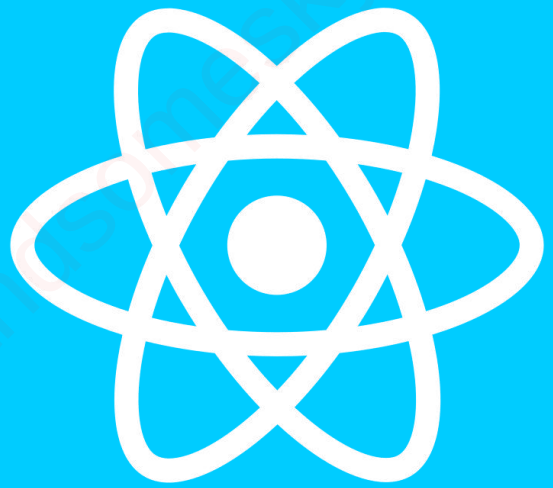
匠人学院
JR ACADEMY



React 基础与进阶

大漠穷秋 2023-08-01

课程前言

1. 本课程内容面向初学者，不要求开发经验。
2. 为了方便初学者入门，本课程采用纯 JS 编码，不引入 TypeScript 。
3. 组件全部采用官方倡导的 functional component 写法，不要求学员有 OO 的概念。
4. 课程内容采用当前最新的 v18 版本。
5. 内容共10章，按照从简单到复杂排列，随着内容的推进，最终构建出一个完整的项目。
6. 每一章内容都先学会使用 API ，再解释原理，部分有难度的内容已加标注，初学者可以暂时忽略。
7. 内容中标注成红色的部分，是实际企业开发中的常用技术点，需要重点掌握。
8. 为了照顾初学者和转码的同学，本课程将尽量详细解释知识点，学生后期回放复习时可以尝试倍速。
9. 本课程所有演示代码都在这个 repo 中： https://gitee.com/mumu-osc/learn-react 。

内容提纲-第1次课

匠人学院 JR ACADEMY

第1章 React 框架的起源与版本演进历程

第2章 Quick Start

2.1 配置开发环境

2.2 快速上手 create-react-app

2.3 初步体验 JSX/useState/useEffect

2.4 拆分组件

第3章 组件基础

3.1 组件化思想

3.2 用 SASS 与 Bootstrap 构建组件的外观

3.3 强大的模板语法 JSX

3.4 props & state

3.5 props 与 state 的区别和联系

3.6 组件间通讯

3.7 综合实例：构建一个结构复杂的页面，理解 React 切分组件的思想

内容提纲-第2次课

第4章 路由与导航

4.1 React Router 快速上手

4.2 嵌套路由

4.3 HashRouter 与 BrowserRouter

4.4 理解 React Router 的运行机制

第5章 处理表单

5.1 非受控表单

5.2 受控表单

5.3 表单校验 Form Validation

内容提纲-第3次课

匠人学院 JR ACADEMY

第6章 与服务端交互

6.1 与服务端交互的几种典型方式介绍(Ajax/Fetch/Axios)

6.2 Axios 快速上手

6.3 用拦截器处理通用的操作

6.4 封装可复用的 Axios 服务

内容提纲-第4次课

匠人学院 JR ACADEMY

第7章 组件进阶（这一章的内容整体偏难，但是面试和实际业务开发都是必备的）

7.1 React 组件的生命周期

7.2 VDOM（本节有难度，初学者有了解即可）

7.3 类组件与函数式组件（以及为什么 functional 模式呼声比较高）

7.4 复合组件

7.5 高阶组件

7.6 开源组件库

7.7 企业开发中的业务组件库

7.8 动态生成页面与 React 低代码

内容提纲-第5次课

匠人学院 JR ACADEMY

第8章 详解 React Hooks

8.1 React 内置的 Hooks

8.2 理解 useState

8.3 理解 useEffect

8.4 自定义 Hooks

8.5 用 useTranslation 钩子实现组件的 i18n

内容提纲-第6次课

匠人学院 JR ACADEMY

第9章 状态管理

9.1 状态管理的基本概念

9.2 Context （React 官方在框架中内置的状态管理工具）

9.3 React Redux （状态管理中间件）

第10章 自动化测试

10.1 单元测试，安装配置 Jest

10.2 单元测试，详解 Jasmine 语法

10.3 集成测试，安装配置 Cypress

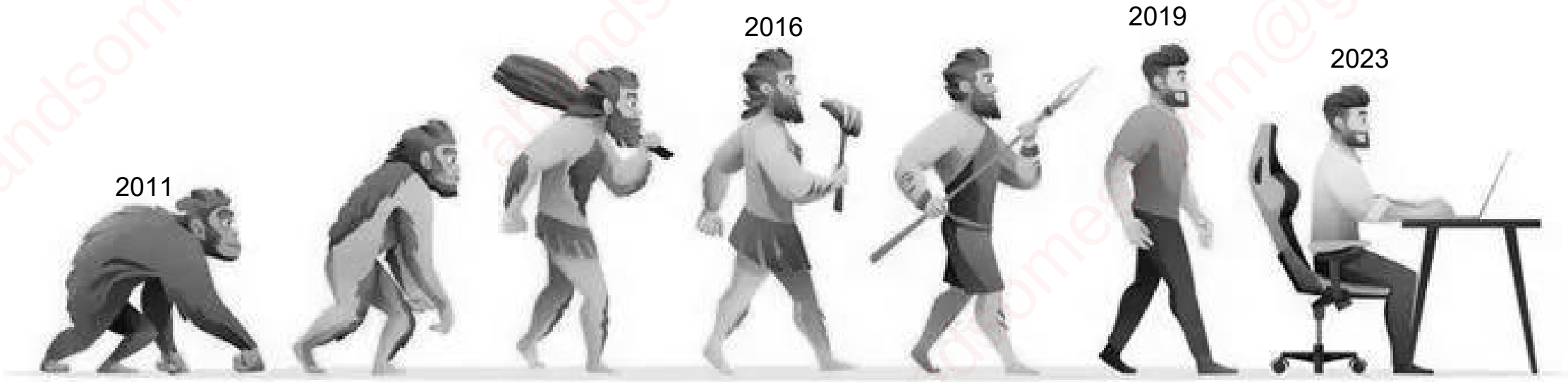
10.4 集成测试，运行集成测试用例

推荐阅读

- https://react.dev
- https://create-react-app.dev/
- https://redux.js.org/
- https://reactrouter.com/en/main
- https://remix.run/docs/en/main/tutorials/blog
- VDOM 处理流程（非常详细的图）：https://medium.com/@rajaraodv/the-inner-workings-of-virtual-dom-666ee7ad47cf
- 分析Diff算法：https://calendar.perfplanet.com/2013/diff/
- https://developer.mozilla.org/en-US/
- NiceFish-React（完整功能，带服务端）：https://gitee.com/mumu-osc/NiceFish-React

匠人学院
JR ACADEMY

第1章 React 框架的起源与版本演进历程



React 起源：Facebook 失败的技术架构

Facebook Abandoning HTML5 to Speed Up iOS App

Wednesday June 27, 2012 11:01 AM PDT by Jordan Golson

Facebook engineers are completely rewriting the social network iOS app to make it faster. The app, which has been criticized by many users for being extremely slow, was built using the HTML5 Web-based programming language.

It makes the app much easier for Facebook's engineers to port to other platforms like Android or Windows Mobile -- but it fails to take advantage of the iOS operating system.

As a result, the app -- which is used for everything from posting status updates to checking in to restaurants -- is much slower than comparable apps written to be a native app from the bottom up. Facebook knows this and is giving the app a complete rewrite in Objective-C, turning it into a true iPhone app and making it much faster. Nick Bilton, reporting for *The New York Times*:

> *I had the opportunity to see the yet-unreleased iPhone application and its fast. Blazing fast. The two developers I spoke with said the new application is currently being tested by Facebook developers and is expected to be updated this summer.*
>
> *Don't get too excited, though. The new Facebook app looks exactly like the old one. Facebook hasn't changed the design, but rather focused on speed.*

According to Bilton, the new and improved Facebook app should be released next month.

Facebook for iOS is available free from the App Store. [Direct Link]

Next Article

Top Stories: iPhone 15 Pro Rumors, iOS 17 Beta 4, Vision Pro Developer Kit, and More

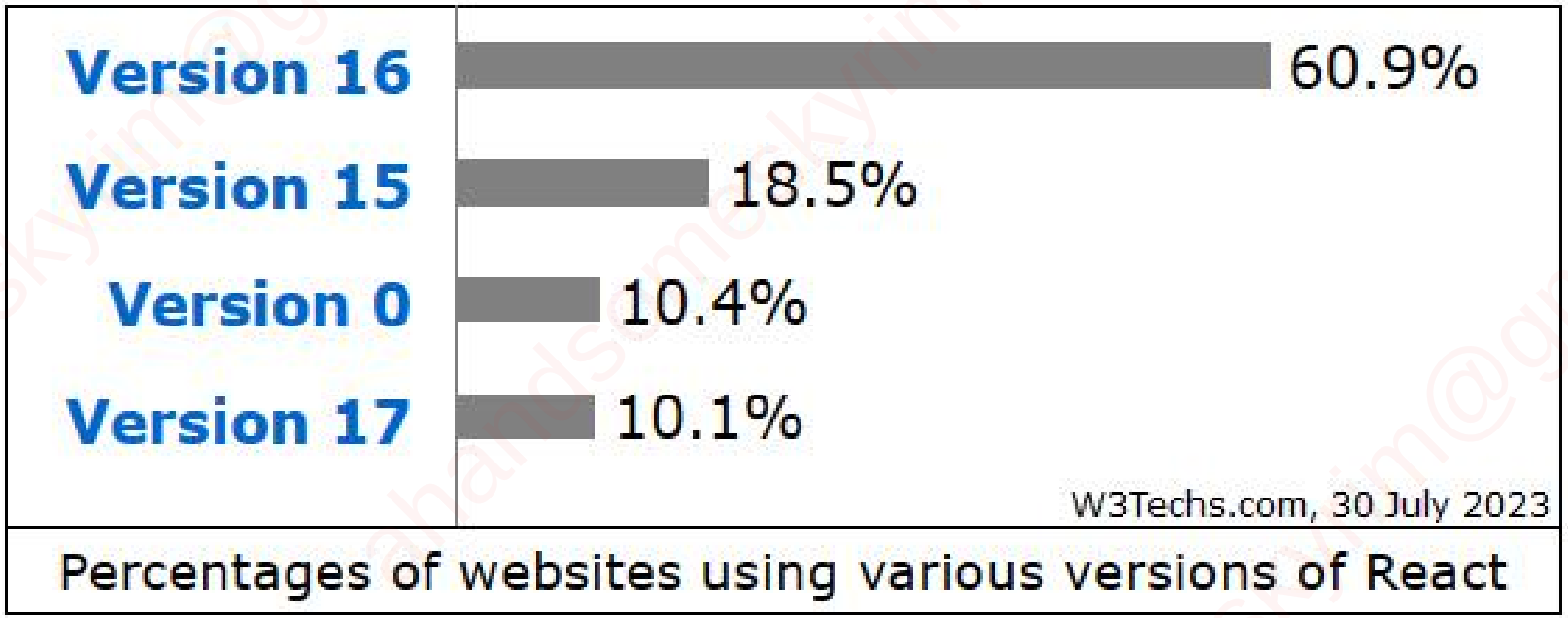
2012年，Facebook 踩了一个大坑，它是第一个尝试用 H5 来改写自家 APP 的一线大厂，最终因为 APP 性能太差被迫放弃。*（PS：目前，腾讯在尝试以 Electron 为基础，实现跨平台目标，或许是一个可行的方法。微软在 React Native 方向也在做重点的投入。但是在 10 年前的 2012 年，无论是硬件还是技术生态，都不太可能实现，FB 明显掉进了大坑。）*

React 版本演进历程（当前时间 2023 年 8 月）

React 的创造者 Jordan Walke

- 2011年，Jordan Walke 发布了一款名叫 FaxJS 的框架，用在 Facebook 的新闻列表模块中。前身是一个名为 XHP 的项目，XHP 是一个 PHP 扩展，用于在服务器端构建用户界面。
- 2012年，FaxJS 被使用在 Instagram 中。
- 2013年5月，Jordan 宣布 FaxJS 在 github 上开源，命名为 React ，此时的版本是 0.3 ，此时仅仅是一些核心的概念和思想，整体还不成熟。
- 2015年3月，React Native 开源，支持一套代码适配多种移动端平台。
- **2016，React v15 发布，这是一个重要里程碑版本，React 框架从此时开始快速流行起来。**
- 2020年8月，React v17 进入候选发布阶段。 ***（PS：这个版本没有 break changes，API 与之前完全一致，这也是 React 非常好的一个特征。）***
- 2021年，Jordan 离开 Facebook 。
- 2022年3月，React v18 发布，主要引入了并发渲染模式。 ***（PS：这个版本有非常多的 API 发生了变更。v18 是当前最新的版本。）***

React 版本使用情况（当前时间 2023 年 8 月）



目前绝大多数线上的站点停留在 v18 之前，本教程中的内容和示例代码采用最新的 v18 版本。

数据来源： https://w3techs.com/technologies/details/js-react

React Core Features（区别与其它框架）

1. VDOM （工作在底层的 VDOM 机制确保了 React 的渲染速度很快）
2. JSX （JavaScript XML 确保了非常灵活强大的模板语法）
3. State Management 状态管理（确保了数据与 UI 的分离，以及单一数据来源）

React 框架有很多技术特性，以上是它最有特色的地方，也是区别于其它前端框架的地方。本教程最核心的内容都围绕以上技术特性展开。

React 生态圈

1. **Next.js**：首次发布于 2017 年，主打服务端渲染（SSR）和 App Router 这2个核心特性。Google 给这个框架贡献了很多代码，很多大企业在自己的产品中使用了这个框架，包括 Apple，Google，**Tiktok** 。2020 年，Vercel 公司提供了 2100 万美元的基金支持。目前（2023-08）Next.js 最新版本是 v13.4，它是 React 官方推荐的工具链。

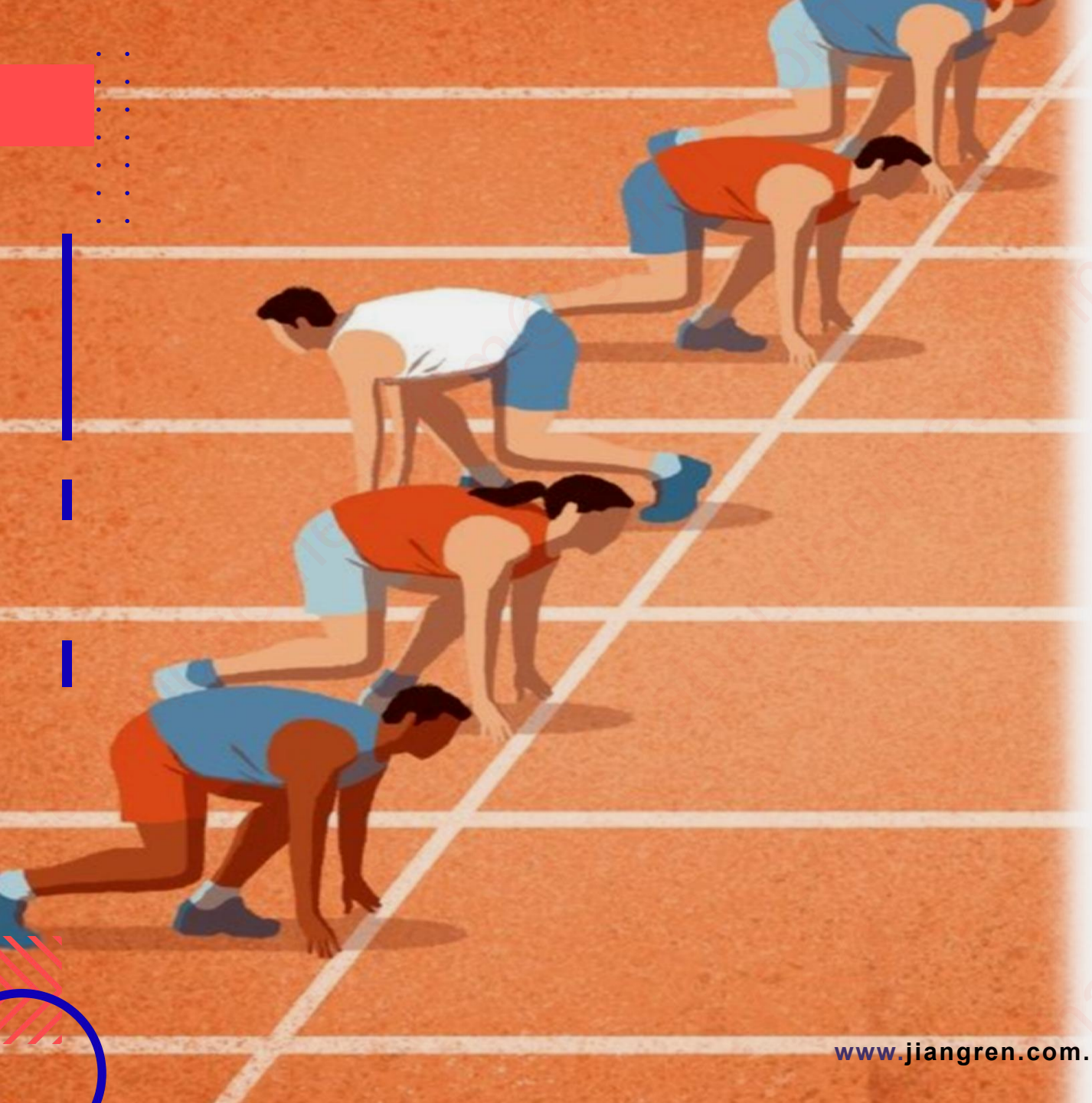
2. **Remix**：由著名的电商平台 Shopify 开发并维护，最早发布于 2021 年 9 月。

3. **Gatsby**：由 Netlify 公司开发并维护，主打 CMS 型的应用。

4. **Expo**：由 Expo 公司开发并维护，主打 Native App，它为 React Native 提供了强大的 SDK 支持，用来开发跨平台的 App 。

5. **React Native Windows**: 目前，微软正在大力推进一个名为 React Native Windows 的项目，当前最新的版本为 0.72，https://microsoft.github.io/react-native-windows/ 。（***据传，微软对 React Native 提供了资金支持，未证实。***）

从 2017 年 开始，React 生态圈迎来了爆发式的发展，各种基于 React 的框架、组件库、模板项目层出不穷，本教程选用 React 官方的 create-react-app 作为入手点，因为它足够简洁，对初学者友好。

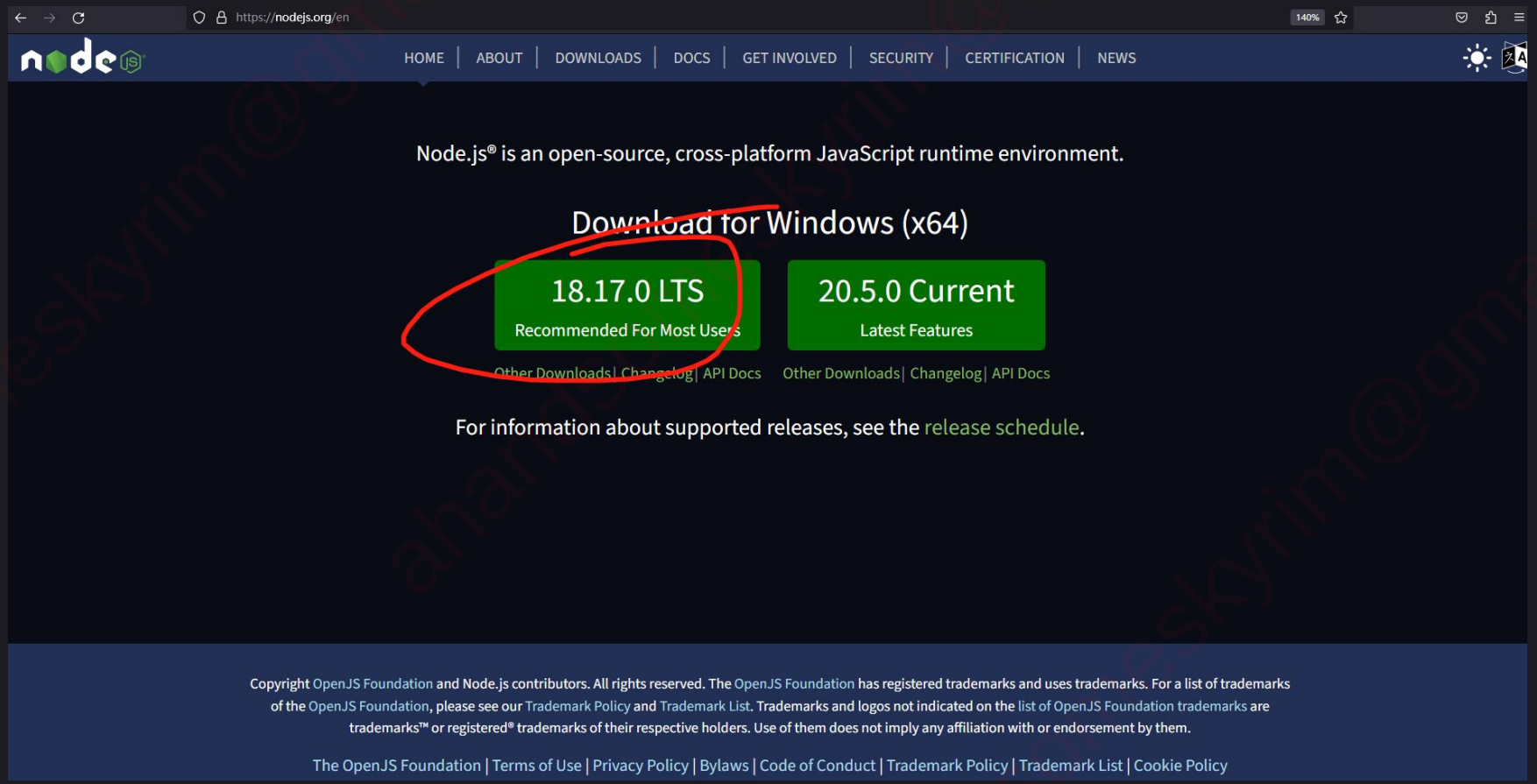


匠人学院
JR ACADEMY

第2章 Quick Start

2.1 配置开发环境

安装 nodejs



选 LTS 版，最新版可能会有坑

确认环境是否 ready

```
F:\>npm
npm <command>

Usage:

npm install        install all the dependencies in your project
npm install <foo>  add the <foo> dependency to your project
npm test           run this project's tests
npm run <foo>      run the script named <foo>
npm <command> -h   quick help on <command>
npm -l             display usage info for all commands
npm help <term>    search for help on <term> (in a browser)
npm help npm       more involved overview (in a browser)

All commands:

    access, adduser, audit, bugs, cache, ci, completion,
    config, dedupe, deprecate, diff, dist-tag, docs, doctor,
    edit, exec, explain, explore, find-dupes, fund, get, help,
    hook, init, install, install-ci-test, install-test, link,
    ll, login, logout, ls, org, outdated, owner, pack, ping,
    pkg, prefix, profile, prune, publish, query, rebuild, repo,
    restart, root, run-script, search, set, shrinkwrap, star,
    stars, start, stop, team, test, token, uninstall, unpublish,
    unstar, update, version, view, whoami

Specify configs in the ini-formatted file:
    C:\Users\felix\.npmrc
or on the command line via: npm <command> --key=value

More configuration info: npm help config
Configuration fields: npm help 7 config

npm@9.5.0 C:\Users\felix\AppData\Roaming\nvm\v18.15.0\node_modules\npm
```

执行 npm 命令，确认环境变量是好的，如果不行，需要把 nodejs 加到环境变量中

nvm-nodejs version manager

```
E:\github>nvm

Running version 1.1.10.

Usage:

  nvm arch                    : Show if node is running in 32 or 64 bit mode.
  nvm current                 : Display active version.
  nvm install <version> [arch] : The version can be a specific version, "latest" for the latest current version, or "lts" for the
                                most recent LTS version. Optionally specify whether to install the 32 or 64 bit version (defaults
                                to system arch). Set [arch] to "all" to install 32 AND 64 bit versions.
                                Add --insecure to the end of this command to bypass SSL validation of the remote download server.
  nvm list [available]        : List the node.js installations. Type "available" at the end to see what can be installed. Aliased as ls.
  nvm on                      : Enable node.js version management.
  nvm off                     : Disable node.js version management.
  nvm proxy [url]             : Set a proxy to use for downloads. Leave [url] blank to see the current proxy.
                                Set [url] to "none" to remove the proxy.
  nvm node_mirror [url]       : Set the node mirror. Defaults to https://nodejs.org/dist/. Leave [url] blank to use default url.
  nvm npm_mirror [url]        : Set the npm mirror. Defaults to https://github.com/npm/cli/archive/. Leave [url] blank to default url.
  nvm uninstall <version>     : The version must be a specific version.
  nvm use [version] [arch]    : Switch to use the specified version. Optionally use "latest", "lts", or "newest".
                                "newest" is the latest installed version. Optionally specify 32/64bit architecture.
                                nvm use <arch> will continue using the selected version, but switch to 32/64 bit mode.
  nvm root [path]             : Set the directory where nvm should store different versions of node.js.
                                If <path> is not set, the current root will be displayed.
  nvm [--]version             : Displays the current running version of nvm for Windows. Aliased as v.


E:\github>nvm list

    19.8.1
  * 18.15.0 (Currently using 64-bit executable)
    14.21.3
```

用 nvm 来管理多个 nodejs 版本
（原因：某些模块限定了 nodejs 版本，不切换版本无法运行）

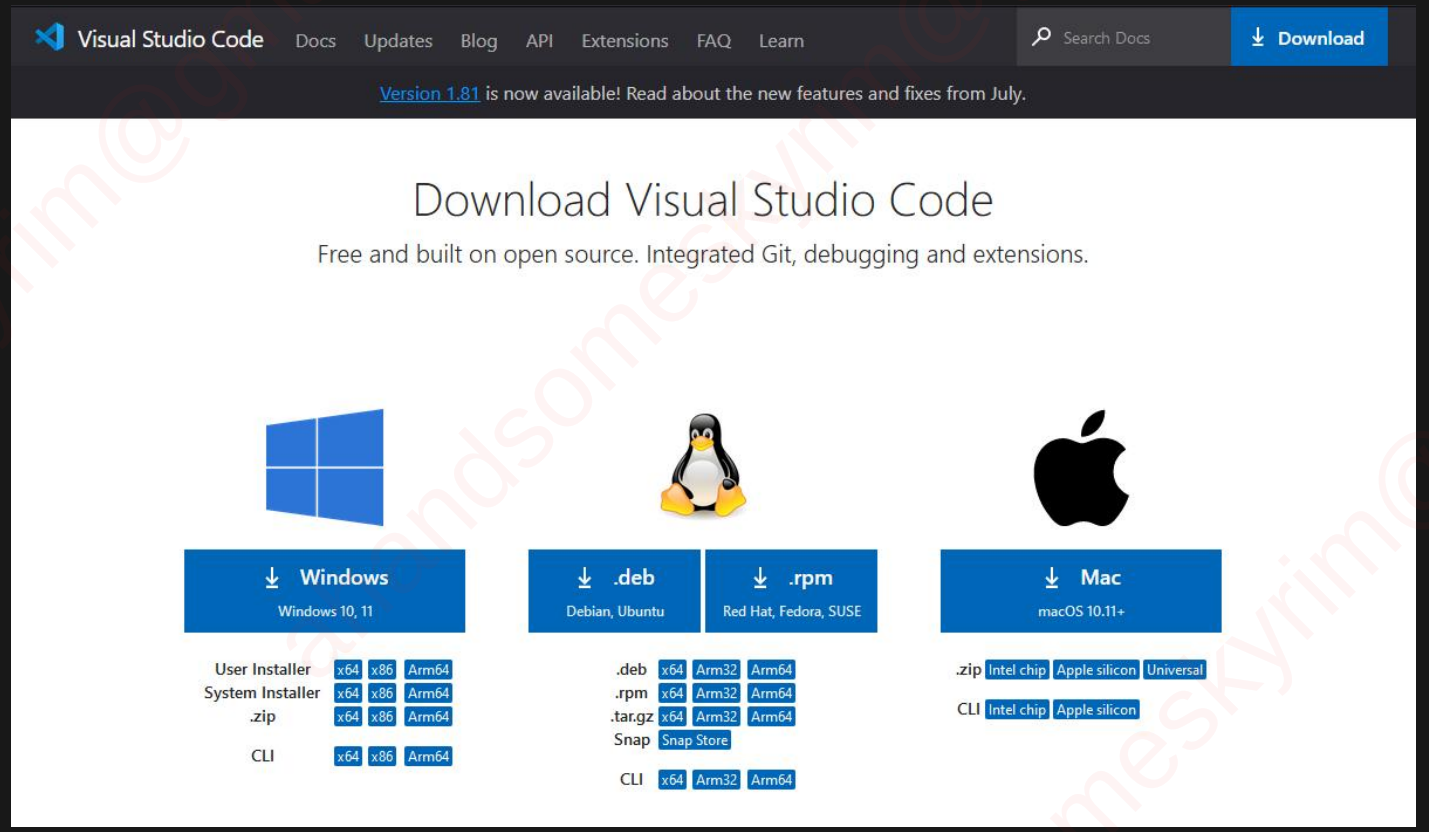
nrm-nodejs registry manager

匠人学院 JR ACADEMY

```
E:\github-my>nrm ls

* npm ---- https://registry.npmjs.org/
  cnpm --- http://r.cnpmjs.org/
  taobao - https://registry.npm.taobao.org/
  nj ----- https://registry.nodejitsu.com/
  rednpm - http://registry.mirror.cqupt.edu.cn/
  npmMirror  https://skimdb.npmjs.com/registry/
  edunpm - http://registry.enpmjs.org/


E:\github-my>nrm test

* npm ---- 816ms
  cnpm --- 461ms
  taobao - 281ms
  nj ----- Fetch Error
  rednpm - Fetch Error
  npmMirror  1194ms
  edunpm - Fetch Error
```

