参照<http://react-china.org/t/webpack-react-react-router-redux-less-flex-css-es6-react-cnode/6332>源码

这里面写得有点乱，看不太明白，所以自己学着边学边做

搭建react项目结构，直接用官网的方法 create-react-app myapp

使用create-react-app 创建的项目默认不支持less，以下增加less配置的步骤<https://segmentfault.com/a/1190000010162614>

布局使用flex.css，移动端flex布局神器，兼容微信，UC，webview等移动端主流浏览器08.10

npm install flex.css –save

<http://huziketang.com/books/react/>

### flex属性大全

dir：主轴方向

top：从上到下

right：从右到左

bottom：从上到下

left：从左到右（默认）

main：主轴对齐方式

right：从右到左

left：从左到右（默认）

justify：两端对齐

center：居中对齐

cross：交叉轴对齐方式

top：从上到下

bottom：从上到下

baseline：跟随内容高度对齐

center：居中对齐

stretch：高度并排铺满（默认）

box：子元素设置

mean：子元素平分空间

first：第一个子元素不要多余空间，其他子元素平分多余空间

last：最后一个子元素不要多余空间，其他子元素平分多余空间

justify：两端第一个元素不要多余空间，其他子元素平分多余空间

1、You should not use <Route> or withRouter() outside a <Router>

源码里面是每个路由导航的Component都包含一个Rooter，我想要整个页面就用一个Footer，但是Footer里面包含了<Route>,<Route>必须包含在<Rooter>里面，所以不可以这样写

1、render(

<Provider store={store}>

<div>

{router}

<Footer/>

</div>

</Provider>,

document.getElementById('root')

);

2、 React.Children.only expected to receive a single React element child.

<Router><Provider>里面只能有一个一级子节点,Provider的store是必须的，Router的history是必须的

History用这种方法创造

import createBrowserHistory from 'history/createBrowserHistory';

const history = createBrowserHistory();

var history = process.env.NODE\_ENV !== 'production' ? browserHistory : hashHistory;

自己探索出来整个项目的结构就是

<Provider store={store}>

<Router history={history}>

<Switch>

<Route key=’’ path=’’ component=’’/>

<Route key=’’ path=’’ component=’’/>

………………………………..

</Switch>

</Router>

</Provider>

3、修改react端口号

打开react项目的 package.json文件

将 scripts中的start键值对

"start": "roadhog server"

修改为

"start": "set PORT=3000&&roadhog server",

4、React并没有一个自己的Component处理网络请求，自己包装

5、render 方法必须要返回一个 JSX 元素。而且必须要用一个外层的 JSX 元素把所有内容包裹起来。返回并列多个 JSX 元素是不合法的

6\{this.props.children}相当于angular的<ui-view>



<https://segmentfault.com/q/1010000009616045>

react-router4没有indexRoute了。 react-router4版本中路由的本质变成了React组件，也就是自定义标签。所以你可以像使用组件一样是用路由。那么嵌套路由无非就是组件嵌套的写法（自定义标签嵌套而已）

