

link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=205 sentences=575, words=3434

1. 什么是同构

- 同构的项目支持客户端渲染和服务端渲染
- 客户端渲染缺点
 - 首屏速度加载慢
 - 不支持 SEO 和搜索引擎优化
 - 首页需要通过请求初始化数据



2.Next.js

- [Next.js 英文文档 \(https://nextjs.org/docs\)](https://nextjs.org/docs) [Next.js 中文文档 \(https://nextjs.frontendx.cn/docs/\)](https://nextjs.frontendx.cn/docs/) 是一个轻量级的 React 服务端渲染应用框架
- 默认支持服务端渲染
- 自动根据页面进行代码分割
- 基于页面的客户端路由方案
- 基于 Webpack 的开发环境, 支持热模块替换
- 可以跟 Koa 或其它 Node.js 服务器进行集成
- 支持 Babel 和 Webpack 的配置项定制
- 静态文件服务 public

3.项目初始化

3.1 创建项目

```
mkdir zhufengnextjs
cd zhufengnextjs
npm init -y
npm install react react-dom next redux react-redux --save
npm install axios express body-parser cors express-session connect-mongo mongoose koa koa-router --save
```

3.2 添加脚本

package.json

```
{
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start"
  }
}
```

[.gitignore \(https://github.com/zeit/next.js/blob/canary/.gitignore\)](https://github.com/zeit/next.js/blob/canary/.gitignore)

3.3 访问服务

- 以 `./pages` 作为服务端的渲染和索引的根目录
- `pages` 是 `next.js` 中非常重要的一个目录，其中每一个 `js` 文件就代表一个页面，但是有两个例外，`_app.js` 和 `_document.js`
- `next.js` 会将 `pages` 下的 `js` 文件根据其路径名和文件名自动生成对应的路由
- `pages` 组件代码自动分割
- `next.js` 项目运行之后会自动生成 `.next` 目录

3.3.1 index.js

```
function Home() {
  return <div>Homediv</div>;
}
export default Home;
```

3.3.2 pages/user.js

pages/user.js

```
function User() {
  return <div>Userdiv</div>;
}
export default User;
```

3.3.3 访问

```
npm run dev
curl http:
```

4. 跑通路由和样式

4.1 知识点

- 绑定 `styled-jsx` 来生成独立作用域的 `CSS`
- 如何支持本地和全局 `css`
- 路由的使用和两种跳转路径的方法

4.2 pages_app.js

- `App` 组件是每个页面的根组件，页面切换时 `App` 不会销毁，但是里面的页面组件会销毁，因此可以利用这个来设置全局属性和样式
- 全局 `css` 只能写在这里，否则会报错
 - 当页面变化时保持页面布局
 - 当路由变化时保持页面状态
 - 注入额外数据到页面里

pages_app.js

```
import App from "next/app";
import Link from "next/link";
import _appStyle from "../_app.module.css";
import "../styles/global.css";
class LayoutApp extends App {
  render() {
    let { Component } = this.props;
    return (
      <div>
        <style jsx>
          {`
            li {
              display: inline-block;
              margin-left: 10px;
              line-height: 31px;
            }
          `}
        <style>
        <header>
          
          <ul>
            <li>
              <Link href="/">首页Link</Link>
            </li>
            <li>
              <Link href="/user">用户管理Link</Link>
            </li>
            <li>
              <Link href="/profile">个人中心Link</Link>
            </li>
          </ul>
        <header>
        <Component />
        <footer style={{ textAlign: "center" }}>©copyright 珠峰架构footer</footer>
      </div>
    );
  }
}
export default LayoutApp;
```

4.3 _app.module.css

pages_app.module.css

```
.logo {
  width: 120px;
  height: 31px;
  float: left;
}
```

4.4 global.css

styles/global.css

```
html,
body {
  padding: 0;
  margin: 0;
}
```

4.5 pages/index.js

pages/index.js

```
function Home() {
  return <div>Homediv</div>;
}
export default Home;
```

4.6 pages/user.js

pages/user.js

```
import Link from "next/link";
function User() {
  return (
    <div>
      <p>Userp</p>
      <Link href="/">首页Link</Link>
    </div>
  );
}
export default User;
```

4.7 pages/profile.js

pages/profile.js

```
import router from "next/router";
function Profile() {
  return (
    <div>
      <p>Profilep</p>
      <button onClick={() => router.back()}>返回button</button>
    </div>
  );
}
export default Profile;
```