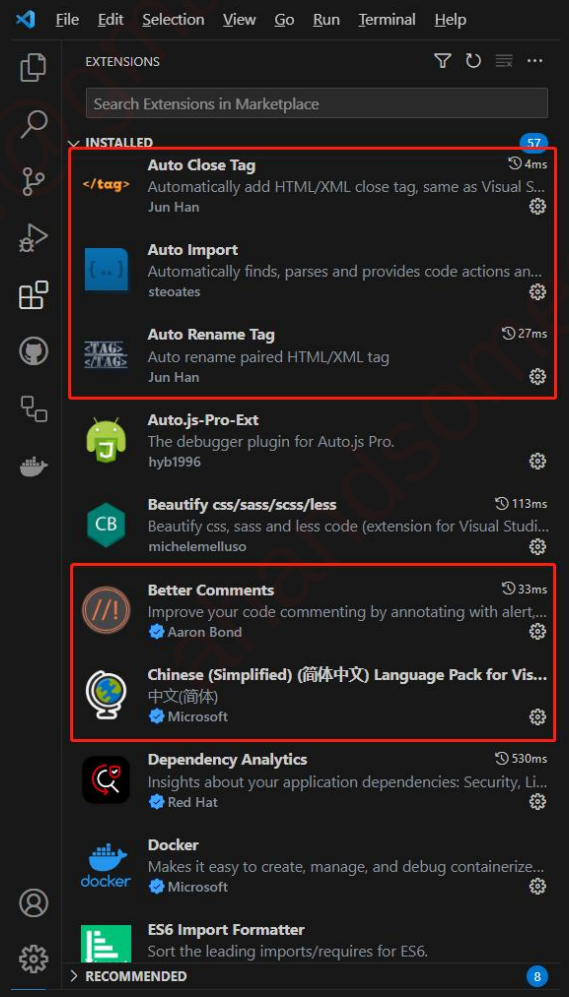
nrm 是中文开发者必备，你懂的

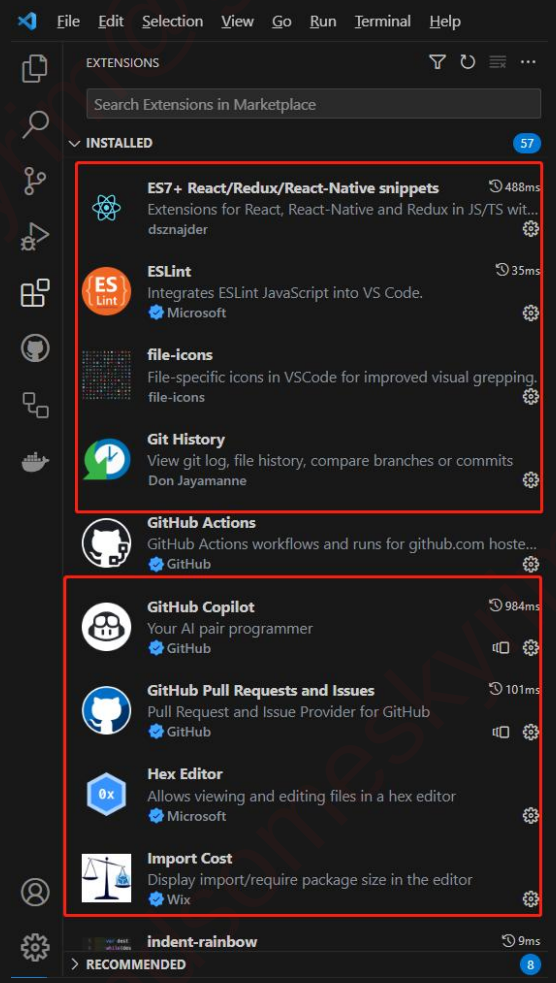
安装 VSCode



目前，VS Code 是最受欢迎的前端开发工具。

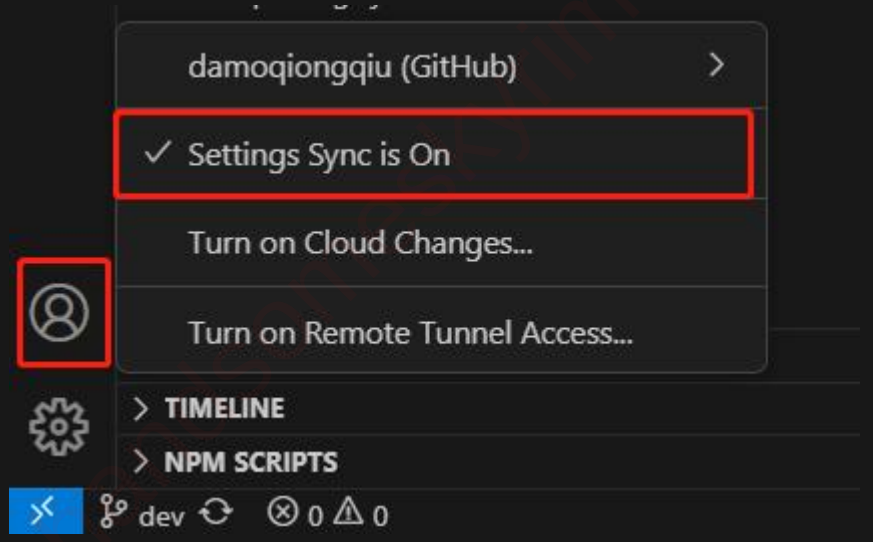
VSCode 必备的一些插件





这些插件会让你的日常开发过程更加高效，也是很多企业组织必备的插件。

VSCode 打开自动同步

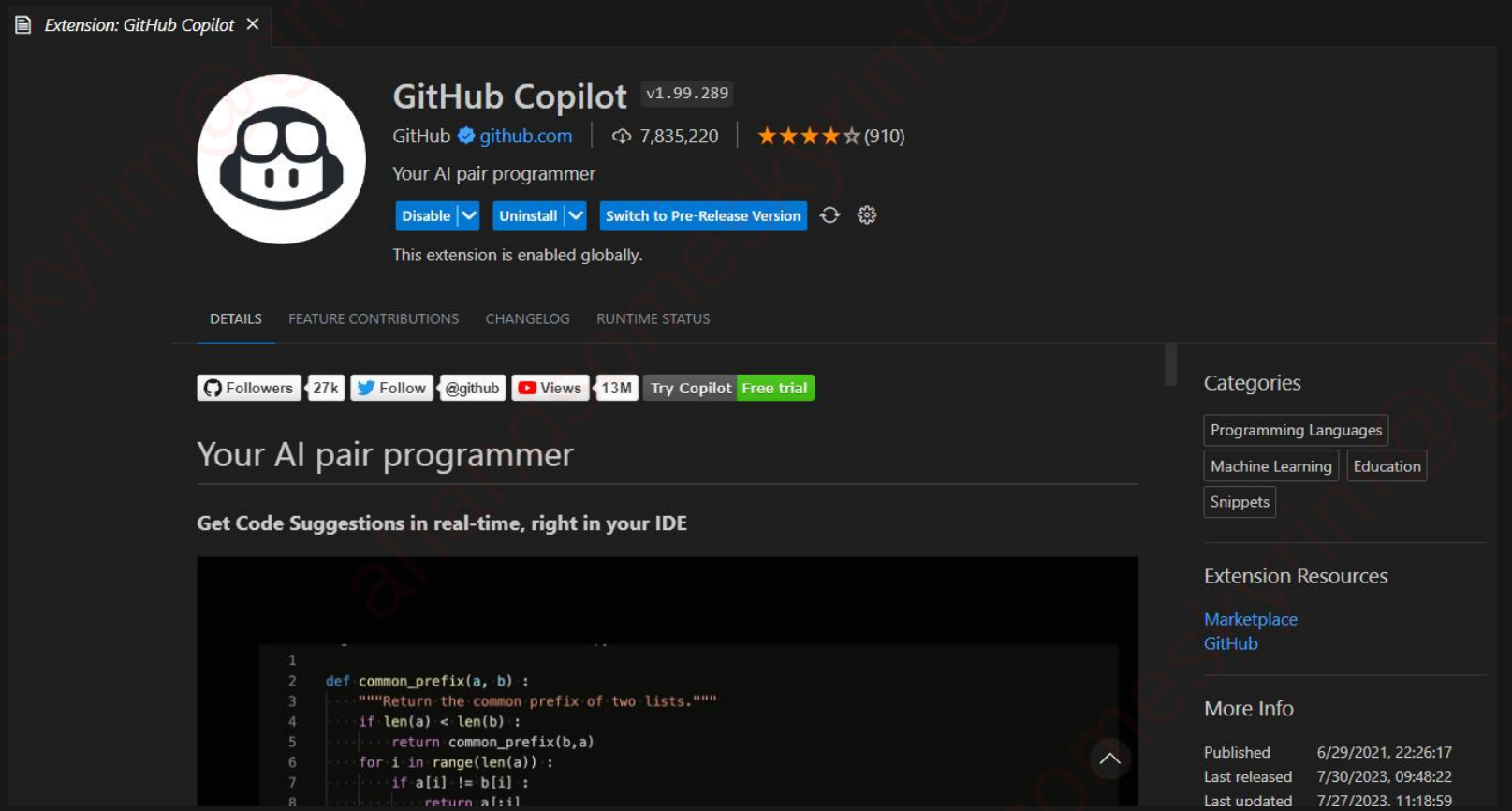


登录并打开自动同步，VS Code 会把你本地的配置都同步到云端

如果你本地的环境被破坏，以后只要登录账号，VS Code 就可以自动用云端的配置来恢复你以前的工作状态

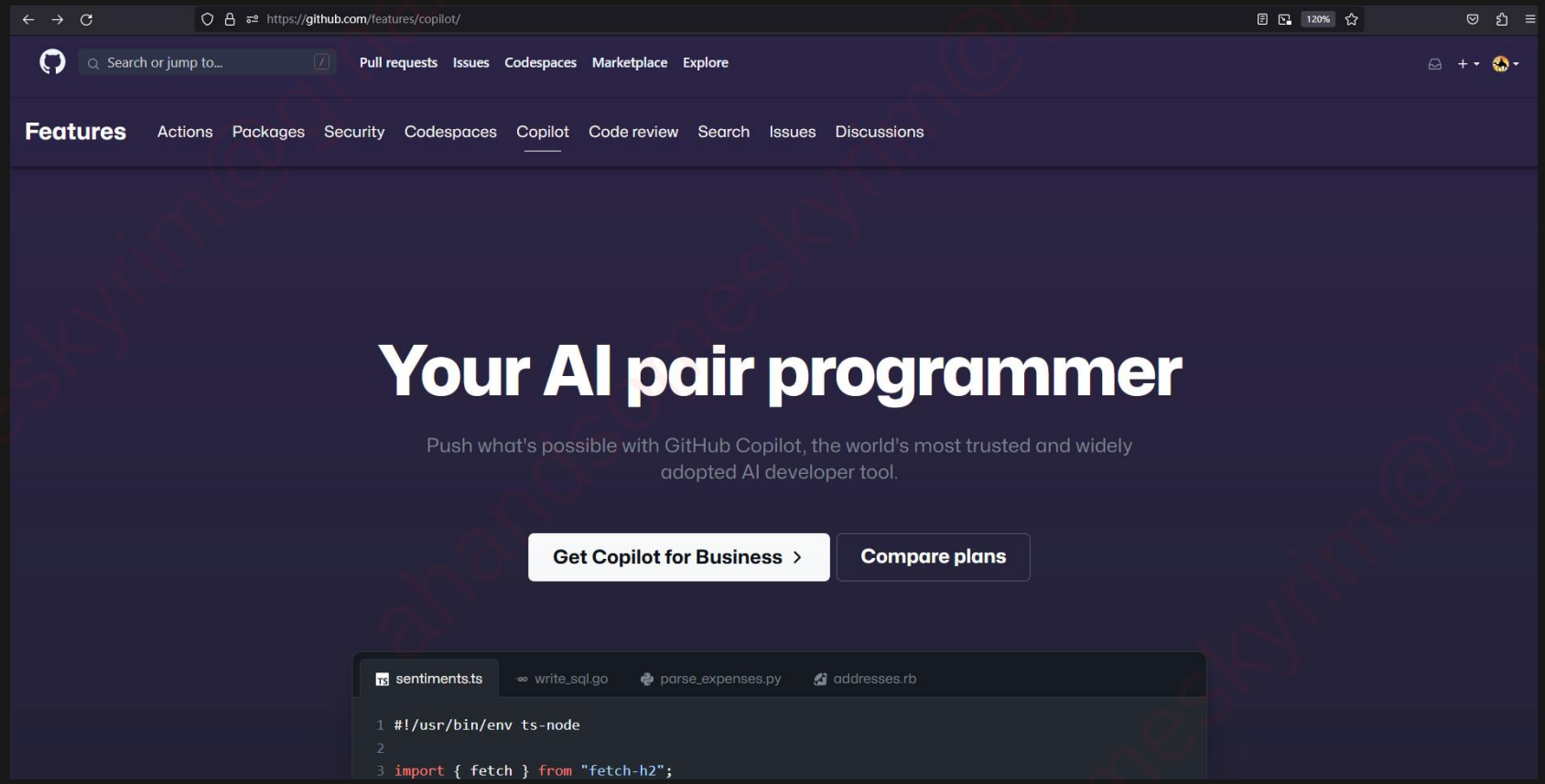
VSCode 安装配置 Copilot

匠人学院 JR ACADEMY



VS Code 安装配置 Copilot，它会帮助你生成很多代码，它甚至还会慢慢学会你的编码风格。

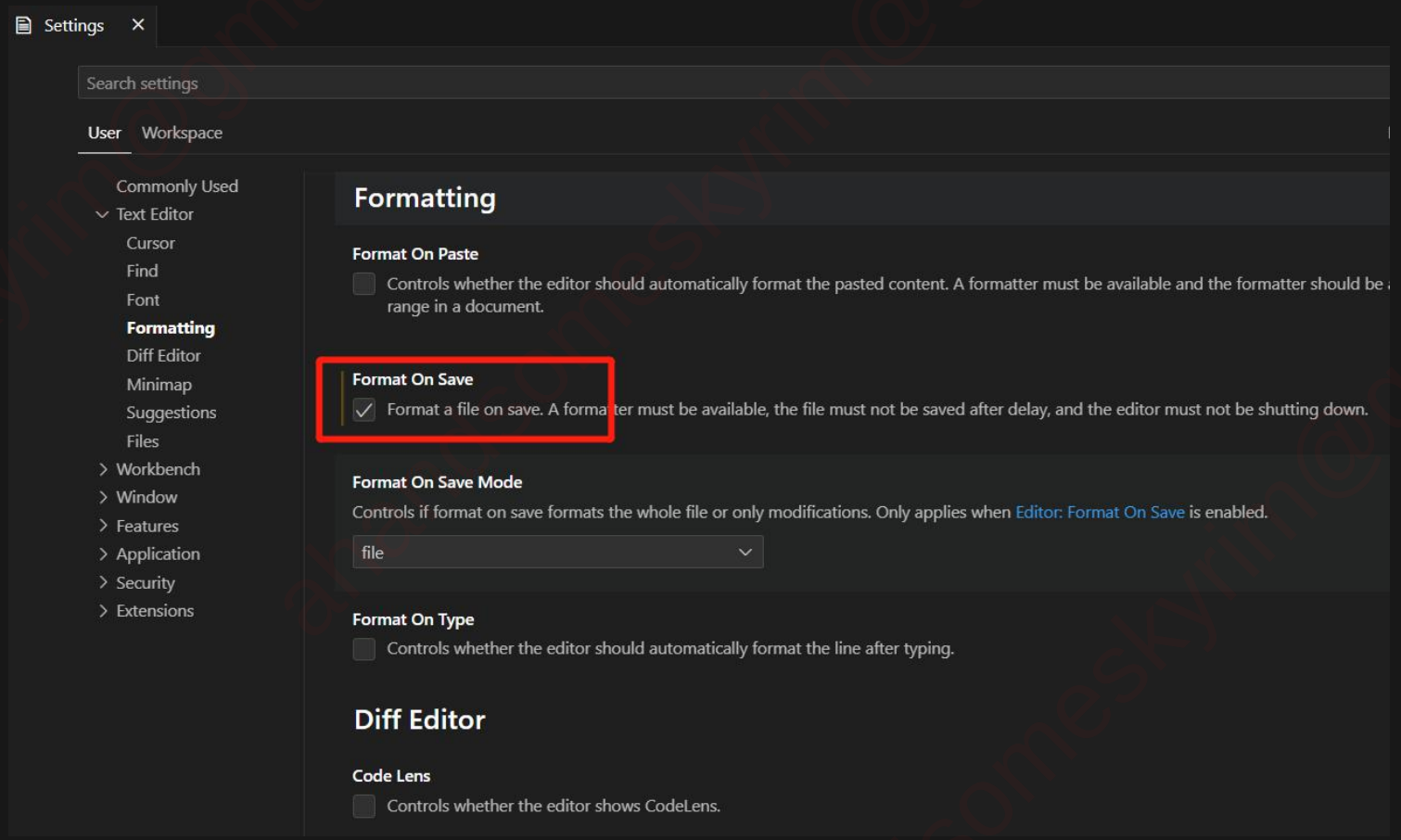
VSCode 安装配置 Copilot



安装完成之后，插件会引导你用 github 账号授权登录，然后排队等待平台为你开通（目前用的人多，可能要等比较久）。

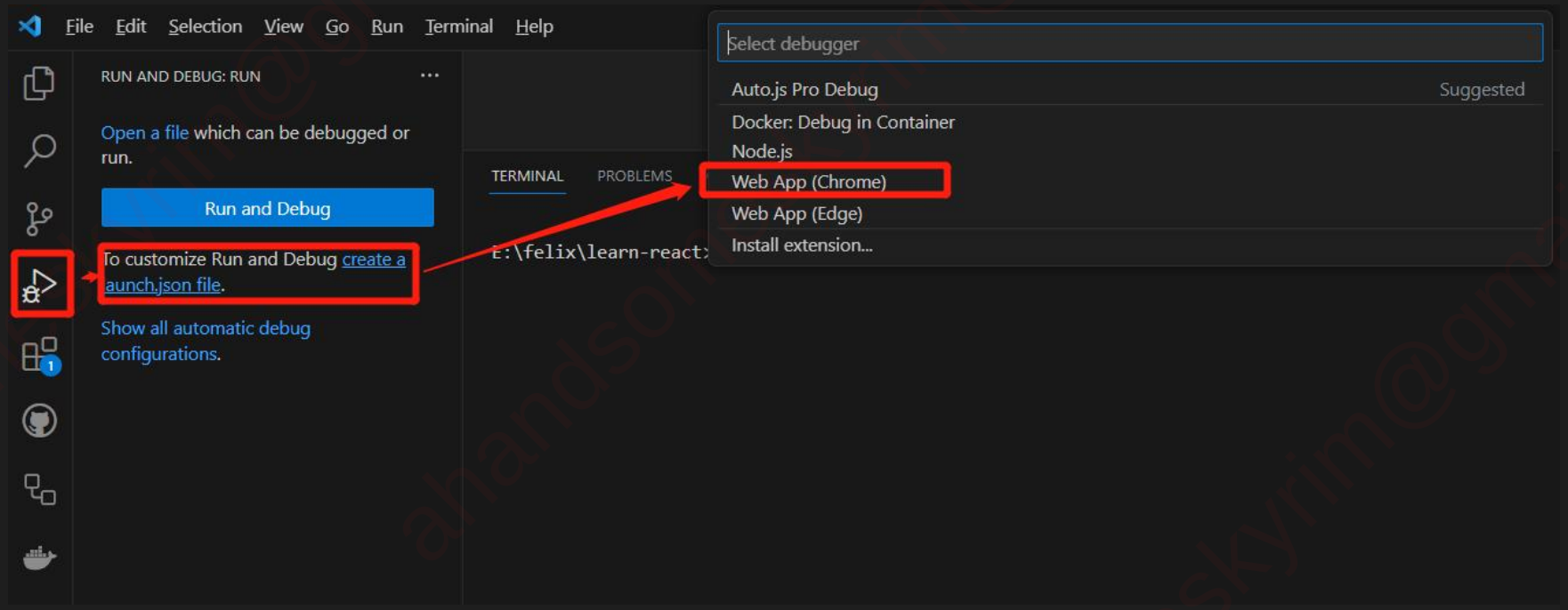
--- 演示 Copilot 的效果 ---

VSCode 开启自动格式化



VS Code 安装 Prettier 插件，开启保存时自动格式化代码，保证整个团队代码风格统一。

VSCode 开启调试

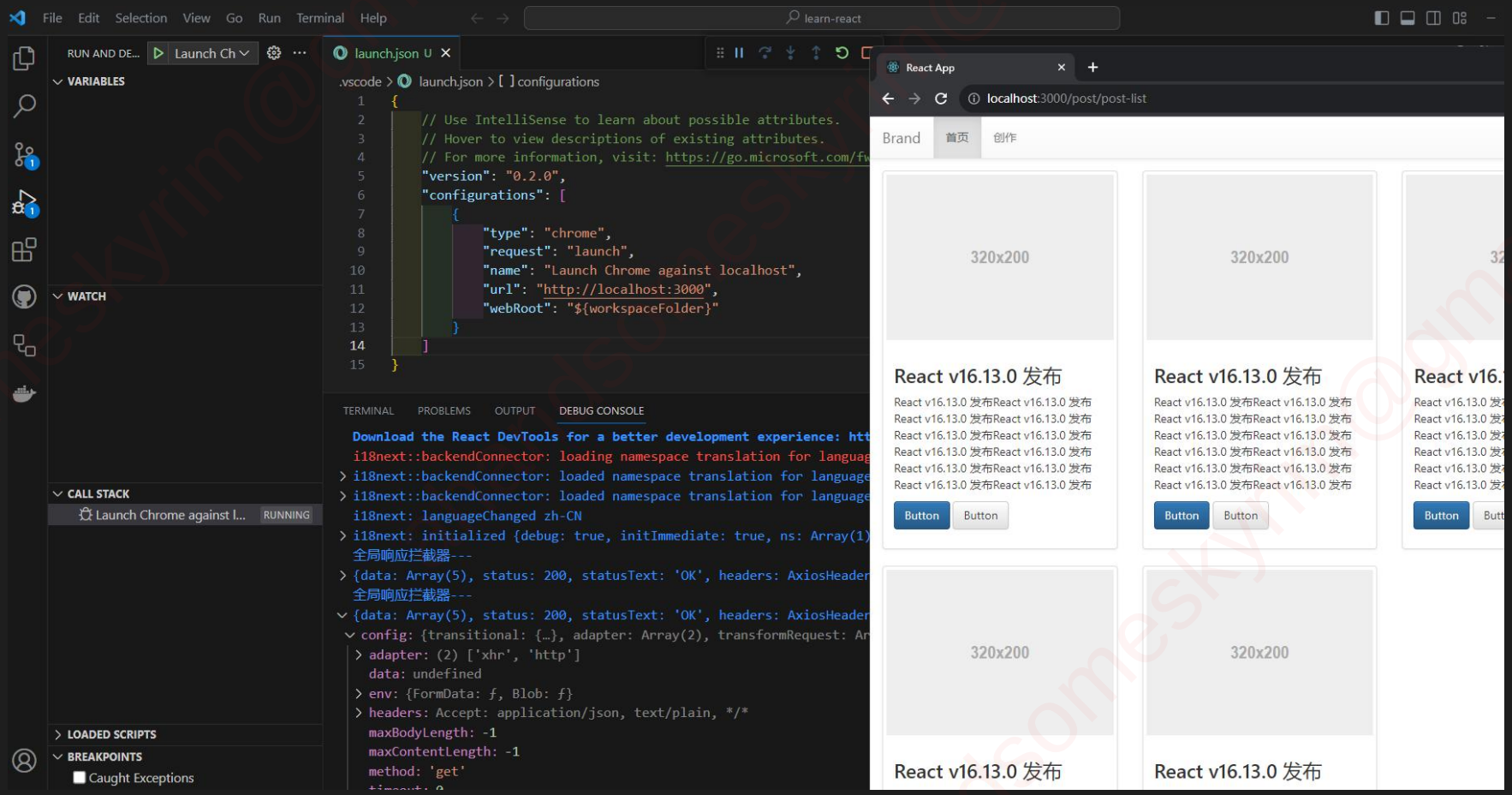


VS Code 开启调试，连接浏览器进行断点调试

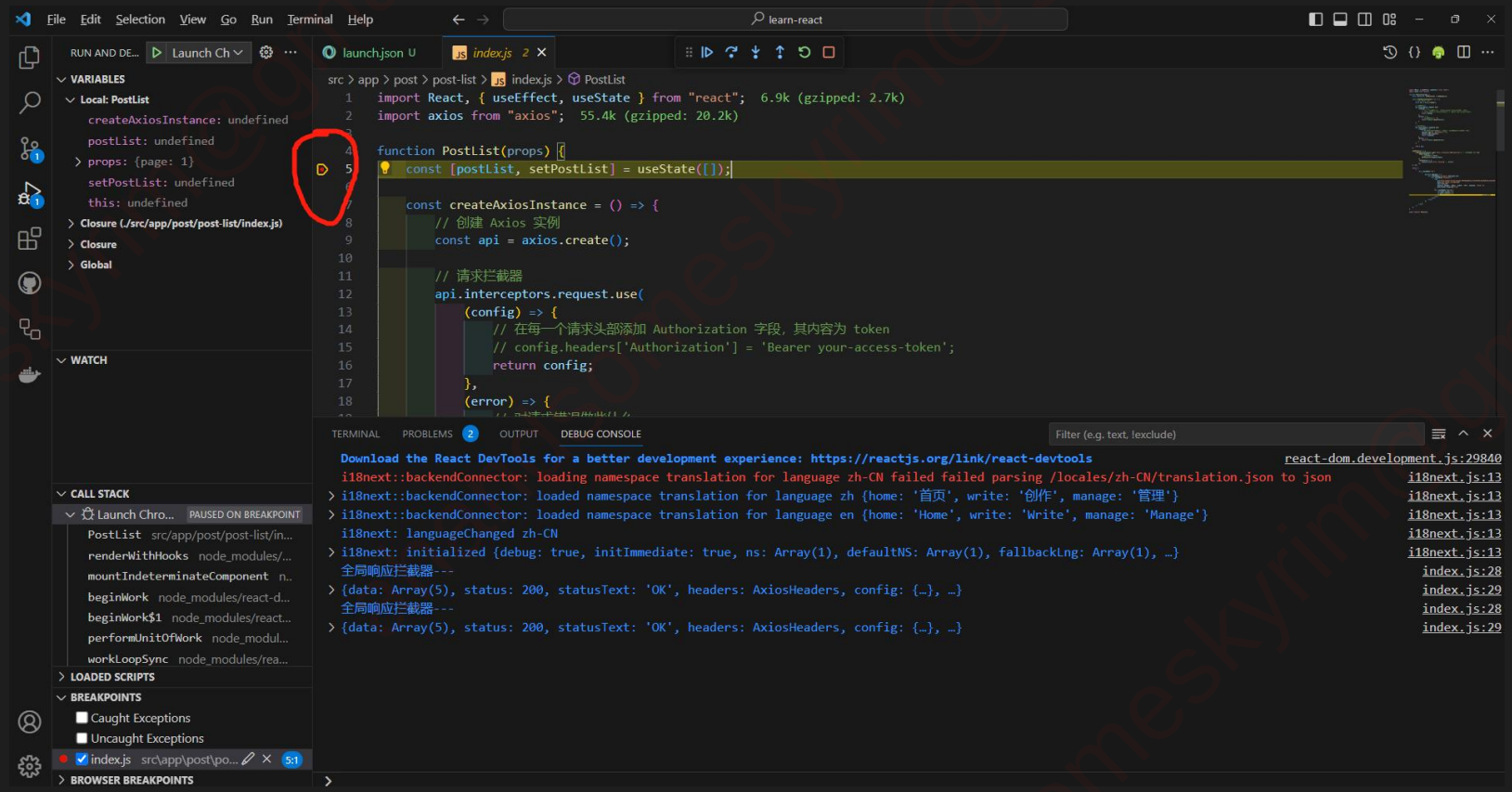
（备注：演示一下，方便初学者理解）

VSCode 开启调试

匠人学院 JR ACADEMY



VSCode 开启调试



在源码里面打断点调式，比在 Chrome 里面调试更方便

初学的同学，可以利用这些工具更好地理解代码的执行过程

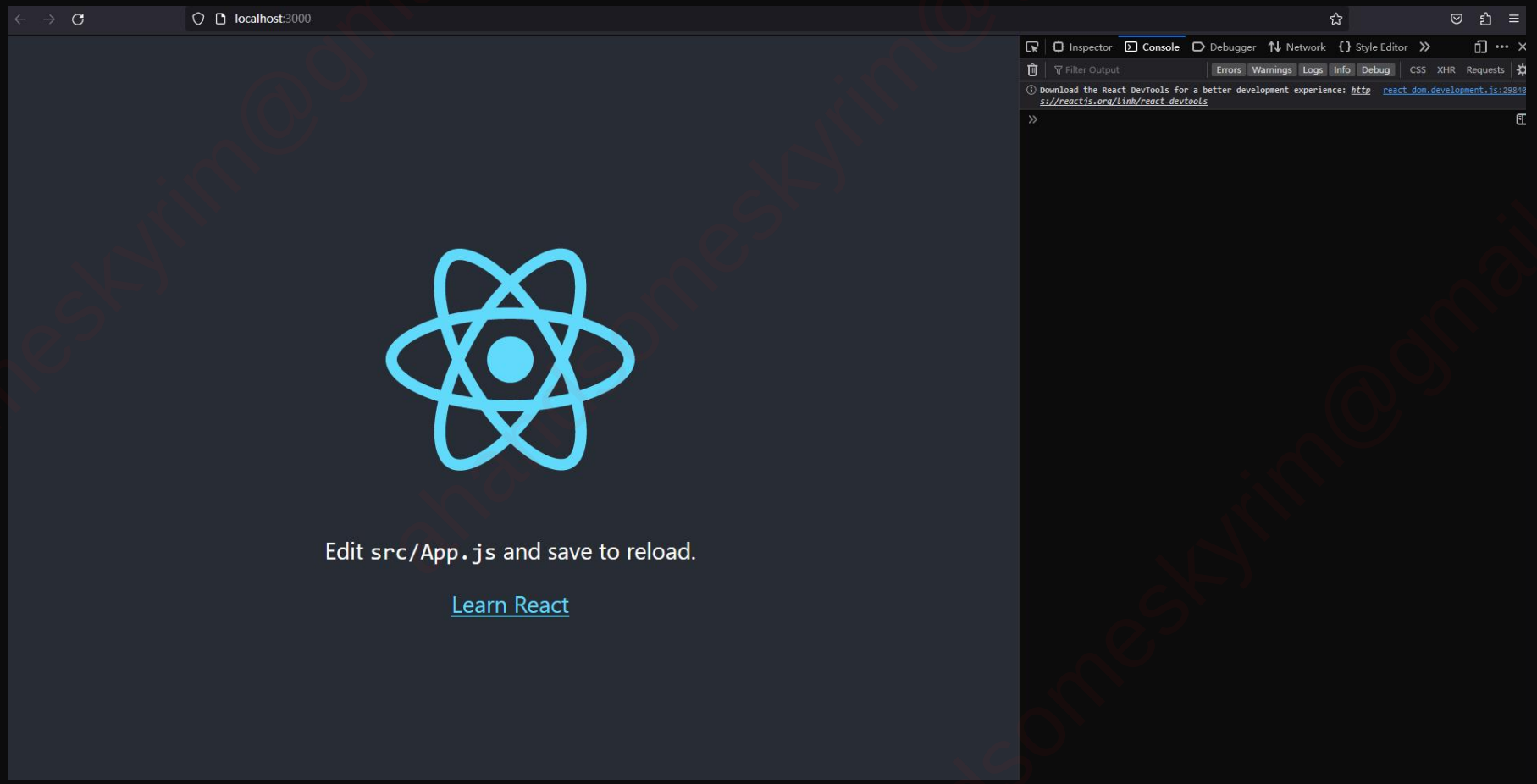
2.2 快速上手 create-react-app

快速上手：create-react-app

```
F:\Temp>npx create-react-app my-react-app

Creating a new React app in F:\Temp\my-react-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


added 1428 packages in 3m

Initialized a git repository.

Installing template dependencies using npm...

added 74 packages, and changed 1 package in 9s
Removing template package using npm...


removed 1 package in 4s

Created git commit.

Success! Created my-react-app at F:\Temp\my-react-app
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd my-react-app
  npm start

Happy hacking!
```

动手演示 & 练习

快速上手：create-react-app



动手演示 & 练习

快速上手：create-react-app

--- 解释目录结构 & 关键文件的代码（备注：面向初学者，每一个点都尽量解释细致一些）---

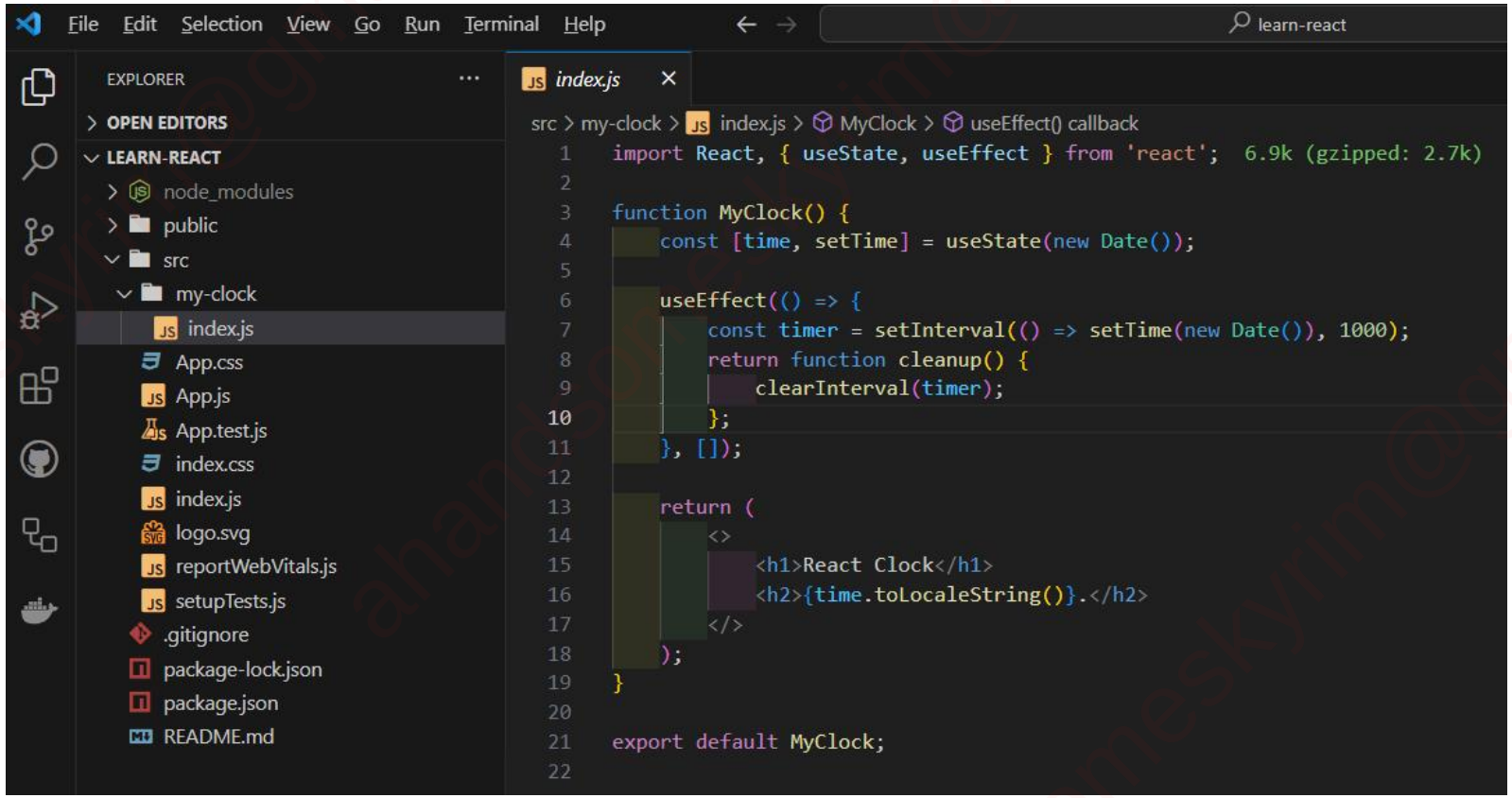
2.3 初步体验 JSX/useState/useEffect

实现一个时钟

动手演示 & 练习

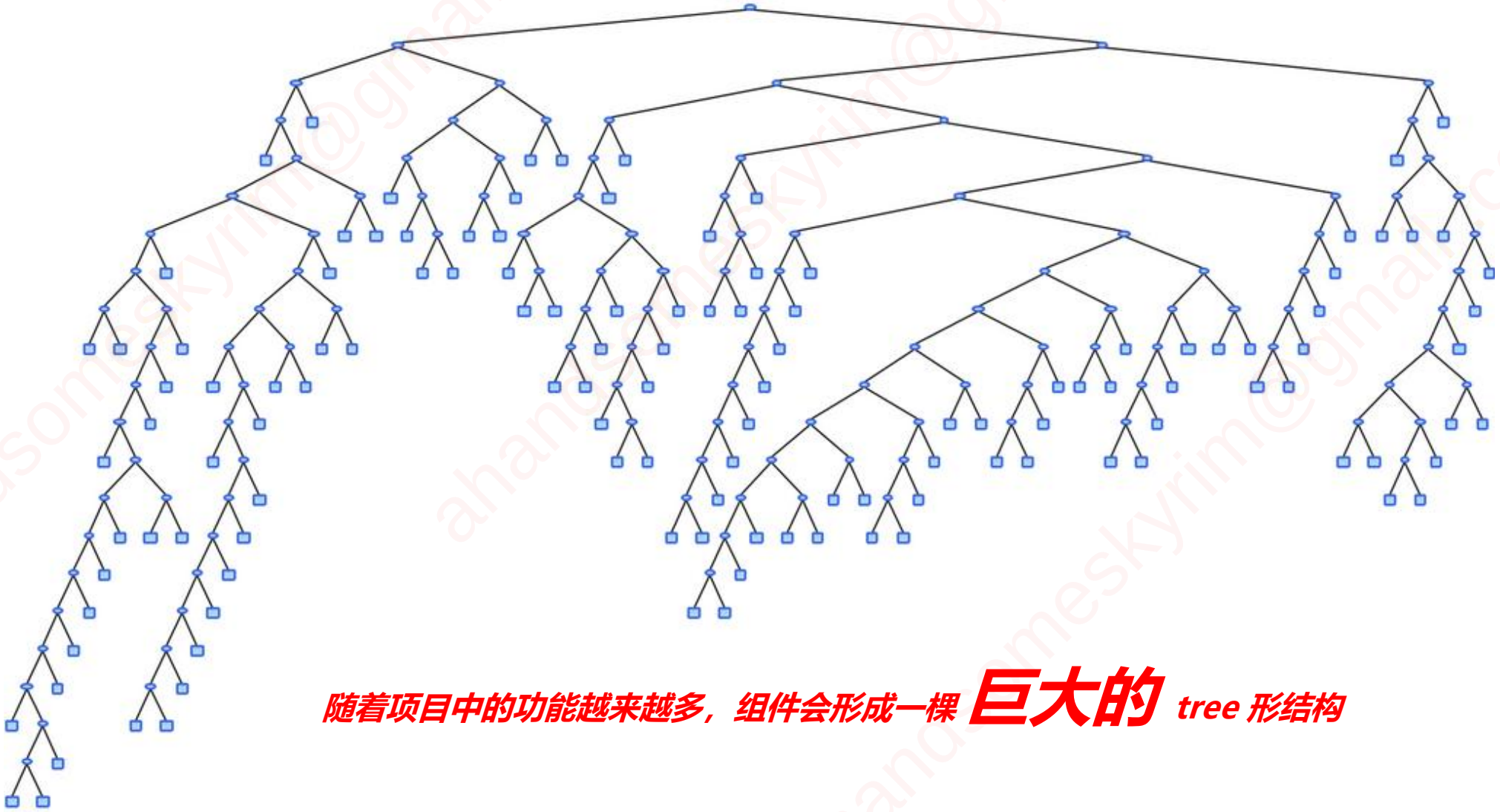
这里先尝试一下手感，在后续的小节中详细解释这些语法

快速上手：拆分组件



动手演示 & 练习

组件树



随着项目中的功能越来越多，组件会形成一棵 ***巨大的*** *tree 形结构*

作业

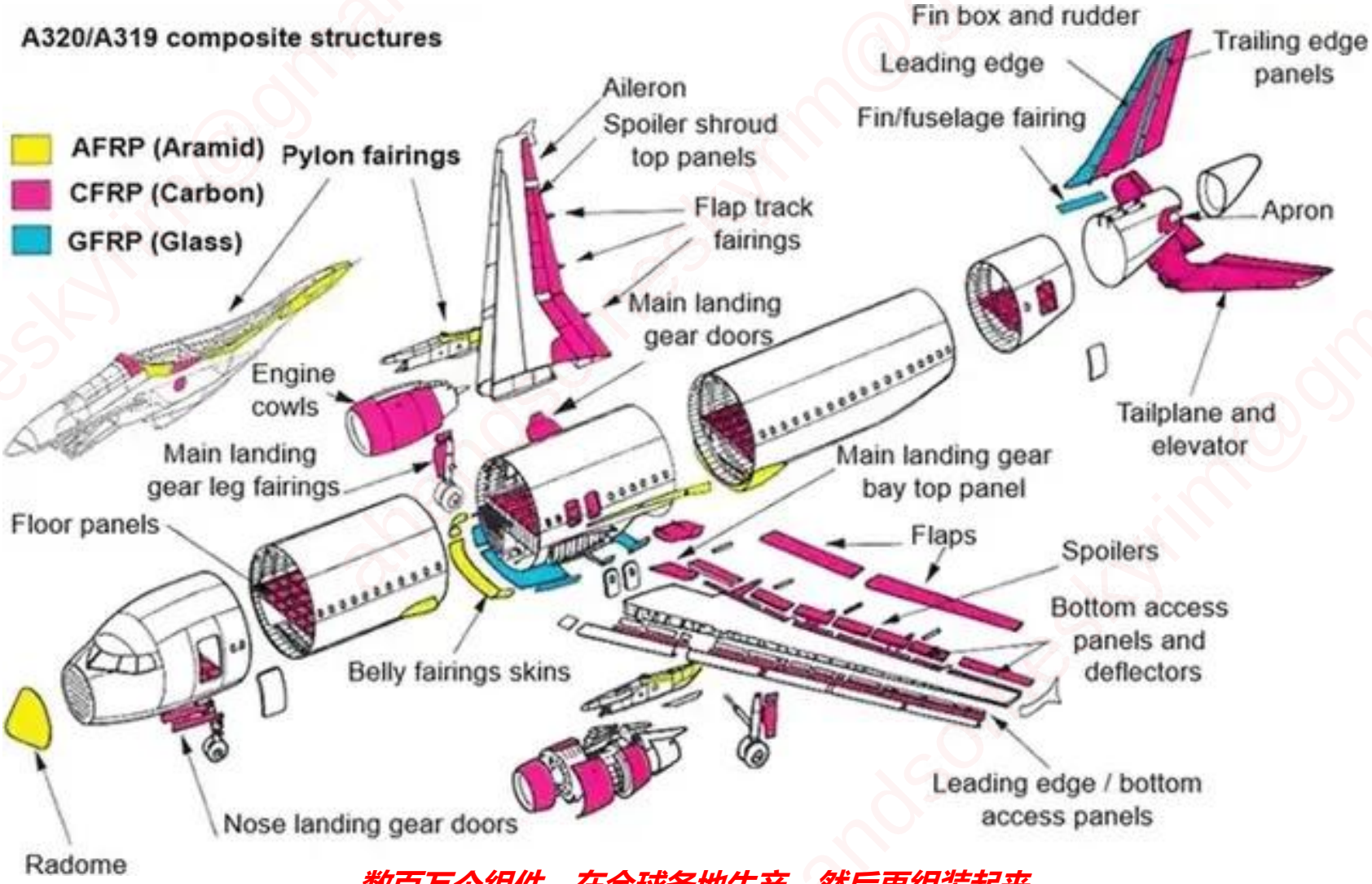
1. 安装配置好以下环境：git, nodejs, nvm, nrm, VSCode, github copilot
2. 安装好 create-react-app 并创建第一个 React 项目 my-react-app (或者你喜欢的名称)
3. 启动你的第一个 React 项目，并看到界面
4. 尝试修改生成的项目模板，实现一个时钟
5. 把时钟相关的逻辑从 App.js 中分离出去，封装成 ReactClock 组件，体验一下 React 的组件化思维模式
6. 打开 React 官方的开发文档，阅读 Quick Start 相关的章节

匠人学院
JR ACADEMY

第3章 组件基础

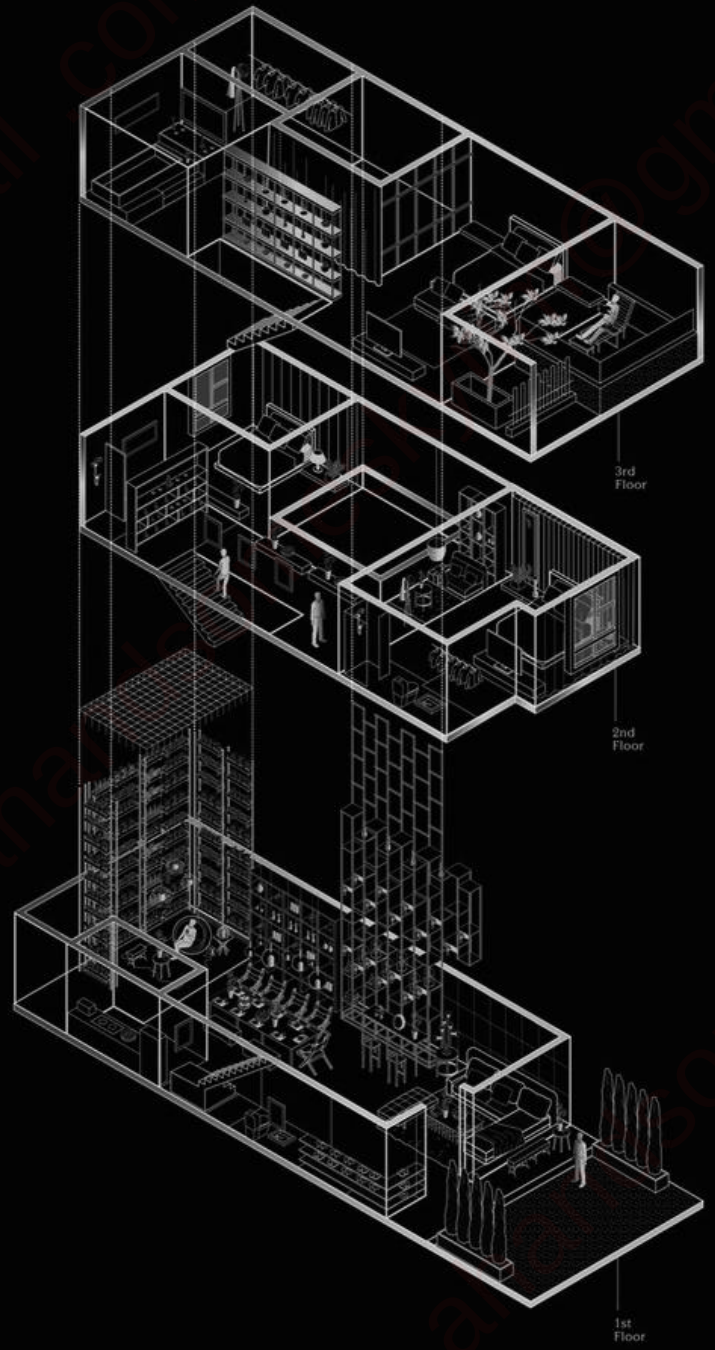
3.1 组件化思想

工业领域的组件化

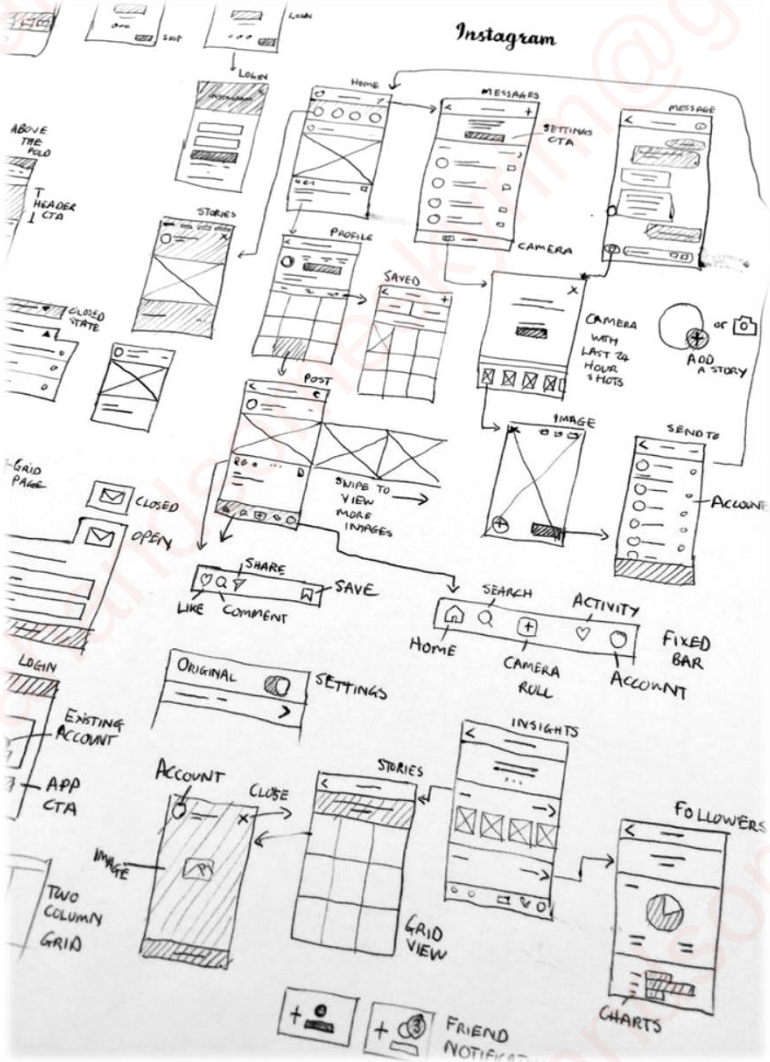


数百万个组件，在全球各地生产，然后再组装起来

建筑领域的组件化



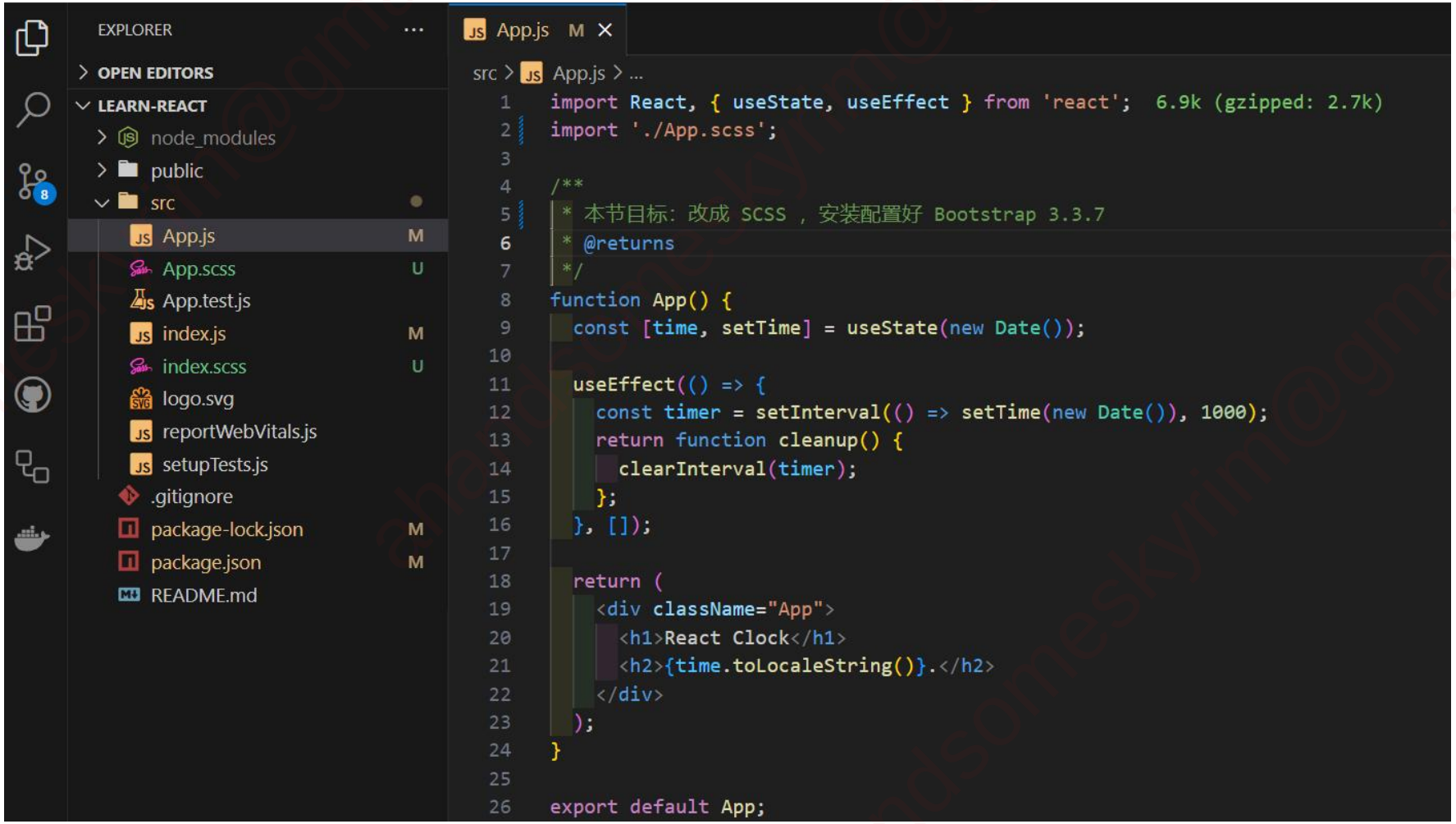
软件开发领域的组件化



现代软件工程也广泛采用“组件化”的思维模式，先开发出一个一个功能有限的小模块，然后再组合起来

3.2 用 SASS 与 Bootstrap 构建组件外观

添加 SASS 支持 （Bootstrap）



动手演示 & 练习

3.3 强大的模板语法 JSX （JavaScript XML）

3.2.1 什么是 JSX（JavaScript XML）？

匠人学院 JR ACADEMY

Search...

