => {} ]
```

When we execute `npm run lint` again, we can see that there is now an error:

```
1. fish /Users/dan/Development/Hands-On-Web-Development-with-Hooks/Chapter09/chapter9_1 (fish)
/Users/dan/Development/Hands-On-Web-Development-with-Hooks/Chapter09/chapter9_1/src/user/Login.js
11:72 error React Hook "useInput" is called conditionally. React Hooks must be
called in the exact same order in every component render react-hooks/rules-of-hooks
30:6 warning React Hook useEffect has a missing dependency: 'dispatch'. Either i
nclude it or remove the dependency array react-hooks/exhaustive-deps

/Users/dan/Development/Hands-On-Web-Development-with-Hooks/Chapter09/chapter9_1/src/u
ser/Register.js
23:8 warning React Hook useEffect has a missing dependency: 'dispatch'. Either in
clude it or remove the dependency array react-hooks/exhaustive-deps

✖ 6 problems (1 error, 5 warnings)
0 errors and 5 warnings potentially fixable with the `--fix` option.

npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! chapter8_5@1.0.0 lint: `npx eslint src/`
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the chapter8_5@1.0.0 lint script.
npm ERR! This is probably not a problem with npm. There is likely additional logging
output above.

npm ERR! A complete log of this run can be found in:
npm ERR! /Users/dan/.npm/_logs/2019-06-23T12_51_52_877Z-debug.log
dan@galaxy ~/D/H/C/chapter9_1 (master) [1]>
```

Executing ESLint after breaking the rules of Hooks

As we can see, `eslint` helps us by forcing us to stick to the rules of Hooks. The linter will throw an error when we violate any rules, and show warnings when Effect Hooks have missing dependencies. Listening to `eslint` will help us to avoid bugs and unexpected behavior, so we should never ignore its errors or warnings.

Exercise 3: Dealing with useEffect dependencies (Chapter9_2)

In addition to enforcing the rules of Hooks, we are also checking whether all the variables that are used in an Effect Hook are passed to its dependency array. This *exhaustive dependencies* rule ensures that whenever something that is used inside the Effect Hook changes (a function, value, and so on), the Hook will trigger again.

As we have seen in the previous section, there are a couple warnings related to the exhaustive dependencies rule when running the linter with `npm run lint`. Often, it has to do with the `dispatch` function or other functions not being part of the dependency array. Usually, these functions should not change, but we can never be sure, so it is better to just add them to the dependencies.

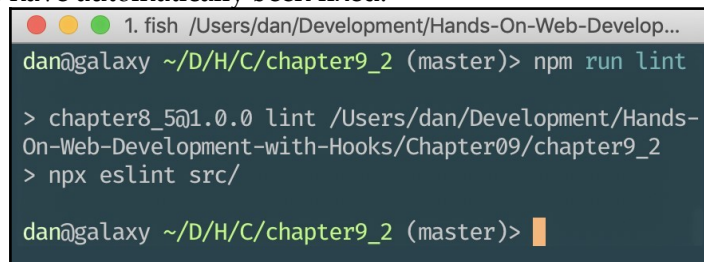
Step 1: Automatically fixing warnings with eslint

As the exhaustive dependencies rule is quite simple and straightforward to fix, we can automatically let `eslint` fix it.

To do so, we need to pass the `--fix` flag to `eslint`. Using `npm run`, we can pass flags by using an additional `--` as a separator, as follows:

```
> npm run lint -- --fix
```

After running the preceding command, we can run `npm run lint` again, and we are going to see that all warnings have automatically been fixed:



```
1. fish /Users/dan/Development/Hands-On-Web-Develop...
dan@galaxy ~/D/H/C/chapter9_2 (master)> npm run lint
> chapter8_5@1.0.0 lint /Users/dan/Development/Hands-On-Web-Development-with-Hooks/Chapter09/chapter9_2
> npx eslint src/

dan@galaxy ~/D/H/C/chapter9_2 (master)> 
```

No warnings after letting eslint fix them

As we can see, `eslint` not only warns us about problems, it can even fix some of them automatically for us!