5. 二级路由

5.1 知识点

- 支持二级路由
- 实现二级布局组件

- 路由跳转传递参数
- 页面组件通过 `getInitialProps` 获取数据

5.2 执行顺序

5.2.1 后台顺序 <#>

- LayoutApp getInitialProps
- UseList getInitialProps
- LayoutApp constructor
- UseList constructor

5.2.2 前台顺序

- 初次渲染
 - LayoutApp constructor
 - UseList constructor
- 路由切换
 - LayoutApp getInitialProps
 - UseList getInitialProps
 - UseList constructor

5.2.3 _app.js

pages_app.js

```
import App from 'next/app';
import Link from 'next/link';
import _appStyle from './_app.module.css';
import '../styles/global.css';
class LayoutApp extends App {
+ static async getInitialProps({ Component, ctx }) {
+   let pageProps = {};
+   if (Component.getInitialProps)
+     pageProps = await Component.getInitialProps(ctx);
+   return { pageProps };
+ }
render() {
+   let { Component, pageProps } = this.props;
+   return (
      {
        `li{
          display:inline-block;
          margin-left:10px;
          line-height:31px;
        }`
      )
      首页
      用户管理
      个人中心
    )
  }
}
export default LayoutApp;
```

5.2.4 user\index.js

pages\user\index.js

```
import Link from "next/link";
function UserLayout(props) {
  return (
    <div>
      <div>
        <ul>
          <li>
            <Link href="/user/list">
              <a>用户列表</a>
            </Link>
          </li>
          <li>
            <Link href="/user/add">
              <a>添加用户</a>
            </Link>
          </li>
        </ul>
        <div>{props.children}</div>
      </div>
    </div>
  );
}
export default UserLayout;
```

5.2.5 user\list.js

pages\user\list.js

```
import Link from "next/link";
import UserLayout from "../";
function UseList(props) {
  return (
    <UserLayout>
      <ul>
        {props.list.map((user) => (
          <li key={user.id}>
            <Link href={` /user/detail/${user.id}`}>{user.name}</Link>
          </li>
        ))}
      </ul>
    </UserLayout>
  );
}
UseList.getInitialProps = async () => {
  let list = [
    { id: 1, name: "张三" },
    { id: 2, name: "李四" },
  ];
  return { list };
};
export default UseList;
```

5.2.6 userAdd.js

pages/userAdd.js

```
import UserLayout from "../";
import React from "react";
function UserAdd() {
  let nameRef = React.useRef();
  let passwordRef = React.useRef();
  let handleSubmit = (event) => {
    event.preventDefault();
    let user = {
      name: nameRef.current.value,
      password: passwordRef.current.value,
    };
  };
  return (
    <UserLayout>
      <form onSubmit={handleSubmit}>
        用户名:
        <input ref={nameRef} />
        密码:
        <input ref={passwordRef} />
        <button type="submit">添加</button>
      </form>
    </UserLayout>
  );
}
export default UserAdd;
```

5.2.7 userDetail[id].js

pages/userDetail[id].js

```
import React from "react";
import UserLayout from "../";
function UserDetail(props) {
  return (
    <UserLayout>
      <p>ID: {props.user.id}</p>
    </UserLayout>
  );
}
UserDetail.getInitialProps = async (ctx) => {
  return { user: { id: ctx.query.id } };
};
export default UserDetail;
```

6.调用接口

- 当服务渲染时, getInitialProps 将会把数据序列化, 就像 JSON.stringify
- 所以确保 getInitialProps 返回的是一个普通 JS 对象, 而不是 Date, Map 或 Set 类型
- 当页面初始化加载时, getInitialProps 只会加载在服务端。只有当路由跳转 (Link 组件跳转或 API 方法跳转) 时, 客户端才会执行 getInitialProps
- getInitialProps 将不能使用在子组件中。只能使用在 pages 页面中

6.1 方法执行顺序

6.1.1 首次访问

服务器端

```
LayoutApp getInitialProps 获取LayoutApp初始属性
UseList getInitialProps 调用页面组件的getInitialProps方法
LayoutApp constructor 根据属性创建LayoutApp的实例
LayoutApp render 调用此实例的render方法, 返回react元素
UseList constructor 创建子组件
UseList render 渲染子组件
```

