

爱彼迎

配置

别名配置

craco

安装：

```
npm install @craco/craco@alpha -D
```

1. 自己在根目录下创建一个craco.config 在里面配置别名
2. 在pack.json里面的scripts 换成craco启动 换成craco 他会把之前的reac配置和自己刚刚配置的进行合并再启动，才可以使别名配置生效

```
const path = require('path')
const resolve = pathname => path.resolve(__dirname, pathname)
module.exports = {
  webpack: {
    // 要传绝对路径 所以在上面const了一下方便下面导入
    alias: {
      '@': resolve('src'),
      'components': resolve('src/components'),
      'utils': resolve('src/utils')
    }
  }
}
```

less配置

安装：`npm i craco-less@2.1.0-alpha.0`

修改：`craco.config.js` 文件如下

```
const CracoLessPlugin = require('craco-less');

module.exports = {
```

```

plugins: [
  {
    plugin: CracoLessPlugin,
    options: {
      lessLoaderOptions: {
        lessOptions: {
          modifyVars: { '@primary-color': '#1DA57A' },
          javascriptEnabled: true,
        },
      },
    },
  },
],
};

```

CSS样式重置

1. `npm install normalize.css`

下载后在index里面引用它

```
import 'normalize.css';
```

2. reset.less

在src/assets/css 里自己创建

```

* {
  padding: 0;
  margin: 0;
}

```

//这段 CSS 代码的作用是为所有 HTML 元素（包括所有子元素）设置统一的
//padding 和 margin 样式，将它们的值全部设为 0。

一些其他的变量我们在当前的文件夹下创建了variables.css

```
@textColor: #484848;  
@textColorSecondary: #222;  
//创建了两个变量分别为对应的颜色
```

在编辑好这些 我们要在重置的less文件中引入这些预先修改的变量，所以要在对应的less上面引入他

```
@import './variables.less'
```

3. 把编辑好的reset引入到主less中 方法同上

Router 路由配置

1. 安装： `npm install react-router-dom`

2. 然后在index.js中导入这个第三方包 `import { HashRouter } from 'react-router-dom'`

3. 在render里面写上标签

```
root.render(  
  <React.StrictMode>  
    <HashRouter>  
      <App/>  
    </HashRouter>  
  </React.StrictMode>  
);
```

针对爱彼迎的项目 Header 和 Footer是固定不变的 所以路由的切换至需要配置其余里面的内容

在App.js中配置大概，然后其余的配置在router文件夹下再引入

```
import React, { memo } from 'react'  
import { useRoutes } from 'react-router-dom'  
import routes from './router'  
  
const App = memo(() => {  
  return (  
    <div className='app'>
```

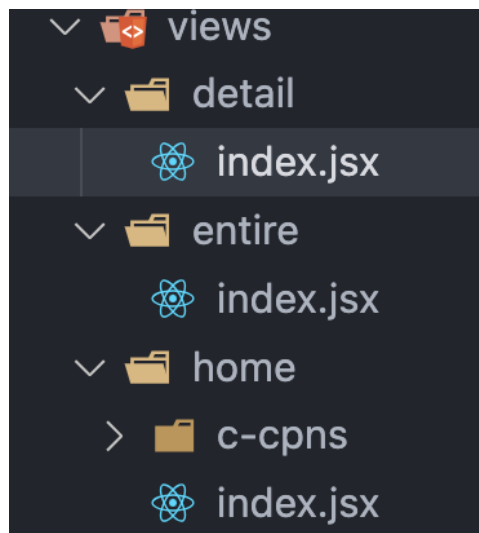
```

    <div className='header'>header</div>
    <div className='page'>
      {useRoutes(routes)}
    </div>
    <div className='footer'>footer</div>
  </div>
)
})

export default App