TABLE OF CONTENTS

Draft / August 4, 2022

JSX

Introduction

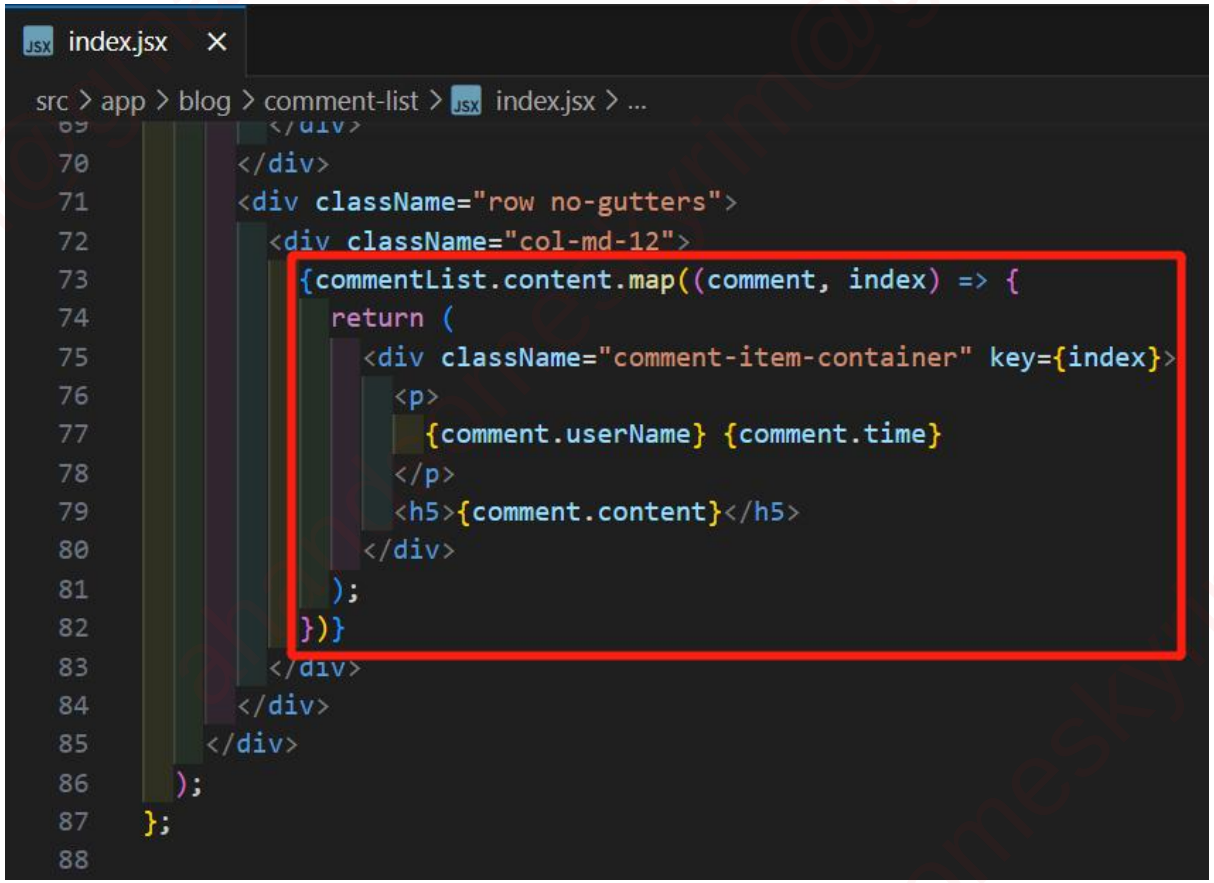
JSX is an XML-like syntax extension to ECMAScript without any defined semantics. It's NOT intended to be implemented by engines or browsers. **It's NOT a proposal to incorporate JSX into the ECMAScript spec itself.** It's intended to be used by various preprocessors (transpilers) to transform these tokens into standard ECMAScript.

```
// Using JSX to express UI components
var dropdown =
  <Dropdown>
    A dropdown list
    <Menu>
      <MenuItem>Do Something</MenuItem>
      <MenuItem>Do Something Fun!</MenuItem>
      <MenuItem>Do Something Else</MenuItem>
    </Menu>
  </Dropdown>;

render(dropdown);
```

http://facebook.github.io/jsx/

这是官方发布的一篇简单的文档，里面详细解释了 what and why，推荐阅读。

3.2.2 JSX 的强大之处在哪里?



这种 HTML 与 JavaScript 混写的模式，极大地增强了 HTML 标签和 JS 语言的表现力。

这是 React 区别于其它框架的一个重要的技术特性。

3.2.3 JSX basic rules

- **标签：** 总是返回单个根标签
- **表达式：**JSX 写标签属性的时候支持动态值、函数调用、基本的表达式（这个特性跟其它前端模板引擎类似）
- **样式：**HTML 标签的 class 属性需要用 className 代替（写 class 编译器会警告，也能运行）
- **样式：**JSX 在编写内联样式的时候全部使用驼峰语法，编译器会处理成原生的标准样式
- **事件：**JSX 的事件语法与原生 HTML 标准不同，JSX 里面使用 onClick、onChange
- **扩展了 HTML 属性：**JSX 在 HTML 标签上扩展了一些自己特有的属性：key/ref/dangerouslySetInnerHTML

JSX 非常灵活，限制性规则不多，以上这些规则是必须掌握的

初学者不用强记，因为编译器都会给出提示，写熟练了之后会自然记得

--- 上手练习 ---

3.4 props & state

3.3.1 什么是 props？什么是 state？

--- 上手练习 ---

3.5 props 与 state 的区别和联系

3.4.1 props 与 state 的区别

匠人学院 JR ACADEMY

Props（属性，英文中还有“财产”的意思）：

- props 用于父子之间传递参数，每个组件都有 props 构造参数。
- props 被设计成只读的，直接修改 props 不会触发重新渲染。

State（状态）：

- state 是可变的，组件可以通过 this.setState() 修改数据，修改之后会触发 React 重新渲染。
- state 用于描述组件的状态和随时间变化的数据。
- state 是组件私有的，其他组件无法直接访问组件的状态数据。

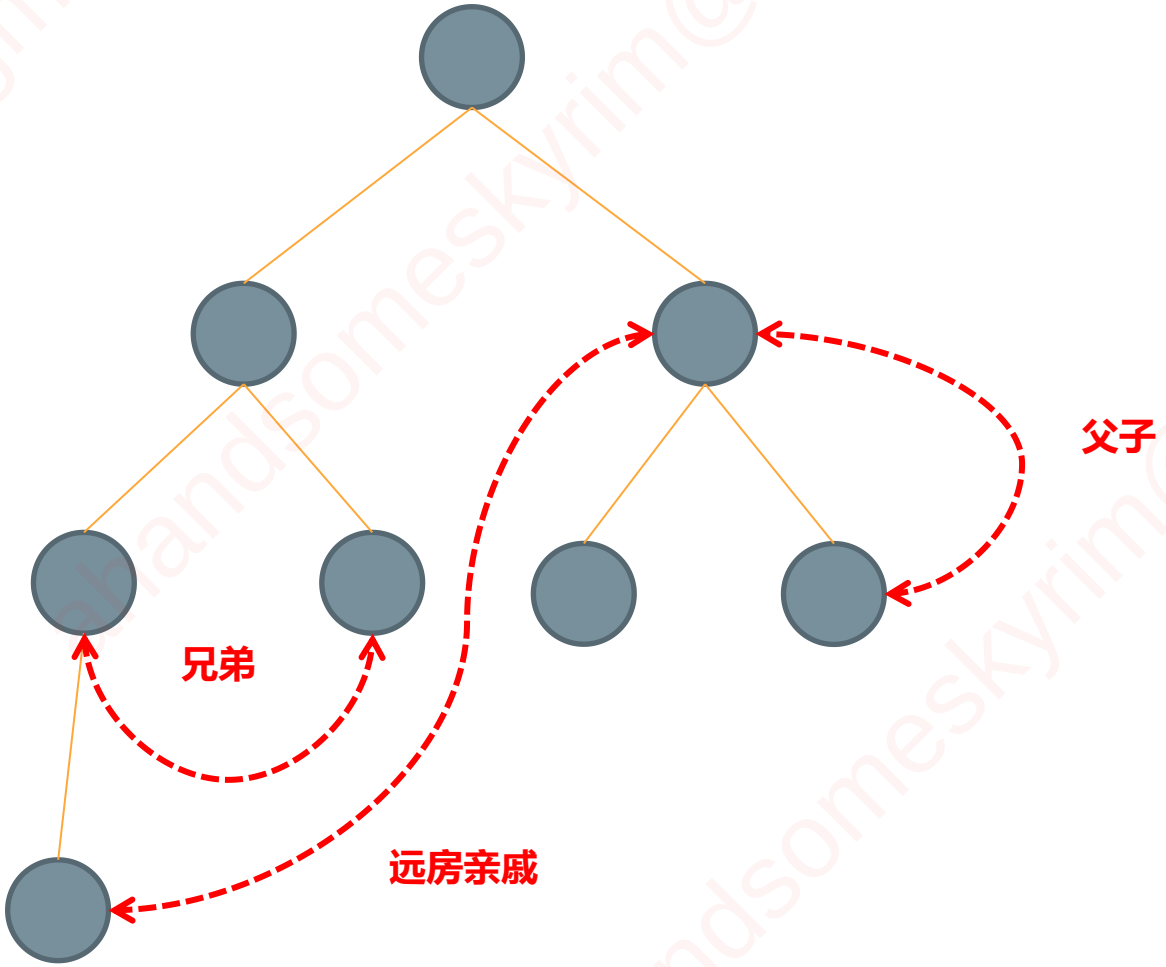
3.4.2 初学者典型的疑惑

- **什么时候用 props？什么时候用 state？** *（参见上一页的表格）*

- **我可以把 props 中的数据全部都丢给 state 吗？** *（技术上没有问题，但是最好不要这样做。在有需要的时候，把 props 中的一部分“复制”给 state，交给 React 去管理是可以的。如果把 props 全部“拷贝”给 state，容易造成数据冗余，也会造成逻辑理解上的困难。）*

3.6 组件间的通讯

组件间通讯

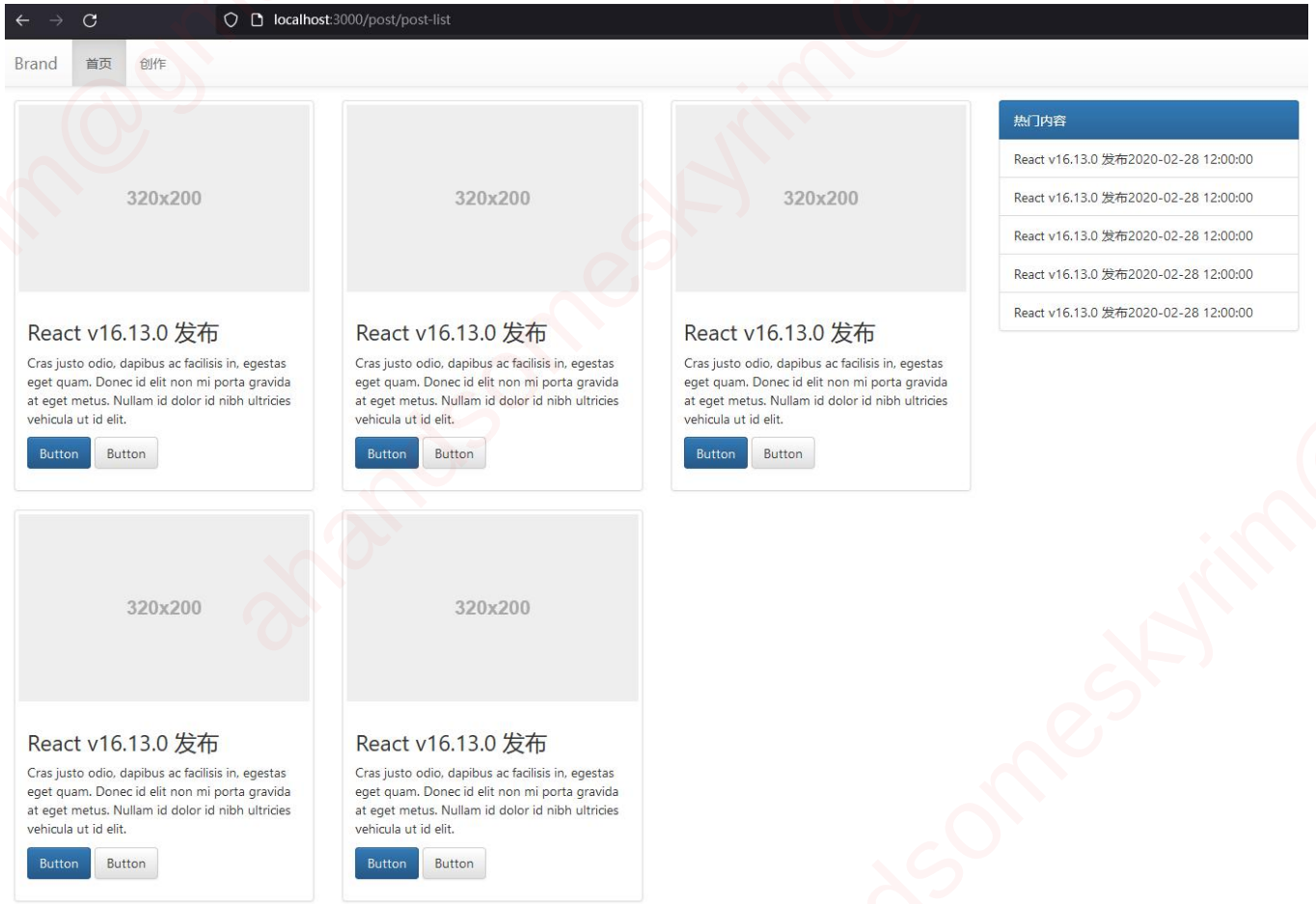


组件间通讯-典型实现方法

- 利用 props 进行通讯
- 利用 EventBus 进行通讯
- 利用 cookie/localStorage 进行通讯
- *利用全局 store 进行通讯*
- 利用路由参数进行通讯
- 利用服务端的 Session

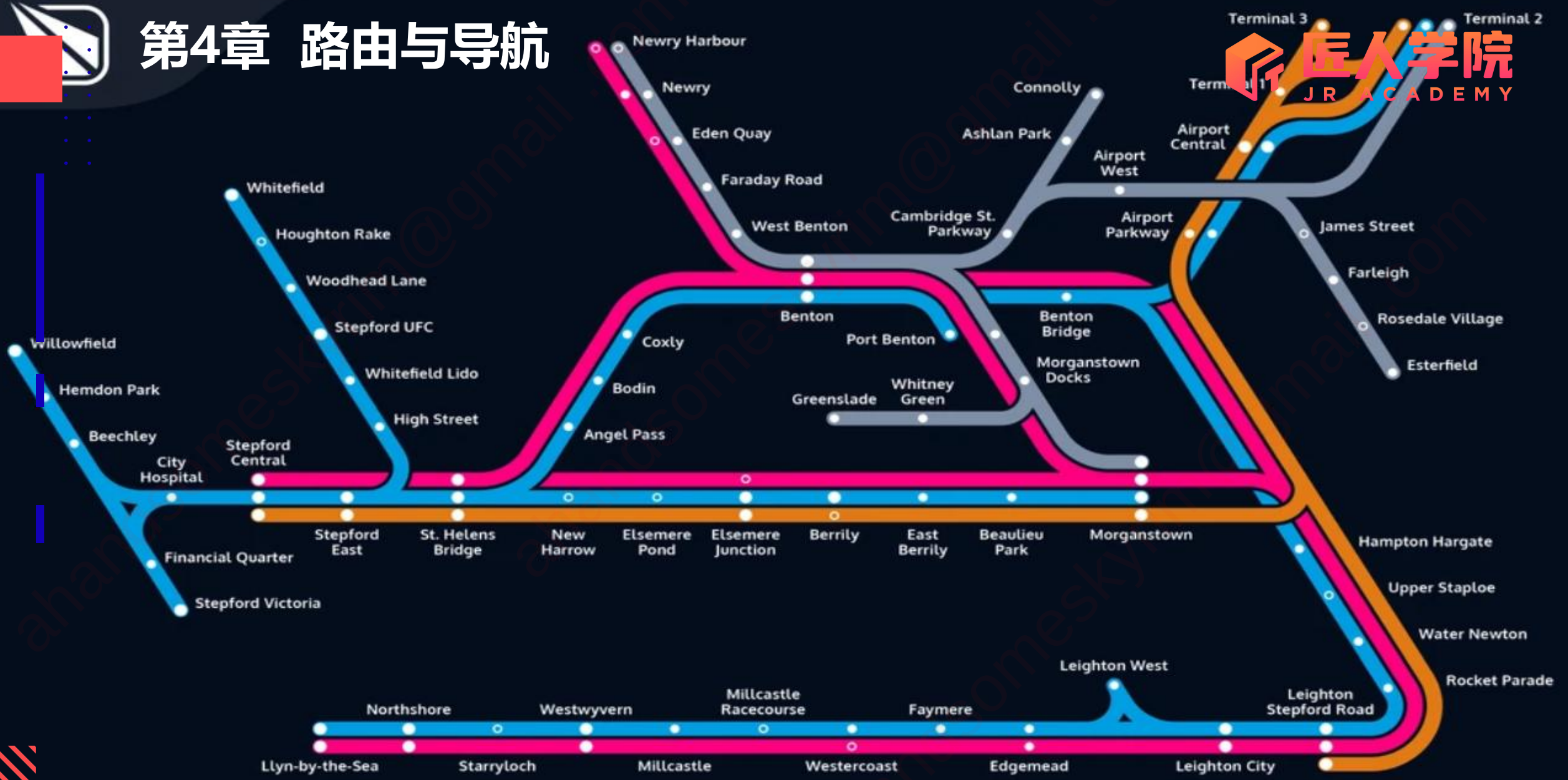
3.7 综合实例：构建一个结构复杂的页面，理解 React 切分组件的思想

我们来构建一个稍微复杂一点的页面布局



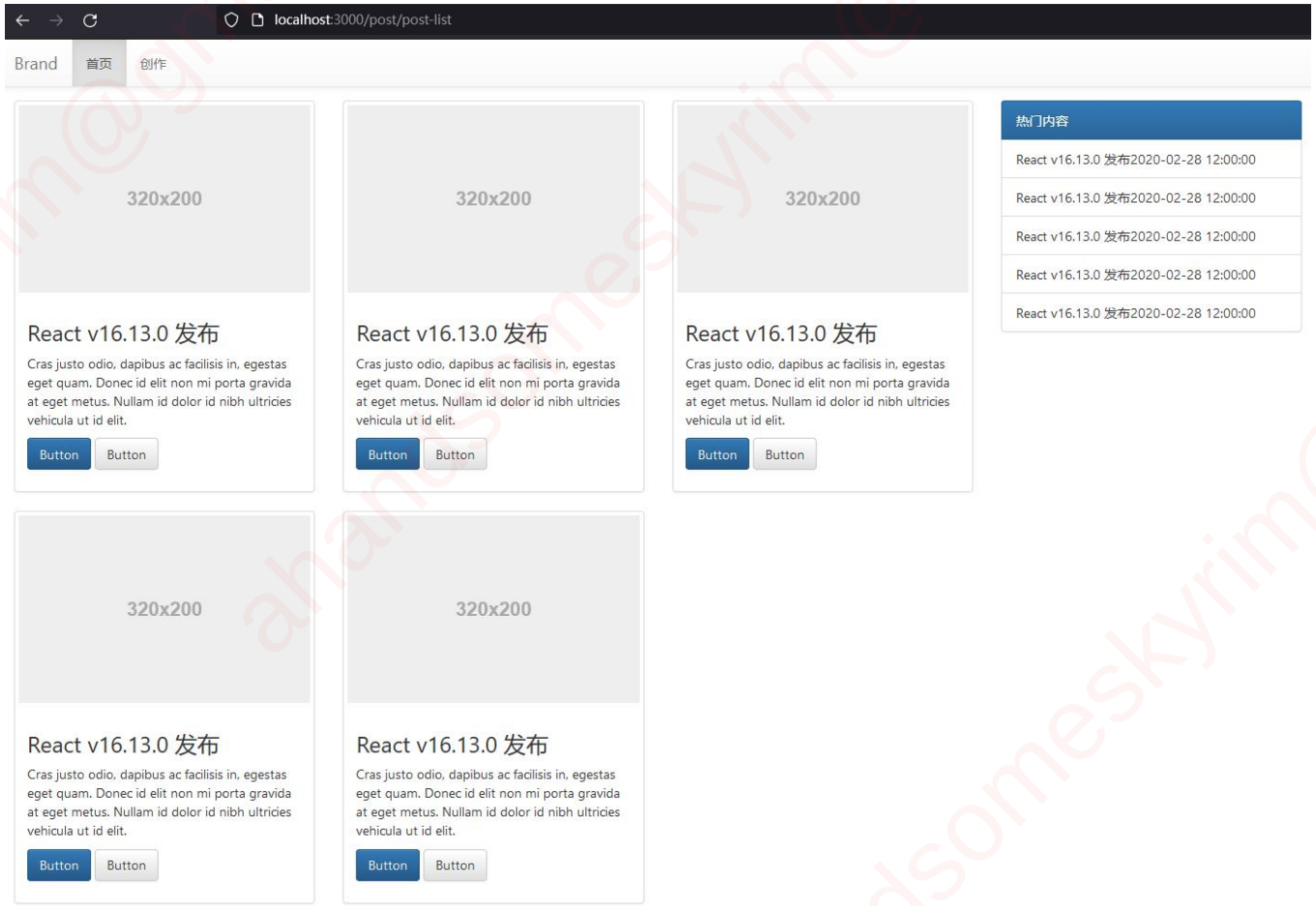
--- 上手练习 ---

第4章 路由与导航


Newry Harbour
Newry
Eden Quay
Faraday Road
West Benton
Connolly
Ashlan Park
Cambridge St. Parkway
Airport West
Terminal 3
Terminal 2
Airport Central
Airport Parkway
James Street
Farleigh
Rosedale Village
Esterfield
Whitefield
Houghton Rake
Woodhead Lane
Stepford UFC
Whitefield Lido
High Street
Willowfield
Hemdon Park
Beechley
City Hospital
Stepford Central
Coxly
Bodin
Angel Pass
Benton
Port Benton
Benton Bridge
Morganstown Docks
Greenslade
Whitney Green
Stepford East
St. Helens Bridge
New Harrow
Elsemere Pond
Elsemere Junction
Berrily
East Berrily
Beaulieu Park
Morganstown
Financial Quarter
Stepford Victoria
Hampton Hargate
Upper Staploe
Water Newton
Rocket Parade
Leighton West
Leighton Stepford Road
Northshore
Westwyvern
Millcastle Racecourse
Faymere
Llyn-by-the-Sea
Starryloch
Millcastle
Westercoast
Edgemead
Leighton City
Most Services
Limited Service

4.1 React Router 快速上手

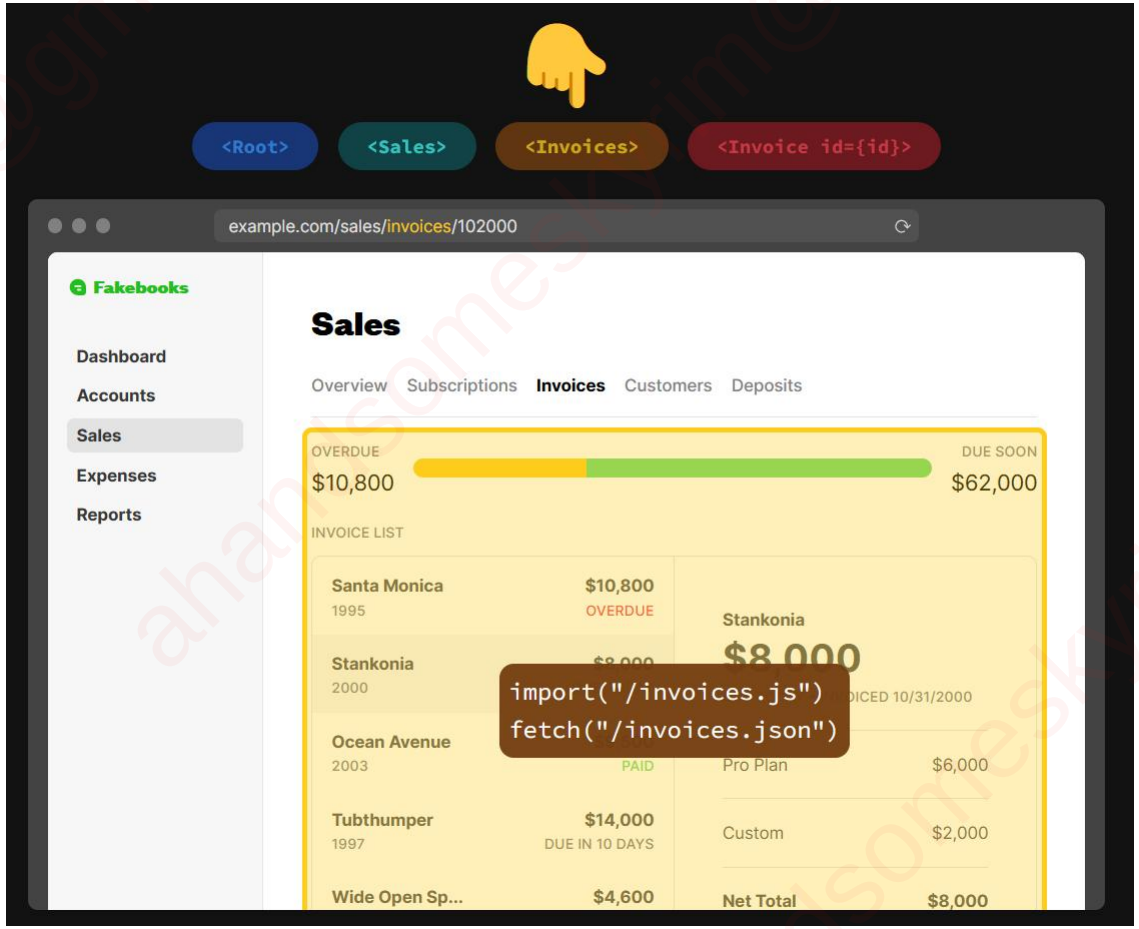
React Router 快速上手



--- 上手练习 ---

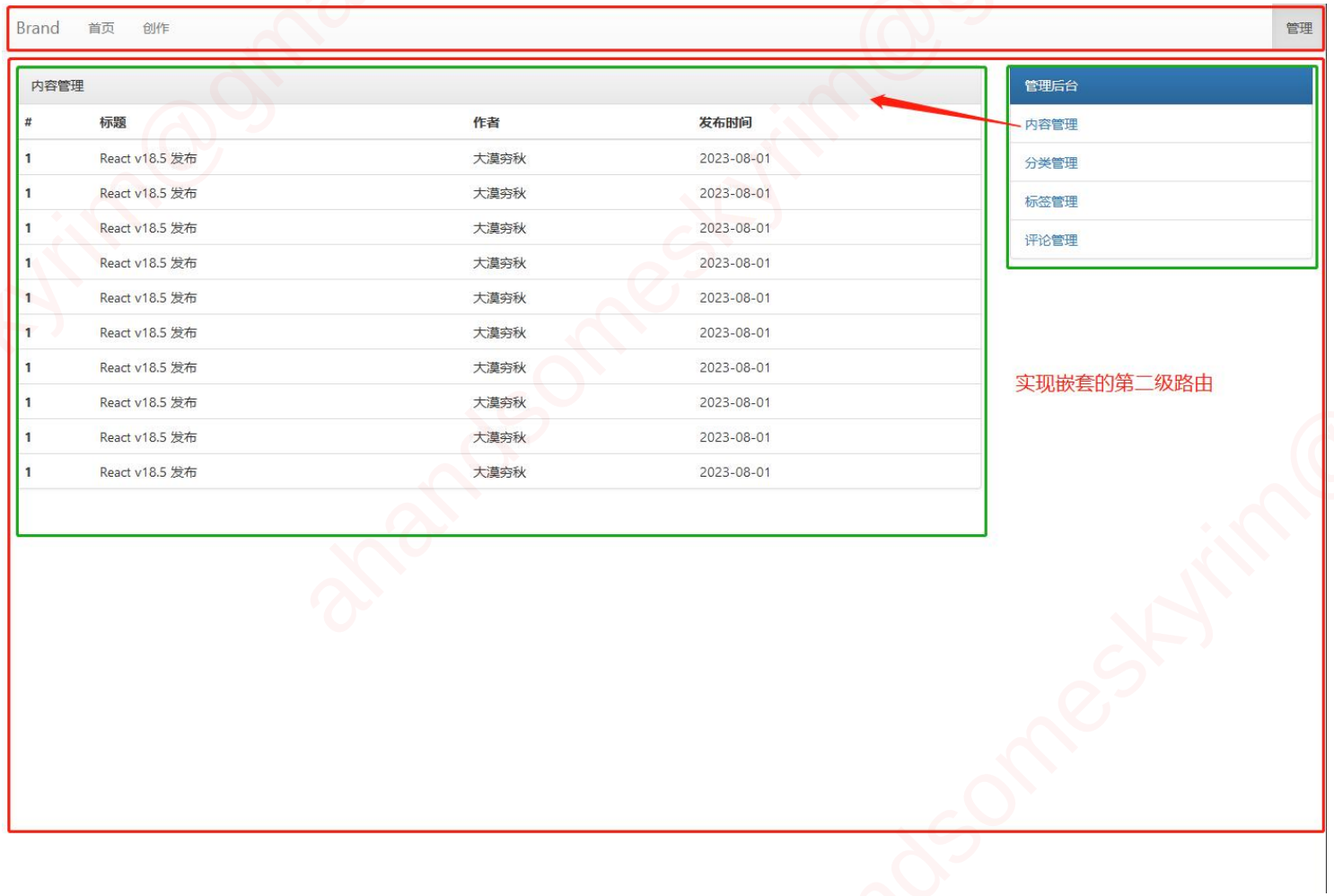
4.2 嵌套路由

URL 与页面布局之间的关系



https://remix.run/_docs/routing

React Router 嵌套路由写法



--- 上手练习 ---

4.3 HashRouter 与 BrowserRouter

4.3 HashRouter 与 BrowserRouter 的区别

匠人学院 JR ACADEMY

HashRouter:

- 使用 URL 中的哈希部分（即 # 后面的部分）来表示路由。
- 适用于不需要服务器配置支持的情况，因为哈希部分的变化不会导致浏览器向服务器发送请求，所以可以在静态网站中使用。
- 适合在不支持 HTML5 History API 的老版本浏览器中使用。

BrowserRouter:

- 使用 HTML5 History API，即通过 pushState 和 replaceState 来改变 URL。
- 需要服务端配合，一般需要修改 HTTP Server 的配置，把 404 页面指向应用首页，从而让前端框架接管路由跳转机制。
- 需要浏览器支持 H5 标准，老的浏览器不行。（*目前2023年，这个问题基本已经不存在了，我们在第一节课已经讲过，webkit 内核已经基本一统江湖。*）

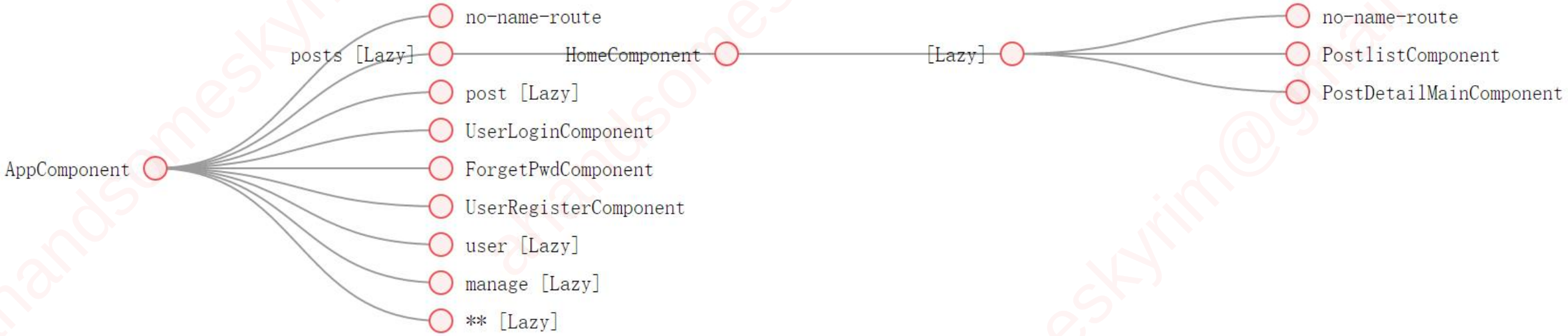
4.4 理解 React Router 的运行机制

--- 对于初学者这一小节有难度，可以不用完全理解，在后续学习的过程中逐步加深 ---

为什么需要路由?