- 然后会把HTML结构和LayoutApp属性对象序列化后发给客户端
- 客户端再把LayoutApp属性反序列化后变成对象

客户端

```
LayoutApp constructor
LayoutApp render
UseList constructor
UseList render
```

6.1.2 切换路由

客户端

```
LayoutApp getInitialProps
UseList getInitialProps
LayoutApp render
UseList constructor
UseList render
```

6.2 pages_app.js

pages_app.js

```
import App from 'next/app';
import Link from 'next/link';
import _appStyle from './_app.module.css';
import './styles/global.css';
class LayoutApp extends App {
+  constructor(props) {
+    super(props)
+    console.log('LayoutApp constructor');
+  }
  static async getInitialProps({ Component, ctx }) {
+  console.log('LayoutApp getInitialProps');
    let pageProps = {};
    if (Component.getInitialProps)
      pageProps = await Component.getInitialProps(ctx);
    return { pageProps };
  }
  render() {
    console.log('LayoutApp render');
    let { Component, pageProps } = this.props;
    return (
      <div>
        <div>
          {
            `li{
              display:inline-block;
              margin-left:10px;
              line-height:31px;
            }`
          }
          首页
          用户管理
          个人中心
        </div>
        @copyright 珠峰架构
      </div>
    )
  }
}
export default LayoutApp;
```

6.3 pages/userList.js

pages/userList.js

```
import React from 'react';
import Link from 'next/link';
import UserLayout from './';
+import request from '../../utils/request';
class UseList extends React.Component {
+  constructor(props) {
+    super(props);
+    console.log('UseList constructor');
+  }
  render() {
    console.log('UseList render');
    return (
      <div>
        {
          this.props.list.map((user) => (
            <div>
              {user.name}
            </div>
          ))
        }
      </div>
    )
  }
}

UseList.getInitialProps = async () => {
+  console.log('UseList getInitialProps');
  let response = await request({ url: '/api/users', method: 'GET' }).then(res => res.data);
  return { list: response.data };
}
export default UseList;
```

6.4 pages/userAdd.js

pages/userAdd.js

```

import UserLayout from '../';
import React from 'react';
+import request from '../utils/request';
import router from 'next/router'
function UserAdd() {
  let nameRef = React.useRef();
  let passwordRef = React.useRef();
  let handleSubmit = async (event) => {
    event.preventDefault();
    const user = { name: nameRef.current.value, password: passwordRef.current.value };
+   let response = await request.post('/api/register', user).then(res => res.data);
+   if (response.success) {
+     router.push('/user/list');
+   } else {
+     alert('添加用户失败');
+   }
  }
  return (
    <div>
      用户名:
      密码:
      添加
    </div>
  )
}
export default UserAdd;

```

6.5 [id].js

pages\user\detail[id].js

```

import React from 'react';
import UserLayout from '../';
+import request from '../utils/request';
function UserDetail(props) {
  return (
    <div>
      ID:{props.user.id}
      NAME:{props.user.name}
    </div>
  )
}
UserDetail.getInitialProps = async (ctx) => {
+ let response = await request({ url: `/api/users/${ctx.query.id}`, method: 'GET' }).then(res => res.data);
+ return { user: response.data };
}
export default UserDetail;

```

6.6 utils\request.js

utils\request.js

```

import axios from "axios";
axios.defaults.withCredentials = true;
const instance = axios.create({
  baseURL: "http://localhost:5000",
});
export default instance;

```

7. 懒加载

7.1 [id].js

pages\user\detail[id].js

```

import React from 'react';
import UserLayout from '../';
import request from '../utils/request';
+import dynamic from 'next/dynamic';
//import UserInfo from '@components/UserInfo';
+const DynamicUserInfo = dynamic(() => import('@components/UserInfo'));
function UserDetail(props) {
  const [show, setShow] = React.useState(false);
  return (
    <div>
      ID:{props.user && props.user.id}
+     setShow(!show)>显示/隐藏
+     {
+       show && props.user &&
+     }
    </div>
  )
}
UserDetail.getInitialProps = async (ctx) => {
  let response = await request({ url: `/api/users/${ctx.query.id}`, method: 'GET' }).then(res => res.data);
  return { user: response.data };
}
export default UserDetail;