```

views文件夹是用来存放编辑的界面目录如下 (c-cpns = children-components)
 jsx是为了后面导出这个组件



开始在router下配置

配置的过程其实就是编辑对应的映射关系

```

import React from 'react'
import { Navigate } from "react-router-dom"

//配置懒加载
const Home = React.lazy(() => import('@views/home'))
const Entire = React.lazy(() => import('@views/entire'))

```

```
const Detail = React.lazy(() => import('@views/detail'))
//这些组件在实际渲染时会被懒加载。
//这意味着在路径匹配时，这些组件会在需要时才异步加载，而不是在应用启动时全部
```

```
const routes = [
  { //当默认进来的时候 我们希望他直接进入home页面
    path: '/',
    element: <Navigate to='/home' />
  },
  {
    path: '/home', //路径
    element: <Home /> , //组件
    //这个组件可以这样写是因为配置了懒加载 懒加载的时候给他const了什么名字
    //这里就写什么
  },
  {
    path: '/entire',
    element: <Entire />
  },
  {
    path: '/detail',
    element: <Detail />
  }
]

export default routes
```

下一步，在src下的index.js配置 suspense 因为是异步懒加载，来确保在没有加载出来的时候显示一些内容

```
import React, { Suspense } from 'react'; //从react中导入suspense
import ReactDOM from 'react-dom/client';
import { HashRouter } from 'react-router-dom'

import App from '@App';
```

```
import './assets/css/index.less'
import 'normalize.css'

const root = ReactDOM.createRoot(document.getElementById('root'))
root.render(
  <React.StrictMode>
    <Suspense fallback='loading'> //把内容用suspense包裹 fallback中
      <HashRouter>
        <App/>
      </HashRouter>
    </Suspense>
  </React.StrictMode>
);
```

Redux的状态管理

方式：`@reduxjs/toolkit` 或 自己搭建

安装：`npm install @reduxjs/toolkit` 、 `npm install react-redux`

在src/store 下新建 index.js进行搭建，代码如下 reducer里面的内容是后面创建完代码后导入进来的

```
import { configureStore } from "@reduxjs/toolkit";
import homeReducer from './modules/home'
import entireReducer from './modules/entire'

const store = configureStore({
  reducer: {
    home: homeReducer, //rtk配置
    entire: entireReducer //手动配置
  }
})
```

```
export default store
```

2. 在store文件夹下新建一个modules用来存放具体的文件 home采用rtk的方法进行搭建，entire采用手动搭建

home的搭建

在module文件夹下新建home.js，代码初始化内容如下

```
import { createSlice } from '@reduxjs/toolkit' //帮助我们创建对应的

const homeSlice = createSlice({
  name: 'home',
  initialState: { //初始化数据，但是现在还不知道需要放什么数据进去

  },
  reducers: {

  }

})

export default homeSlice.reducer
```

在src下的index.js中引入 `import { Provider } from 'react-redux'` 并在下方对内容进行包裹

```
root.render(
  <React.StrictMode>
    <Suspense fallback='loading'>
      <Provider store={store}> //这个store就是上面所配置的store
        <HashRouter>
```

```

        <App/>
      </HashRouter>
    </Provider>
  </Suspense>
</React.StrictMode>
);

```

网络请求 Axios

1. 首先在services文件夹下创建index.js 作为整个services文件的出口
2. 在该文件夹下创建request，对Axios进行封装，config文件是用来存放常量
3. 在该文件夹下新建modules文件夹——每个模块都有自己独立的文件来管理对应的网络请求

封装axios

安装：`npm install axios`

在request下新建index.js 和 config.js。 config用来放后面的baseUrl

index的代码如下：

```

import axios from "axios";
import { BASE_URL, TIME_OUT } from "../config";
class HyRequest {
  constructor(baseUrl, timeout) {
    this.instance = axios.create({
      baseUrl,
      timeout
    });

    this.instance.interceptors.response.use((res) => {
      return res.data
    }, err => {
      return err
    })
  }
}