7、<Route path="/" component={App} />

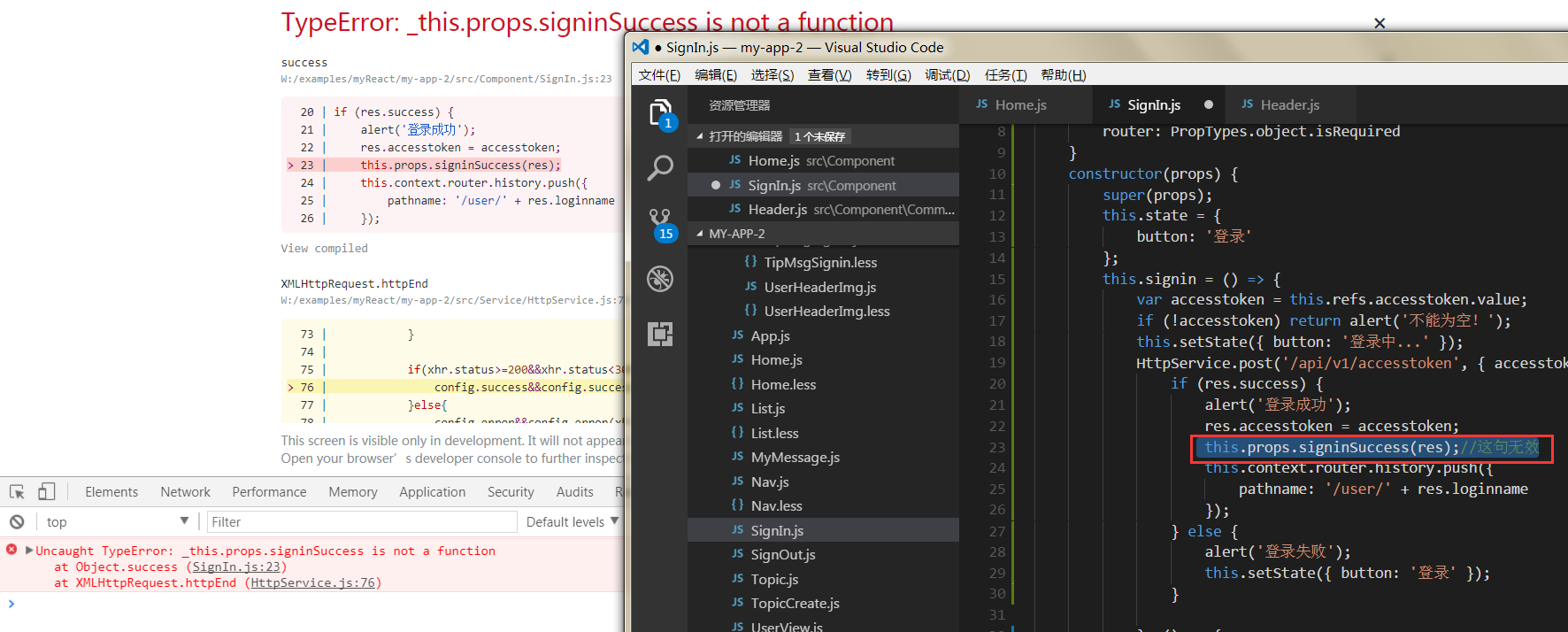
<Route exact path="/" component={Home} />

<Route path="topics" component={Topics} />

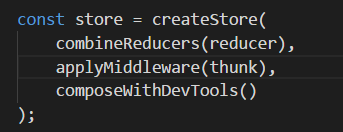
<Route path="/topics/:id" component={Topic} />

用{this.props.children}渲染不出组件，用的是react-router-dom

8、没有传入action 之前，会出现这个错误



9、proxyConsole.js:56 The previous state received by the reducer has unexpected type of "Function". Expected argument to be an object with the following keys: "User"



原因见以下链接

<https://stackoverflow.com/questions/38074154/redux-the-previous-state-received-by-the-reducer-has-unexpected-type-of-functi>

改成

const store = createStore(

combineReducers(reducer),

compose(applyMiddleware(thunk),composeWithDevTools())

);

10、在reducer里面返回新的state,就是store

reducer只有在createStore的时候用到，其他地方没有用到，不用管。

Connet的时候就是组件丛store里面取出自己需要的东西，仅仅是自己需要的数据。

只有触发的时候用到dispatch，其他时候没有用到。

刚开始想像一般推荐的那种方法那样弄两个文件夹Component和Container，试了一天，最后发现store是放共享的东西，我为什么什么都要往里面放。还有一个组件写两个文件很繁琐，直接像源码里面在组件最后connect一下就好了

总体流程就是

Reducer

createStore(reducer)

action

connect(props,action)

this.props.（action.type）(arg)

手动调用this.props.（action.type）(arg)

react-redux自动对应到reducer

react-redux自动更新props

11、前往某个制定的页面

this.context.router.history.push({

pathname: '/user/' + res.loginname

});

12 mapDispatchToProps可以返回一个对象，action('User')返回对象,每个元素都是函数

{

'signinSuccess':( target)=>{

\_ID: User,

target: target,

type: 'signinSuccess'

},

'signin': ( target)=>{

\_ID: User,

target: target,

type: 'signin'