```

7.2 components\UserInfo.js

UserInfo.js

```
import React, { useState } from "react";
function UserInfo(props) {
  const [createdAt, setCreatedAt] = useState(props.user.createdAt);
  async function changeFormat() {
    const moment = await import("moment");
    setCreatedAt(moment.default(createdAt).fromNow());
  }
  return (
    <>
      <p>NAME:{props.user.name}</p>
      <p>创建时间:{createdAt}</p>
      <button onClick={changeFormat}>切换为相对时间</button>
    </>
  );
}
export default UserInfo;
```

7.3 jsconfig.json

jsconfig.json

```
{
  "compilerOptions": {
    "baseUrl": ".",
    "paths": {
      "@/*": ["/*"]
    }
  }
}
```

8.集成 redux

8.1 知识点

- 集成redux
- cookie保存和存递

8.2 pages_app.js

pages_app.js


```
import { createStore } from 'redux';
import reducer from './reducer';

export default function (initialState) {
  return createStore(reducer, initialState);
}
```

8.6 login.js

pages/login.js

```
import React from 'react';
import request from '@utils/request';
import router from 'next/router';
import { connect } from 'react-redux';
import * as types from '@store/action-types';

function Login(props) {
  let nameRef = React.useRef();
  let passwordRef = React.useRef();
  let handleSubmit = async (event) => {
    event.preventDefault();
    let user = { name: nameRef.current.value, password: passwordRef.current.value };
    let response = await request.post('/api/login', user).then(res => res.data);
    if (response.success) {
      props.dispatch({ type: types.SET_USER_INFO, payload: response.data });
      router.push('/');
    } else {
      alert('登录失败');
    }
  }

  return (
    <form onSubmit={handleSubmit}>
      用户名:<input ref={nameRef} />
      密码:<input ref={passwordRef} />
      <button type="submit">登录button</button>
    </form>
  )
}

export default connect(state => state)(Login);
```

9.loading

- [路由事件 \(https://nextjs.frontendx.cn/docs/#%E8%B7%AF%E7%94%B1\)](https://nextjs.frontendx.cn/docs/#%E8%B7%AF%E7%94%B1)

事件 触发时机 routeChangeStart(url) 路由开始切换时触发 routeChangeComplete(url) 完成路由切换时触发 routeChangeError(err, url) 路由切换报错时触发 beforeHistoryChange(url) 浏览器 history 模式开始切换时触发 hashChangeStart(url) 开始切换 hash 值但是没有切换页面路由时触发 hashChangeComplete(url) 完成切换 hash 值但是没有切换页面路由时触发

9.1 _app.js

pages/_app.js

```

import App from 'next/app';
import Link from 'next/link';
import _appStyle from './_app.module.css';
import '../styles/global.css';
import { Provider } from 'react-redux';
import request from '../utils/request';
import createStore from '../store';
import * as types from '../store/action-types';
+import router from 'next/router';
function getStore(initialState) {
  if (typeof window
    return createStore(initialState); //如果是服务器端,每次都返回新的仓库
  } else {
    if (!window._REDUX_STORE_) {
      window._REDUX_STORE_ = createStore(initialState);
    }
    return window._REDUX_STORE_;
  }
}
}
class LayoutApp extends App {
  constructor(props) {
    super(props)
    + this.state = { loading: false }
    this.store = getStore(props.initialState);
    console.log('LayoutApp constructor');
  }
  static async getInitialProps({ Component, ctx }) {
    console.log('LayoutApp getInitialProps');
    let store = getStore(); //1.后台创建新仓库 5.每次切换路由都会执行此方法获取老仓库
    if (typeof window == 'undefined') { //2.后台获取用户信息
      let options = { url: '/api/validate' };
      if (ctx.req && ctx.req.headers.cookie) {
        options.headers = options.headers || {};
        options.headers.cookie = ctx.req.headers.cookie;
      }
      let response = await request(options).then(res => res.data);
      if (response.success) {
        store.dispatch({ type: types.SET_USER_INFO, payload: response.data });
      }
    }
    let pageProps = {};
    if (Component.getInitialProps)
      pageProps = await Component.getInitialProps(ctx);
    const props = { pageProps };
    if (typeof window == 'undefined') { //后台获取用赋值状态
      props.initialState = store.getState();
    }
    return props;
  }
  + componentDidMount() {
    + this.routeChangeStart = (url) => {
      + this.setState({ loading: true });
    };
    + router.events.on('routeChangeStart', this.routeChangeStart);
    + this.routeChangeComplete = (url) => {
      + this.setState({ loading: false });
    };
    + router.events.on('routeChangeComplete', this.routeChangeComplete);
  }
  + componentWillUnmount() {
    + router.events.off('routeChangeStart', this.routeChangeStart)
    + router.events.off('routeChangeComplete', this.routeChangeComplete)
  }
  render() {
    console.log('LayoutApp render');
    let state = this.store.getState();
    let { Component, pageProps } = this.props;
    return (
      {
        `li{
          display:inline-block;
          margin-left:10px;
          line-height:31px;
        }`
      }
      首页
      用户管理
      个人中心
      {
        state.currentUser ? {state.currentUser.name} : 登录
      }
      {
        + this.state.loading ? 切换中..... :
      }
      @copyright 珠峰架构
    )
  }
}
export default LayoutApp;

