This is your **last** free member-only story this month. Sign up for Medium and get an extra one

Mohammad Faisal
May 6, 2021 · 3 min read · Listen

Apply the Dependency Inversion Principle in React

Your React component knows too much. Let's fix that

Photo by Isaiah.Rivera from Flickr.

The *dependency inversion* principle is one of the famous SOLID principles. Also, it is one of the most important ones.

Today, we will see how to solve a very common mistake that novice React developers make using this principle.

I will try to keep it very simple. Let's get started!

. . .

What Does This Principle Tell Us?

In terms of object-oriented programming, the main idea behind this principle is to always have a high-level code interface with abstraction rather than an implementation detail.

Hold on! I know what you are thinking: "I am a simple frontend developer. Why are you bothering me with these complex terms?"

Let me state it simply for you. For a React application, this principle means:

> "No component or function should care about how a particular thing is done."

Still not clear? OK, let's get our hands dirty with some code!

. . .

A Practical Example

Let's take a very common use case. We are going to make an API call from our component to get some data from a remote source. An implementation can look like this:

```
import React from 'react';

const REMOTE_URL = 'https://jsonplaceholder.typicode.com/users'

export const Users = () => {

  const [users, setUsers] = useState([])

  useEffect(() => {

    fetch(URL)
      .then(response => response.json())
      .then(json => setUsers(json))

  },[])

  return <>
    <div> Users list</div>
    {Object.keys(users).map(user => <div>{user.name}</div>)}
  </>
}
```

Users.js hosted with ❤ by GitHub view raw

Users.js

Look at this component. It depends on some remote data that is fetched right inside the component.

Our `users` component's main responsibility is to render the data. It should not care about how data is fetched or where the data comes from.

This component knows too much — and that's a problem.

. . .

Why?

Well, let's say you have ten other components and all of them fetch their own data.

Now your manager comes along and tells you to use `axios` instead of `fetch`

You are in trouble! Now you have to go into each file and refactor the logic to use `axios`.

But life is not so simple! After a few days, your manager comes again and tells you to implement caching.

You have to do the same thing once again.

Thus, it increases the chance of introducing a bug in your software. Also, the code becomes unmaintainable and valuable time is wasted.

. . .

So What Should We Do Then?

Let's introduce a data-fetching Hook and abstract away our logic outside our component because that's exactly what this principle tells us. To depend on abstraction, remember?

```
import {useSwfetch} from "swsr";

export const useFetch = (URL) => {

  const [data , setData] = useState([])

  useEffect(() => {

    fetch(URL)
      .then(response => response.json())
      .then(json => setData(json))

  },[])

  return data;

}
```

useFetch.js hosted with ❤ by GitHub view raw

useFetch.js

Now use this Hook inside our `users` component:

```
import React from 'react';

const REMOTE_URL = 'https://jsonplaceholder.typicode.com/users'

export const Users = () => {

  const users = useFetch(REMOTE_URL)

  return <>
    <div> Users list</div>
    {Object.keys(users).map(user => <div>{user.name}</div>)}
  </>
}
```

Users.js hosted with ❤ by GitHub view raw

Users

Notice a great thing about this solution: Your `useFetch` Hook doesn't care about who is calling it. It just takes a `URL` as an input and returns the data.

Now all other components can take advantage of the Hook that we just wrote. And our `users` component no longer depends on the concrete details on how the data is coming back or which library is being used!

. . .

More Advanced Usage

Now let's satisfy your manager with basic caching functionality:

```
import {useSwfetch} from "swsr";

export const useFetch = (URL) => {

  const [data , setData] = useState([])

  useEffect(() => {

    const cachedata = localstorage.getItem(URL)

    if(cachedata!= {
      setData(JSON.parse(cachedata))
    },[])
    else{
      fetch(URL)
      .then(response => response.json())
      .then(json => setData(json)
      }

  },[URL])

  useEffect(() => {
    localstorage.setItem(URL , JSON.stringify(data))
  },[data])

  return users;

}
```

useFetch.js hosted with ❤ by GitHub view raw

useFetch.js

You have to change the code is only one place now. That's great! Let's say you need to show API errors as a toast. Can you do that now? If so, then you got my point.