},

‘setState’: ( target)=>{

\_ID: User,

target: target,

type: ‘setState’

},

}

export default (\_ID) => {

var action = {};

var arr = [

'signinSuccess', //登录成功

'signin', //退出登录

'setState' //设置状态

];

for (let i = 0; i < arr.length; i++) {

action[arr[i]] = (target) => {

return {

\_ID: \_ID,

target: target,

type: arr[i]

};

}

}

return action;

}

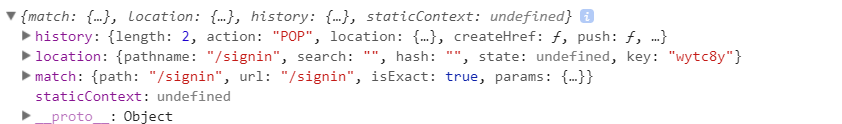
export default connect((state) => { return { User: state.User }; }, action('User'))(SignIn); //连接redux

connect([mapStateToProps], [mapDispatchToProps], [mergeProps], [options])(组件)

[mapDispatchToProps(dispatch, [ownProps]): dispatchProps] (Object or Function): 如果传递的是一个对象，那么每个定义在该对象的函数都将被当作 Redux action creator，而且这个对象会与 Redux store 绑定在一起，其中所定义的方法名将作为属性名，合并到组件的 props 中。

13、TypeError: \_\_WEBPACK\_IMPORTED\_MODULE\_0\_\_Service\_Tool\_\_.a.localItem is not a function

Class里面定义函数要加权限，static,public,private等等

14，在signin里面不connect的时候super(props)输出来的props是这样的，默认的props是

history:

* 1. action:"POP"/"PUSH"
  2. block:*ƒ block()*
  3. createHref:*ƒ createHref(location)*
  4. go:*ƒ go(n)*
  5. goBack:*ƒ goBack()*
  6. goForward:*ƒ goForward()*
  7. length:2//浏览器历史列表中的 URL 数量
  8. listen:*ƒ listen(listener)*
  9. location:{

pathname: "/signin", //同window.location.pathname

search: "", //同window.location.search

hash: "",

state: undefined,// 一个捆绑在这个地址上的object对象

 key: "wytc8y"}

* 1. push:*ƒ push(path, state)*
  2. replace:*ƒ replace(path, state)*

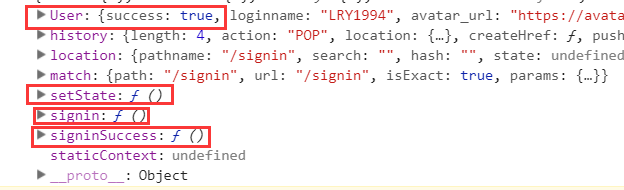
location:和history.location内容一样

match:

* 1. isExact:true //路由extract设置为true
  2. params:{}
  3. path:"/signin"
  4. url:"/signin"