```

10.受保护路由

10.1 profile.js

pages/profile.js

```

import router from 'next/router';
import { connect } from 'react-redux';
import request from '../utils/request';
function Profile(props) {
  let { currentUser } = props;
  return (
    <div>
      <p>当前登录用户:{currentUser.name}</p>
      <button onClick={() => router.back()}>返回button</button>
    </div>
  )
}
Profile.getInitialProps = async function (ctx) {
  let options = { url: '/api/validate' };
  if (ctx.req && ctx.req.headers.cookie) {
    options.headers = options.headers || {};
    options.headers.cookie = ctx.req.headers.cookie;
  }
  let response = await request(options).then(res=>res.data);
  if (response.success) {
    return {currentUser:response.data};
  } else {
    if (ctx.req) {
      ctx.res.writeHead(303, { Location: '/login' })
      ctx.res.end()
    } else {
      router.push('/login');
    }
    return {};
  }
}
const WrappedProfile = connect(
  state => state
)(Profile);
export default WrappedProfile;

```

11.自定义Document

11.1 pages_document.js

pages_document.js

```

import Document, { Html, Head, Main, NextScript } from 'next/document';
class CustomDocument extends Document {
  static async getInitialProps(ctx) {
    const props = await Document.getInitialProps(ctx);
    return { ...props };
  }
  render() {
    return (
      <Html>
        <Head>
          <style>
            {
              `
              *{
                padding:0;
                margin:0;
              }
            `
          </style>
          <Head>
            <body>
              <Main />
              <NextScript />
            </body>
          </Html>
        </Head>
      </Html>
    )
  }
}
export default CustomDocument;

```

11.2 pagesindex.js

pagesindex.js

```

import Head from 'next/head'
export default function (props) {
  return (
    <div>
      <Head>
        <title>首页title</title>
        <meta name="description" content="这是首页" />
      </Head>
      <p>Homep</p>
    </div>
  )
}

```

12. getServerSideProps

- [data-fetching \(https://nextjs.org/docs/basic-features/data-fetching\)](https://nextjs.org/docs/basic-features/data-fetching)
 - getServerSideProps (Server-side Rendering): Fetch data on each request

12.1 list.js

pagesuser/list.js

```

import React from 'react';
import Link from 'next/link';
import UserLayout from './';
import request from '@/utils/request';
class UseList extends React.Component {
  constructor(props) {
    super(props);
    console.log('UseList constructor');
  }
  render() {
    console.log('UseList render');
    return (
      <div>
        {
          this.props.list.map((user) => (
            <div>
              {user.name}
            </div>
          ))
        }
      </div>
    )
  }
}

-UseList.getInitialProps = async () => {
-  console.log('UseList getInitialProps');
-  let response = await request({ url: '/api/users', method: 'GET' }).then(res => res.data);
-  return { list: response.data };
-}

//每个请求都会调用
+export async function getServerSideProps() {
+  const res = await request.get('http://localhost:5000/api/users')
+  return {
+    props: {
+      list: res.data.data
+    },
+  }
+}
export default UseList;

```

13. getStaticProps

- [data-fetching \(https://nextjs.org/docs/basic-features/data-fetching\)](https://nextjs.org/docs/basic-features/data-fetching)
 - getStaticProps (Static Generation): Fetch data at build time
 - getStaticPaths (Static Generation): Specify dynamic routes to pre-render based on data