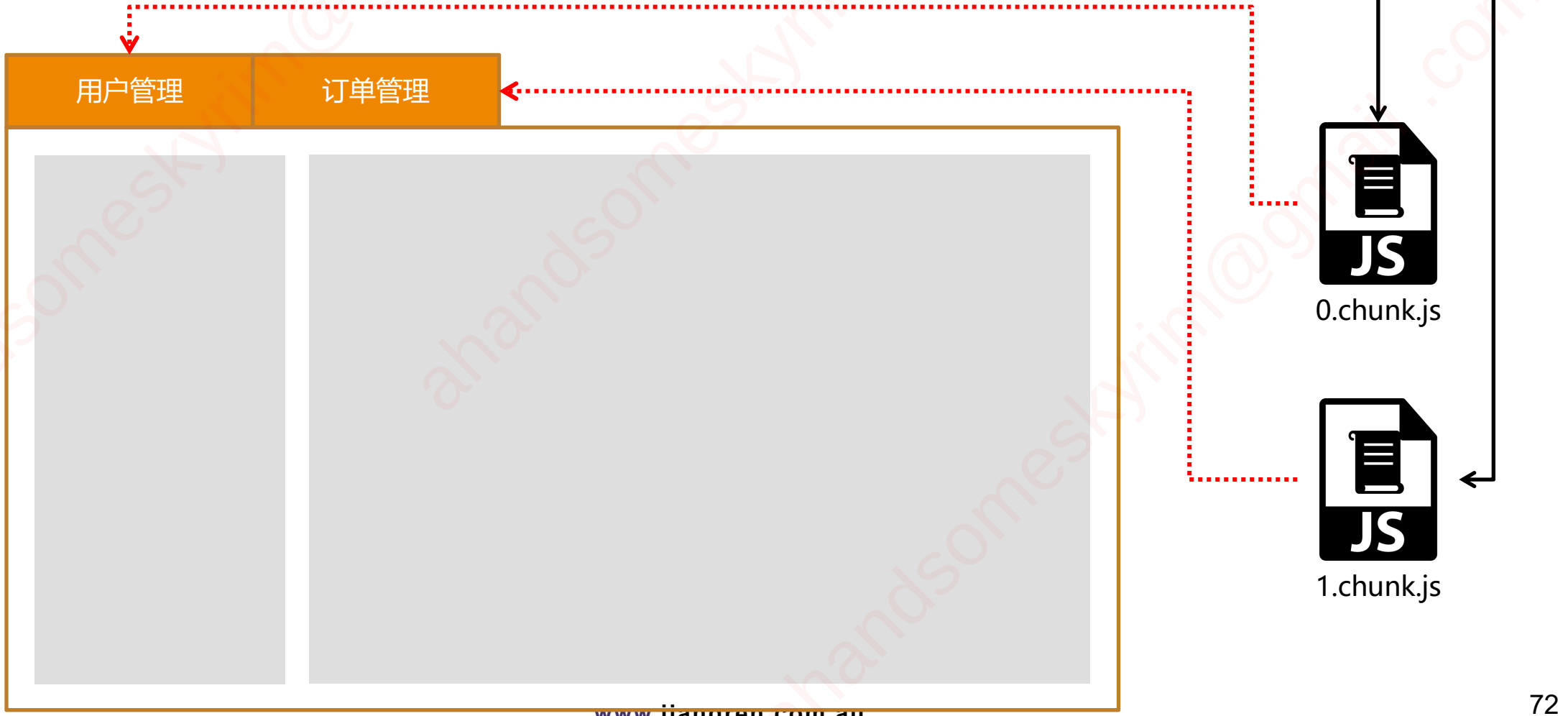
1. 目前，主流的前端应用都是 SPA（Single Page Application） 形式的，如果没有Router，浏览器的前进后退按钮没法用，因为整个应用只有1页，并没有页面跳转，所以浏览器的 history 机制就不起作用了。
2. 如果没有前端路由机制，我们就不得不借助于服务端的 URL 路径进行页面跳转，这样前端的功能会非常受限。
3. 如果没有Router，你将无法把URL拷贝并分享给你的朋友

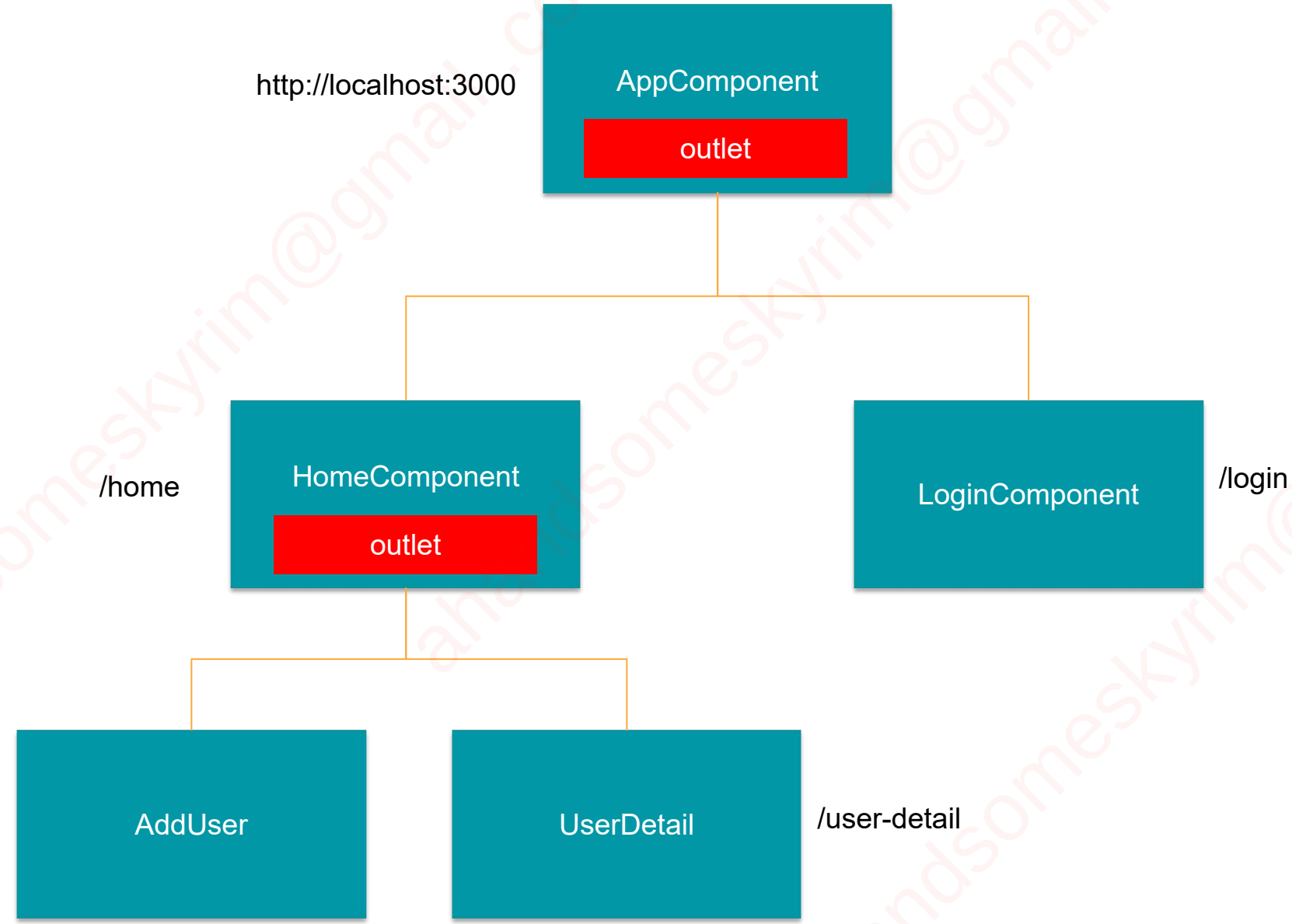
路由的本质是应用当前的一个状态



http://localhost:4200/ordermng

http://localhost:4200/usermng

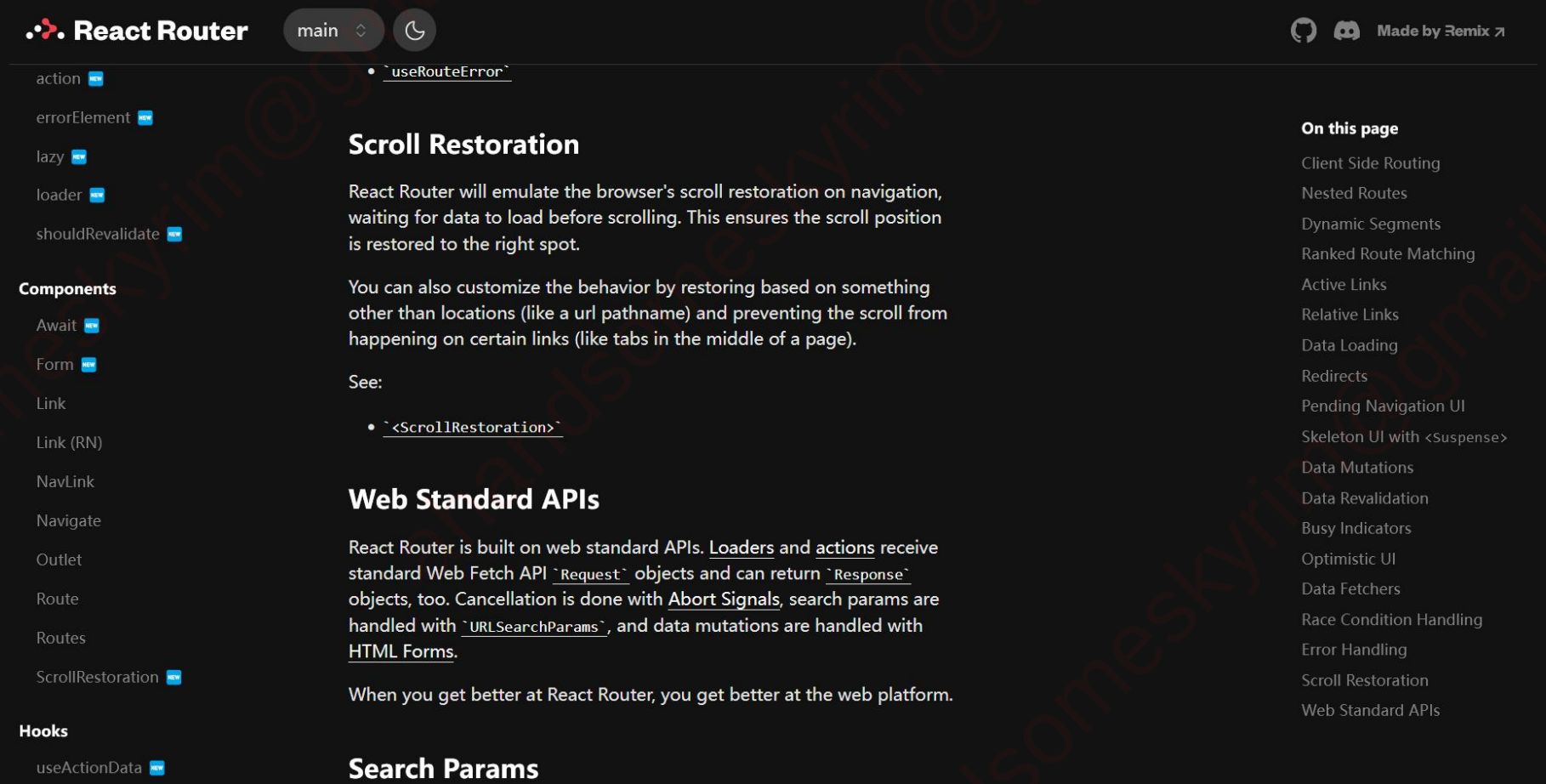




路由与导航-React Router（推荐用这款）

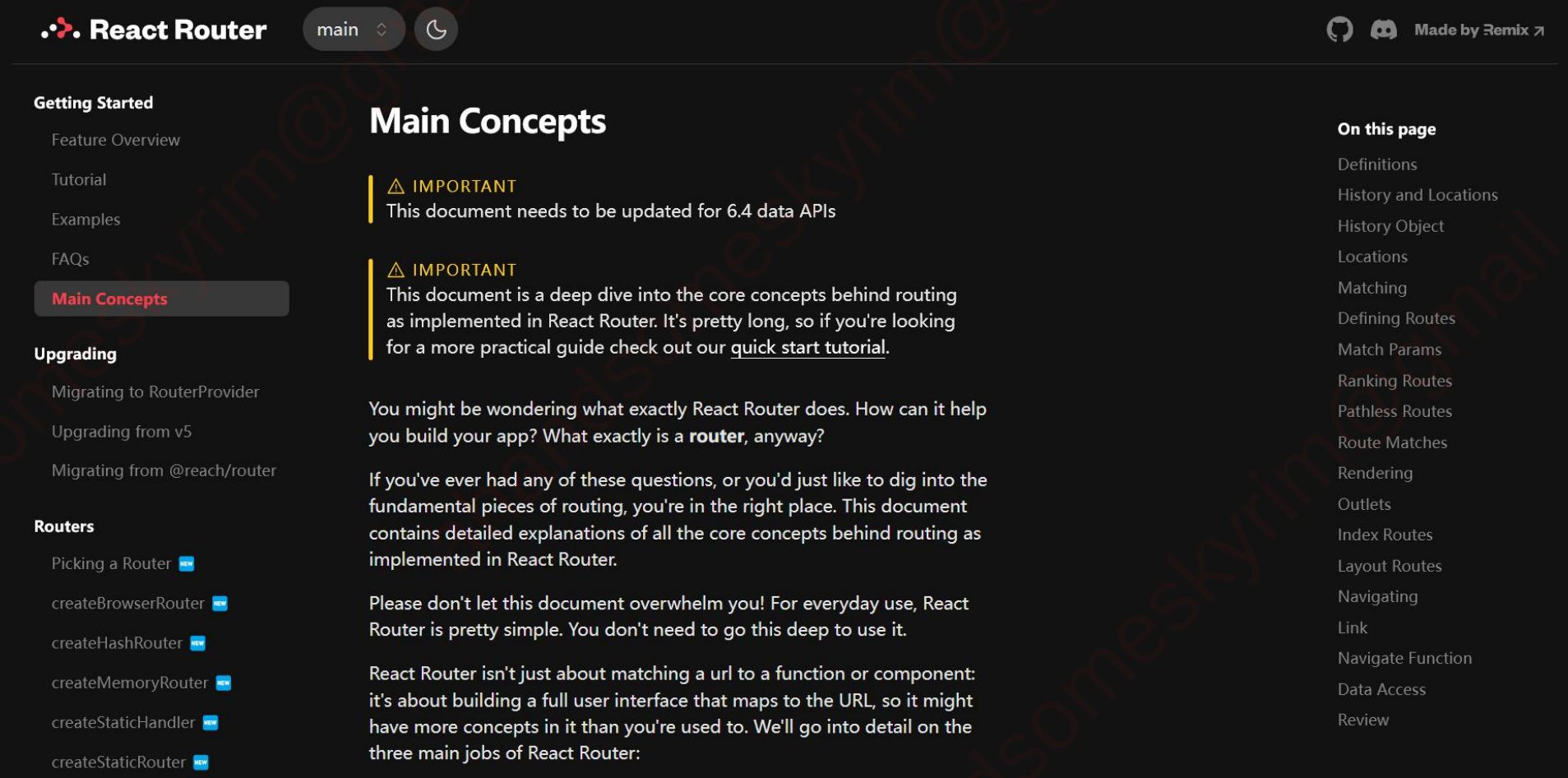
React Router

Router Router core features overview



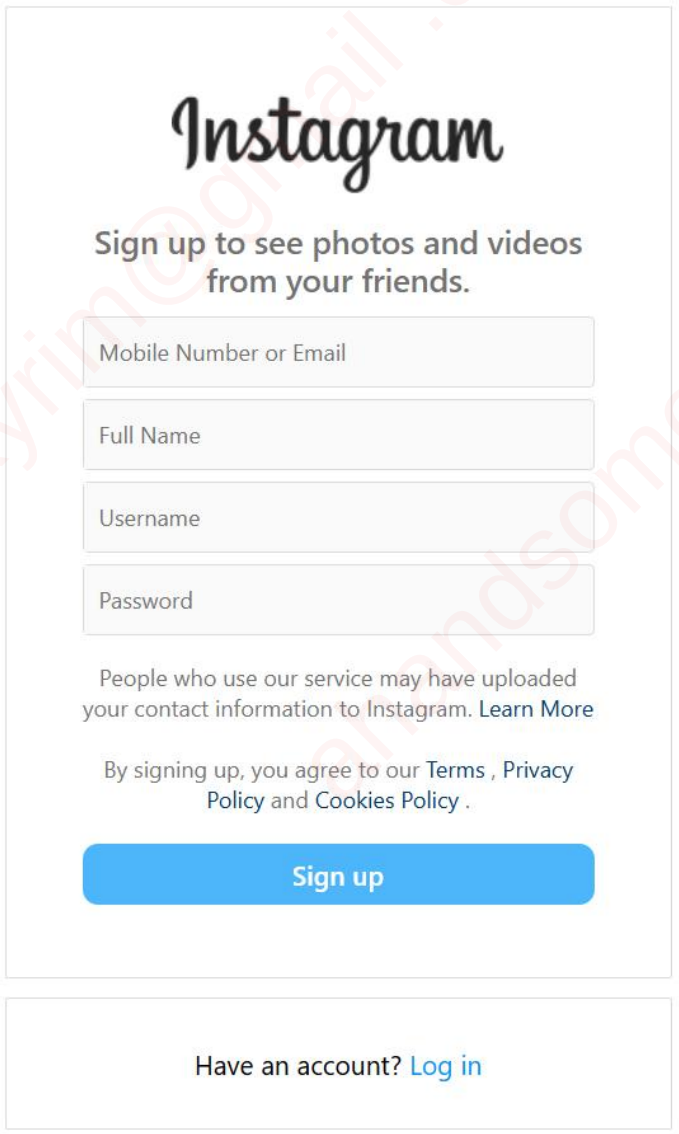
https://reactrouter.com/en/main/start/overview

Main Concepts



https://reactrouter.com/en/main/start/concepts

匠人学院
JR ACADEMY



第5章 处理表单

在实际的企业开发中，我们会有超过一半的时间消耗在处理 Form 上面

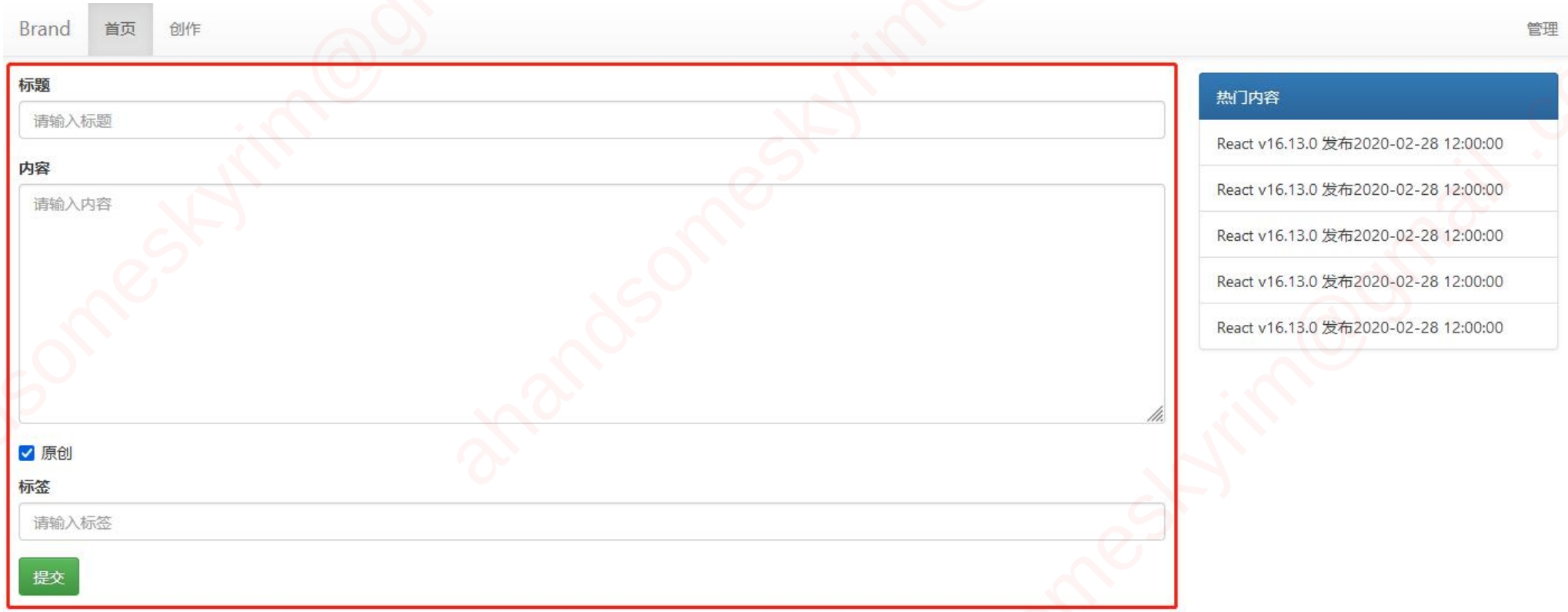
5.1 非受控表单

非受控表单

https://react.dev/reference/react-dom/components/input

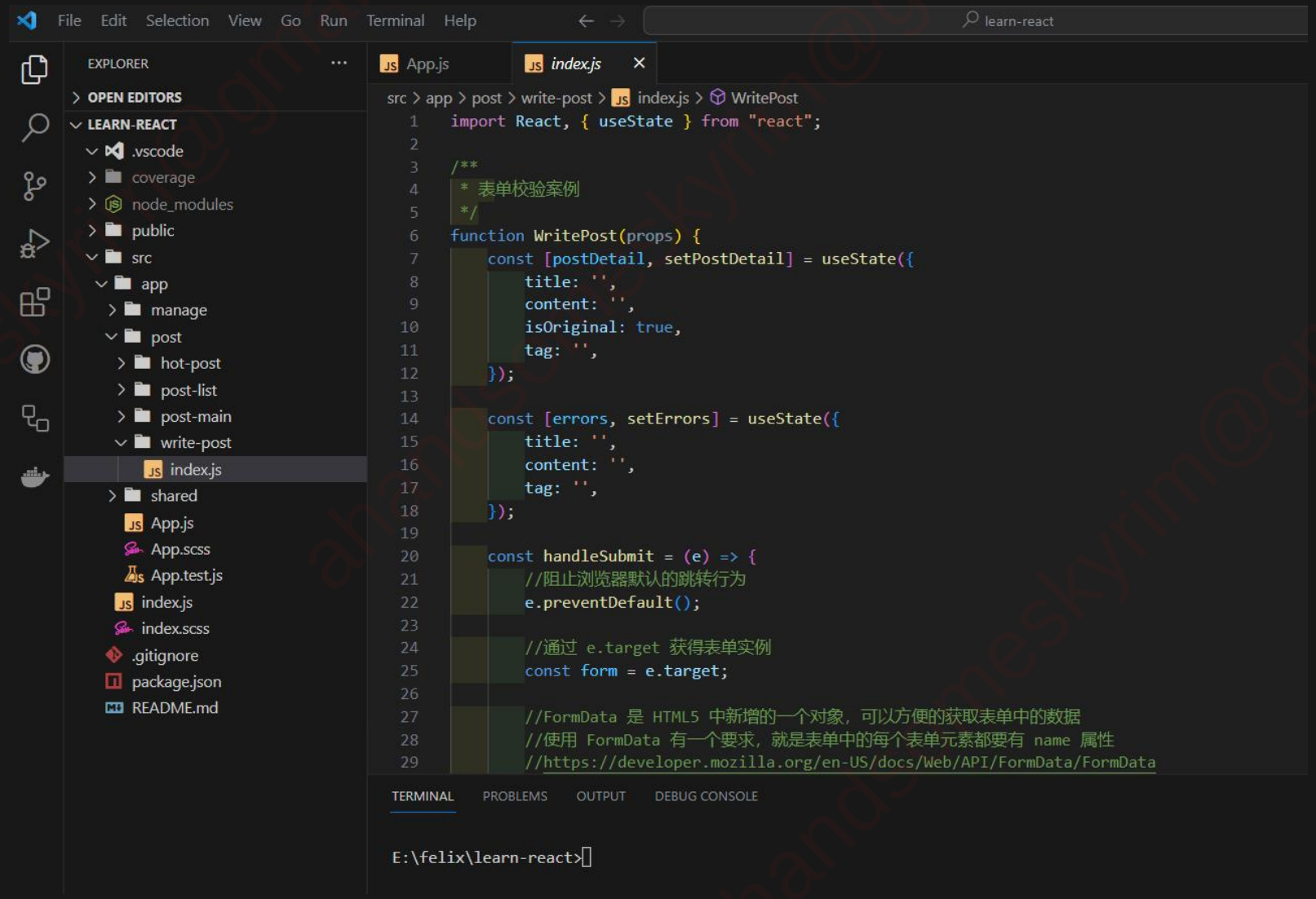
5.2 受控表单

受控表单

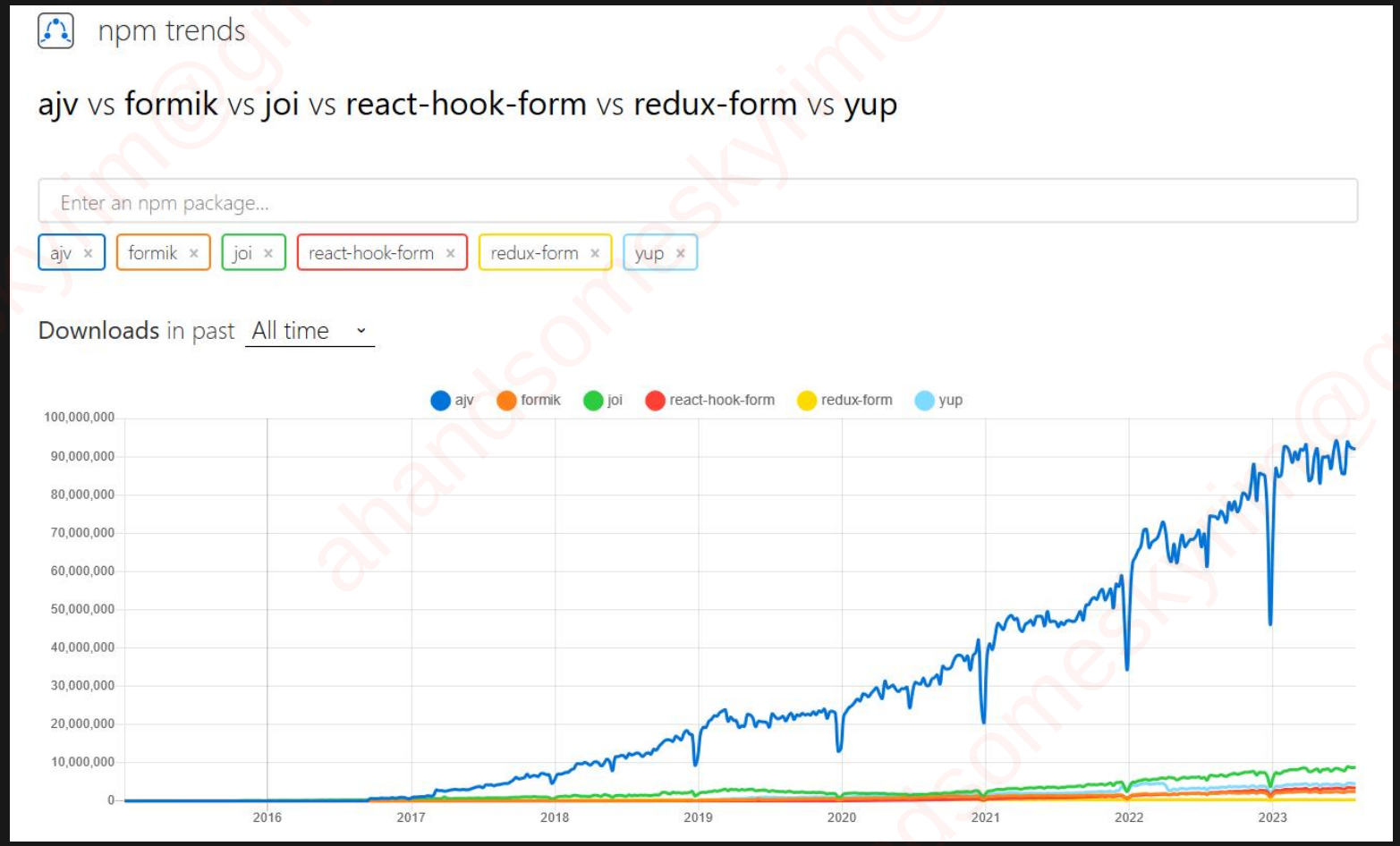


5.3 表单校验 Form Validation

表单校验基础-不使用任何第三方库



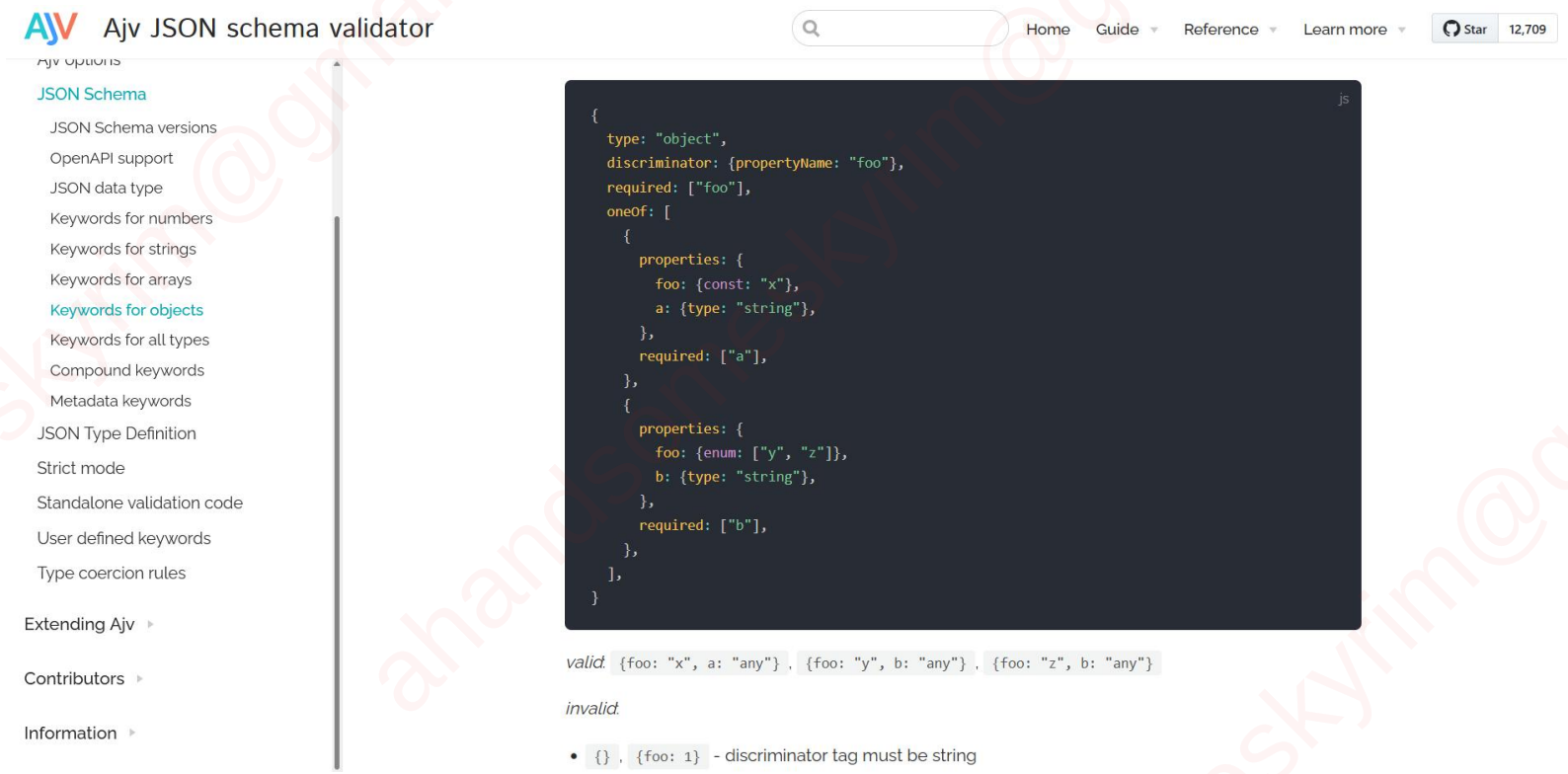
使用 ajv.js 来辅助校验表单



使用 ajv.js 来辅助校验表单

```
// 表单输入项数据规格定义
const schema = {
    "type": "object",
    "properties": {
        "title": {
            "type": "string",
            "minLength": 1,
            "maxLength": 32,
            "errorMessage": "标题长度在 1 到 32 个字符之间。"
        },
        "content": {
            "type": "string",
            "minLength": 10,
            "maxLength": 200,
            "errorMessage": "内容长度在 10 到 200 个字符之间。"
        },
        "isOriginal": {
            "type": "boolean"
        },
        "tag": {
            "tagValidator": true,
            "errorMessage": "标签长度在 2 到 12 个字符之间。",
        }
    },
    "required": ["title", "content"]
}
```

关于 ajv.js 的更多细节



1. **AJV 只是一个基于 JSON schema 的校验库，它与具体的框架无关。也就是说，你可以在任意框架中使用它。**
2. AJV 的语法非常强大，它完全能覆盖日常的业务校验规则。
3. AJV 的性能非常好，根据 ajv 官方给出的数据，它的性能比同类的校验库高 50% 以上。
4. AJV 非常流行，从之前的 npm 安装量数据可以看出来。

第6章 与服务端交互

6.1 与服务端交互的几种典型方式介绍(Ajax/Fetch/Axios)

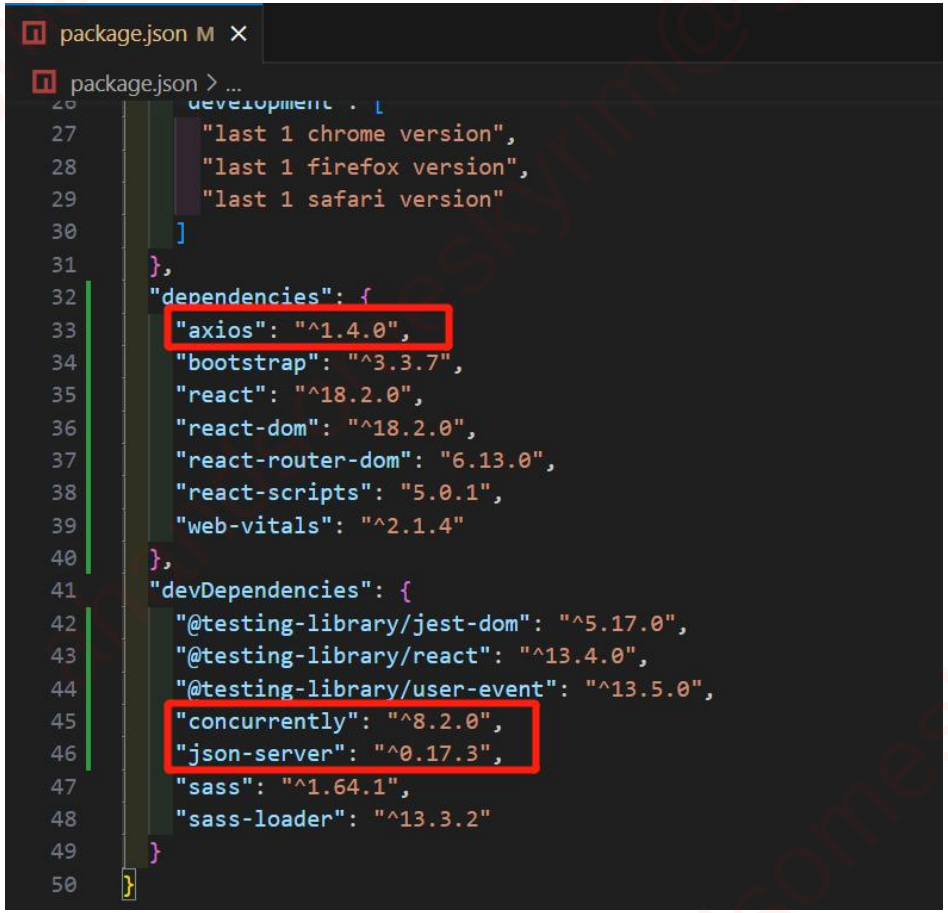
功能对比

名称	API	用法	平台兼容性	功能和扩展性
Ajax	Ajax 是一种**浏览器内置**的技术，通过XMLHttpRequest对象来发送请求。	Ajax 的**使用方式相对较老旧**，需要手动创建 XMLHttpRequest 对象，编写回调函数来处理响应。	Ajax 是一个浏览器内置的技术，**几乎所有现代浏览器都支持。**	Ajax 是浏览器原生支持的技术，**功能相对有限**，通常需要借助其他库来处理复杂场景。
Fetch	Fetch 是基于 Promise 的现代API，它提供了一种更简洁和现代化的方式来进行网络请求。	Fetch 提供了更简单、更直观的 API，使用 fetch() 方法发送请求，返回一个 Promise，并可以使用 then() 处理响应。	**Fetch 是 ES6 的标准，需要较新的浏览器支持。对于旧版本的浏览器需要使用 polyfill 进行支持。**	Fetch 是一个基本的网络请求 API，提供了基本的功能，但在某些场景下可能需要借助其他库或进行自定义扩展。
Axios	**Axios 是一个第三方库，它是一个基于 Promise 的 HTTP 客户端，可以在浏览器和 Node.js 环境中使用。**	Axios 也提供了简单易用的 API，支持 Promise，并且可以使用拦截器(interceptors)、取消请求等高级功能。	Axios 可以在大多数现代浏览器中运行，并且支持 Node.js 环境，可以用于服务器端请求。	Axios 是一个功能强大、灵活的 HTTP 客户端，支持拦截器、请求/响应转换、错误处理等高级功能。

--- 目前，实际的企业开发过程中，大家都选择使用 Axios ，因此本教程内容直接针对 Axios ，不再演示过时的内容 ---

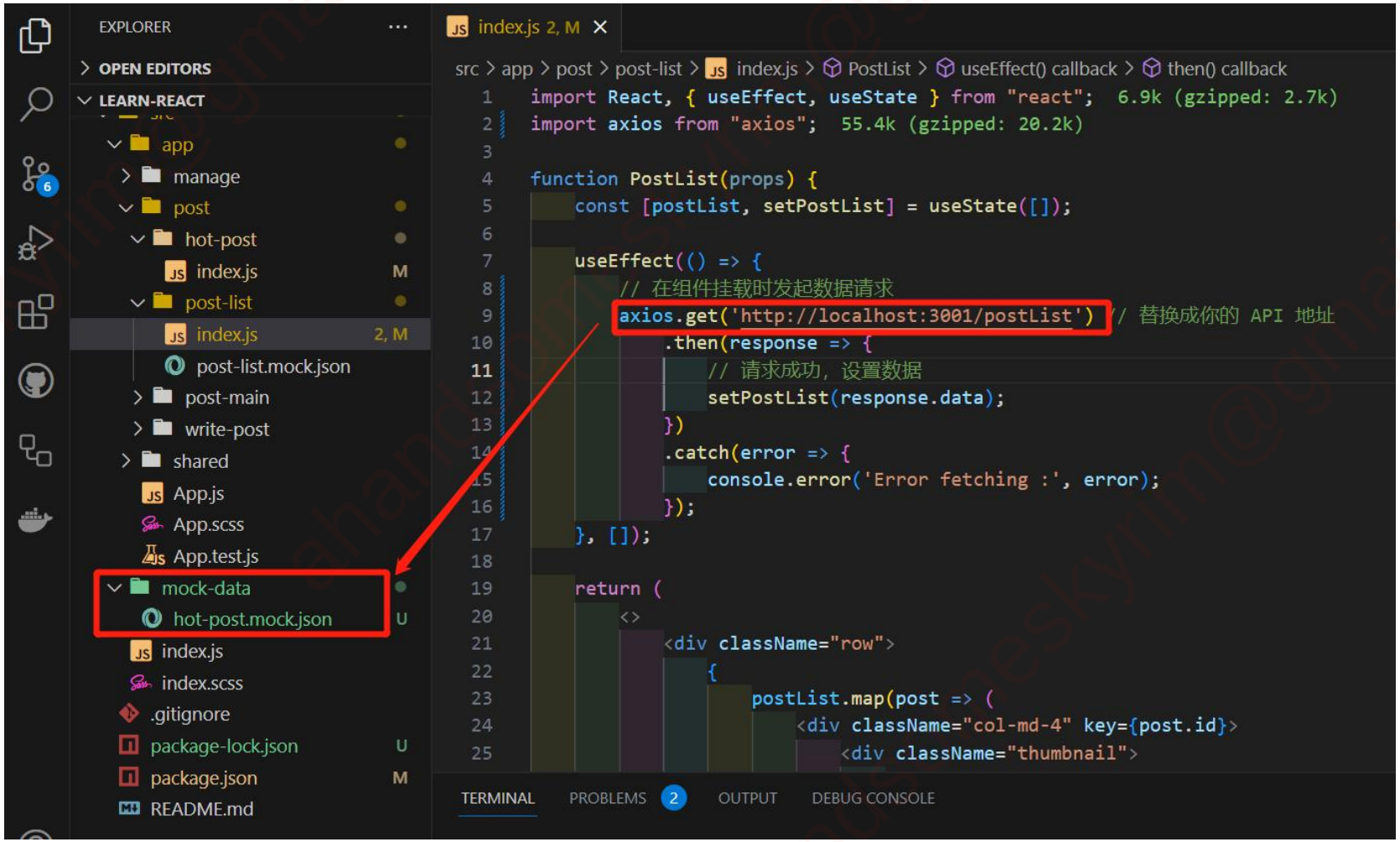
6.2 Axios 快速上手

配置环境



--- 配置 json-server，用来提供 mock 数据（这种模式在实际的企业应用开发中也很常用） ---

配置环境



--- 用 axios 加载数据 ---

6.3 用拦截器处理通用的操作

拦截器

```
8   const createAxiosInstance = () => {
9       // 创建 Axios 实例
10      const api = axios.create();
11
12      // 请求拦截器
13      api.interceptors.request.use(
14          (config) => {
15              // 在每一个请求头部添加 Authorization 字段，其内容为 token
16              // config.headers['Authorization'] = 'Bearer your-access-token';
17              return config;
18          },
19          (error) => {
20              // 对请求错误做些什么
21              return Promise.reject(error);
22          }
23      );
24
25      // 响应拦截器
26      api.interceptors.response.use(
27          (response) => {
28              // 这里可以对响应的数据进行处理，所有请求的响应都会先经过这里
29              console.log("全局响应拦截器---");
30              console.log(response);
31              return response;
32          },
33          (error) => {
34              return Promise.reject(error);
35          }
36      );
37
38      return api;
39  };
```

--- 上手练习 ---

6.4 封装可复用的 Axios 服务

封装成服务

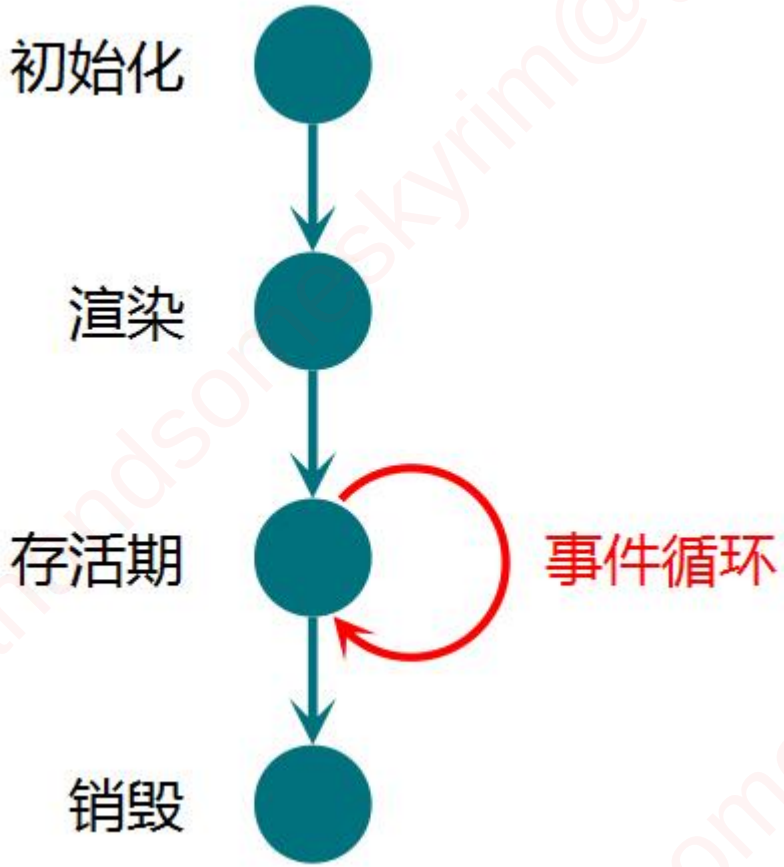
匠人学院
JR ACADEMY

--- 上手练习 ---

第7章 组件进阶

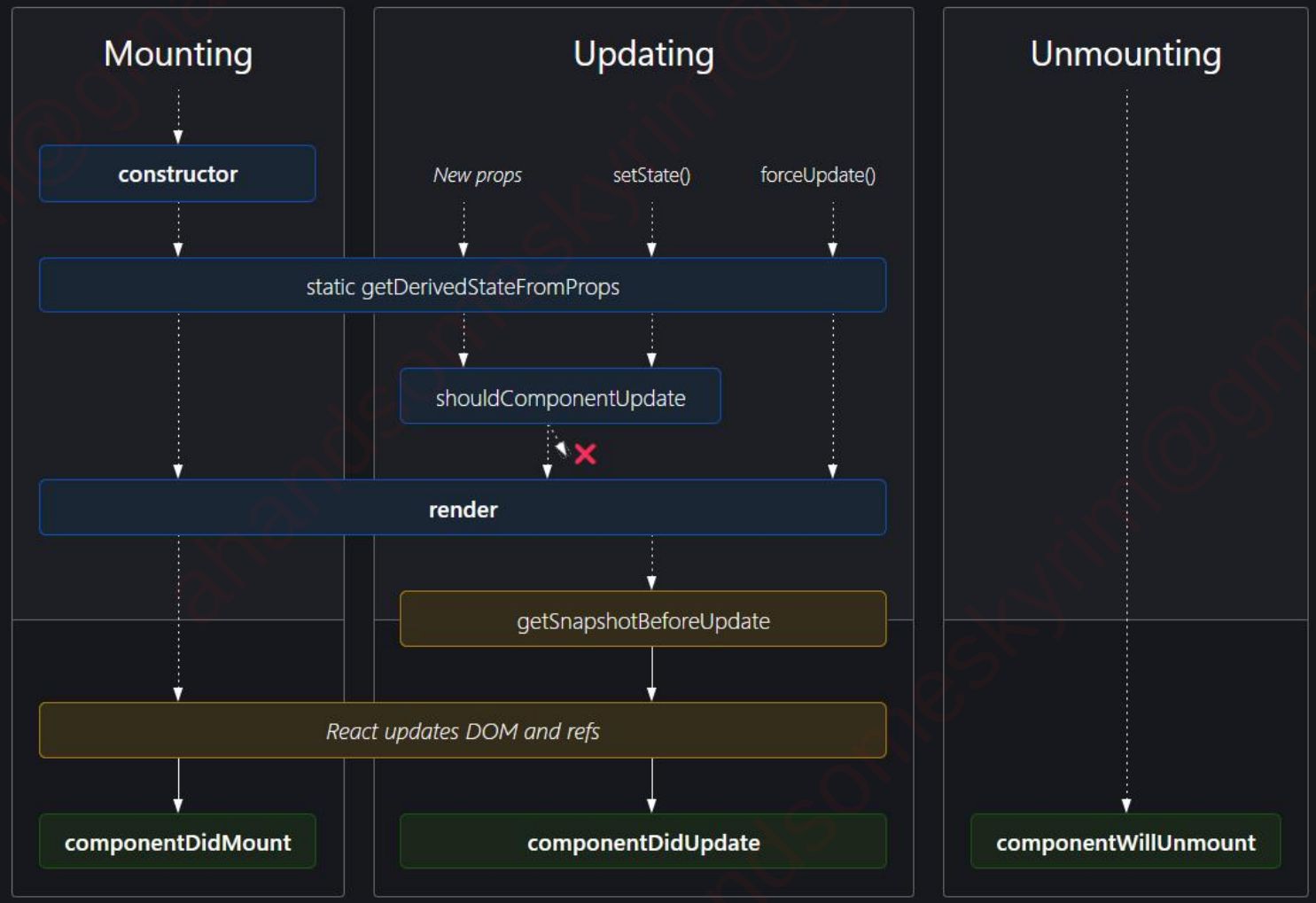
7.1 React 组件的生命周期

什么是生命周期?