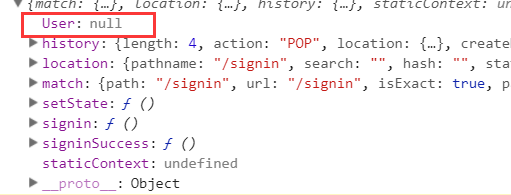
staticContext:undefined

connect之后多了User,和三个action setState,signin,signinSuccess

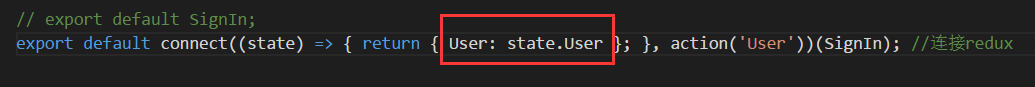


我把default改成return null,User值会变化



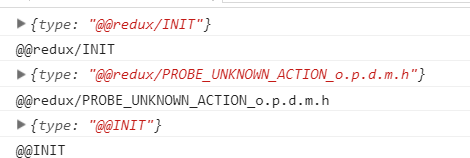


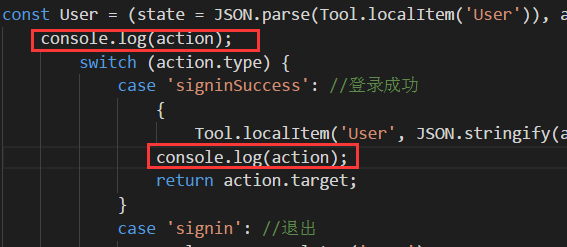
User为什么在props上？因为connect的时候取名叫做User



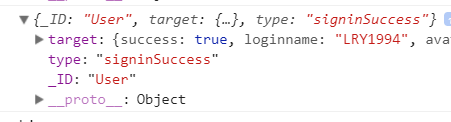
13在user reducer这里输出看看action有什么

刚开始是这样的，应该是combineReducer的时候先占位，但是还没有装入



登录成功后两处输出是一样的，

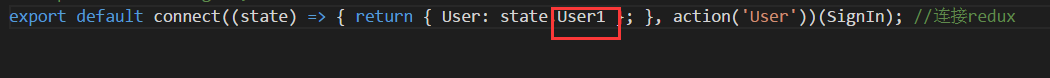
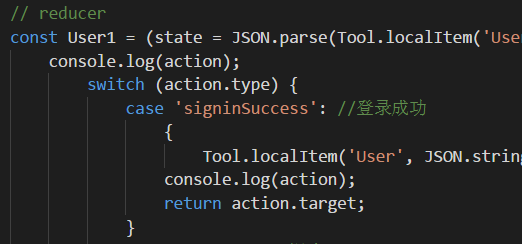
Action



14:



验证成功，把reducer函数名称改一下



15

高阶组件是一个函数（而不是组件）,GetNextPage.jx相当于IndexList的高阶组件，用到了get-next-page插件

她自己自动监听了这些事件，不用自己写触发事件，比如下拉的时候回自动拉取下一页数据

this.monitorEvent = ['DOMContentLoaded', 'load', 'click', 'touchstart', 'touchend', 'haschange', 'online', 'pageshow', 'popstate', 'resize', 'storage', 'mousewheel', 'scroll'];

只需要写好el,拉取的数据就会放在el

使用方法：import GetNextPage from ‘get-next-page’

new GetNextPage(el,setting)

setting参数如下

/\*元素在可视区位置，符合其中一个条件就会触发加载机制\*/

top元素在顶部伸出的距离才加载

right //元素在右边伸出的距离才加载

bottom //元素在底部伸出的距离才加载

left//元素在左边伸出的距离才加载

/\*

发送到服务器的相关数据

\*/

url //发送到服务器的地址

data //发送到服务器的数据

pageName //分页的参数名称，用来加载完成后+1

/\*

回调方法

\*/

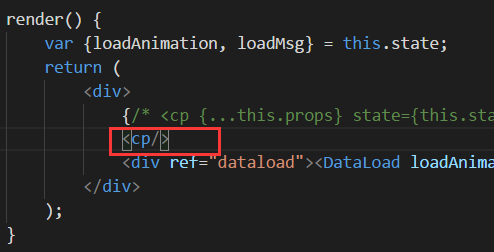
start //开始加载时调用方法

load //加载成功时调用方法

error //开始加载时调用方法

end //加载完成时调用方法

15 组件必须大写字母开头，不然会当做html的标签



16、

this.refs.dataload

<div ref="dataload">

17知识点

#### Mounting

These methods are called when an instance of a component is being created and inserted into the DOM:

* [constructor()](https://facebook.github.io/react/docs/react-component.html#constructor)
* [componentWillMount()](https://facebook.github.io/react/docs/react-component.html#componentwillmount)
* [render()](https://facebook.github.io/react/docs/react-component.html#render)
* [componentDidMount()](https://facebook.github.io/react/docs/react-component.html#componentdidmount)