13.1 pages/user/list.js

pages/user/list.js

```

import React from 'react';
import Link from 'next/link';
import UserLayout from './';
import request from '@/utils/request';

class UseList extends React.Component {
  constructor(props) {
    super(props);
    console.log('UseList constructor');
  }
  render() {
    console.log('UseList render');
    return (
      <UserLayout>
        <ul>
          {
            this.props.list.map((user) => (
              <li key={user.id}>
                <Link href={`/${user/detail}/${user.id}`}>{user.name}</Link>
              </li>
            ))
          }
        </ul>
      </UserLayout>
    )
  }
}

export async function getServerSideProps() {
  const res = await request.get('http://localhost:5000/api/users')
  return {
    props: {
      list: res.data.data
    },
  }
}

export async function getStaticProps() {
  const res = await request.get('http://localhost:5000/api/users');
  return {
    props: {
      list: res.data.data
    },
  }
}
export default UseList;

```

13.2 pages/user/detail[id].js

pages/user/detail[id].js

```
import React from 'react';
import UserLayout from '../';
import request from '@/utils/request';
import dynamic from 'next/dynamic';

const DynamicUserInfo = dynamic(() => import('@/components/UserInfo'));
function UserDetail(props) {
  const [show, setShow] = React.useState(false);
  return (
    <UserLayout>
      <p>ID:{props.user && props.user.id}</p>
      <button onClick={() => setShow(!show)}>显示/隐藏</button>
      {
        show && props.user && <DynamicUserInfo user={props.user} />
      }
    </UserLayout>
  )
}
UserDetail.getInitialProps = async (ctx) => {
  let response = await request.get(`/api/users/${ctx.query.id}`)
  return { user: response.data.data };
}
export async function getStaticPaths() {
  const res = await request.get('http://localhost:5000/api/users');
  const users = res.data;
  const paths = users.map(user => `/user/detail/${user.id}`);
  return { paths, fallback: false }
}
export async function getStaticProps({ params }) {
  console.log('params', params, new Date().toLocaleTimeString());
  const res = await request.get('http://localhost:5000/api/users/${params.id}');
  return {
    props: {
      user: res.data
    }
  }
}
export default UserDetail;
```

14. 部署

14.1 直接部署

```
npm run build
npm run start
```

14.2 集成express部署

14.2.1 编译

```
npm run build
```

14.2.2 start.js

start.js

```
const next = require('next');
const app = next({ dev:false });
const handler = app.getRequestHandler();
app.prepare().then(() => {
  let express = require("express");
  let bodyParser = require("body-parser");
  let {UserModel} = require('./model');
  let session = require("express-session");
  let config = require('./config');
  let MongoStore = require('connect-mongo')(session);
  let app = express();

  app.get('*', async (req, res) => {
    await handler(req, res);
  })
  app.listen(5000, () => {
    console.log('服务器在5000端口启动!');
  });
});
```

15.api.js

```
const express = require('express');
const cors = require('cors');
const session = require('express-session');
const app = express();
app.use(
  cors({
    origin: ['http://localhost:3000'],
    credentials: true,
    allowedHeaders: "Content-Type,Authorization",
    methods: "GET,HEAD,PUT,PATCH,POST,DELETE,OPTIONS"
  })
);
app.use(session({
  saveUninitialized: true,
  resave: true,
  secret: 'zhufeng'
}));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
const users = [];
app.get('/api/users', (req, res) => {
  res.json({
    success: true,
    data: users
  });
});
app.post('/api/register', (req, res) => {
  const user = req.body;
  user.id = Date.now() + "";
  user.createdAt = new Date().toISOString();
  users.push(user);
  res.json({
    success: true,
    data: user
  });
});
app.get('/api/users/:id', (req, res) => {
  const id = req.params.id;
  const user = users.find(user => user.id === id);
  res.json({
    success: true,
    data: user
  });
});
app.post('/api/login', (req, res) => {
  const user = req.body;
  req.session.user = user;
  res.json({
    success: true,
    data: user
  });
});
app.get('/api/logout', (req, res) => {
  req.session.user = null;
  res.json({
    success: true,
    data: null
  });
});
app.get('/api/validate', (req, res) => {
  const user = req.session.user;
  if (user) {
    res.json({
      success: true,
      data: user
    });
  } else {
    res.json({
      success: false,
      error: `用户未登录`
    });
  }
});
app.listen(5000, () => console.log('api server started on port 5000'));
```