**JS异步练习**

*//异步原理*

setTimeout(function(){

console.log('开始')

});

new Promise(function(resolve){

console.log('执行for任务');

for(var i = 0; i < 10000; i++){

i == 99 && resolve();

}

}).then(function(){

console.log('执行then函数')

});

console.log('结束');

预计流程为：

1.读取到settimeout，将settimeout加入异步队列

2.读取到promise，执行实际函数，输出“执行for循环”，等待，将then函数加入队列

3.读取到结束，输出“结束”

4.开始执行异步队列，依次输出“开始”和“执行then函数”

即 执行for循环 结束 开始 执行then函数

执行结果为：

执行for循环

结束

执行then函数

开始

开始和执行then函数的顺序颠倒了。查找资料发现，settimeout会将内容作为一个大任务（等同于全局）推入异步队列，而每当一个大任务被完成时，会优先检测异步队列中的小任务（promise等提供的），小任务全部执行完毕后，再按顺序读取异步队列中的大任务（成为新的全局），形成循环；可以理解为存在两个异步队列，按照 大1 全部小 大2 全部小 大3 全部小....的顺序执行或者理解为 每一个大任务自带一个存储小任务的异步队列，大任务在执行完自身异步队列后才算结束，之后顺序执行其他的大任务；而settimeout提供大任务，promise提供小任务。

*//验证部分*

console.log('开始大任务')

setTimeout(function(){

console.log('开始SetTimeOut1任务')

new Promise(function(resolve){

console.log('小任务1');

for(var i = 0; i < 10000; i++){

if(i==20)

{

console.log('小任务1计时');

resolve();

}

}

}).then(function(){

console.log('执行小任务1then函数')

});

setTimeout(function(){

console.log('SetTimeOut1中的settimeout')

})

new Promise(function(resolve){

console.log('小任务2');

for(var i = 0; i < 10000; i++){

if(i==20)

{

console.log('小任务2计时');

resolve();

}

}

}).then(function(){

console.log('执行小任务2then函数')

});

});

new Promise(function(resolve){

console.log('执行大任务中promise');

for(var i = 0; i < 10000; i++){

if(i==10)

{

console.log('大任务计时');

resolve();

}

}

}).then(function(){

console.log('执行then函数')

});

console.log('大任务结束');

setTimeout(function(){

console.log('开始SetTimeOut2任务')

new Promise(function(resolve){

console.log('小任务3');

for(var i = 0; i < 10000; i++){

if(i==20)

{

console.log('小任务3计时');

resolve();

}

}

}).then(function(){

console.log('执行小任务3then函数')

});

new Promise(function(resolve){

console.log('小任务4');

for(var i = 0; i < 10000; i++){

if(i==20)

{

console.log('小任务4计时');

resolve();

}

}

}).then(function(){

console.log('执行小任务4then函数')

});

});

执行结果:

开始（全局）大任务

（此时插入了settimeout1）

执行（全局）大任务中promise

（全局）大任务计时

（全局）大任务结束

（此时插入了settimeout2）

执行then函数(大任务中的小任务)

开始SetTimeOut1任务（大任务）

小任务1

小任务1计时

（此时插入了settimeout3）

小任务2

小任务2计时

执行小任务1then函数（大任务中的两个小任务）

执行小任务2then函数

开始SetTimeOut2任务（大任务）

小任务3

小任务3计时

小任务4

小任务4计时

执行小任务3then函数（大任务中的两个小任务）

执行小任务4then函数

SetTimeOut1中的settimeout（settimeout1中的settimeout作为大任务排到了settimeout2之后）

*//练习*

console.log('1');

setTimeout(function() {

console.log('2');

process.nextTick(function() {

console.log('3');

})

new Promise(function(resolve) {

console.log('4');

resolve();

}).then(function() {

console.log('5')

})

})

process.nextTick(function() {

console.log('6');

})

new Promise(function(resolve) {

console.log('7');

resolve();

}).then(function() {

console.log('8')

})

setTimeout(function() {

console.log('9');

process.nextTick(function() {

console.log('10');

})

new Promise(function(resolve) {

console.log('11');

resolve();

}).then(function() {

console.log('12')