#### Updating

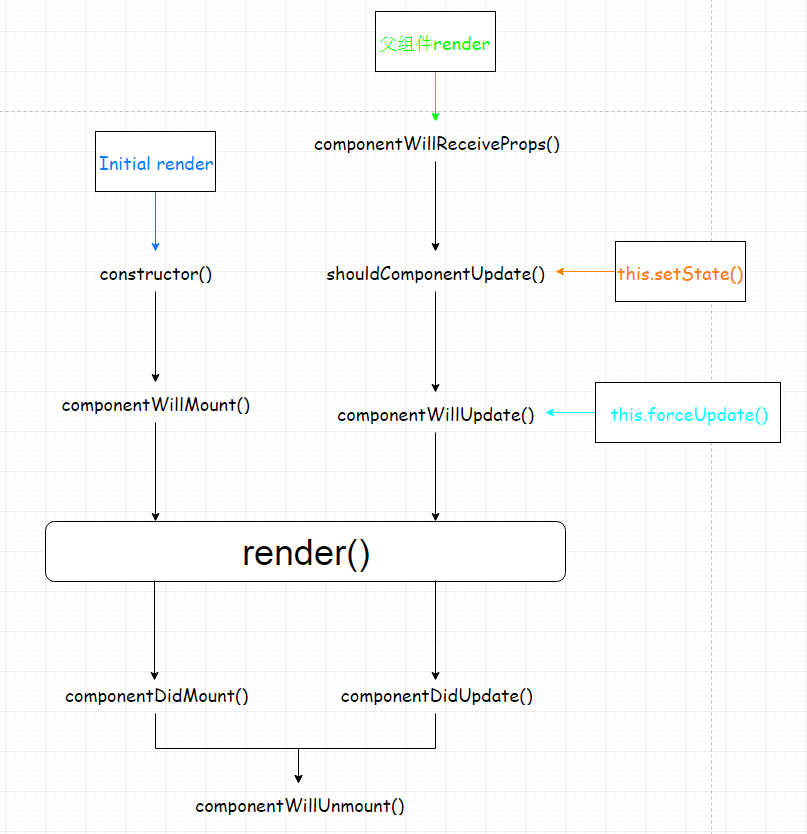
These methods are called when a component is being re-rendered:

* [componentWillReceiveProps()](https://facebook.github.io/react/docs/react-component.html#componentwillreceiveprops)
* [shouldComponentUpdate()](https://facebook.github.io/react/docs/react-component.html#shouldcomponentupdate)
* [componentWillUpdate()](https://facebook.github.io/react/docs/react-component.html#componentwillupdate)
* [render()](https://facebook.github.io/react/docs/react-component.html#render)
* [componentDidUpdate()](https://facebook.github.io/react/docs/react-component.html#componentdidupdate)

#### Unmounting

This method is called when a component is being removed from the DOM:

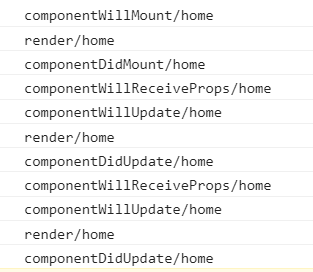
* [componentWillUnmount()](https://facebook.github.io/react/docs/react-component.html#componentwillunmount)

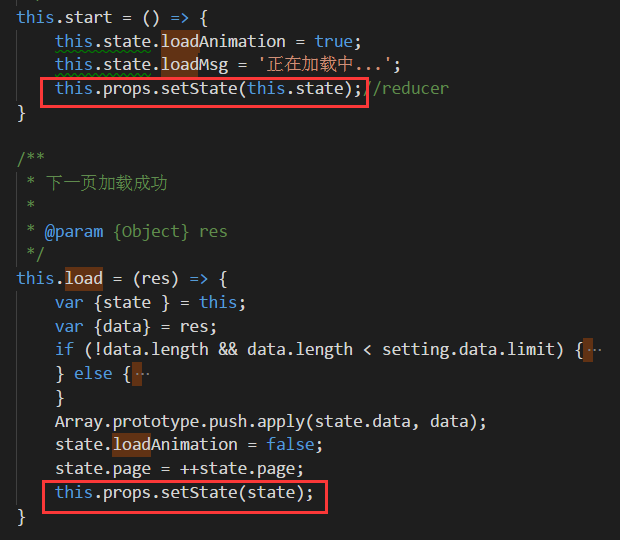
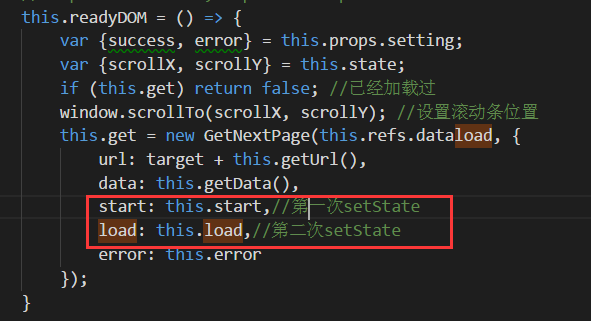