```



```

request(config) {
  return this.instance.request(config)
}

get(config){
  return this.request({...config, method: 'get'})
}
post(config) {
  return this.request({...config, method:'post'})
}
}

export default new HyRequest(baseUrl, TIME_OUT)

```

axios的config文件

```

//常量通常全部都用大写
export const BASE_URL = 'http://codercba.com:1888/airbnb/api' //
export const TIME_OUT = 1000

```

在出口文件index中引入

```

import hyRequest from './request'

export default hyRequest

```

现在我们就可以在对应的页面开始进行网络请求了。举例一个home的，他是在对应的窗口页面下的index.jsx进行修改

```

import React, { memo, useEffect } from 'react'
import hyRequest from '@services'

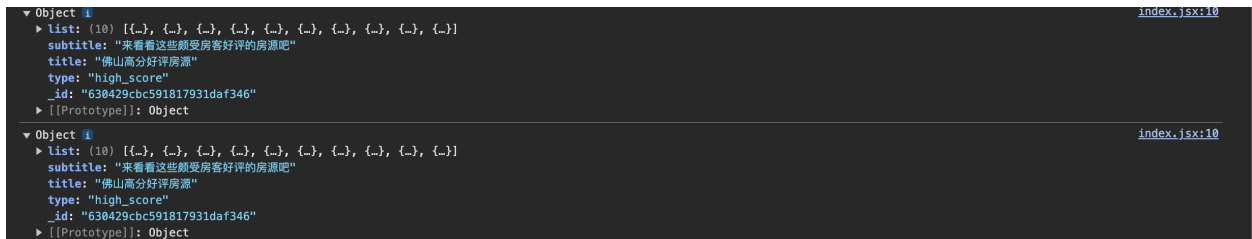
const Home = memo(() => {

```

```
//网络请求代码
useEffect(() =>{
  hyRequest.get({url: '/home/highscore'}).then(res=>{
    console.log(res)
  })
}, [])

return (
  <div>Home</div>
)
})

export default Home
```



```
Object
  list: (10) [Object, Object, Object, Object, Object, Object, Object, Object, Object, Object]
  subtitle: "来看看这些颇受房客好评的房源吧"
  title: "佛山高分好评房源"
  type: "high_score"
  _id: "630429cbc591817931daf346"
  [[Prototype]]: Object
index.jsx:10

Object
  list: (10) [Object, Object, Object, Object, Object, Object, Object, Object, Object, Object]
  subtitle: "来看看这些颇受房客好评的房源吧"
  title: "佛山高分好评房源"
  type: "high_score"
  _id: "630429cbc591817931daf346"
  [[Prototype]]: Object
index.jsx:10
```

网络请求执行了两次是因为现在是严格模式，去根目录下的index.js中删掉严格模式就可以了

Header

各个页面共享一个Header。由于header的东西比较多因此我们在components文件夹下开发好再在App.jsx中引用过来

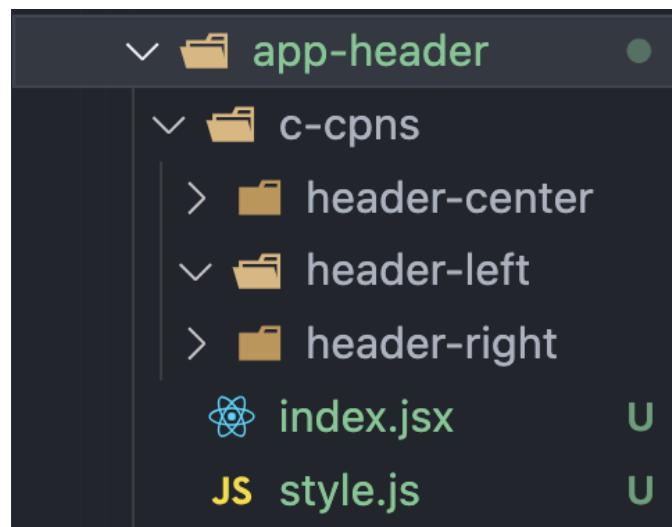
样式开发

1. 安装：`npm install styled-components`
2. 在header的文件夹下创建style.js 用来存放样式，大致代码如下

```
import styled from 'styled-components'
```

```
//这里导出的实际上也是一个标签 用来包裹index中想让它为这个样式的对应元素
//支持样式嵌套
export const HeaderWrapper = styled.div`
```