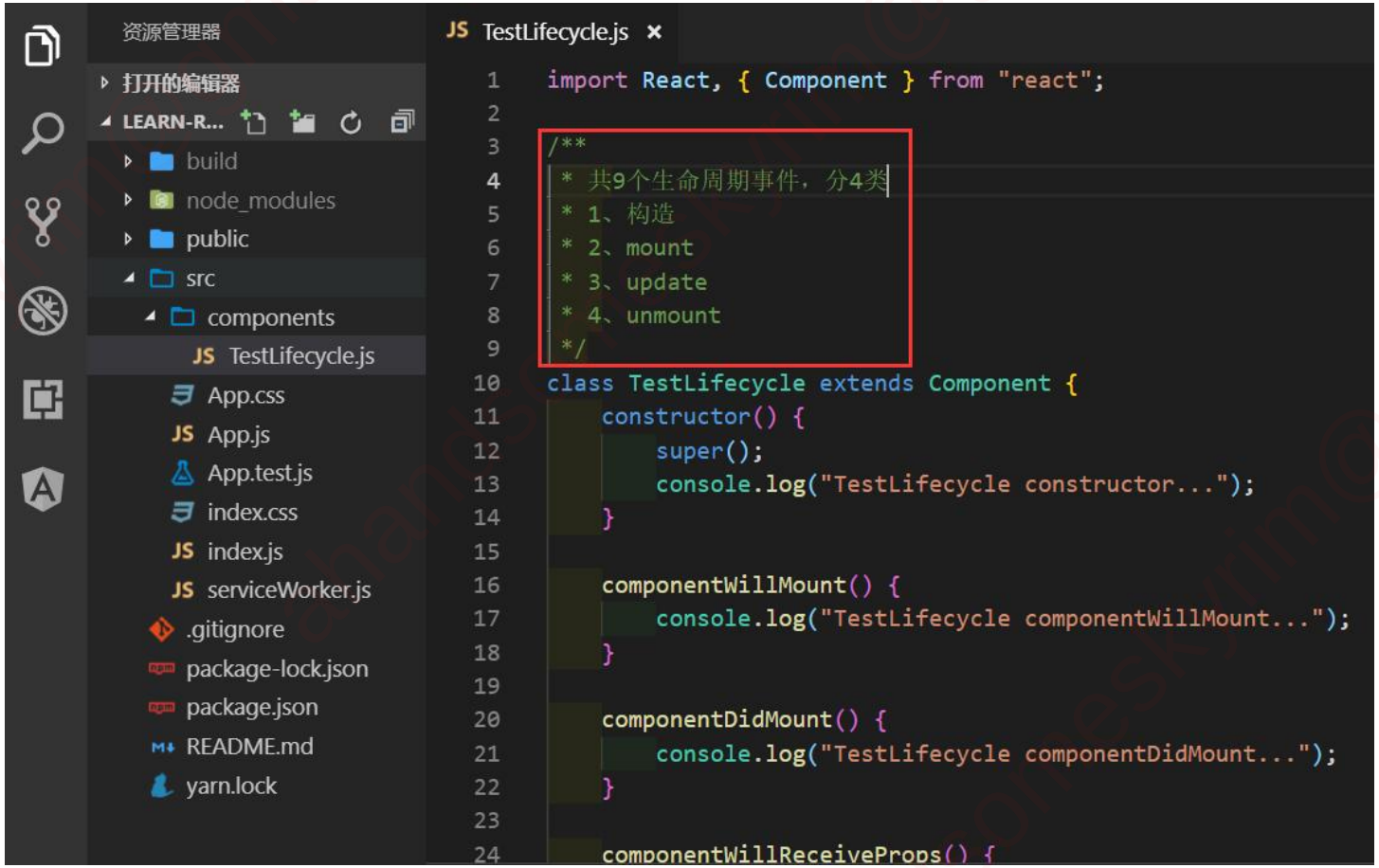
无论使用什么语言，什么框架，UI 组件的生命周期设计都大体类似，只有一些实现细节上的差异
这些概念都不是新事物，在数十年前开发 Windows 和 Linux 操作系统时，这些思想就已经出现了

React 组件的生命周期



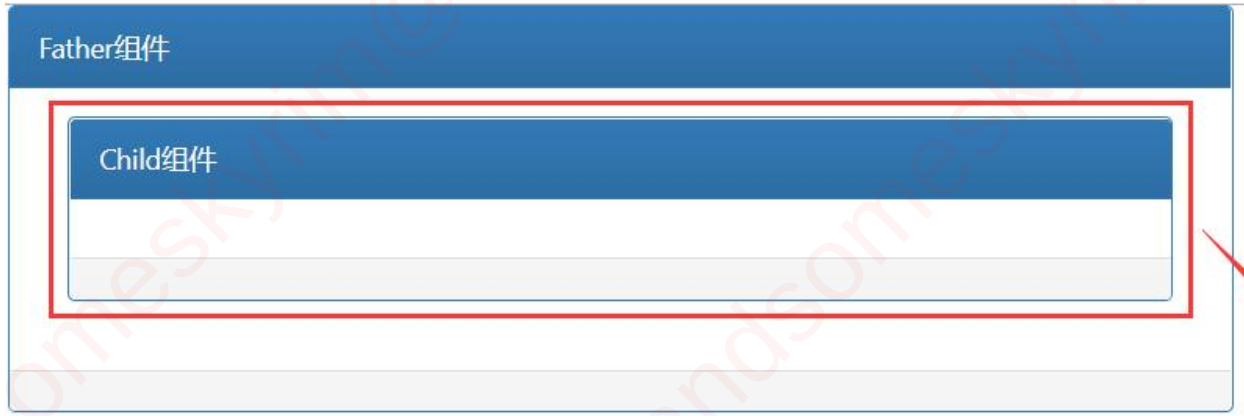
https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

组件的生命周期



--- 上手练习 ---

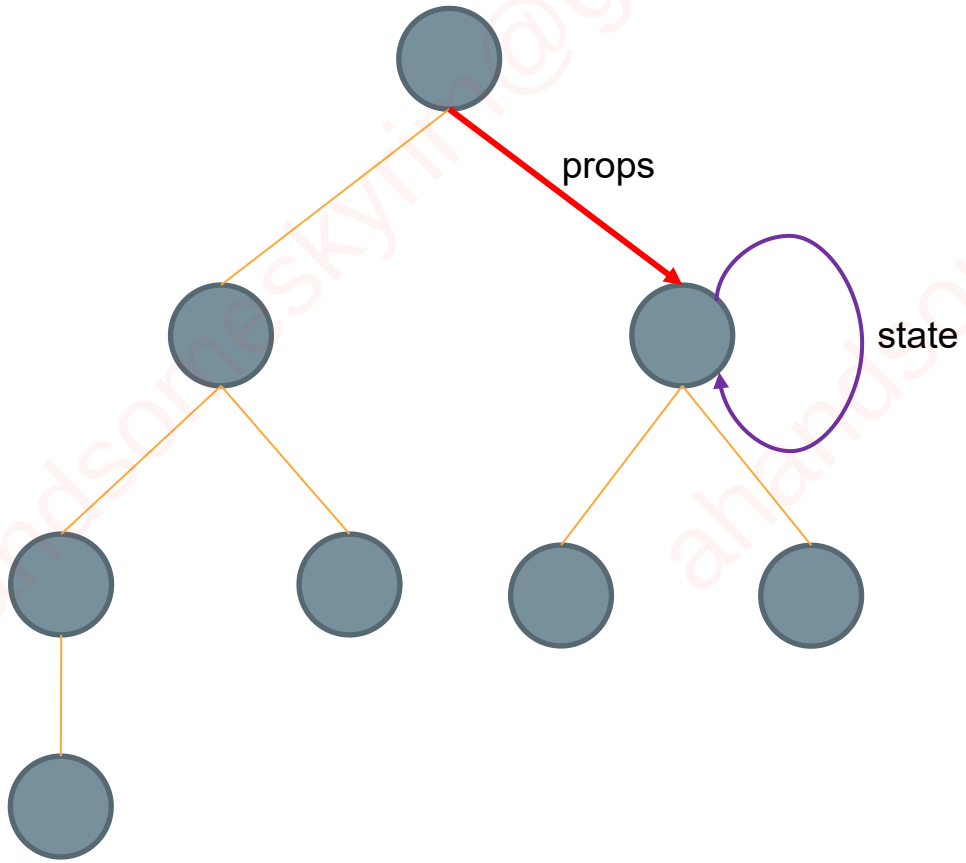
组件的生命周期-组件存在嵌套时的执行顺序



父层组件先render到内存，然后沿着组件树依次向下render

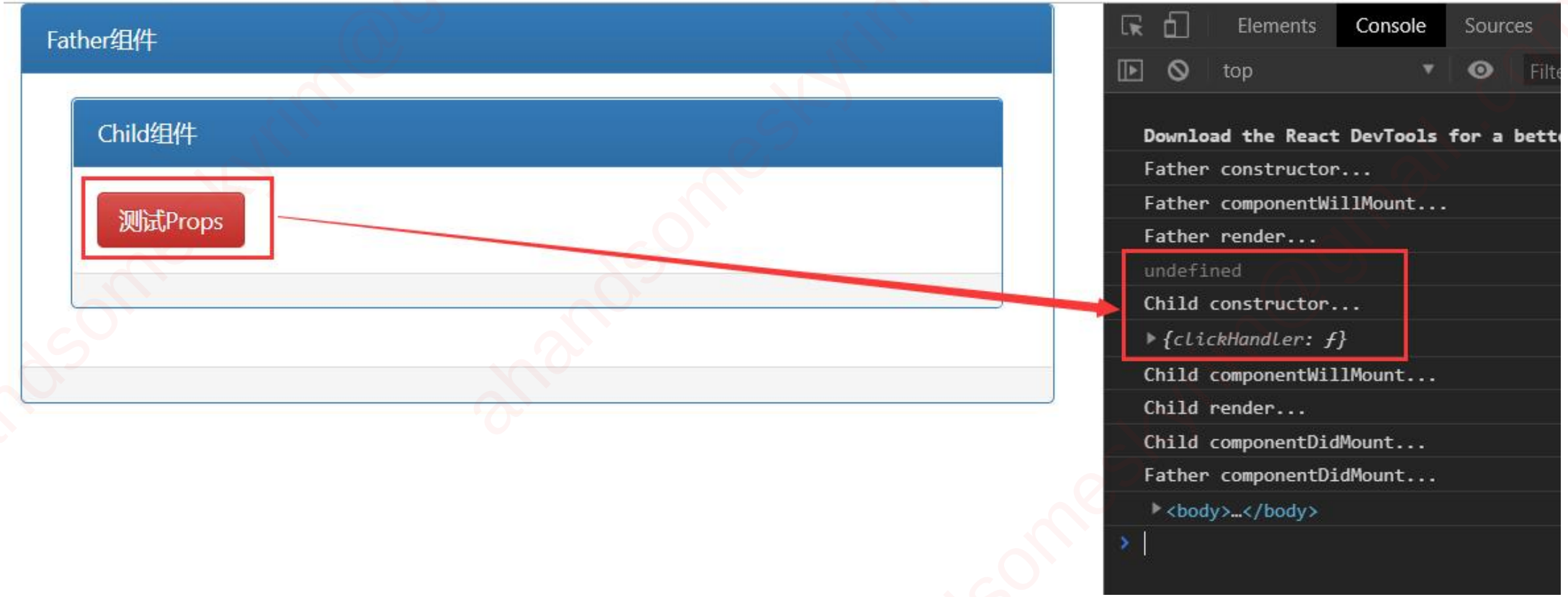
--- 上手练习 ---

数据流



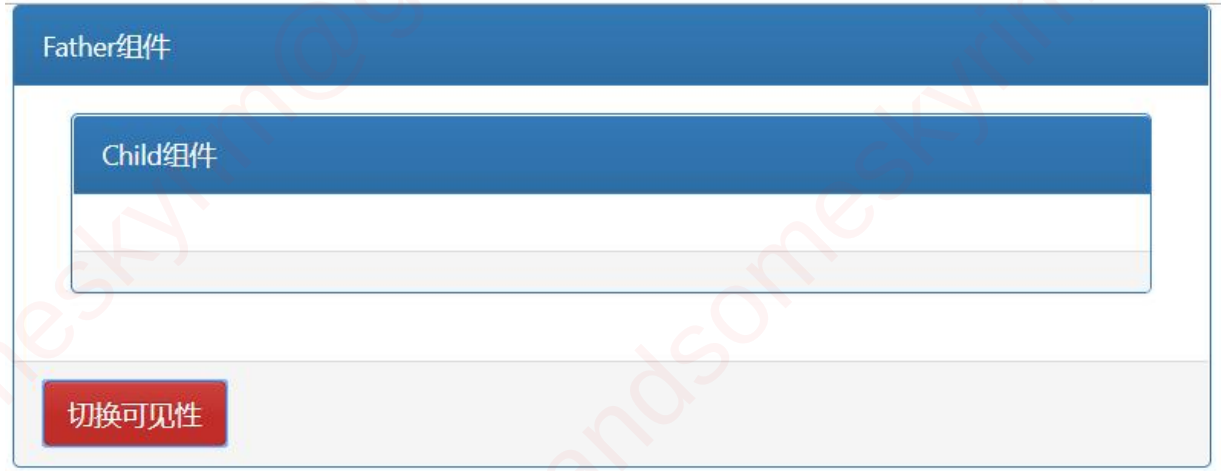
- React 采用了单向数据流的设计哲学
- props：用于父层组件向下层传递参数**（组件不能修改自己的props）**
- state：组件用来维护自己的状态**（只能用setState()函数进行修改，否则数据不会刷新）**

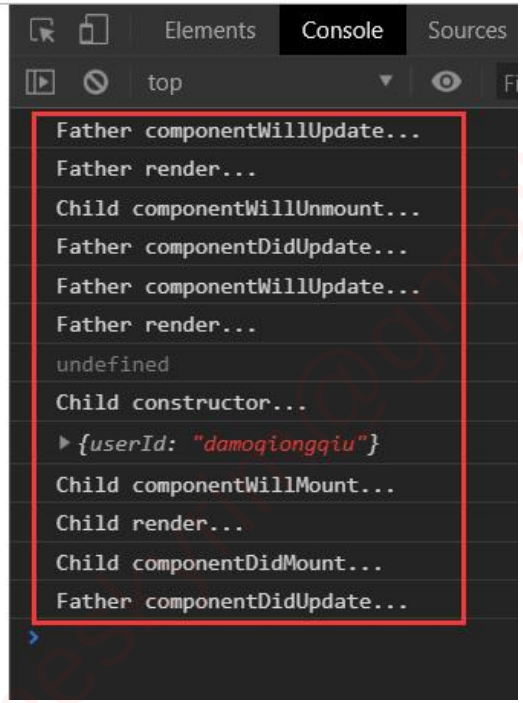
props-初始化的时机



注意：this.props在构造的时候还没有赋值，构造函数执行完成之后this.props已经被赋值

state/setState()-React中最重要的概念





注意观察父子两层上面生命周期方法执行的顺序

7.2 VDOM

--- 对于初学者这一小节有难度，可以不用完全理解，在后续学习的过程中逐步加深 ---

浏览器渲染页面的真实过程

匠人学院 JR ACADEMY

VIDEO

当我们讨论前端性能时我们在讨论什么？

匠人学院 JR ACADEMY

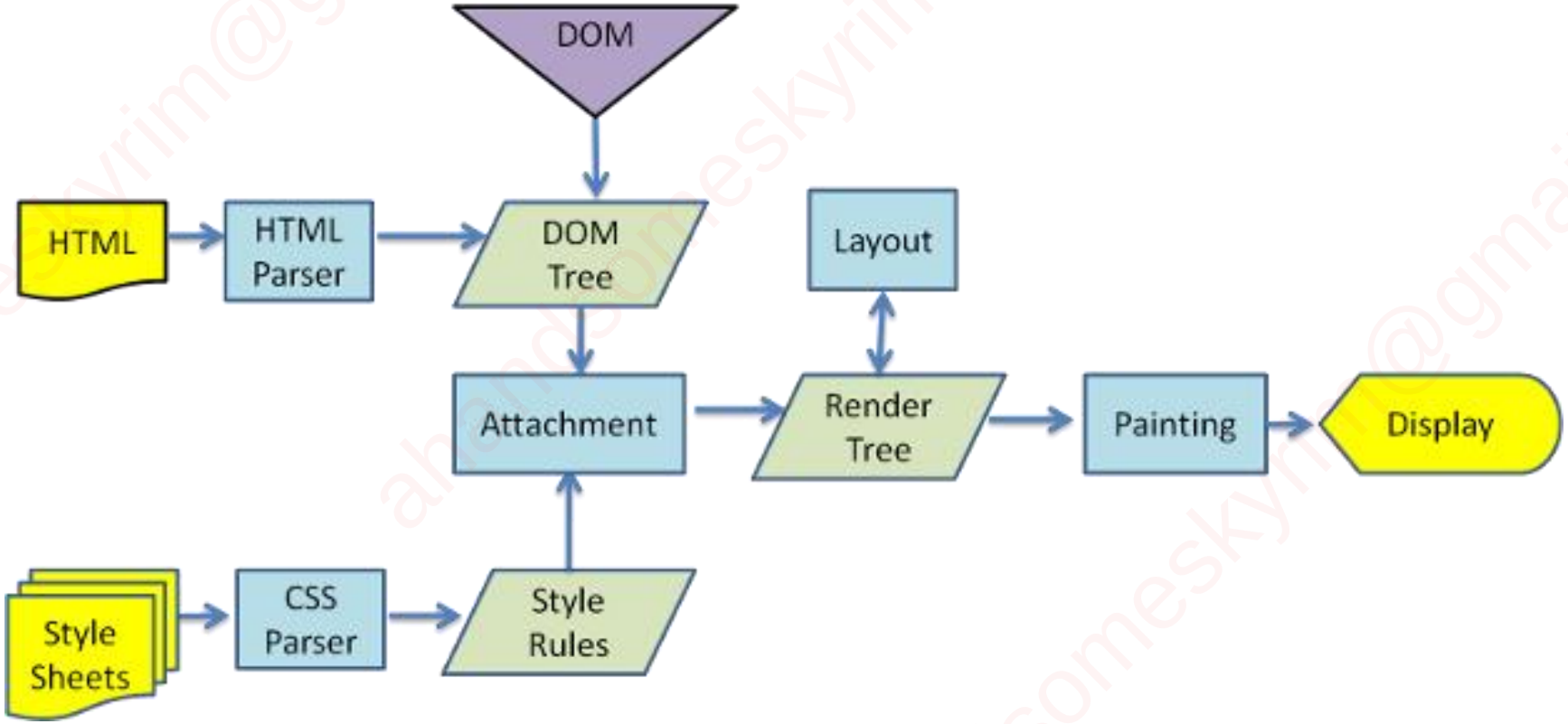
1. **DOM 是树结构**：DOM 是一个树状结构，每个节点都可能有子节点和兄弟节点。当对 DOM 进行操作时，需要对整个树进行重新构建和渲染，特别是在操作大型或复杂的 DOM 结构时（对于结构复杂的页面，DOM 节点的数量会达到数十万），这种操作会变得非常耗时（因为需要频繁递归）。

2. **频繁的重绘和重排**：DOM 操作会触发页面的重新渲染（repaint）和重排（reflow）过程。重新渲染会导致浏览器重新绘制整个页面，而回流会导致浏览器重新计算页面元素的位置和大小。这些操作都是非常消耗性能的。当对DOM进行频繁的修改时，会频繁触发页面重绘和回流，从而导致页面性能下降，使页面出现闪烁、卡顿等问题。

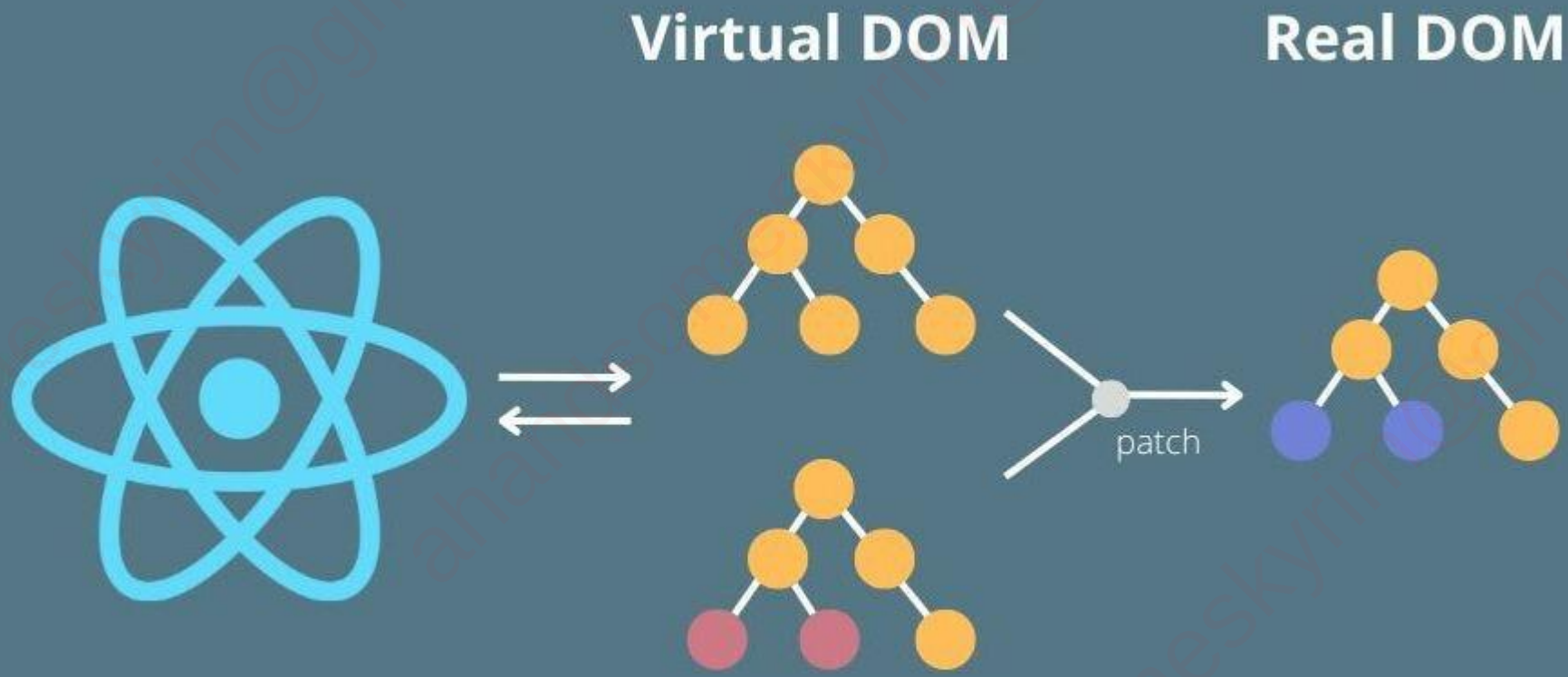
3. **IO 性能**：DOM 操作通常需要与浏览器的渲染引擎进行交互，涉及到 IO 操作，这些操作相对较慢。

4. **JavaScript 和 DOM 的性能差异**：JS 运行在自己的引擎线程中，而 DOM 操作是在主线程上进行的。经过多年的优化，尤其是 Chrome V8 团队的持续努力，**目前 JS 引擎的运行性能已经非常快。但是 DOM 操作相对较慢，当 JavaScript 频繁修改 DOM 时，性能瓶颈就会出现在 DOM 操作上。（如果你对浏览器本身的机制感兴趣，请访问 Chrome V8 的站点：https://v8.dev/ ）**

浏览器渲染页面的简要流程



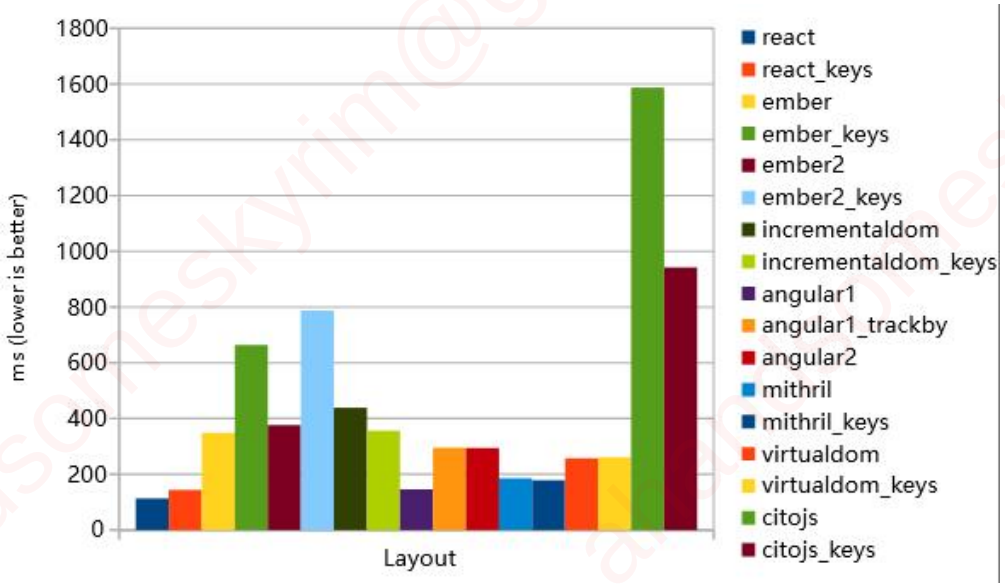
React VDOM 带来的巨大改进

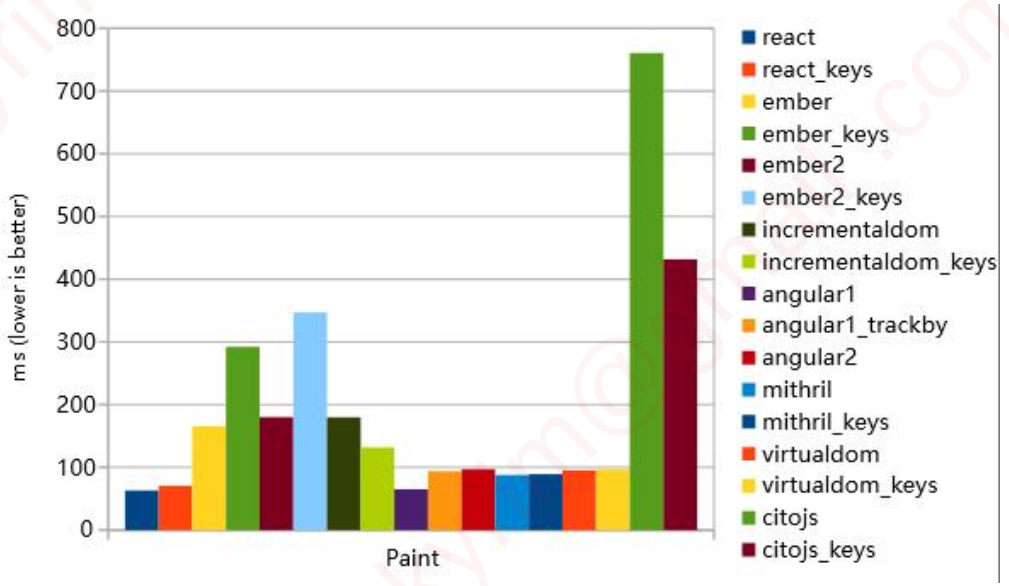


我们之所以要好好研究 VDOM，是因为它是第一个在前端开发领域实现这种思路的架构方案

对于前端框架来说，这是一个巨大的进步，VDOM 的思想让 DOM 操作效率获得了极大的提升（应该是2个数量级）

React VDOM 带来的巨大改进

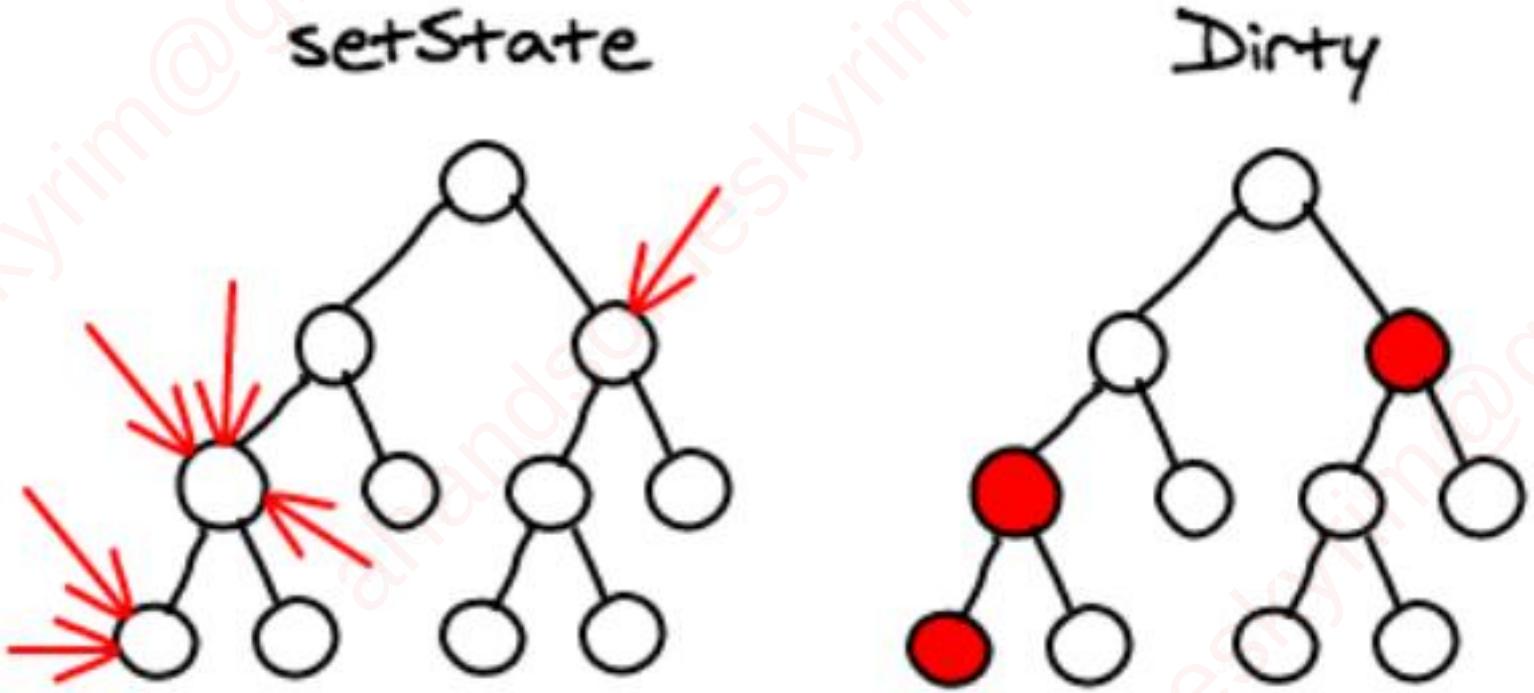




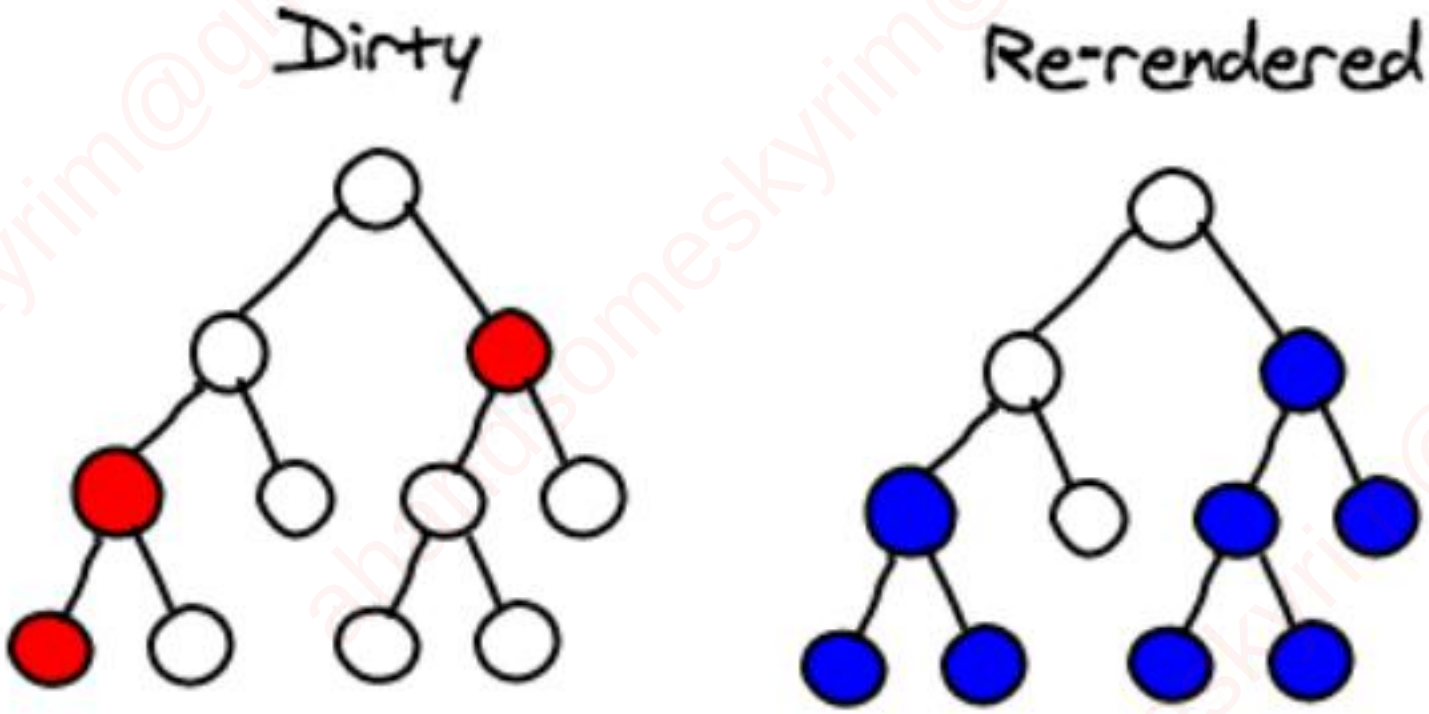
柱子越短，说明花费的时间越少，无论从哪个维度，React 的性能都是最佳

(这份 benchmark 报告时间比较久，网上有最新的 benchmark 跑分结果)

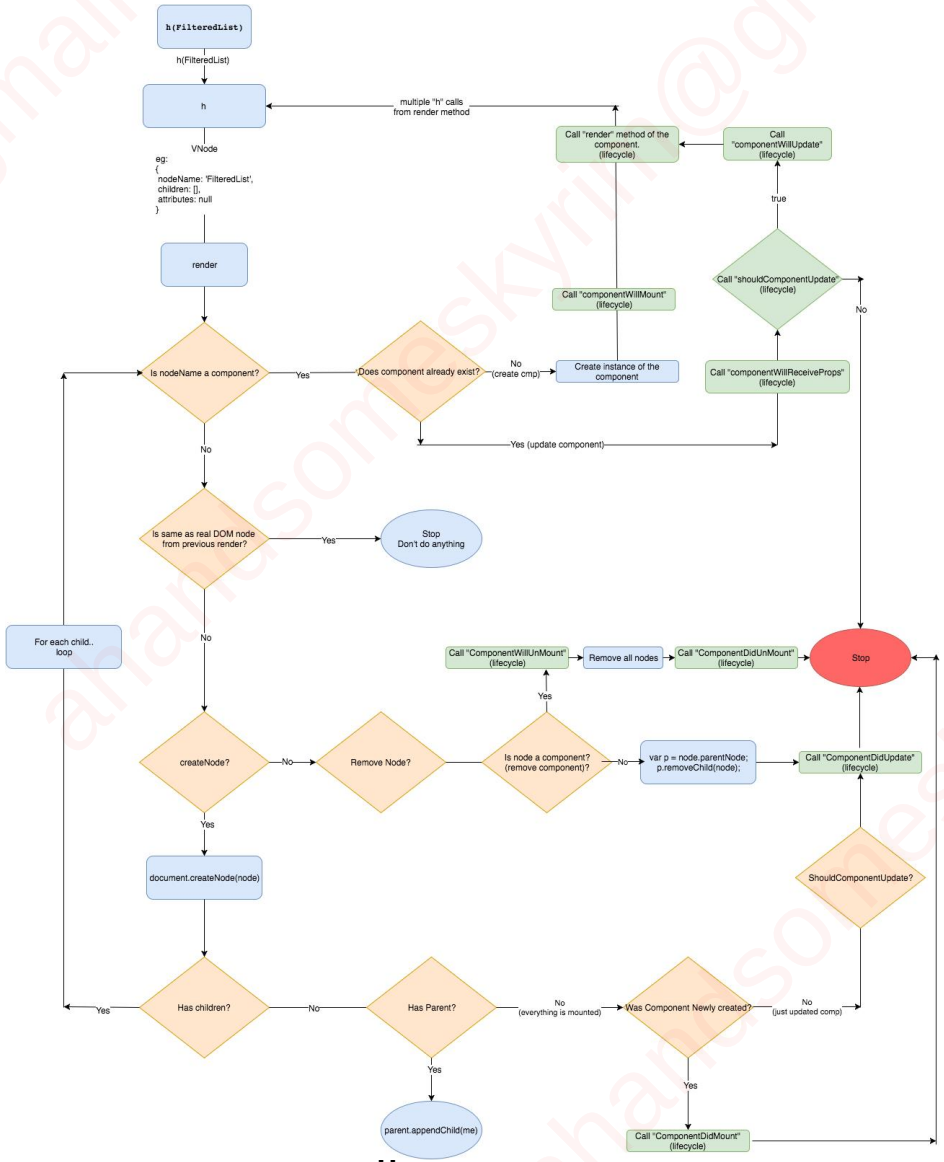
React VDOM 带来的巨大改进



React VDOM 带来的巨大改进



React VDOM-完整处理流程图



7.3 类组件与函数式组件

函数式组件

```
import React from 'react';

class HelloWorld extends React.Component {
  constructor(props) {
    super(props);
  }

  sayHi(event) {
    alert(`Hi ${this.props.name}`);
  }

  render() {
    return (
      <div>
        <a
          href="#"
          onClick={this.sayHi.bind(this)}>Say Hi</a>
      </div>
    );
  }
}

HelloWorld.propTypes = {
  name: React.PropTypes.string.isRequired
};

export default HelloWorld;
```

```
import React from 'react';

const HelloWorld = ({name}) => {
  const sayHi = (event) => {
    alert(`Hi ${name}`);
  };

  return (
    <div>
      <a
        href="#"
        onClick={sayHi}>Say Hi</a>
    </div>
  );
};

HelloWorld.propTypes = {
  name: React.PropTypes.string.isRequired
};

export default HelloWorld;
```

函数式组件-注意点

- 弱化的组件写法，只有 render 方法
- *不支持state、生命周期方法、refs、findDomNode等（注意，在 Hooks 版本之后，已经不存在这些问题。）*
- 一般把无状态的组件写成函数，语法更加简洁