打开localhost的时候，/home初始化，因为调用了两次setState,所以调用了两次componentWillReceiveProps,

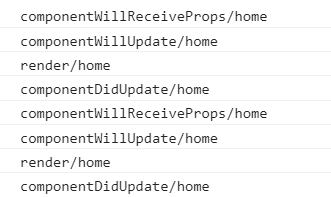
Prop改变会自动触发componentWillReceiveProps, shouldComponentUpdate ,componentWillUpdate ,render,componentDidUpdate,每一次改变prop就会触发这些

State改变需要使用setState方法设置，然后会自动调用shouldComponentUpdate,componentWillUpdate,render, componentDidUpdate()。跟Prop改变不同的是，state改变可能不会立即触发，他会和其他合并之后再触发

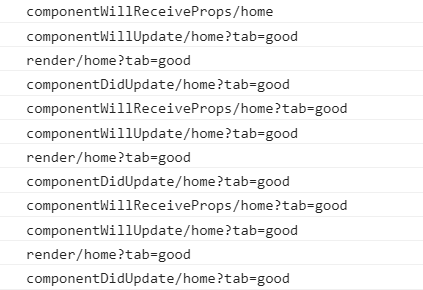
原因



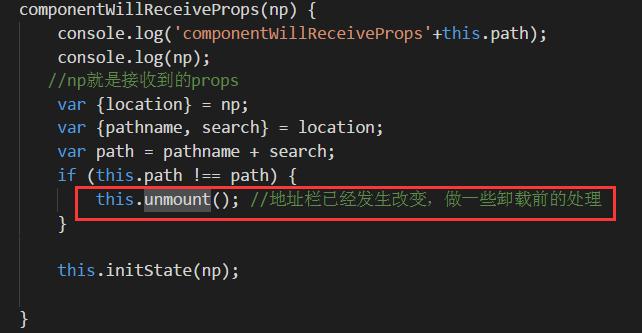
下拉的时候同样是两次



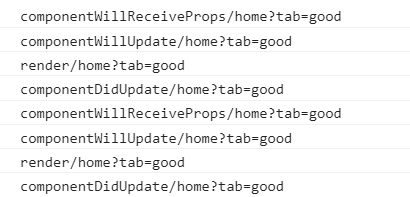
接着切换为tab=good



这里多了一次是URL改变也触发了一次setState



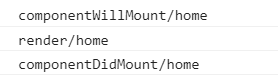
tab=good的时候下拉



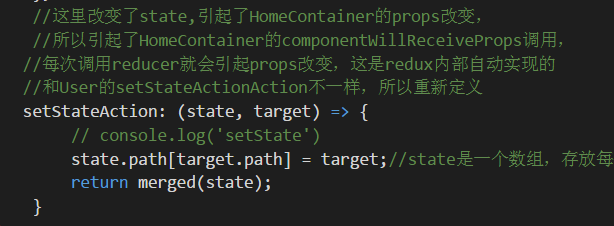
点击“发表”



再回来“首页”，

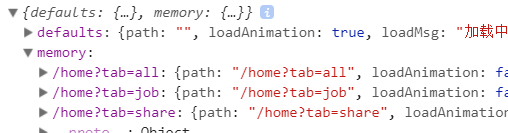


但是什么原理导致HomeContainer的prop改变？？ 每次调用action就会引起props改变，这是redux内部自动实现的



17\super()和super(props)的区别在于能否在子类prototype的constructor中调用this,props

18首页用一个数组存储每个tab之前浏览的情况,但是这里有个情况，/home和/home?tab=all区别存储了



5cff5f1d-f168-44d3-a5c0-b59b9f7e01ca