由于整个header的内容过多 我们采取对相应的内容进行封装



3. 对header进行flex布局

首先编辑整个header的样式

```
import styled from 'styled-components'

//这里导出的实际上也是一个标签 用来包裹index中想让它为这个样式的对应元素

export const HeaderWrapper = styled.div`
  display: flex;
  align-items: center;
  height: 80px;
  border-bottom: 1px solid #eee;
`
```

我们要做的是将left center right都放在他们对应的位置

方法：将left和right的left都修改为1px，right再进行其他修改来让right不会影响到center的位置

```
export const RightWrapper = styled.div`
  flex: 1px;
  display: flex;
  justify-content: flex-end;
`
```

现在布局步骤完成

编辑HeaderLeft

SVG的图标处理这里不做记述——我们将每个图片都封装成了组件存储在了assets下的svg文件夹下。在headerleft中直接引入对应图片组件就好。由于是svg它的图片颜色属性调整我们在headerleft的style里进行调整

```
export const LeftWrapper = styled.div`
  flex: 1px;
  color: red;
`
```

需要实现将鼠标移到logo上指针为指向效果

在headerleft中 把logo用div包裹起来，并把这个div编辑指针移动变换样式

```
const HeaderLeft = memo(() => {
  return (
    <LeftWrapper>
      <div className='logo'>
        <IconLogo />
      </div>
    </LeftWrapper>
  )
})
```

```
.logo {
  margin-left: 25px;
  cursor: pointer;
}
```

编辑主题色

把所有主题色都抽取出来在一个地方进行管理

1. 在assets中新建theme index.js 在这里编

```
const theme = {
  color: {
    primaryColor: '#ff385c',
    secondaryColor: '#00848A'
  }
}
export default theme
```

2. 返回根目录下的index.js 将组件包裹起来

```
<ThemeProvider theme={theme}>
  <HashRouter>
    <App />
  </HashRouter>
</ThemeProvider>
```

3. 返回headerleft 的style里面引入对应的主题色

```
import styled from "styled-components";

export const LeftWrapper = styled.div`
  flex: 1px;
```

```

display: flex;
color: ${props => props.theme.color.primaryColor}; //svg log的颜色

.logo {
  margin-left: 24px;
  cursor: pointer;
}

```

HeaderRight

1. svg的引入不做赘述 首先是搭建基础框架在index.jsx中

```

import React, { memo } from 'react'
import { RightWrapper } from './style'
import IconGlobal from '@assets/svg/icon_global'
import IconMenu from '@assets/svg/icon_menu'
import IconAvatar from '@assets/svg/icon_avatar'

const HeaderRight = memo(() => {
  return (
    <RightWrapper>
      <div className='btns'>
        <span className='btn'>登录</span>
        <span className='btn'>注册</span>
        <span className='btn'>
          <IconGlobal />
        </span>
      </div>

      <div className='profile'>
        <IconMenu />
        <IconAvatar />
      </div>
    </RightWrapper>
  )
})

```

```
      </RightWrapper>
    )
  })

export default HeaderRight
```

2. 添加样式

```
import styled from "styled-components";
//在theme中又添加了一个text样式

export const RightWrapper = styled.div`
  flex: 1px;
  display: flex;
  justify-content: flex-end;
  align-items: center;

  color:${props => props.theme.text.primary};
  font-size: 14px;
  font-weight: 600;

  .btns {
    display: flex;
    box-sizing: content-box;

    .btn {
      height: 18px;
      line-height: 18px;
      padding: 12px 15px;
      border-radius: 22px;
      cursor: pointer;
      box-sizing: content-box;

      &:hover {
```

```

        background-color: #f5f5f5;
    }
}
}