7.4 复合组件

复合（嵌套）组件

```
import React, { Fragment } from "react";
import { Query } from "./Query";
import {
  Loading,
  ErrorMessage,
  EndpointList
} from "./common";

// This component subscribes to to nearest parent Query component
const DetachedEndpointList = () => (
  <Query.Consumer>
    {({ state: { data, loading, error }, actions }) => (
      <Fragment>
        <h2>QueryWithConsumingComponent</h2>
        {loading && <Loading />}
        {error && <ErrorMessage error={error} />}
        {data && <EndpointList endpoints={data} />}
      </Fragment>
    )}
  </Query.Consumer>
);

export const QueryWithConsumingComponent = () => (
  <Query url="https://api.github.com">
    <section>
      {/* Consume your query result in nested components */}
      <DetachedEndpointList />
    </section>
  </Query>
);
```

自定义的组件可以像原生HTML标签一样使用（React在幕后做了解析操作）

7.5 高阶组件

高阶组件-High Order Component

匠人学院 JR ACADEMY

High Order Function

"A function that either takes a function(s) as argument or returns a function."

$$h(x)=f(g(x))$$

高阶组件的概念来源于数学里面的"高阶函数"

高阶组件-典型用法


```
const getRouter = () => {
    return (
        <Router
            getSceneStyle={() => {
                return styles.routerStyle
            }}
            backAndroidHandler={BackUtils()}>
            <Lightbox>
                <Scene key="main">
                    <Scene key="WelcomePage" component={WelcomePage} hideNavBar hideTabBar hide />
                </Scene>
                <Scene key="LoginPage" component={LoginPage}
                    showLabel={false}
                    hideNavBar />
                <Scene key="root"
                    navigationBarStyle={styles.navigationBar}
                    titleStyle={{ color: Constant.titleTextColor }}>
                    <Scene key="mainTabPage"
                        tabs
                        lazy
                        wrap={false}
                        showLabel={false}
                        tabBarPosition={"bottom"}
                        title={I18n('appName')}
                        renderRightButton={
                            () => <SearchButton />
```

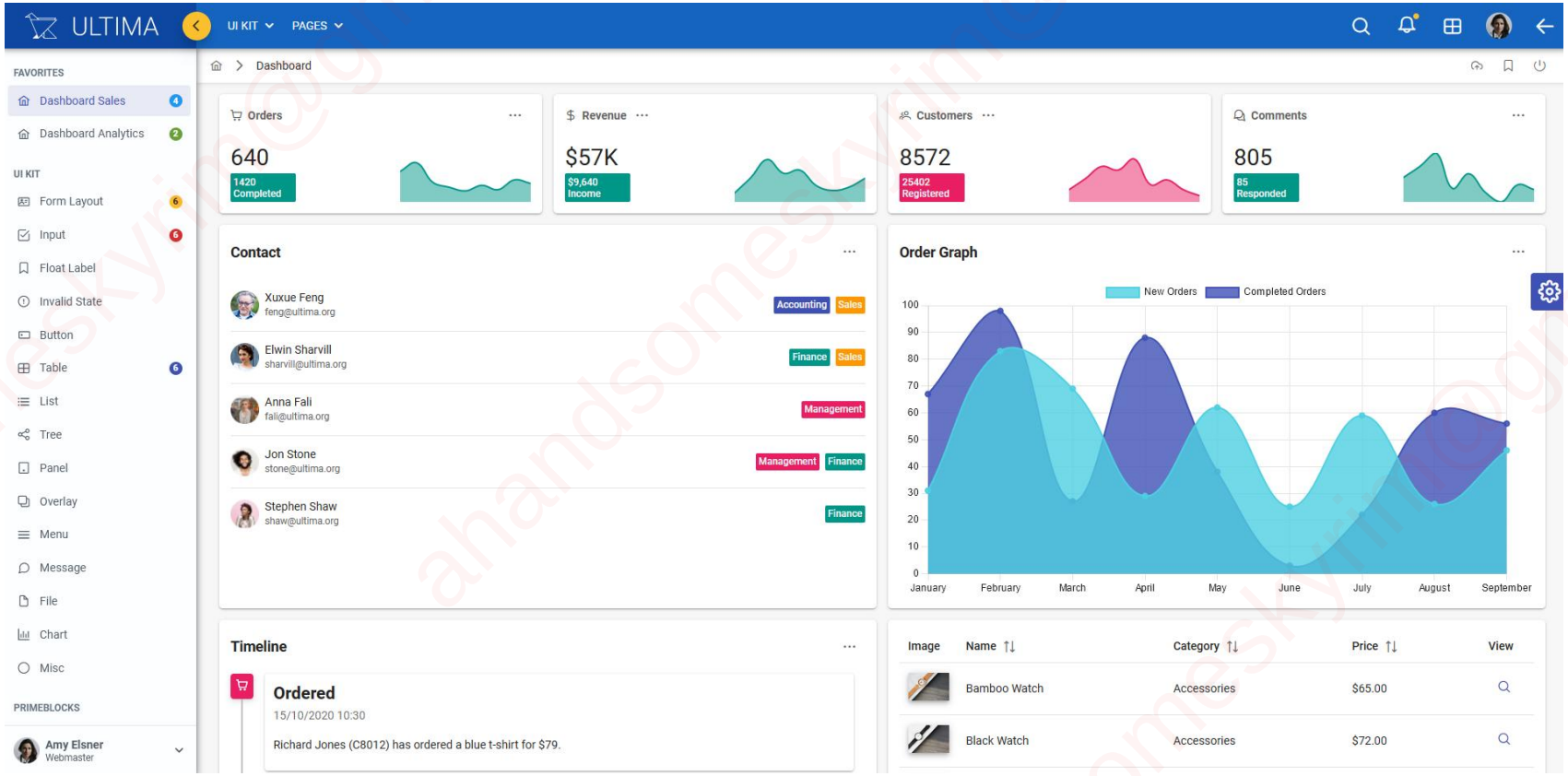
特别注意：“组件”是一个抽象的概念，组件并不一定带有UI，很多纯功能性的东西也叫“组件”，例如：路由、Store、Ajax组件等。

7.6 开源组件库

开源组件库（按照推荐程度排序）

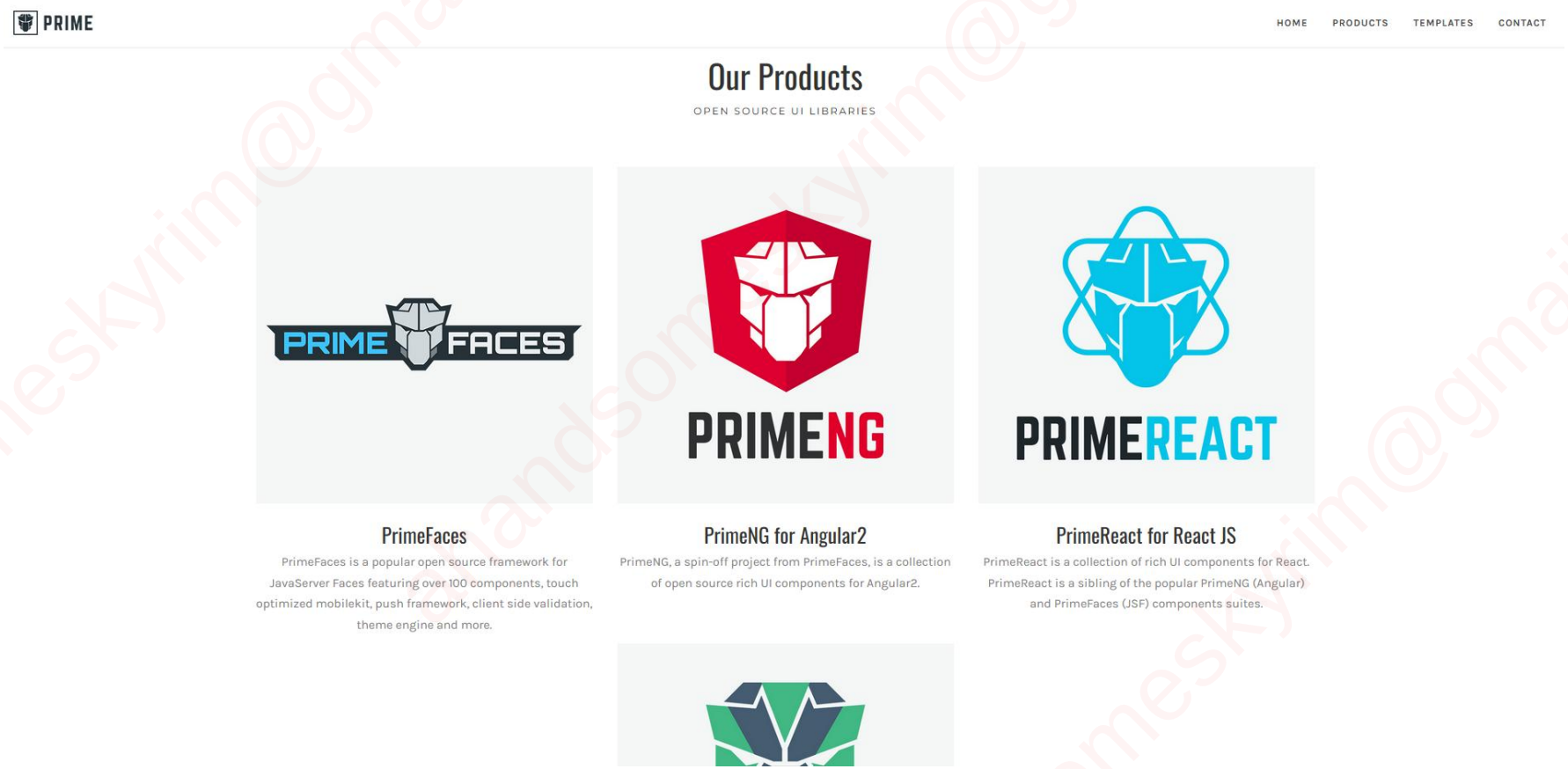
1. **PrimeReact（重点推荐）：** https://github.com/primefaces/primereact
2. **ExtReact（付费，组件质量高）：** https://www.sencha.com/products/extreact/#app
3. **AntD Pro：** https://pro.ant.design/
4. **React Bootstrap：** https://github.com/react-bootstrap/react-bootstrap
5. **Semantic-UI-React：** https://github.com/Semantic-Org/Semantic-UI-React

Prime React（重点推荐）



https://primereact.org

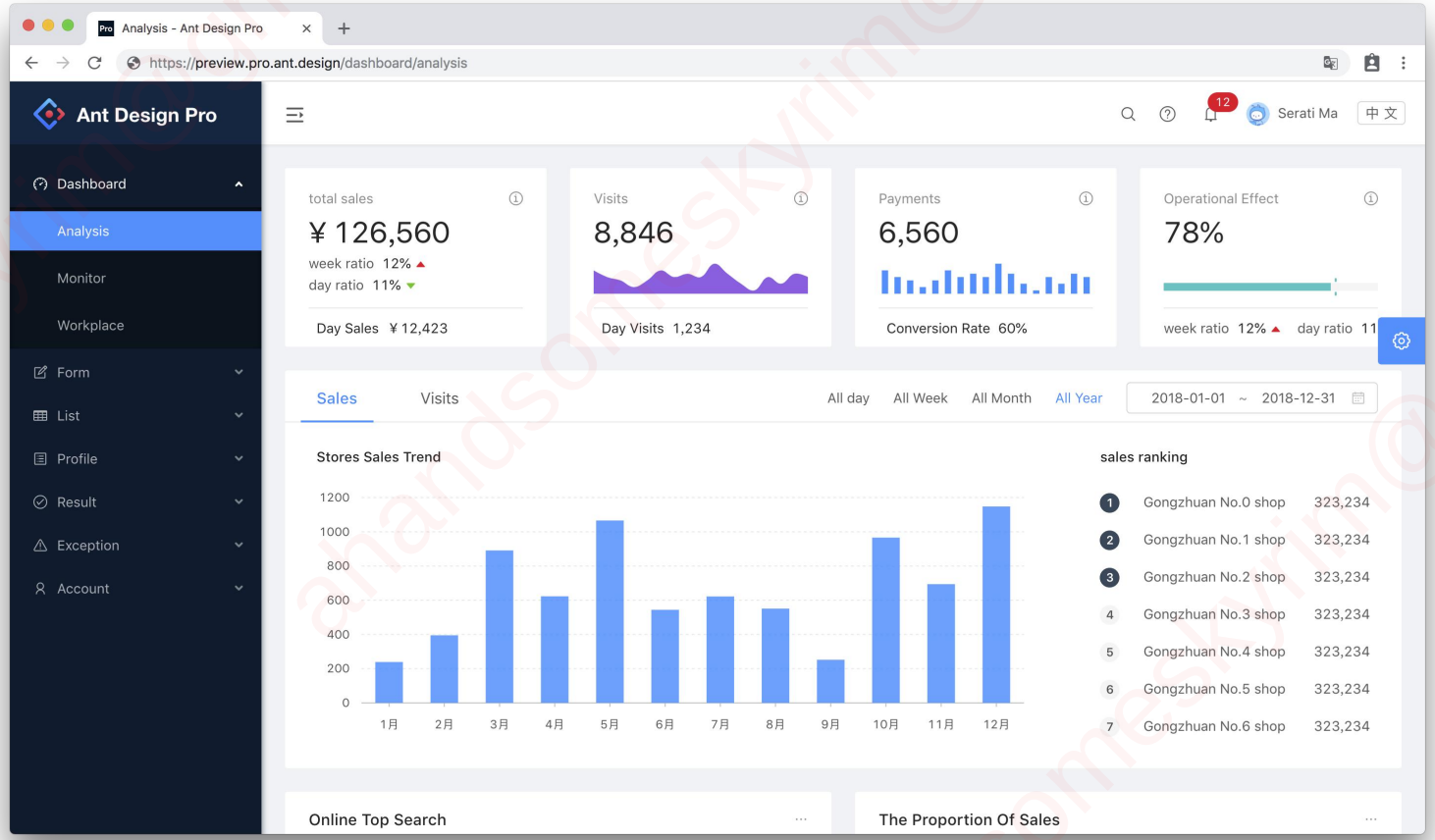
Prime React（重点推荐）



https://www.primetek.com.tr/#products

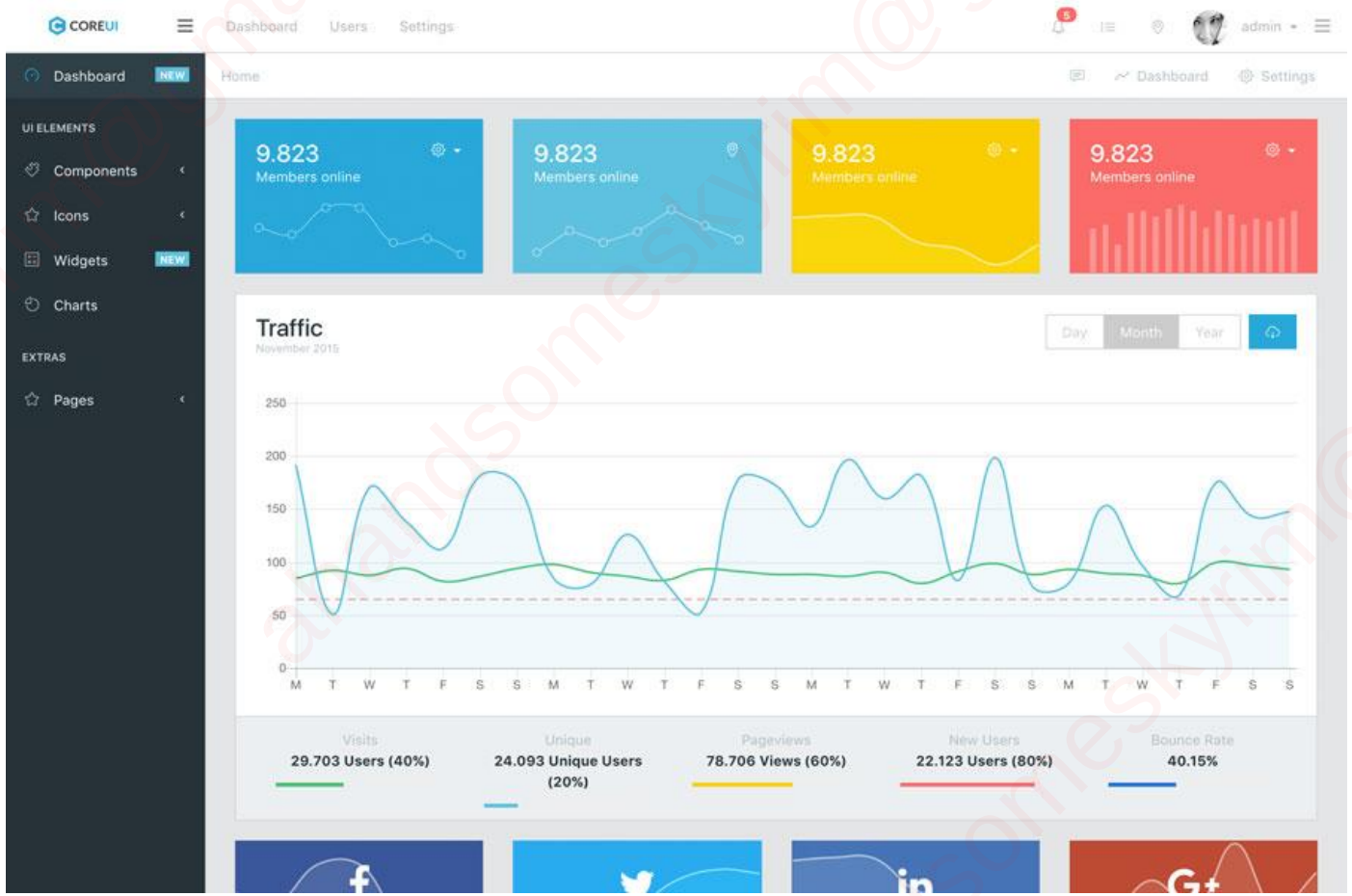
PrimeTek 是一家土耳其软件公司，他们家专门做各种 UI 组件，而且是开源的。

Ant Design Pro



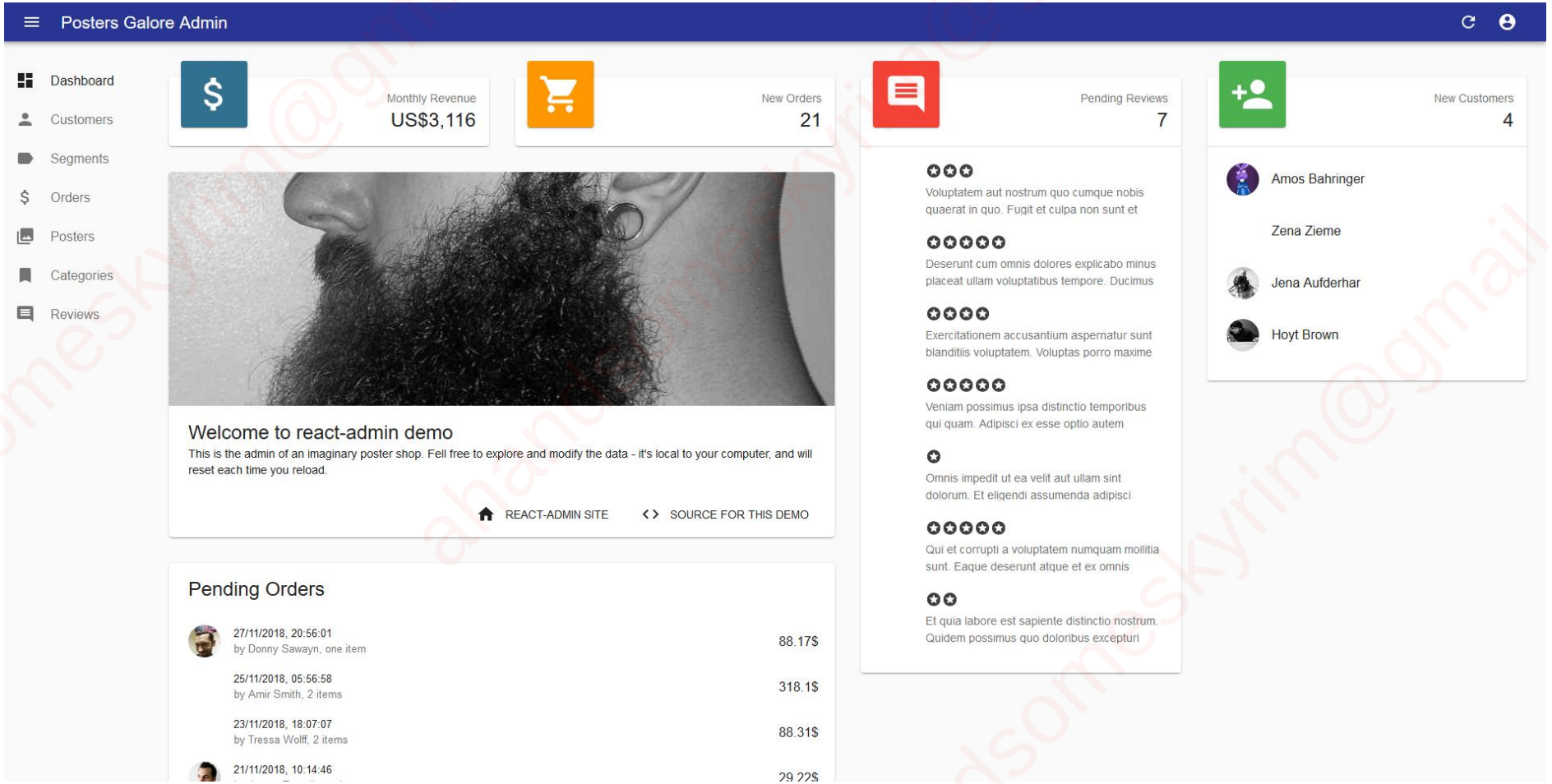
https://github.com/ant-design/ant-design-pro

CoreUI-Bootstrap风格



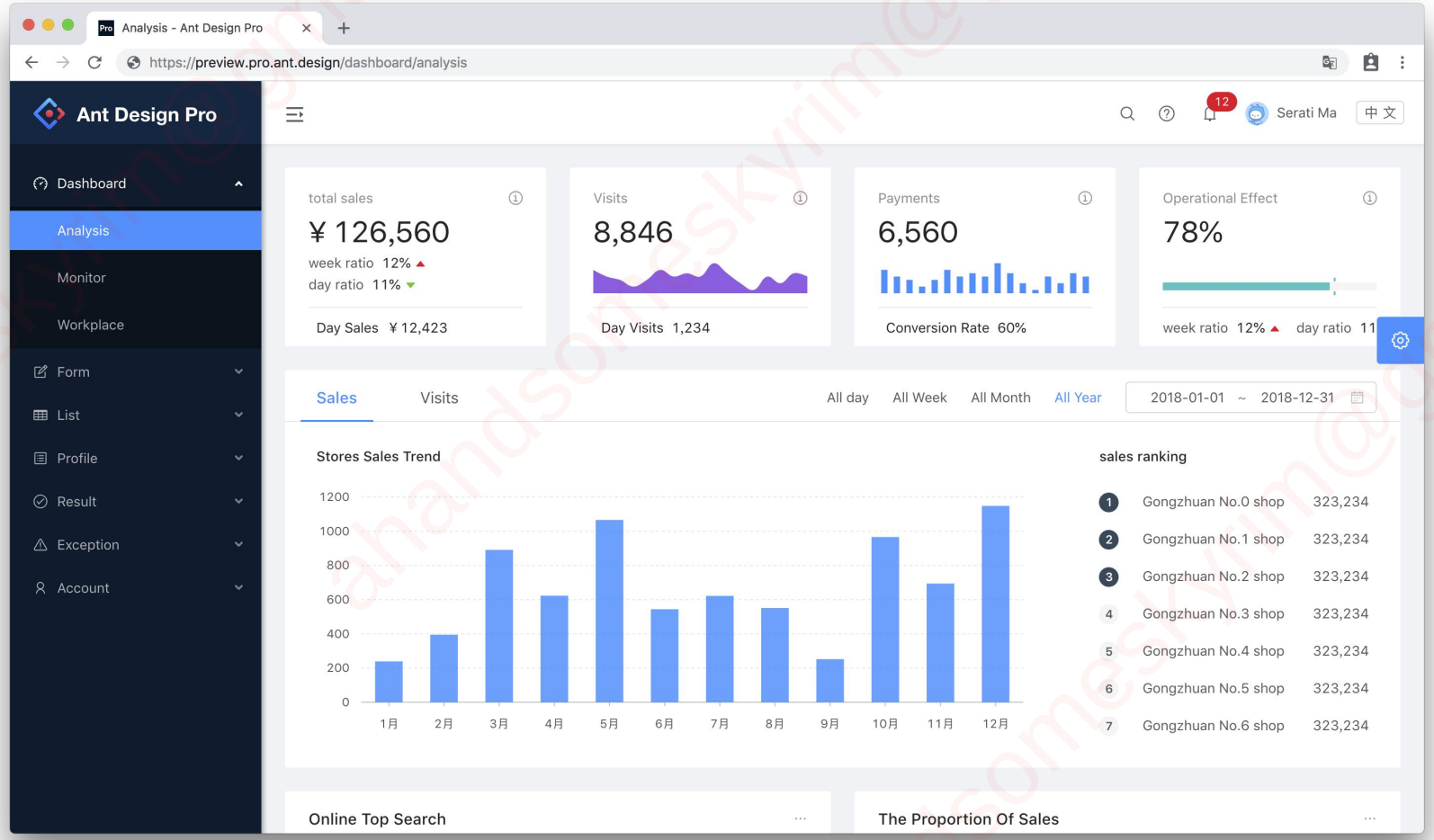
https://github.com/coreui/coreui-free-bootstrap-admin-template

Material 风格



https://github.com/marmelab/react-admin

AntD Pro-模板工程



https://pro.ant.design/index-cn

AntD Pro-浏览器支持

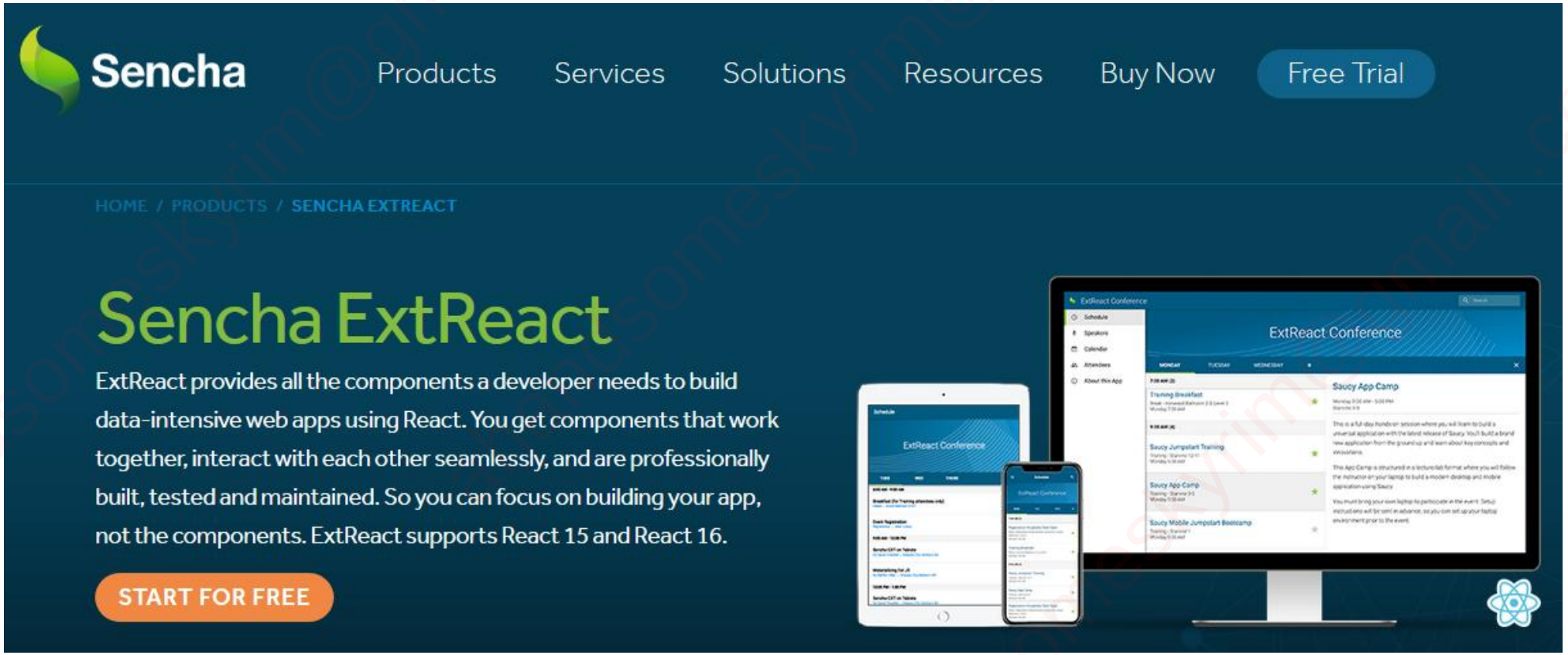
IE ~~& Edge~~	Firefox	Chrome	Safari	Opera
IE11, Edge	last 2 versions	last 2 versions	last 2 versions	last 2 versions

AntD Pro-代码结构

```
├── config                  # umi 配置，包含路由，构建等配置
├── mock                    # 本地模拟数据
├── public
│   └── favicon.png         # Favicon
├── src
│   ├── assets              # 本地静态资源
│   ├── components          # 业务通用组件
│   ├── e2e                 # 集成测试用例
│   ├── layouts             # 通用布局
│   ├── models              # 全局 dva model
│   ├── pages               # 业务页面入口和常用模板
│   ├── services            # 后台接口服务
│   ├── utils               # 工具库
│   ├── locales             # 国际化资源
│   ├── global.less         # 全局样式
│   └── global.js           # 全局 JS
├── tests                   # 测试工具
├── README.md
└── package.json
```

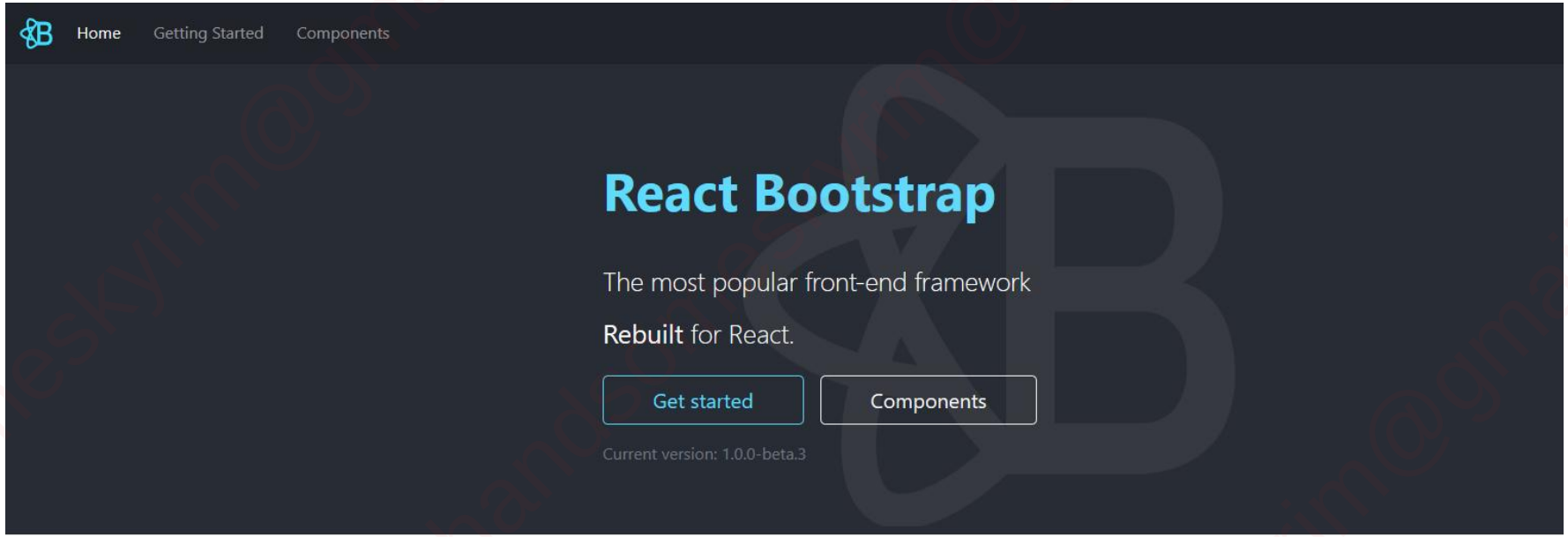
https://pro.ant.design/docs/getting-started-cn

ExtReact-付费版



https://www.sencha.com/products/extreact/#app

React Bootstrap



Rebuilt with React

React bootstrap replaces the Bootstrap javascript. Each component has been built from scratch as true React components, without uneeded dependencies like jQuery.

As one of the oldest React libraries, react bootstrap has evolved and grown along-side React, making it an exellent choice as your UI foundation.

Bootstrap at its core

Built with compatibility in mind, we embrace our bootstrap core and strive to be compatible with the world's largest UI ecosystem.

By relying entirely on the Bootstrap stylesheet, React bootstrap, just works with the thousands of bootstrap themes you already love.

Accessible by default

The React component model gives us more control over form and function of each component.

Each component is implemented with accessibilty in mind. The result is a set of accessible-by-default components, over what is possible from plain Bootstrap.

Semantic-UI-React

Semantic UI React

gitter join chat | build passing | coverage 99% | dependencies up to date | npm v0.83.0

综合对比表格

匠人学院 JR ACADEMY

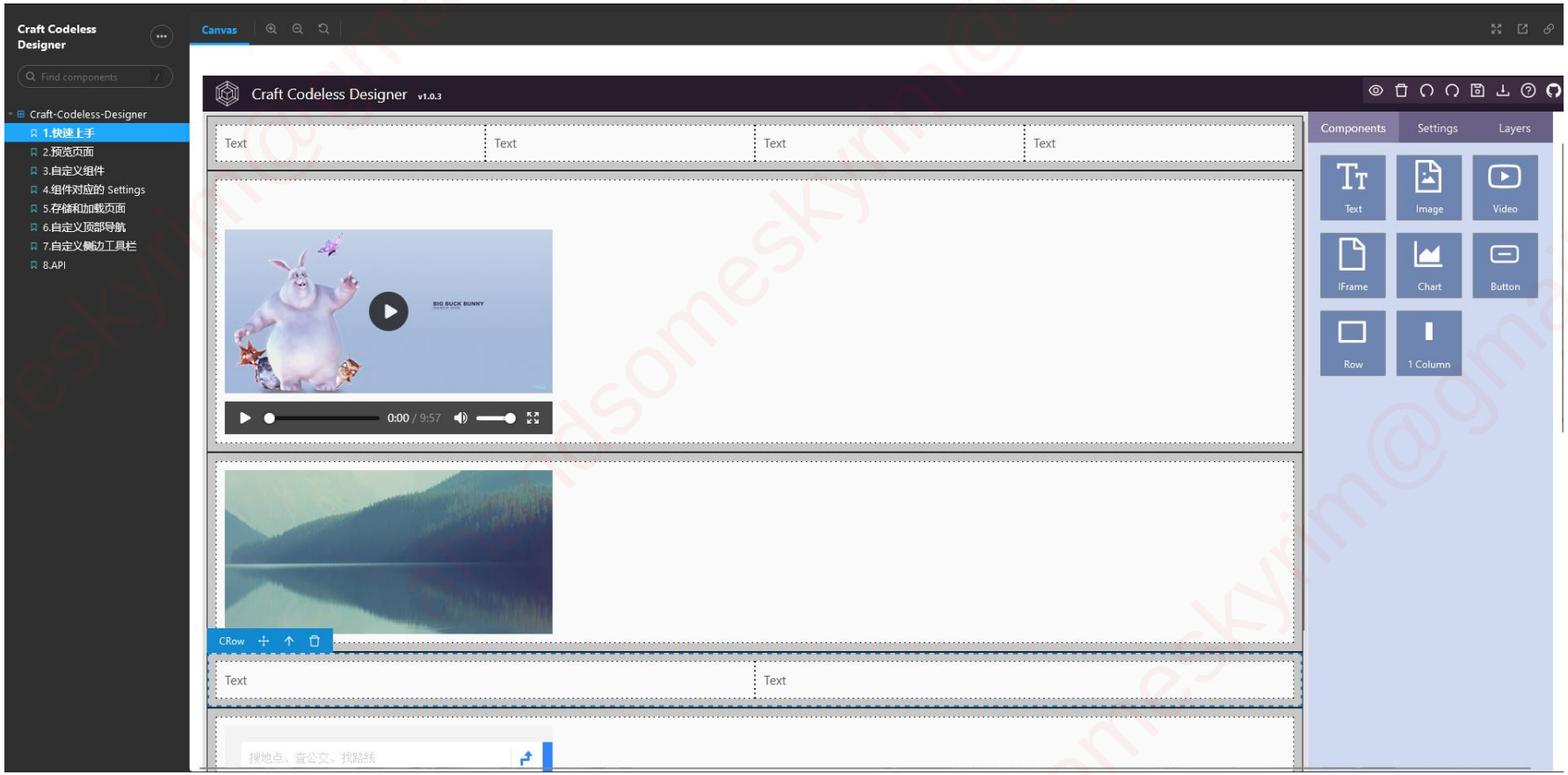
Framework / library	Design language / styling	Number of components	Maintained by	Github Contributors / Stars	Supported browsers
Material-UI	Google Material Design	30+	Call-Em-All, Community	440 / 27K	Crome 49+, Firefox 45+, IE 11, Edge 14+, Safari 10+
Ant Design	Proprietary design language	50	AFX, Community	300 / 16K	All modern browsers, IE 9+ (with polyfills)
React-Bootstrap	Twitter Bootstrap styling	30+ (Bootstrap 3 components)	Community	183 / 10.6K	All modern browsers, IE 9+ (with polyfills)
React-Toolbox	Google Material Design	28+	Community	206 / 6.7K	All modern browsers, IE 11
Blueprint	Proprietary minimalist styling	30+	Palantir	53 / 6.7K	Chrome, Firefox, Safari, IE 11, Edge
Semantic-UI-React	Proprietary minimalist styling	30+	Semantic Org, Community	111 / 3.8K	Latest 2 versions modern browsers; IE 11+
Fabric	Office Design Language (Microsoft)	40	Microsoft, Community	146 / 1.3K	Latest 2 versions of all modern browsers, IE 11
Grommet	Hewlett Packard Enterprise	70+	Hewlett Packard, Community	89 / 2.2K	All modern browsers, IE 11
Rebass	Custom minimalist styling	64	Jxnblk / Brent Jackson, Community	25 / 2.9K	Flexbox-friendly browsers (no IE)
React-MD	Google Material Design	40	Mikkel Laursen, Community	20 / 1.2K	All modern browsers, IE 11
React Desktop	macOS, Windows 10 UI elements	19 (macOS), 12 (win 10)	Community	11 / 6.4K	All modern browsers

https://agileengine.com/react-component-libraries-and-frameworks-to-watch-on-github/

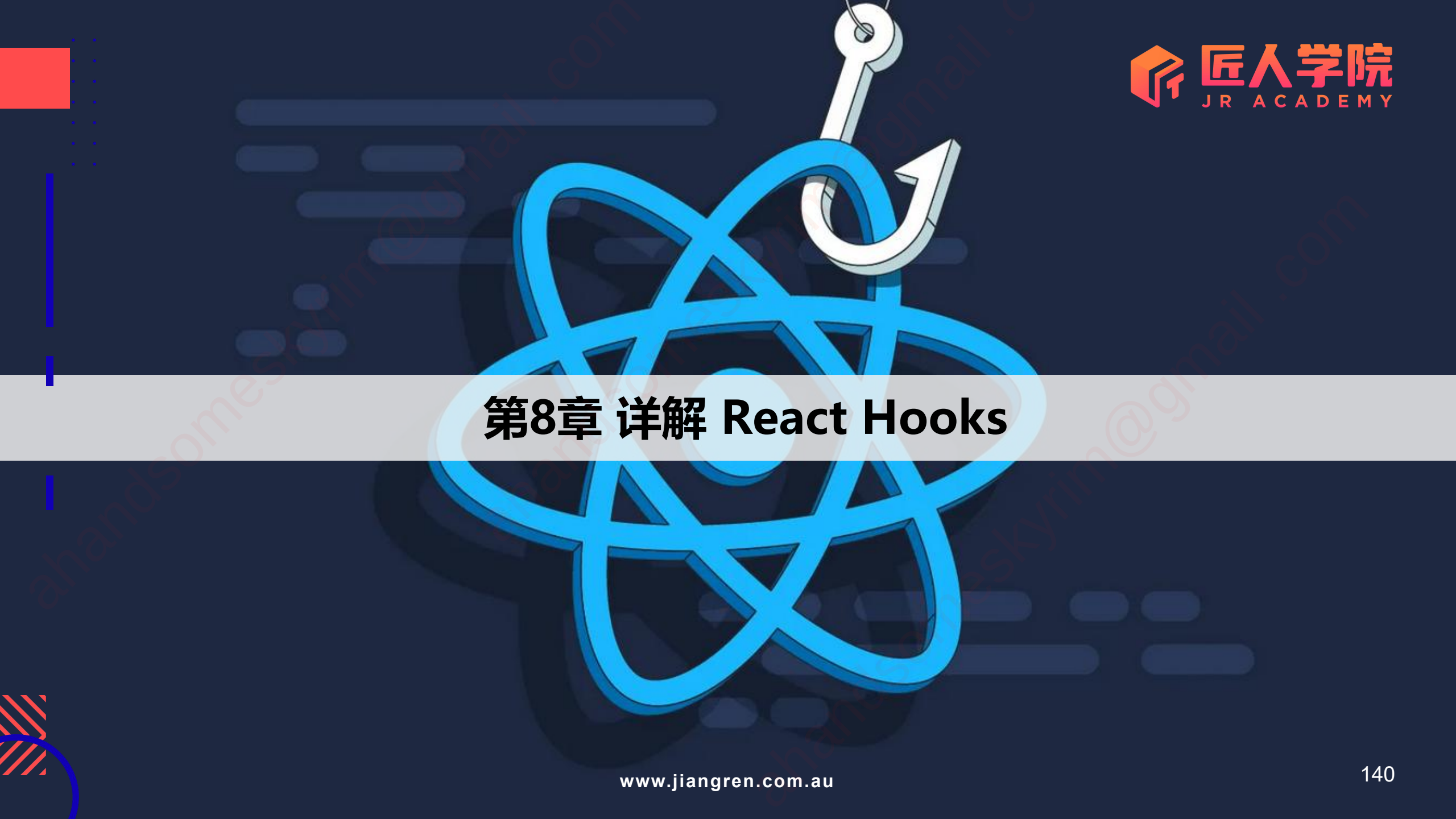
7.7 企业开发中的业务组件库

7.8 动态生成页面与低代码系统

低代码与零代码



https://gitee.com/craft-codeless-designer/craft-codeless-designer



匠人学院
JR ACADEMY

第8章 详解 React Hooks

匠人学院
JR ACADEMY

8.1 React 内置的 Hooks

hooks 特性是从 v16.8 开始引入的，2019年2月发布，

此前的版本没有 hooks 概念，函数式组件里面也不能使用生命周期函数

React 内置的 Hooks

1. **useState**：用于在函数组件中引入状态管理。它返回一个状态变量和一个更新状态的函数。
2. **useEffect**：用于在函数组件中执行副作用操作，比如订阅和取消订阅、数据获取和修改DOM等。
3. **useContext**：用于在函数组件中获取上下文对象。
4. **useReducer**：类似于Redux中的Reducer，用于处理复杂的状态逻辑。它返回状态和一个分发action的函数。
5. **useCallback**：用于缓存回调函数，以避免不必要的重新创建。
6. **useMemo**：用于缓存计算结果，以避免不必要的重复计算。
7. **useRef**：用于获取DOM元素或在渲染之间保存值。
8. **useImperativeHandle**：用于自定义暴露给父组件的实例值。
9. **useLayoutEffect**：类似于useEffect，但会在DOM更新之后同步执行。
10. **useDebugValue**：用于在自定义Hooks中显示自定义标签值。

--- FIXME: 需要更新到最新版本 ---

8.2 理解 useState

useState & useEffect 用法

```
1  import React, { useEffect, useState } from "react";  6.9k (gzipped: 2.7k)
2  import postListMock from "src/mock-date/post-list-mock.json";
3
4  function PostList(props) {
5      const [postList, setPostList] = useState([]);
6
7      useEffect(() => {
8          setPostList(postListMock);
9      }, []);
10
11     return (
12         <ul className="list-group">
13             <li className="list-group-item active">
14                 热门内容
15             </li>
16             {
17                 postList.map((post, index) => (
18                     <li className="list-group-item" key={post.id}>
19                         <div className="d-flex justify-content-between">
20                             <span>{post.title}</span>
21                             <span>{post.updateTime}</span>
22                         </div>
23                     </li>
24                 ))
25             }
26         </ul>
27     );
28 }
29
30 export default PostList;
```

理解 useState

8.3 理解 useEffect

理解 useEffect

Purity: Components as formulas

In computer science (and especially the world of functional programming), a pure function is a function with the following characteristics:

- **It minds its own business.** It does not change any objects or variables that existed before it was called.
- **Same inputs, same output.** Given the same inputs, a pure function should always return the same result.