})

})

预计结果

1 7 6 8 2 4 3 5 9 11 10 12

实际输出

1

7

6

8

2

4

3

5

9

11

10

12

一致

**Ant design Pro学习**

├── 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

整个git下来以后,除了以上目录外的都是无关文件,可以删除

(npm install之后就要小心了,还有各类插件的配置文件)

目前接触了的部分：

config/

router.config.js 路由管理 路由添加删除都是在这里而不是router.js(在router.config.js改了以后框架会自动帮你在这配置菜单信息,实际还是这里)

mork/

模拟的数据就放在这里

src/

layout用于存放各类布局

locales用于给代码中的变量一个名字(各国本地化),各个部件对应了一个文件

models (dva流的中转站?)mork时需要在这里为prop搭建桥梁,

page 在antd-pro1.x中叫做router;用于存放各种页面及其样式(还有一些通用文件?);

services 用于存放各类请求,mork时需要在创建函数返回一个api请求

~~整个工程牵一发动全身,随意改动很有可能就报错,又不知道哪里错了/怎么改回去~~

强烈建议改动前备份一份

**一.添加一个页面**

官方的说明文档有点坑,至少没有经验的人,极难按照说明文档成功添加

**官网文档+个人注释**

菜单跳转到外部地址#(尝试失败,添加路由之后没有按钮出现,之后再尝试)

(1.在src/page下添加对应的js和less文件)

新增页面#

(2.添加路由)

(找准位置添加,次级之下无法再添加)(格式参考自带部分)

(注意:不能添加到404页面之后,否则无论如何新页面都是404,原理不祥).

(官方的代码怎么改都不好用,放上最后自己的)

// newpage

{

path: "/experiment",(你命名的路由地址,可以随便起会自动创建)

icon: "file",(文件图标,有很多自带的,Pro的说明文档里没有,去antd里找)

name: "experiment",(菜单中显示的名字,受本地化local影响)

routes: [

{

path: "/experiment/newPage",

name: "newPage",,

component: "./experiment/NewPage",(文件真实存在的地方)

}

],

},

加好后，会默认生成相关的路由及导航。

如有需要还可以添加

如果需要隐藏某条菜单栏 可以在该条数据中增加hideInMenu参数

HideInMeun:True

如果需要隐藏面包屑中的某个层级，可以使用hideInBreadcrumb

hideInBreadcrumb:true

设置authority属性 配置该项菜单的准入权限,权限等级不足则不显示

Authority:’admin’

**2.原有代码迁移**

原有教程中的代码迁移进来会eslint疯狂报错,但是不影响使用

0. 添加"babel-preset-es2015": "^6.14.0","babel-preset-react": "^6.11.1",没有任何作用

1.添加/\*eslint-disable\*/可以忽略报错.可以在eslint设置中改变,但又可能影响正常的代码校对,很麻烦,最好还是跟着antdpro的规范写

2.网上的解决方法:airbnb 的插件 jsx-a11y 暂时不能使用v6版本,降低 eslint-plugin-jsx-a11y 版本为 5.1.0.好像并没有什么用

原有的一些布局会被忽略如col row

~~试了很多方法,暂时放置, 使用/\*eslint-disable\*/,打得时候小心点~~

心态炸了,把Eslint的配置文件直接删了,之后找到合适的配置再弄回来吧,终于没有烦人的红线了(然而这样不太对)

后续:

eslint只是一个辅助,多人开发有用,现在先不用管;小心一点代码照写

**3.Mork**

Antd的底层框架umi自带了代理请求功能，通过代理请求就能够轻松处理数据模拟的功能。

官方文档讲了重点,不过依旧不适合新手

1.在mork文件夹下添加需要的数据,以表格为例

const morktest = [{

key:'1',

name:'xiaoming',

age:16,

address:'1h'

}, {

key:'2',

name:'wanglihong',

age:28,

address:'2b'

},

{

key:'3',

name:'caoxueqing',

age:49,

address:'136589663d'

}];

//还需要添加api到最后的export中

'GET /api/home': morktest,

2.在src/servers下的api中加入请求数据端口的函数

export async function queryHome() {

return request('/api/home');