```

```

.profile {
  display: flex;
  align-items: center;
}

```

鼠标放到profile上出现二级页面

在index的profile标签内新建一个div classname 为panel

具体的搭建和css样式不做赘述，这里重点说一下怎么实现点击显示

1. 在index页写一个showState的hook

```

import React, { memo, useState } from 'react'
import { RightWrapper } from './style'
import IconGlobal from '@assets/svg/icon_global'
import IconMenu from '@assets/svg/icon_menu'
import IconAvatar from '@assets/svg/icon_avatar'

const HeaderRight = memo(() => {
  const [showPanel, setShowPanel] = useState(false)
  return (
    <RightWrapper>
      <div className='btns'>

```

2. 再在下面做一个逻辑运算 把需要点击显示的内容扩起来

```

<div className='profile'>
  <IconMenu />

```



```

<IconAvatar />
{showPanel && (
  <div className='panel'>
    <div className='top'>

      <div className='item register'>注册</div>
      <div className='item login'>登录</div>
    </div>
    <div className='bottom'>
      <div className='item'>出租房源</div>
      <div className='item'>开展体验</div>
      <div className='item'>帮助</div>
    </div>
  </div>
)}

</div>

```

3. 给profile监听点击事件

```
<div className='profile' onClick={profilcClickHandle}>
```

`profilcClickHandle` 是执行的对应函数

4. 写函数的对应内容

```

function profilcClickHandle() {
  setShowPanel(true)
}

```

5. 再点击整个window的任何一个地方，panel消失

```

useEffect(()=>{
  window.addEventListener('click', ()=> {
    setShowPanel(false)
  }, true)
}, [])

```

HeaderCenter

headercenter实现的是一个搜索框的功能

样式的具体编辑不做赘述

```
//index.jsx
import IconSearchBar from '@assets/svg/icon-search-bar'
import React, { memo } from 'react'
import { CenterWrapper } from './style'

const HeaderComponent = memo(() => {
  return (
    <CenterWrapper>
      <div className='search-bar'>
        <div className='text'>
          搜索房源和体验
        </div>
        <div className='icon'>
          <IconSearchBar />
        </div>
      </div>
    </CenterWrapper>
  )
})

export default HeaderComponent
```

```
//style.js
import styled from "styled-components";

export const CenterWrapper = styled.div`
  .search-bar {
    display: flex;
    justify-content: space-between;
```

```

align-items: center;
width: 300px;
height: 48px;
box-sizing: border-box;
padding: 0 8px;
border: 1px solid #ddd;
border-radius: 24px;
cursor: pointer;
${props => props.theme.mixin.boxShadow};

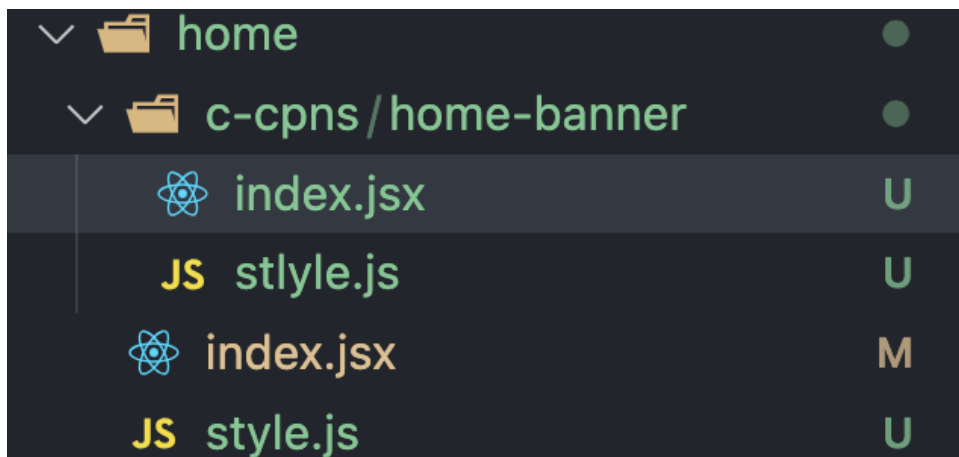
.text {
  padding: 0 16px;
  color: #222;
  font-weight: 600;
}

.icon {
  display: flex;
  align-items: center;
  justify-content: center;
  width: 32px;
  height: 32px;
  border-radius: 50%;
  color: #fff;
  background-color: ${props => props.theme.color.primaryColor};
}