You might already be familiar with one example of pure functions: formulas in math.

Consider this math formula: $y = 2x$.

If $x = 2$ then $y = 4$. Always.

If $x = 3$ then $y = 6$. Always.

If $x = 3$, y won't sometimes be 9 or -1 or 2.5 depending on the time of day or the state of the stock market.

If $y = 2x$ and $x = 3$, y will *always* be 6.

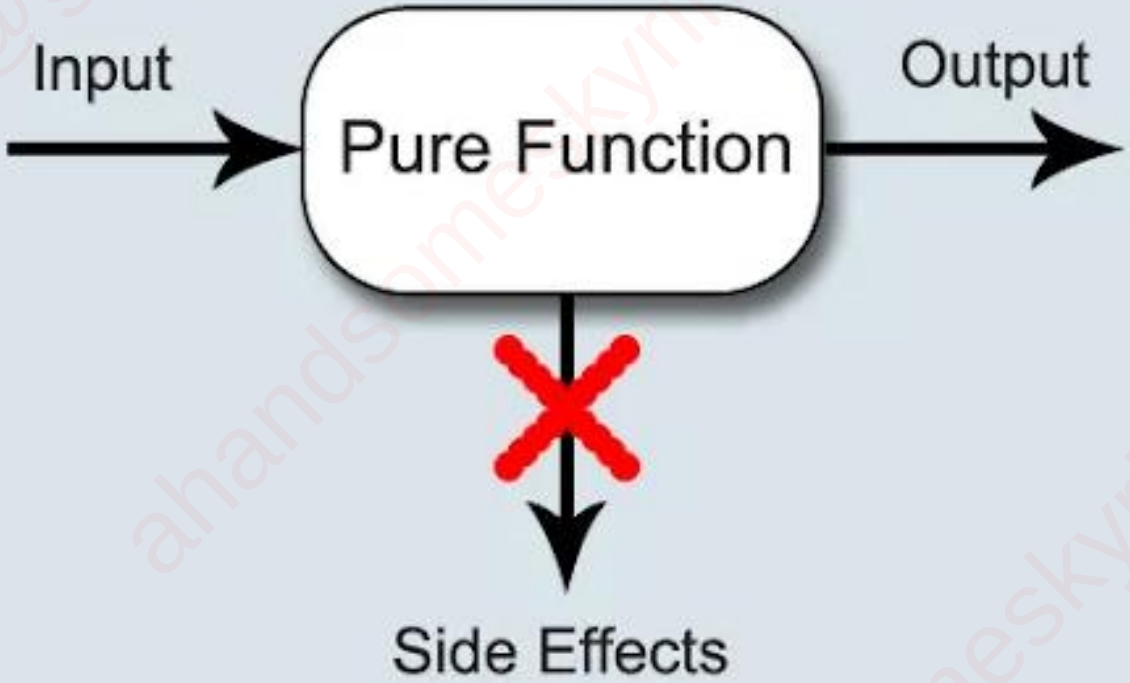
If we made this into a JavaScript function, it would look like this:

```
function double(number) {
  return 2 * number;
}
```

"纯函数"是 React v16 之后的一个重要的设计思想，"纯函数"的思想带来了极大的便利，比如：可以非常方便地构造"高阶组件"。

随着学习的深入，你将会逐步体会到这个设计思想的强大之处

理解 useEffect



“纯函数”是 React v16 之后的一个重要的设计思想，“纯函数”的思想带来了极大的便利，比如：可以非常方便地构造“高阶组件”。

随着学习的深入，你将会逐步体会到这个设计思想的强大之处

理解 useEffect

匠人学院 JR ACADEMY

- **从字面意思理解：**Effect 是 sideeffect 的缩写，sideeffect 的字面意思是“副作用”，副作用是相对于“纯函数”而定义的。所谓的“副作用”，指的是那些不可预知的效果。很显然，对于 React 组件来说，开发者希望它们总是“纯粹的”，对于相同的输入，总是有相同的输出，而不要出现那些不在预期之中的作用。

- **从 React 内部机制上理解：**useEffect 可以看成把 componentDidMount, componentDidUpdate, componentWillUnmount 这3个生命周期的功能合在了一起。

- **所以：**对于那些影响到组件“纯粹性”的逻辑，都可以看成“副作用”。一般来说，在项目开发实践中，开发者应该把以下这些操作放在 useEffect 中：**获取服务端数据、DOM 操作、修改组件的状态、在组件卸载时做的清理资源操作。**

理解 useEffect-只在 mount 时运行一次的副作用

```
7   useEffect(() => {
8       const fetchPostList = async () => {
9           try {
10              const response = await fetch('http://localhost:3001/postList');//这里也可以使用我们自己封装的 axios-service
11              const data = await response.json();
12              setPostList(data);
13              setIsLoading(false);
14          } catch (error) {
15              console.error('Error fetching post list:', error);
16              setIsLoading(false);
17          }
18      };
19      fetchPostList();
20  }, []);
21
```

传一个空数组，内部的代码只在 mount 时执行一次
如果有返回的函数，React 会尝试在组件 unmount 时执行它，用来清理资源

只在 mount 时运行一次的“副作用”

理解 useEffect-有条件执行的副作用

有条件执行的“副作用”

理解 useEffect-返回清理函数

```jsx
import React, { useState, useEffect } from 'react';  6.9k (gzipped: 2.7k)

const MyHookExample = () => {
    const [count, setCount] = useState(0);

    // useEffect 副作用函数
    useEffect(() => {
        // 此处是副作用操作
        console.log('Component mounted');

        // 返回一个清理函数，在组件卸载时执行
        return () => {
            console.log('Component unmounted');
            // 在这里可以进行一些清理操作，比如取消订阅、清除定时器等
        };
    }, []); // 依赖项数组为空，表示只在组件 mount 时执行一次

    return (
        <div>
            <p>Count: {count}</p>
            <button onClick={() => setCount(count + 1)}>Increment</button>
        </div>
    );
};

export default MyHookExample;
```

8.4 自定义 Hooks

自定义 Hooks

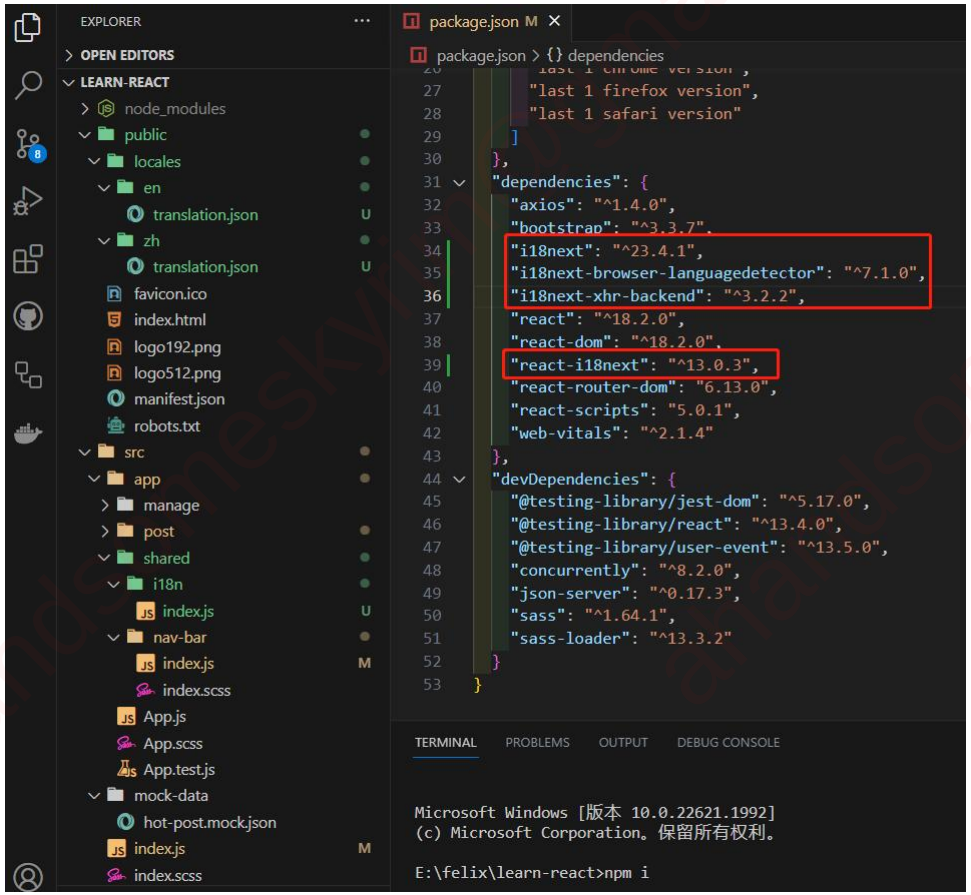
use-post-list.js U

src > app > post > hot-post > use-post-list.js > default

```
1   import { useState, useEffect } from 'react';  4.2k (gzipped: 1.8k)
2
3   const usePostList = () => {
4       const [postList, setPostList] = useState([]);
5       const [isLoading, setIsLoading] = useState(true);
6
7       useEffect(() => {
8           const fetchPostList = async () => {
9               try {
10                  const response = await fetch('http://localhost:3001/postList');//这里也可以使用我们自己封装的 axios-service
11                  const data = await response.json();
12                  setPostList(data);
13                  setIsLoading(false);
14              } catch (error) {
15                  console.error('Error fetching post list:', error);
16                  setIsLoading(false);
17              }
18          };
19          fetchPostList();
20      }, []);
21
22      return { postList, isLoading };
23  };
24
25  export default usePostList;
```

8.5 用 useTranslation 钩子实现组件的 i18n

step1: 安装 i18n 工具模块



npm i **i18next i18next-browser-languagedetector i18next-xhr-backend react-i18next** --save

1. **i18next**：国际化工具库，它和具体的框架无关，是一个通用的工具库。
2. **react-i18next**：针对 React 进行了封装，方便 React 组件调用。
3. **i18next-browser-languagedetector**：工具库，用来检测浏览器当前的语言。
4. **i18next-xhr-backend**：工具库，用来异步加载 i18n 资源文件。

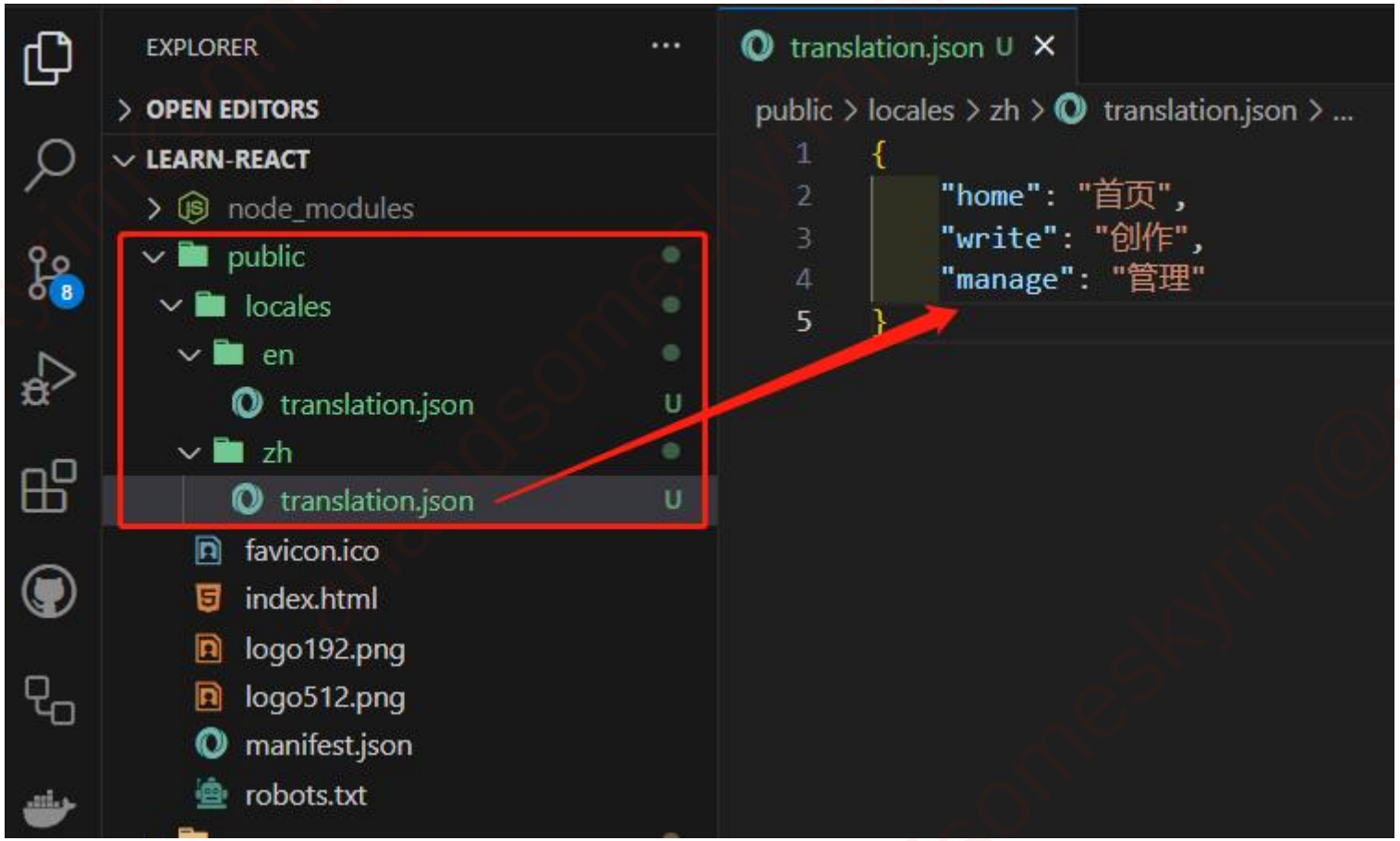
step2: 封装 i18n 模块，提供初始化参数

src > app > shared > i18n > JS index.js > ...

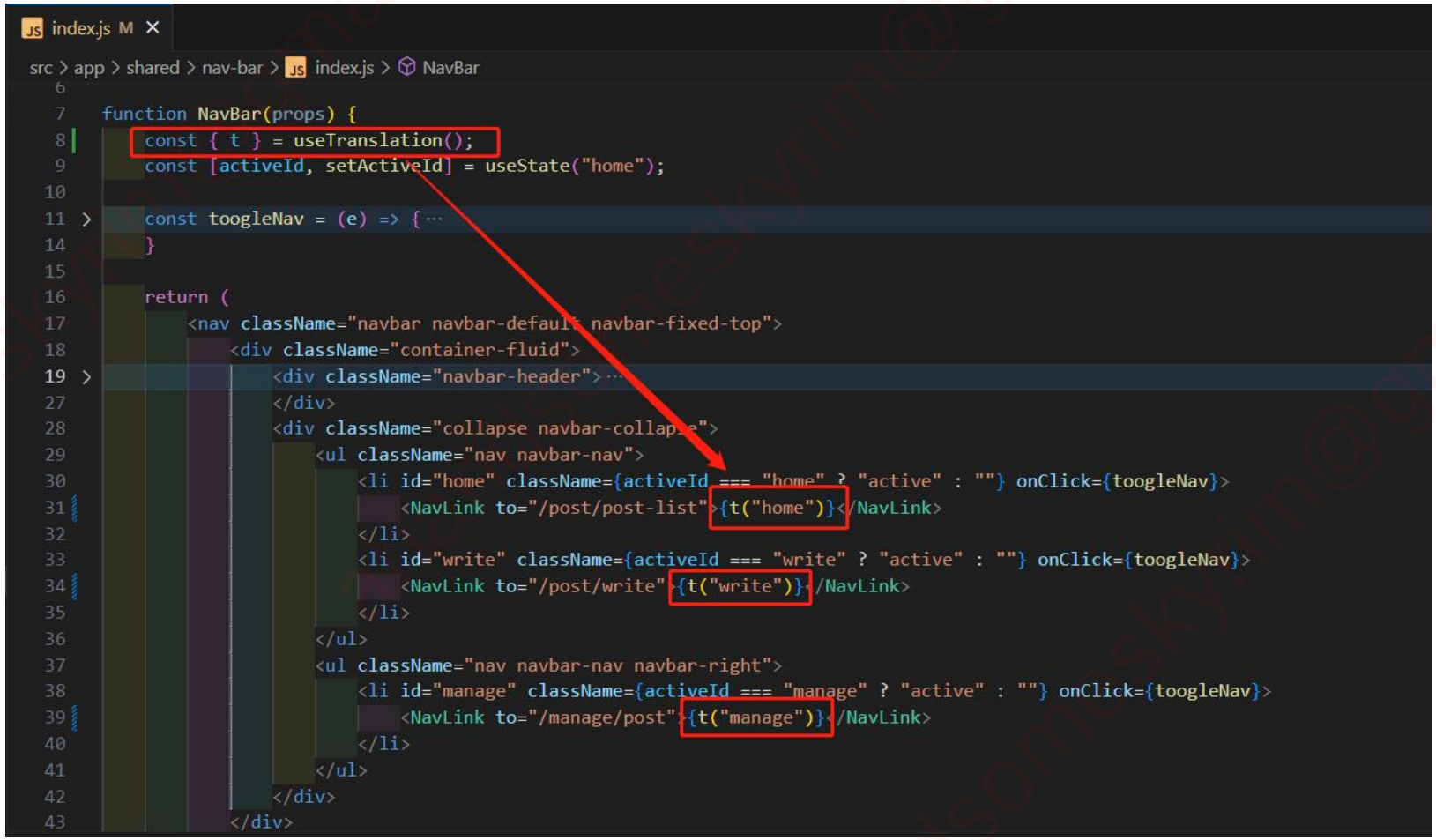
```
1   import i18n from 'i18next';  47.4k (gzipped: 13.6k)
2   import Backend from 'i18next-xhr-backend';  3.9k (gzipped: 1.6k)
3   import LanguageDetector from 'i18next-browser-languagedetector';  7.5k (gzipped: 2.5k)
4   import { initReactI18next } from 'react-i18next';  1.6k (gzipped: 895)
5
6   i18n
7       .use(Backend)
8       .use(LanguageDetector)
9       .use(initReactI18next)
10      .init({
11          lng: navigator.language,    //动态指定语言使用 changeLanguage 函数
12          fallbackLng: 'en',
13          debug: true,
14          interpolation: {
15              escapeValue: false,
16          }
17      });
18  export default i18n;
```

在应用的入口文件 index.js 中引用这份封装好的模块

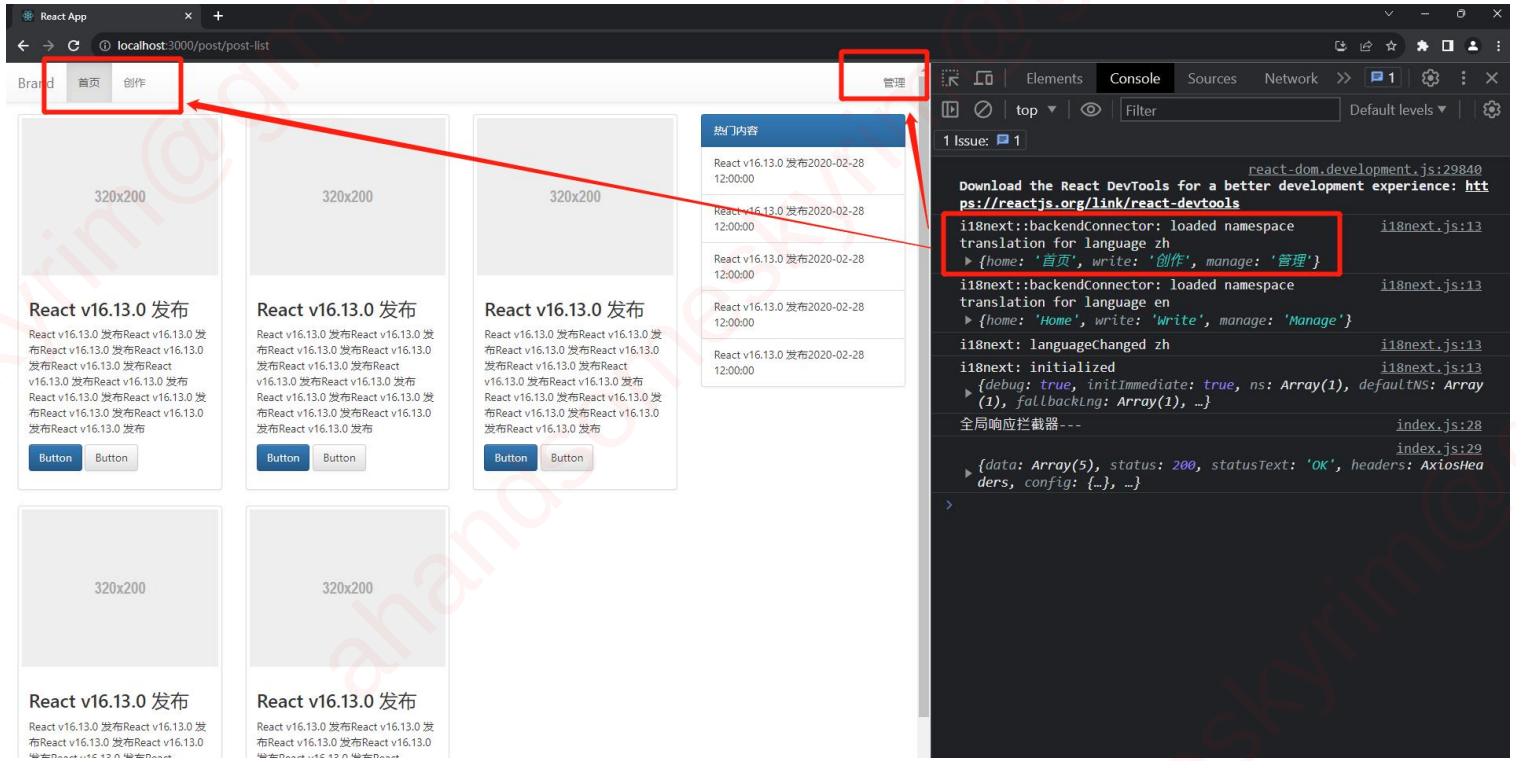
step3: 提供翻译好的字符串资源文件



step4: 在组件中使用国际化字符串



step5: 测试 i18n 效果



注意：修改了浏览器的语言设置之后，需要关闭所有浏览器标签，然后重启浏览器才能生效，这是浏览器本身的机制决定的。

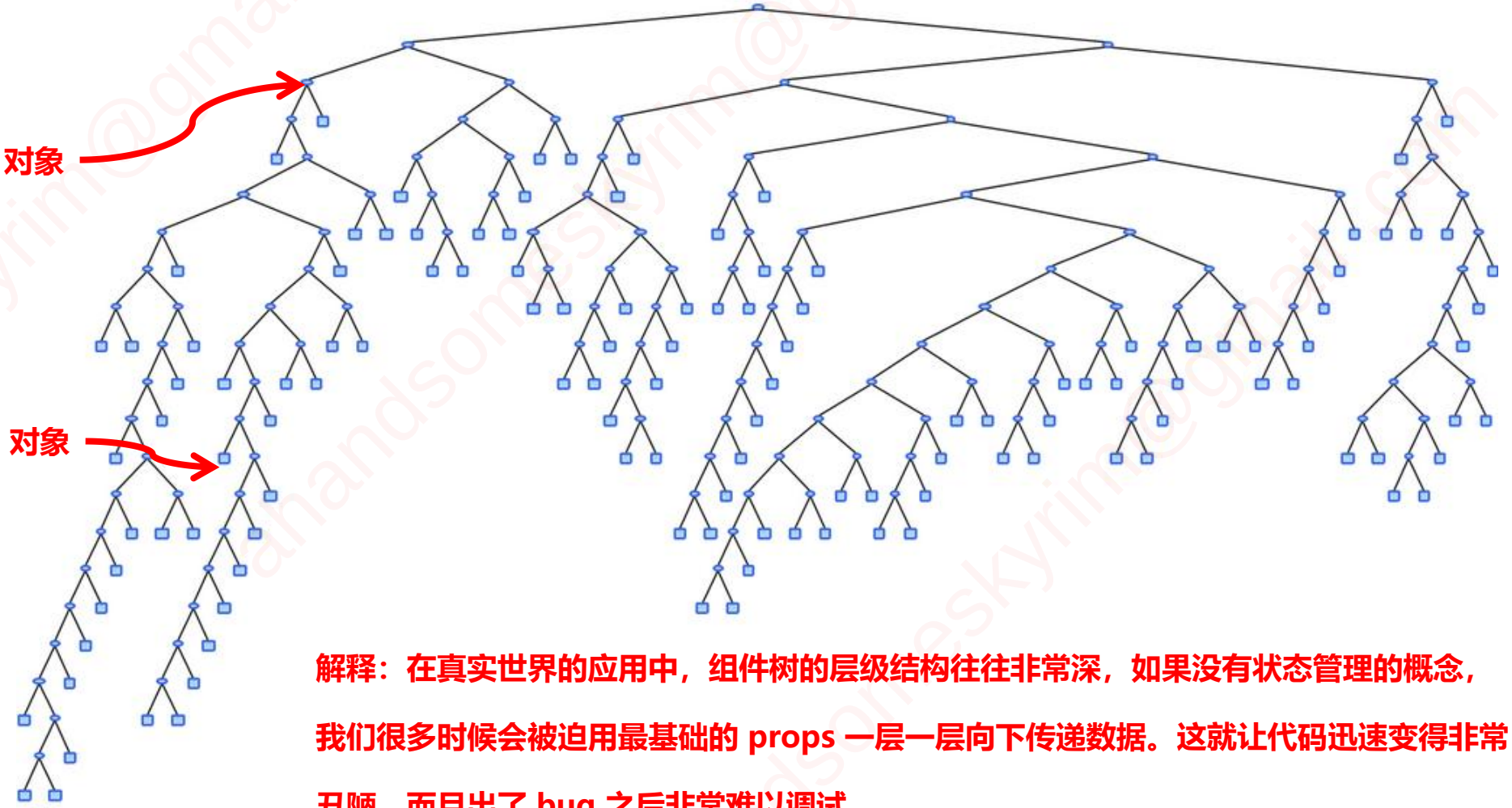
如果需要用编程的方式动态修改语言，调用 i18n.changeLanguage 函数

第9章 状态管理


LIQUID
MELTING
FREEZING
EVAPORATION
CONDENSATION
DEPOSITION
SUBLIMATION
SOLID
GAS

先解决一个问题：为什么要状态管理？

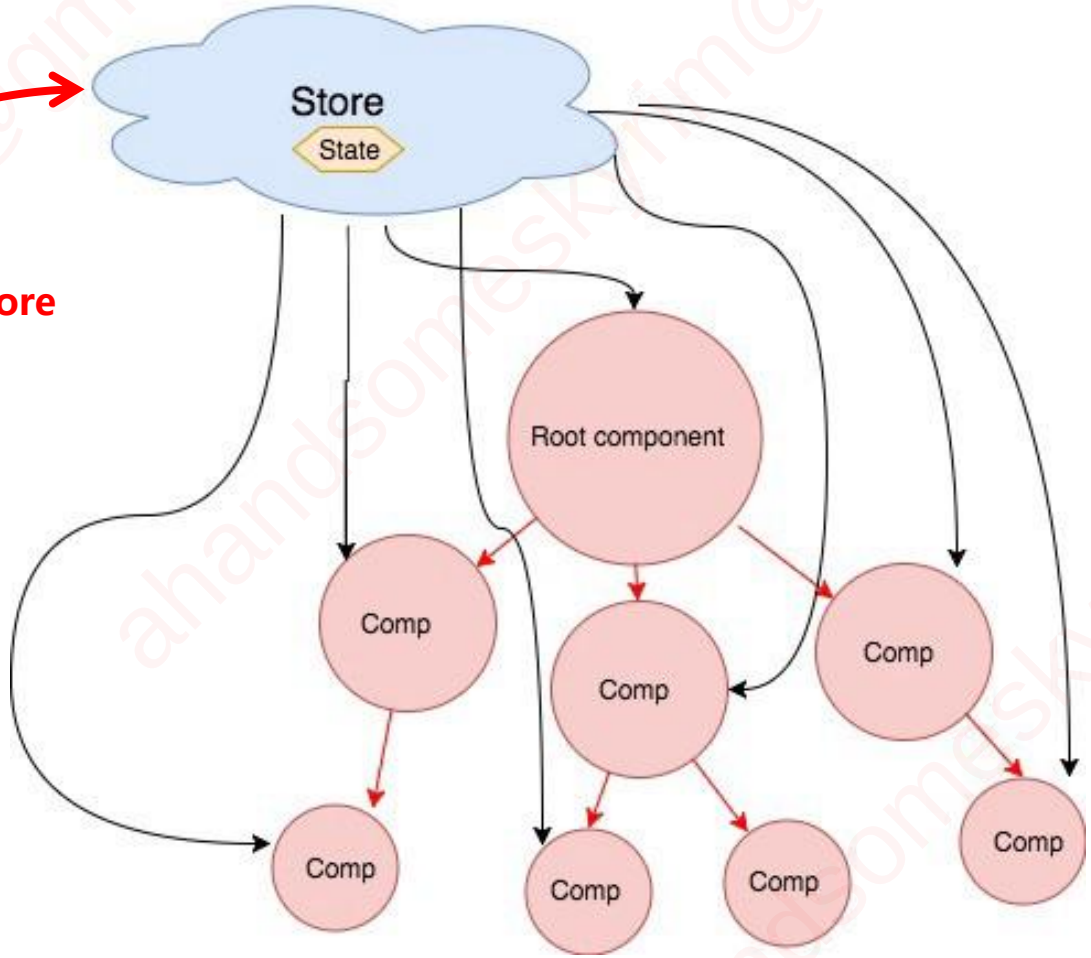
为什么要状态管理？没有行不行？



解释：在真实世界的应用中，组件树的层级结构往往非常深，如果没有状态管理的概念，我们很多时候会被迫用最基础的 props 一层一层向下传递数据。这就让代码迅速变得非常丑陋，而且出了 bug 之后非常难以调试。

状态管理的本质是什么?

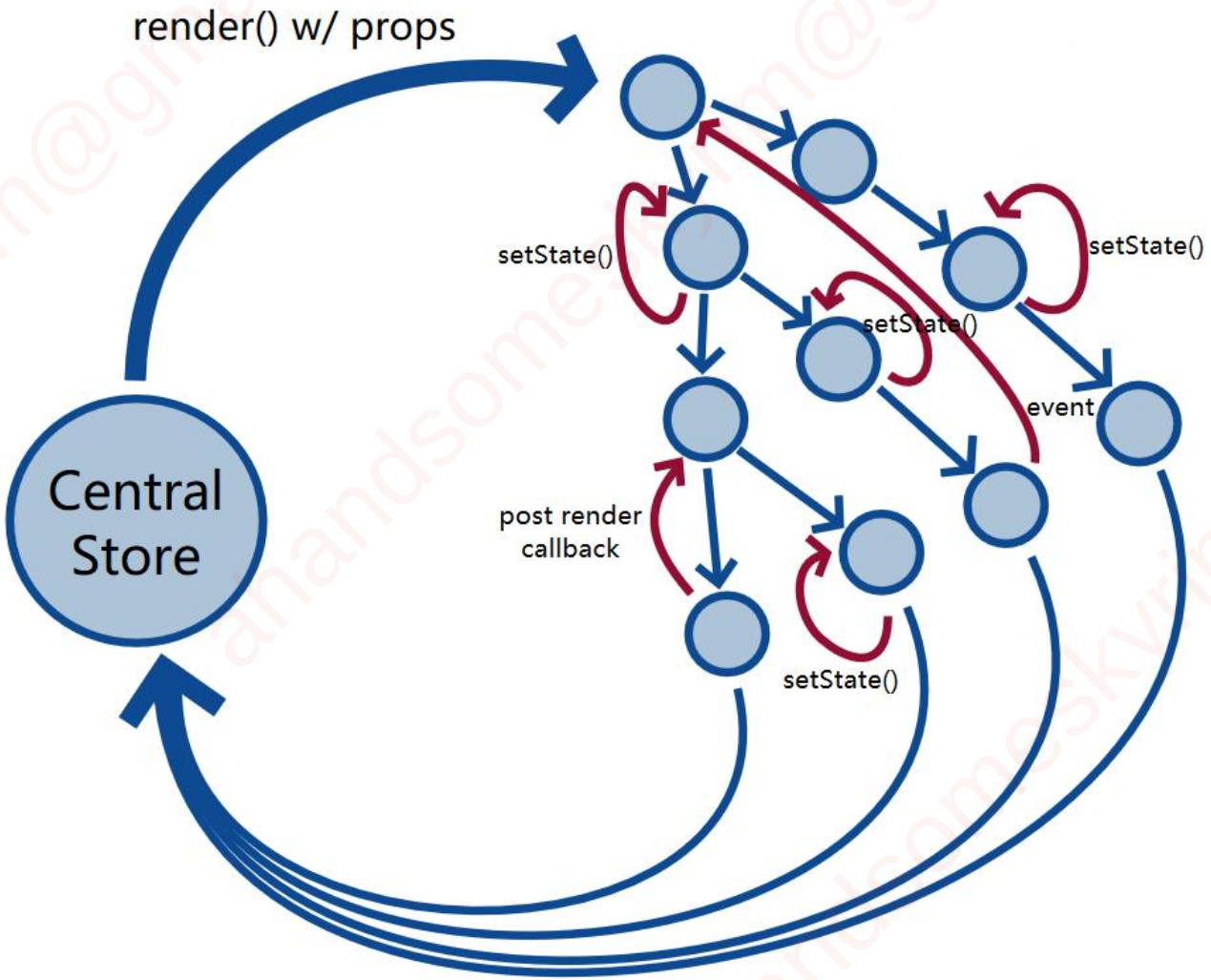
状态管理的本质



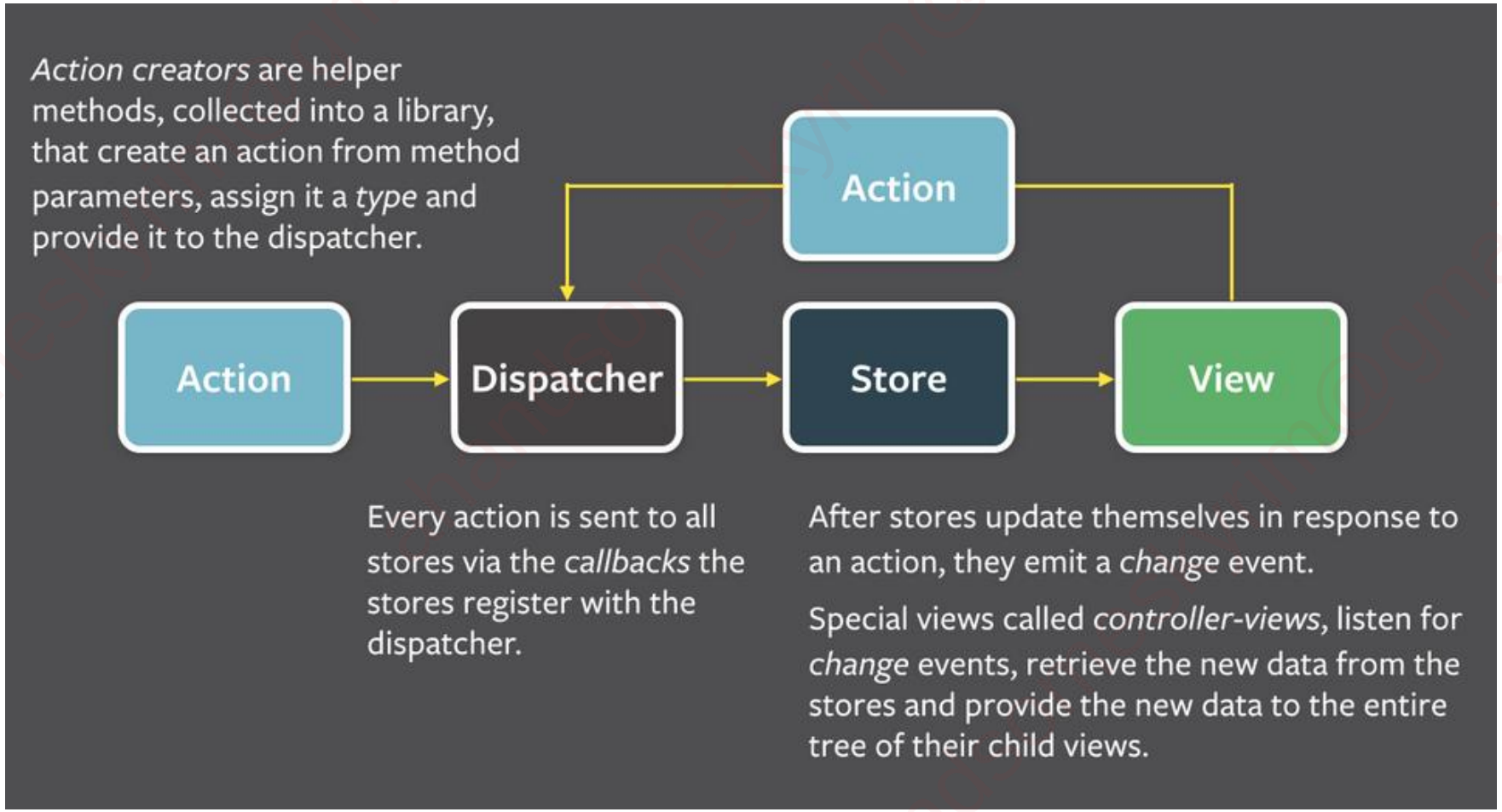
所有组件都能“共享”到这个 Store

注意：Store 也是有层级的，对于复杂的应用，不要把所有数据都“堆”在根 store 上面。

状态管理的本质



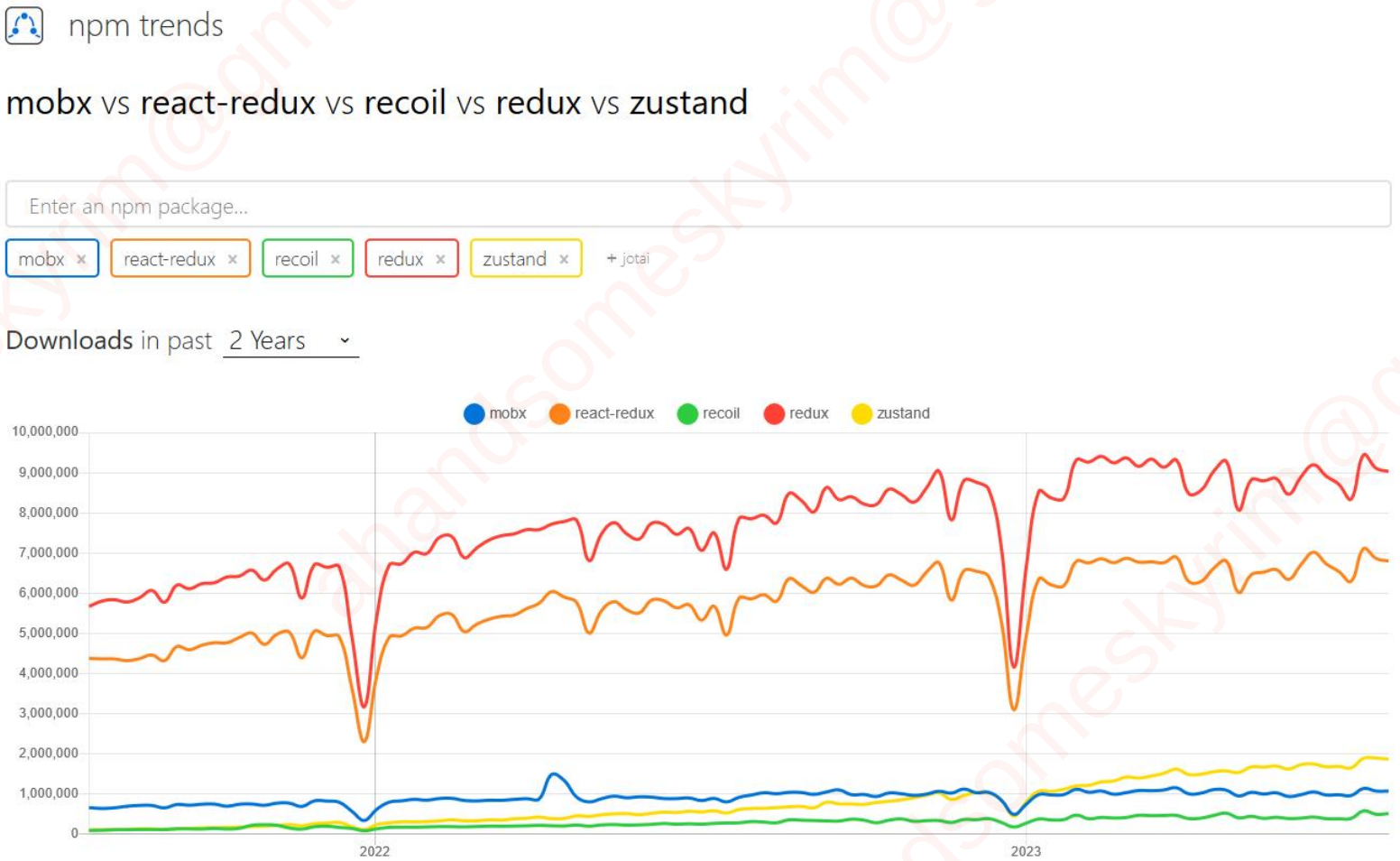
Facebook 官方提倡的 Flux 架构



https://facebook.github.io/flux/docs/in-depth-overview.html

市面上可用的状态管理组件有哪些?