. . .

How To Detect It

- In most cases, if you are violating the single-responsibility principle, then you might also be violating the dependency inversion principle.
- For any component, look into the import section at the top. If you are importing some library that's not responsible for displaying something (e.g. a toast or modal), then you might be violating the principle.

Previous Articles in this Series

1. Single Responsibility Principle
2. Open Closed Principle
3. Liskov Substitution Principle
4. Interface Segregation Principle

That's all for today. I hope you enjoyed this article as well as this series.

Have something to say? Get in touch with me via LinkedIn

Top 7 Libraries for Blazingly Fast React.JS Applications
Some must-have tools for a rock-star developer
betterprogramming.pub

Thanks to Anupam Chugh

350 · 10

350 · 10

Sign up for Coffee Bytes
By Better Programming
A newsletter covering the best programming articles published across Medium Take a look.
Notifications may be sent to you according to our privacy notice.
Get this newsletter

More from Better Programming · Follow
Advice for programmers.

Jennifer Fu · May 5, 2021
Is YAML Ready For JavaScript?
Compare XML, JSON, and YAML to see the trends for data formatting — What is the world's most popular programming language? DevOps vendor Datree.io said it is YAML. After analyzing millions of open source GitHub…
Programming · 10 min read

Share your ideas with millions of readers. Write on Medium

Claire Q. Costa · May 5, 2021
9 Must-Read Python Books in 2021
Which book would you recommend someone to read? — "The joy of coding Python should be in seeing short, concise, readable classes that express a lot of action in a small amount of clear code — not in reams of trivial cod…
Programming · 8 min read

Rakia Ben Sassi · May 5, 2021
What Wikipedia Can't Tell You About Serverless
The unconventional guide to the post-virtual machine and post-container paradigm — What is serverless? This question puzzled me for a long while. When I started learning about it, the first lie I've discovered was that, only…
Programming · 11 min read

Jesús Castro · May 5, 2021
3 Lines of Code That Really Messed Up My Day
Just a few lines that made me yell at random things on my desk — Coding is a lot of fun — usually. Sometimes you just sit there and wonder if this line of code was really worth all the trouble that it caused. …
Programming · 4 min read

Samuel Martins · May 5, 2021
5 Tips To Make Better Technical Decisions as a Software Engineer
Add this underrated skill to your arsenal — How do you make technical decisions? That is a tough question to answer. It had me scratching my he…
Programming · 5 min read

Read more from Better Programming

Recommended from Medium

Dr. Derek Aug… in Better Progra… **Compiling vs. Polyfilling in JavaScript**
Robin Jongs **Forgotten Callbacks, Getting a Closure**
Mohammad… in JavaScript in Plai… **11 Ways to Improve Your Technical Resume**
Ishandhanani **Promises vs SetTimeout | JAVASCRIPT**
Anagha Kumbhojkar **ASCII and UTF-16**
Nadhiraa Dayini **Don't use Redux!**
Amit Pandya in TechvsBoys **Moving A Laravel Webpack Project To Vite**
Dhananjay Nandkumar Chavan **Basic Lightning Web Component Quiz App**

Get started · Sign in
Search
Mohammad Faisal
3.4K Followers
Software Developer | https://www.linkedin.com/in/faisal-ibn
Follow
More from Medium
Chris… in Better… Top React Development Trends for 2022
Kim… in JavaScr… Tips to Improve Your React Code
Ha… in Web De… JavaScript concepts you need to know in…
Coels… in Floatin… How to Manage the State for a DropDownList…
Help Status Writers Blog Careers Privacy Terms About Knowable