}

3.在src/models中创建接收action, 更新 state 并传递给 props,调用fetch函数即可请求mork数据

import { queryHome } from '@/services/api';

export default {

namespace: 'home',

state: {

list: [],

},

effects: {

\*fetch({ payload }, { call, put }) {

const response = yield call(queryHome, payload);

yield put({

type: 'show',

payload: response,

});

},

},

reducers: {

show(state, action) {

return {

...state,

list: action.payload,

};

},

},

};

4.在文件中请求数据

//首先链接文件

@connect(({ home, loading }) => ({

home,

loading: loading.models.home,

}))

//再在didmount中得到props

componentDidMount() {

const { dispatch } = this.props;

//可以为dispatch添加settimeout模拟数据延迟

dispatch({

type: 'home/fetch', // action的类型，由home命名空间和其下面的fetch方法构成

}); // 派发这个action就会调用home中的fetch函数，然后就会请求数据

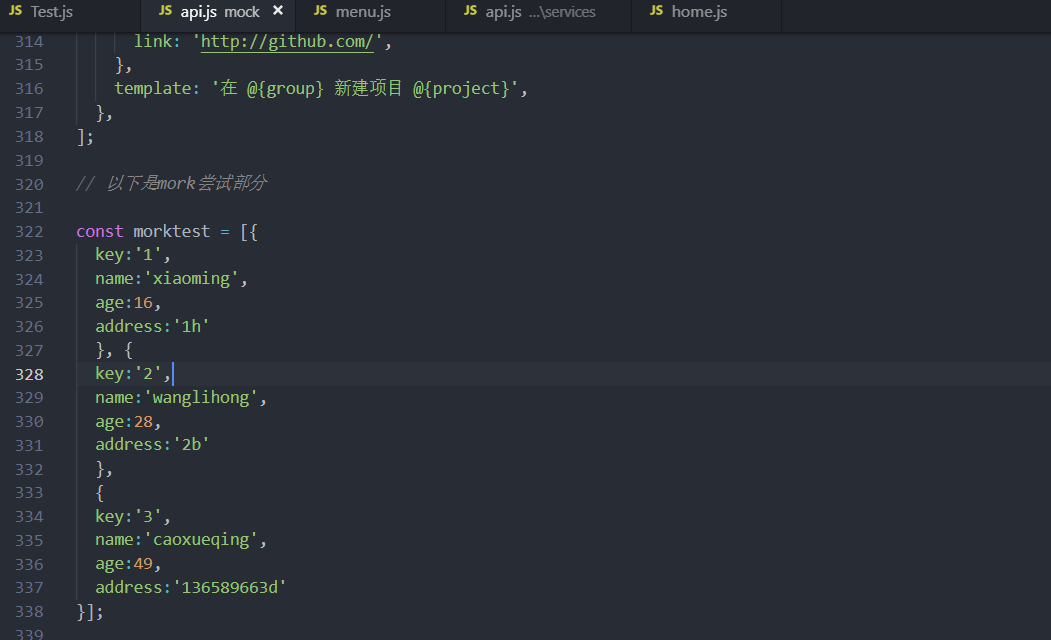
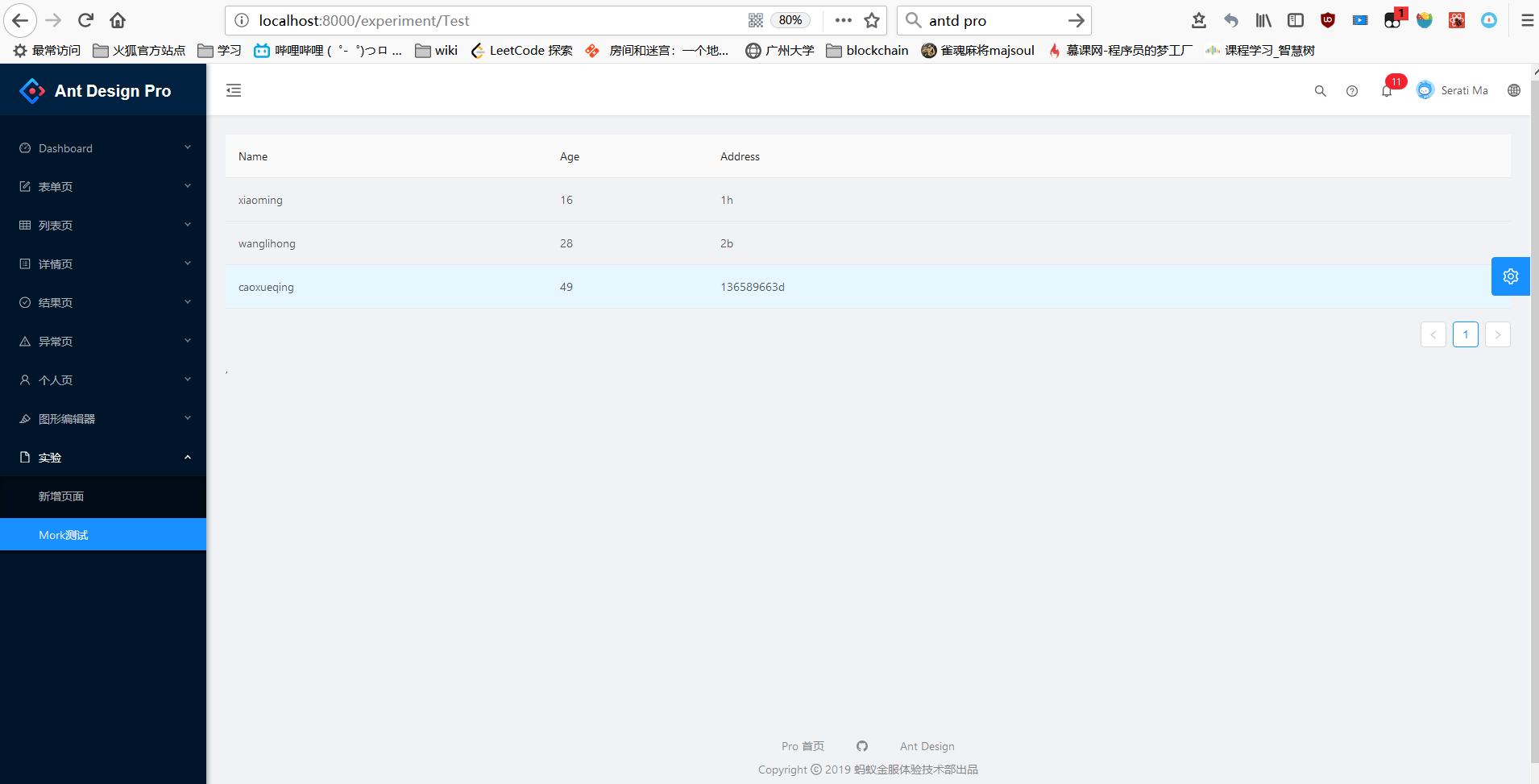
}