React 状态管理库流行趋势

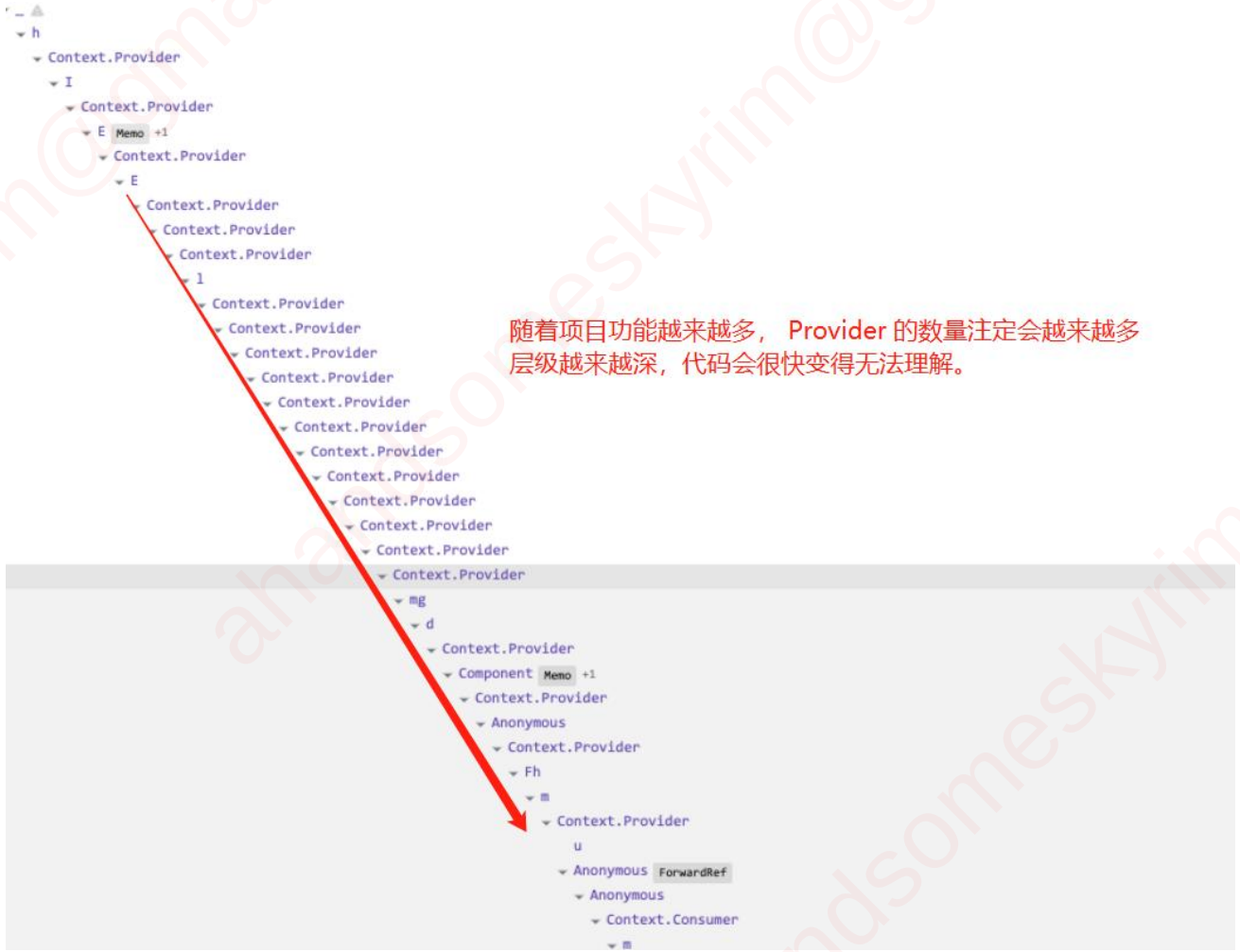


https://npmtrends.com/mobx-vs-react-redux-vs-recoil-vs-redux-vs-zustand

React 16.03 内置了 Context

从 16.03 开始，React 内置了Context可以用来做状态管理

Context 地狱

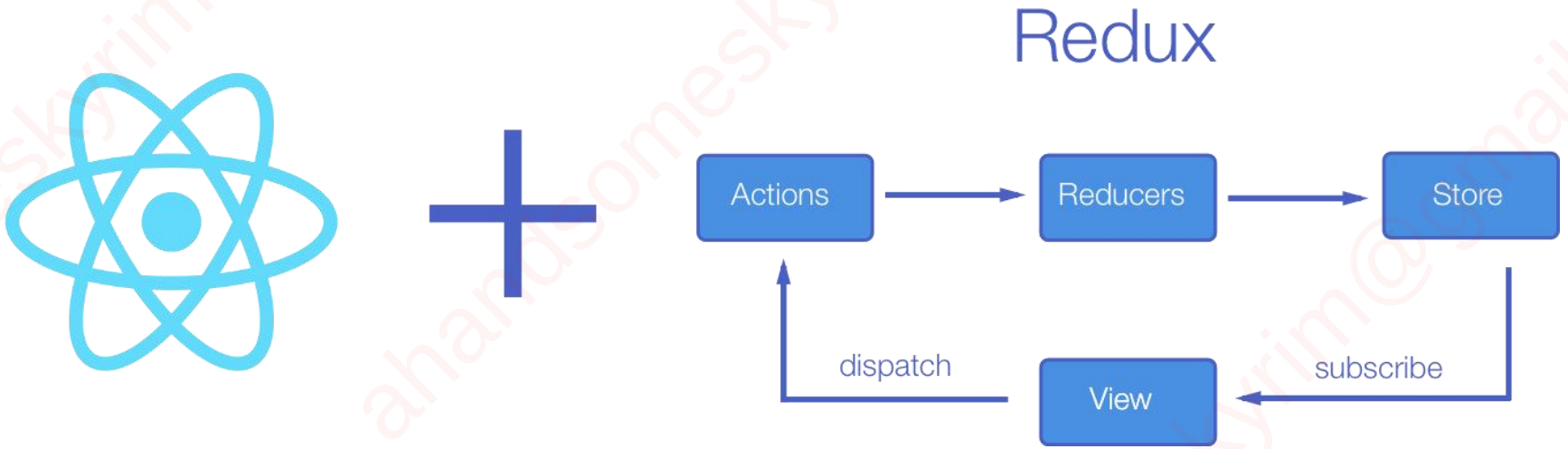


随着项目功能越来越多， Provider 的数量注定会越来越多
层级越来越深，代码会很快变得无法理解。

React 内置的 Context 理解起来非常简单，但是在一些复杂的场景下，它就有点吃力了。

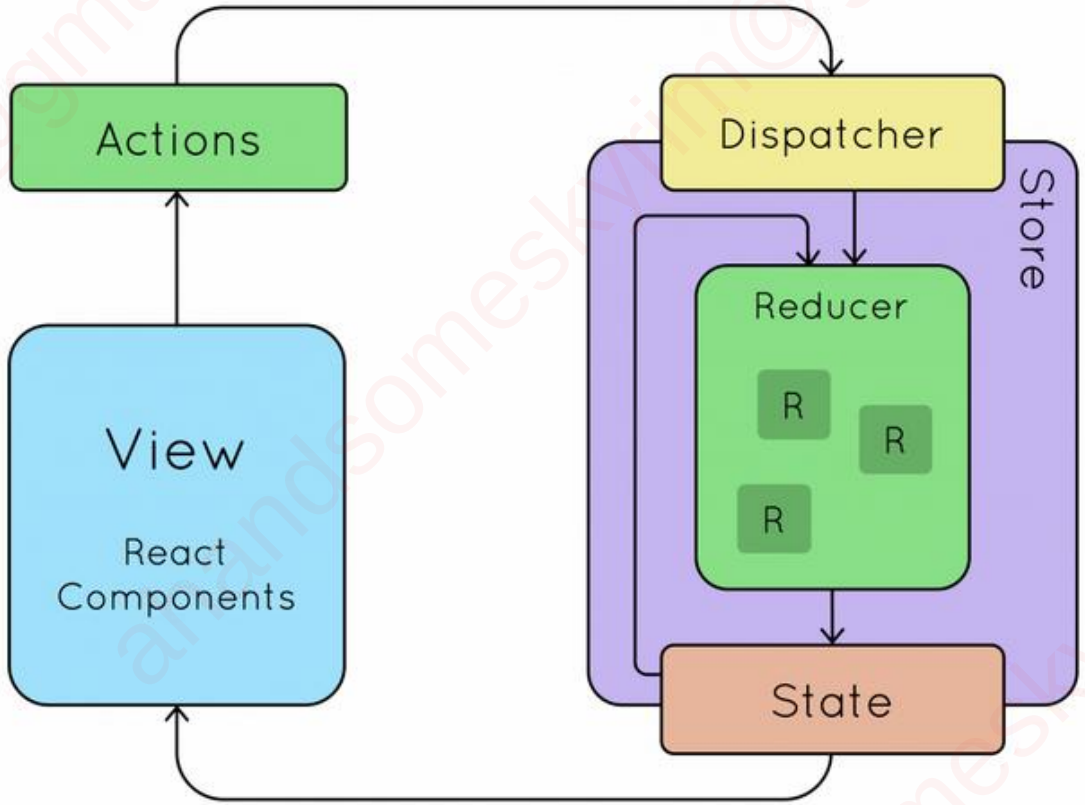
详解 React Redux

状态管理框架-Redux



如果你不确定是否要引入状态管理，那就是不要--- 某大牛

详解 React Redux



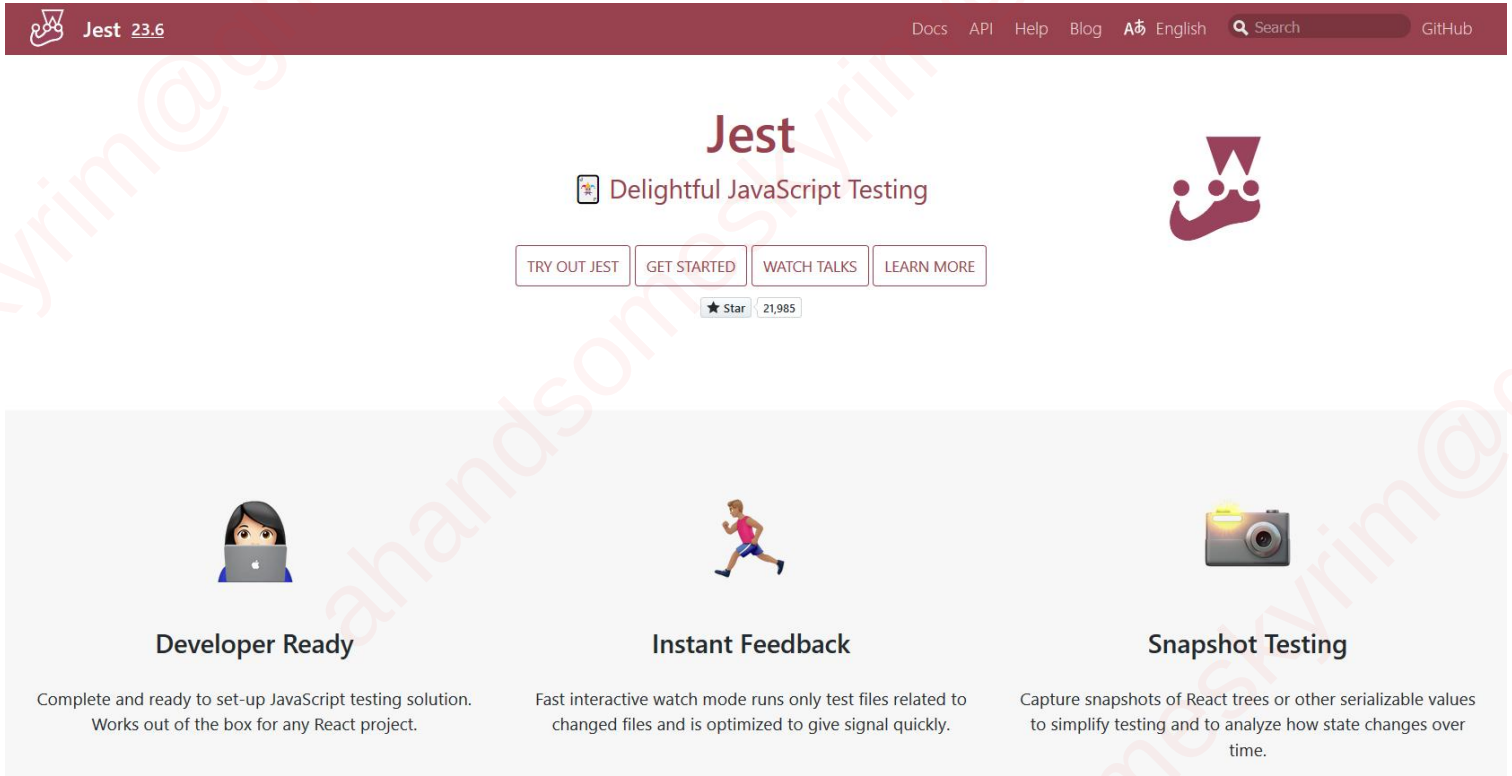
- React Redux **8.x** requires **React 16.8.3** or later / **React Native 0.59** or later, in order to make use of React Hooks.
- https://react-redux.js.org
- https://react-redux.js.org/tutorials/quick-start

匠人学院
JR ACADEMY

第10章 自动化测试

10.1 单元测试，安装配置 Jest

来自 Facebook 官方的测试工具 Jest



https://jestjs.io/en/

Jest 覆盖率

JS suite1.js

```
1  import add from '../../../src/common/utils/math'
2
3  test('adds 1 + 2 to equal 3', () => {
4      expect(add(1, 2)).toBe(3);
5  });
```

问题 输出 调试控制台 终端

```
E:\github-my\learn-react>npm run test

> learn-react@0.1.0 test E:\github-my\learn-react
> jest __jest__/__tests__

 PASS  __jest__/__tests__/utils/suite1.js
  √ adds 1 + 2 to equal 3 (5ms)

-------------------|----------|----------|----------|----------|-------------------|
File               |  % Stmts | % Branch |  % Funcs |  % Lines | Uncovered Line #s |
-------------------|----------|----------|----------|----------|-------------------|
All files          |     1.85 |        0 |     5.56 |     1.85 |                   |
 src               |        0 |        0 |        0 |        0 |                   |
  App.js           |        0 |        0 |        0 |        0 |... 24,27,28,55,59 |
  index.js         |        0 |      100 |      100 |        0 |      1,2,3,4,5,12 |
  serviceWorker.js |        0 |        0 |        0 |        0 |... 23,130,131,132 |
 src/common/utils  |      100 |      100 |      100 |      100 |                   |
  math.js          |      100 |      100 |      100 |      100 |                   |
-------------------|----------|----------|----------|----------|-------------------|
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        5.654s
Ran all test suites matching /__jest__\\__tests__/i.
```

10.2 单元测试，详解 Jasmine 语法

Jasmine 的发展历程

- **2008年**：Jasmine 的最早版本于 2008 年由 Pivotal Labs 的一群工程师创建。它的灵感来自于 RSpec（Ruby 的 BDD 测试框架），旨在为 JavaScript 提供类似的 BDD 风格的测试框架。 **（Pivotal 是著名的 Java 框架 Spring 的持有者，这家公司由 VMWare， EMC 等在 2013 年合资成立，大家在 Jasmine 里面会找到熟悉的 Junit 的味道。）**
- **2009年**：Jasmine 于 2009 年首次正式发布。它的版本 1.0 是一个比较简单的测试框架，但已经具备了基本的 BDD 测试功能，包括 describe 和 it 块、expect 断言和匹配器等。
- **2012年**：Jasmine 发布了版本 2.0，这个版本引入了异步测试支持，提供了 beforeAll、afterAll 和 pending 等钩子函数，增强了测试框架的功能和灵活性。
- **2015年**：Jasmine 2.4 发布，带来了一些改进和新特性，包括更好的 ES6/ES2015 支持和 jasmine.any 匹配器。
- **2018年**：Jasmine 3.0 发布，引入了一些重要的改进，包括删除已弃用的功能、增强异步测试支持和修复了一些 Bug。
- **2019年**：Jasmine 3.5 发布，继续增强异步测试支持，引入了 jasmine.anything 匹配器，并提供了更好的 TypeScript 支持。
- **2020年**：Jasmine 3.6 发布，增加了对 ArrayBuffer 和 DataView 类型的支持，改进了对 jasmine.objectContaining 的支持。
- **2021年**：Jasmine 3.7 发布，增加了 jasmine.arrayWithExactContents 匹配器，用于检查数组是否具有精确相同的内容。
- **当前**：Jasmine 成为了 JavaScript 社区中最受欢迎和广泛使用的测试框架之一。 **（几乎已成为事实标准，无论使用何种前端框架，都会看到它的身影。）**

Jasmine 基本用法概览

匠人学院 JR ACADEMY

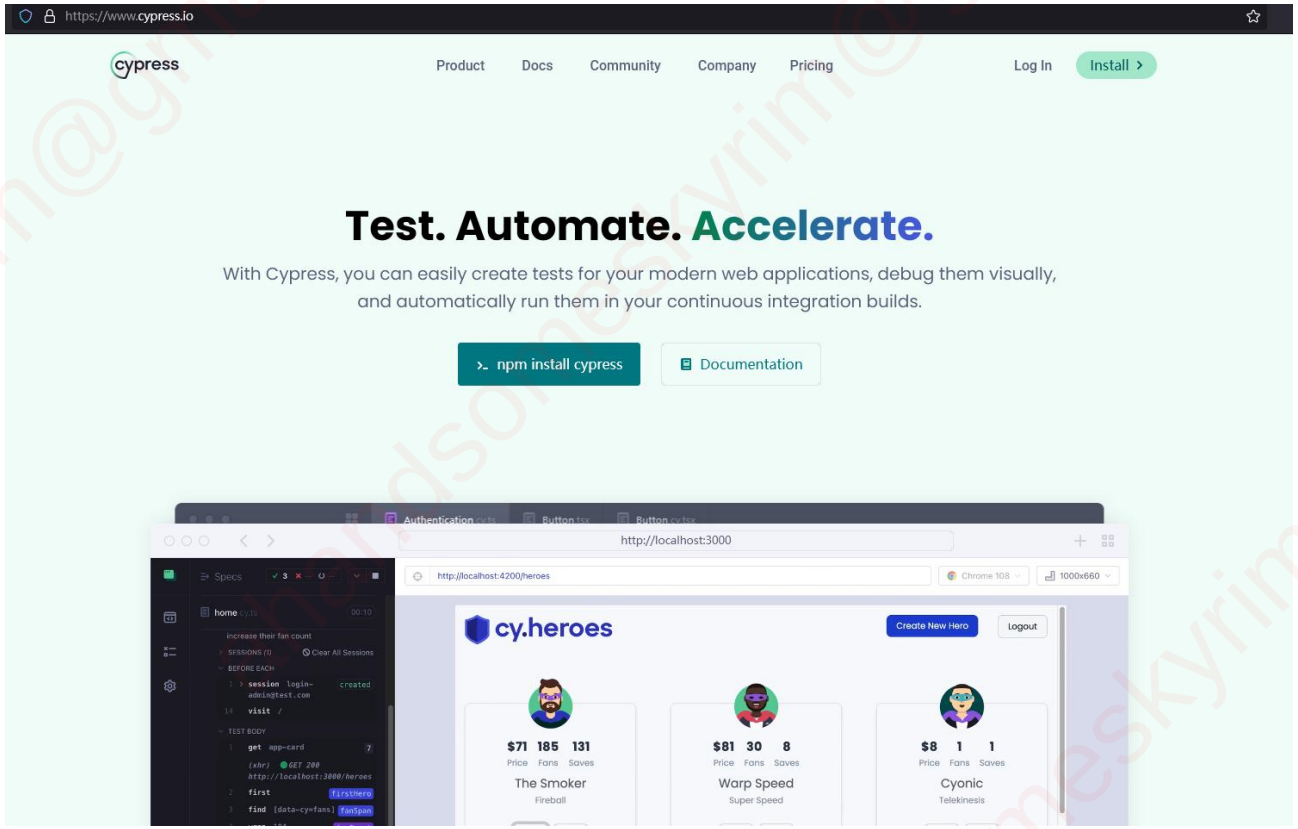
src > App.test.js > ...

```
1  import { render, screen } from '@testing-library/react';  247.2k (gzipped: 54.7k)
2  import App from './App';
3
4  test('renders learn react link', () => {
5    render(<App />);
6    const linkElement = screen.getByText(/learn react/i);
7    expect(linkElement).toBeInTheDocument();
8  });
9
```

断言式语法： expect ... toBe***

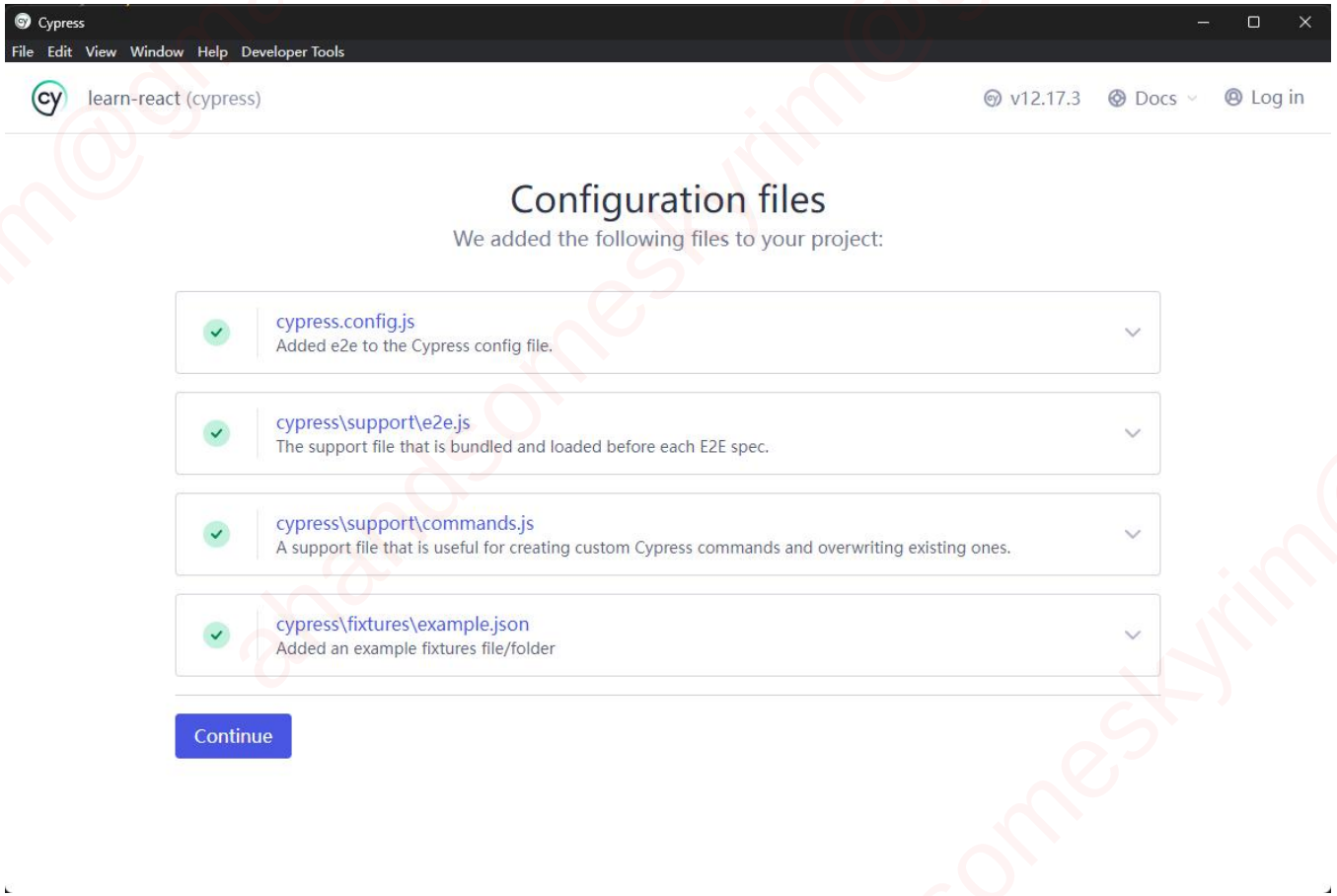
10.3 集成测试，安装配置 Cypress

Cypress 安装配置



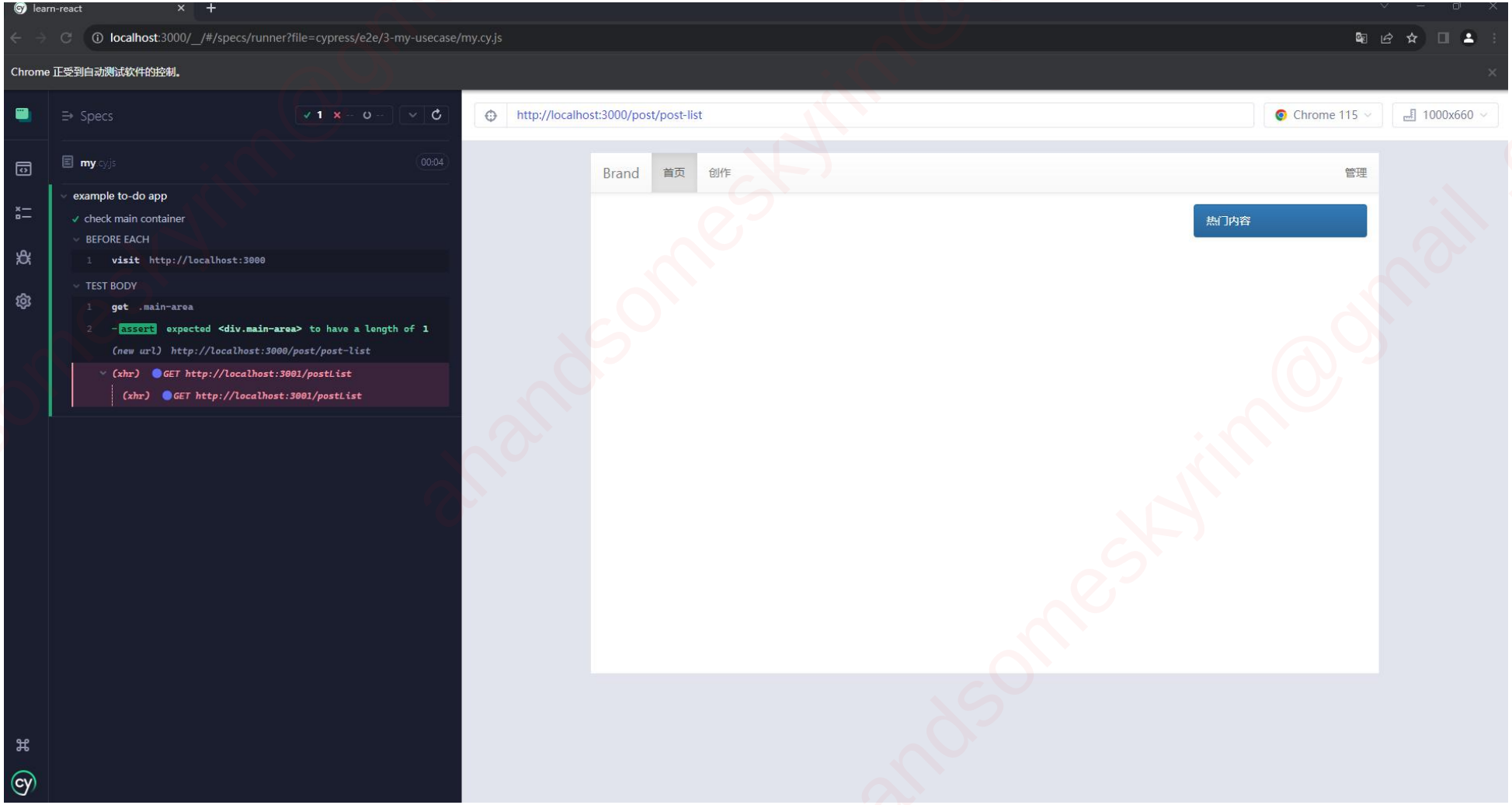
https://www.cypress.io/

Cypress 安装配置

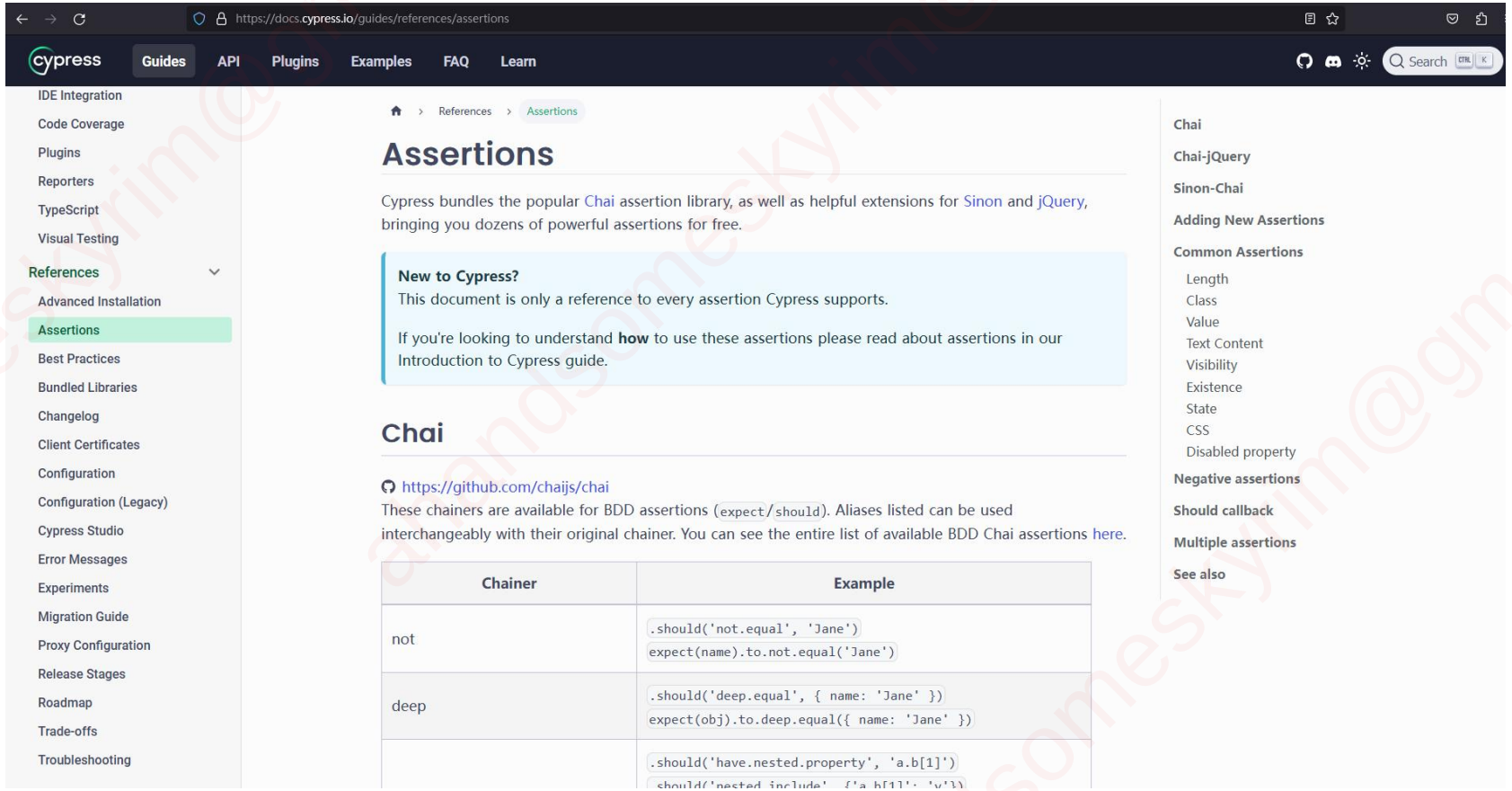


安装完 Cypress 之后，用 npx cypress open 启动，如果 Cypress 发现你是第一次在项目中运行，它会引导你进行配置

上手编写第一个测试用例



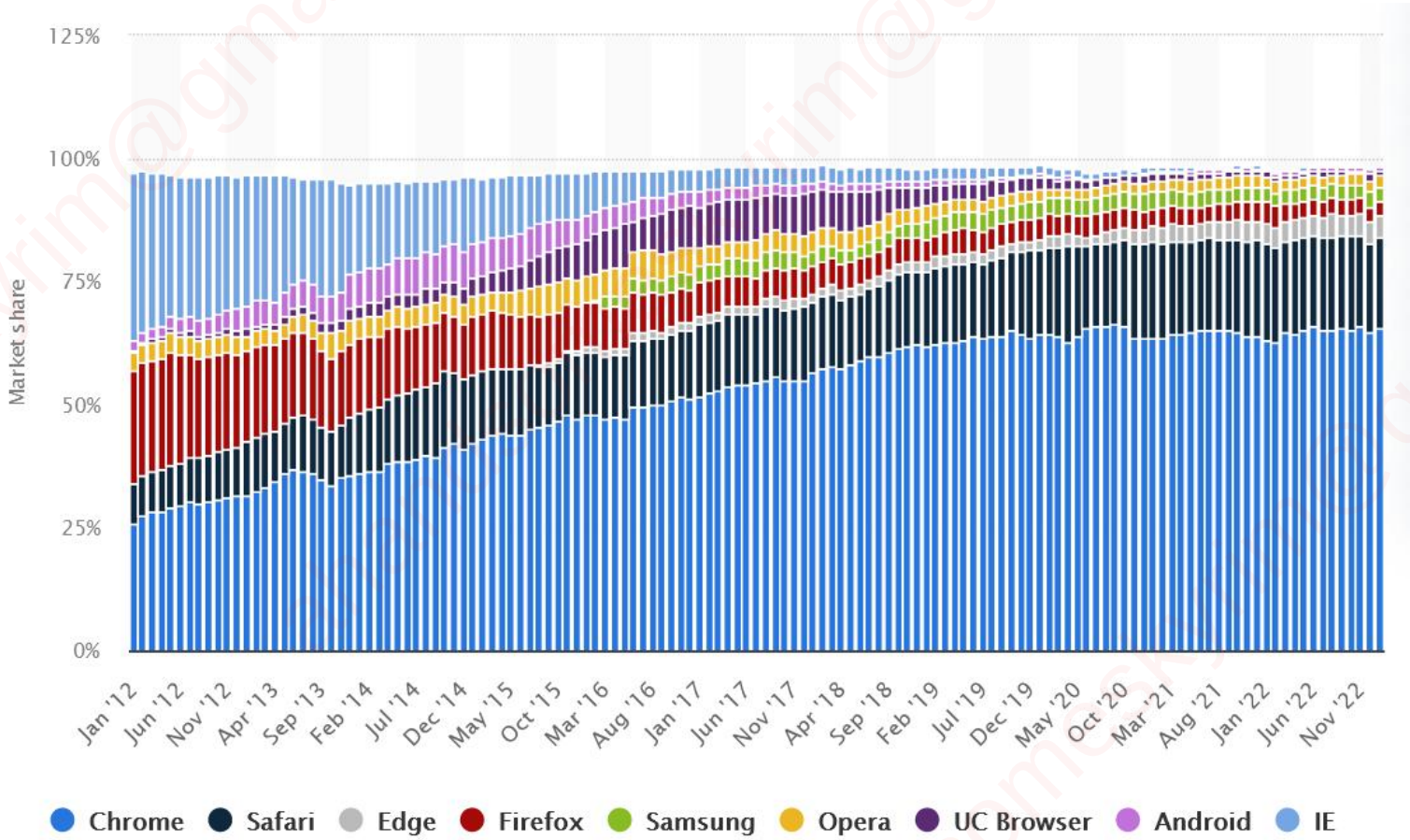
Cypress 断言语法（与 Jasmine 类似）



https://docs.cypress.io/guides/references/assertions

10.4 集成测试，运行集成测试用例

关于浏览器兼容性



在 2023 年的今天，webkit 内核的浏览器已经占有超过 90% 的市场份额

如果没有特殊的需要，强烈建议放弃支持老浏览器，兼容成本实在太高了

云端浏览器集群-大批量测试浏览器兼容性

https://github.com/alibaba/f2etest

匠人学院
JR ACADEMY

Thank You!

Q&A

本课程所有演示代码都在这个 repo 中：https://gitee.com/mumu-osc/learn-react