```

首页

由于首页也是由每一个子组件构成的 因此我们将其封装为HomeBanner



图片的引入

```
import styled from 'styled-components'
import coverImg from '@assets/img/cover_01.jpeg'

export const BannerWrapper = styled.div`
height: 529px;
background: url(${coverImg}) center/cover;
`
```

要import 引入进来

goodprice栏的内容展示

从后端引用数据，先从store/modules/home.js中接收数据

1. 首先在redux中写好存储数据的内容

```
const homeSlice = createSlice({
  name: 'home',
  initialState: { //初始化数据，但是现在还不知道需要放什么数据进去
    goodPriceInfo: {}
  },
  reducers: {
    changeGoodPriceAction(state, {payload}) {
      state.goodPriceInfo = payload
    }
  }
})
```

```

    }
  }

}))

export const { changeGoodePriceAction } = homeSlice.actions

export default homeSlice.reducer

```

2. 发送网络请求，发起网络请求写在对应的组件里，发送的真正逻辑写在redux中

页面上的展示

在components文件夹下，新建为selection-header

服务器信息请求大致同上的网络请求



整套loft · 1室1卫1床

【闲梦】韩系ins纯白民宿/一次性毛巾/复式loft/落地窗/拍照纪念/...

¥194/晚

★★★★★ 2 · 超赞房东

下面是这个组件的搭建，由于这个组件在多个页面下都会被调用，因此在componen下封装出来

```

import PropTypes from 'prop-types'
import React, { memo } from 'react'

const RoomItem = memo((props) => {
  const { itemData } = props //从外部传入的数据在这里拿到

  return (
    <div>{itemData.name}</div> //展示
  )
})

RoomItem.propTypes = { //从外部传入进来的数据
  itemData: PropTypes.object
}

export default RoomItem

```

页面上的展示就可以这样写 `<RoomItem itemData={item} key={ item.id }/>`

由于这个组件我们在后台请求数据时，每一个房间是被一包裹起来，因此我们给每一个ul编辑classname为room-list 这样方便编写样式

```

export const HomeWrapper = styled.div`
  > .content {
    width: 1032px;
    margin: 0 auto;
  }

  .good-price {
    margin-top: 30px;

    .room-list {
      display: flex;
      flex-wrap: wrap; //一行不够自动换行
    }
  }

```

```
}
```

现在在页面上的布局已经写完了，现在回去编辑组件的样式

```
import PropTypes from 'prop-types'
import React, { memo } from 'react'
import { ItemWrapper } from './style'
const RoomItem = memo((props) => {
  const { itemData } = props //从外部传入的数据在这里拿到

  return (
    <ItemWrapper>
      <div className='inner'>
        <div className='cover'>
          <img src = {itemData.picture_url}/>
        </div>
        <div className='desc'>
          {itemData.verify_info.messages.join('.')}</div>
        <div className='name'>{itemData.name}</div>
        <div className='price'>¥{itemData.price}/晚</div>
      </div>
    </ItemWrapper> //展示
  )
})

RoomItem.propTypes = { //从外部传入进来的数据
  itemData: PropTypes.object
}

export default RoomItem
```

style的编辑不做赘述