const { home: { list } } = this.props; // 将home中的this.props.list命名为list

5.就可以使用啦

const dataSource = [...list];

放上成功后的图

**疑问:**

1.并没有使用到官方文档中的mork.js

2.. 官方文档中”手动添加 setTimeout 模拟延迟#

你可以重写请求的代理方法，在其中添加模拟延迟的处理，如：

'POST /api/forms': (req, res) => {

setTimeout(() => {

res.send('Ok');

}, 1000);

},”

没找到这段该加在那里,或者没用mork.js用不了?

预设框架中也没有例子,都是mork数据瞬间到达,页面延迟更新的

所以我也采用了延迟dispatch的方法来模拟数据延迟,而不是真的延迟了mork数据的传输

备用部分

Antd-Pro2.0版本如何修改代理，让Mock变为真实服务器接口

<https://www.cnblogs.com/soyxiaobi/p/10557665.html>

**4.Antd Table组件**

表格应该会成为非常常用的组件

不得不说antd提供了功能齐全,外观优美的许多表格样式

看完发现没什么好写的,参照官方文档就好了

几点需要注意的:

1. 控制表格大小需要在布局里控制,全表宽度似乎是一定占满布局空间的
2. 单列宽度的单位可以是px或百分比
3. 滚动和自定义大小 一起使用会导致原因不明的崩溃