

百度智客前端串讲

FE:Guo,Zhipeng

guozhipeng03@baidu.com

概要

- 业务简介
- 技术介绍
- 流程控制

从数据本身的角度结合视觉交互去帮助理解业务，
从而设计一种合适的可视化形态。

业务介绍

客流情况

主要维度： 时间, 顾客属性（新老顾客）
动作（到店时长, 到店频次, 店铺关联）

整场客流

楼层客流

店铺客流

客流趋势

平均进店率

到店时段

住店时长

业态客流占比

客流占比

客流趋势

门店关联

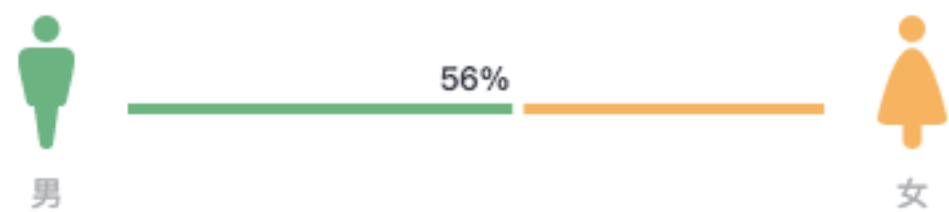
顾客动向

洞察顾客特征与偏好

顾客属性 2017/08/17-2017/08/23

时间: 昨天 近7天 近30天 2017/08/17 - 2017/08/23 | 顾客属性: 客流 到店顾客 新顾客 老顾客

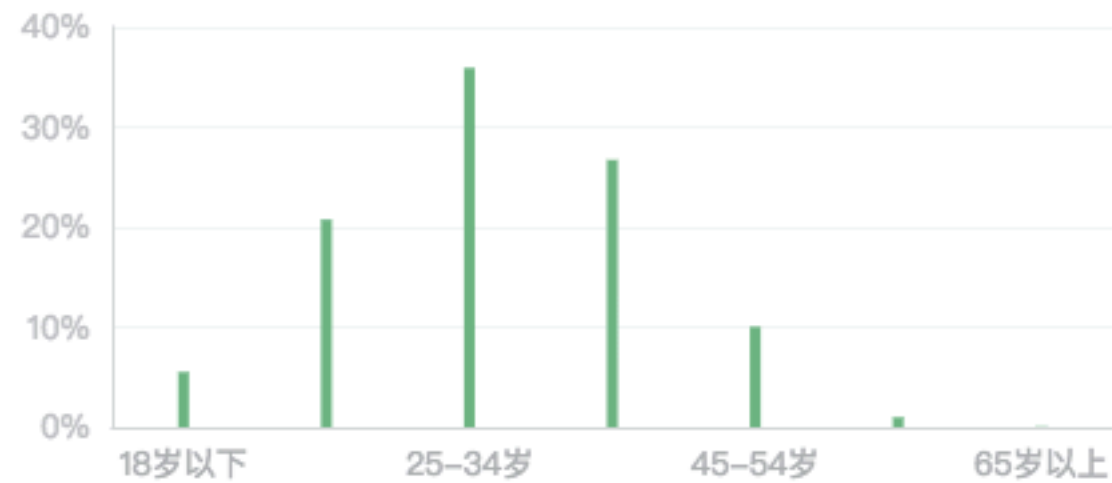
性别比例



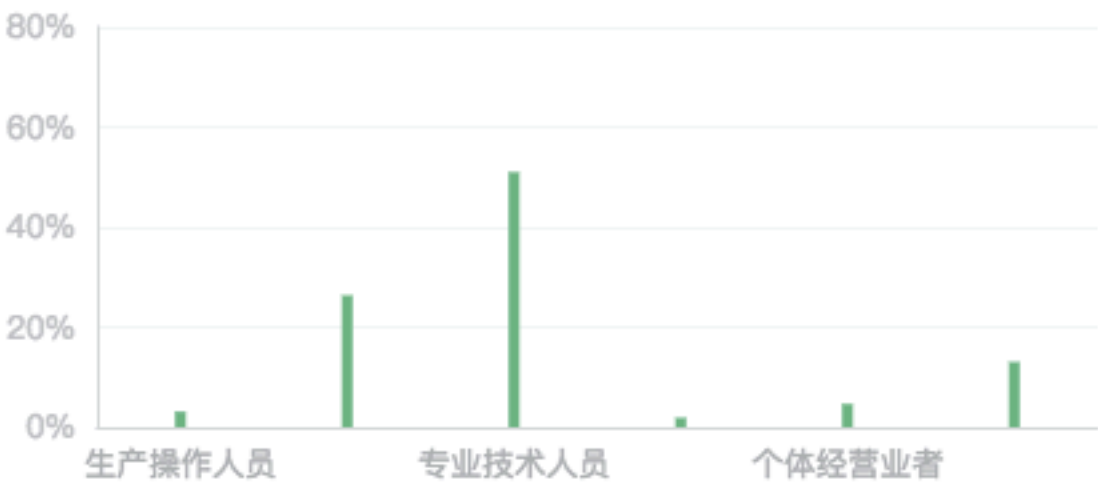
学历分布



年龄分布



职业分布



消费水平

私家车拥有情况

维度：

顾客属性（新老顾客，客流，到店顾客），
时间（昨天，近7天，30天，其他）

展示数据：

对性别，学历，有无车，职业，消费水平进行比较
兴趣爱好，搜索关注之类

基于地理位置数据，商圈客群分布

维度：

顾客属性（新老顾客，客流，到店顾客），
时间（昨天，近7天，30天，其他）
性别，年龄

商圈覆盖展示数据：

基于热力图以及客流占比以及渗透度数据展示

商圈洞察展示数据（基于距离）：

展示消费属性 人口属性

根据重合顾客的占比，来源，属性等进行对比

竞品: 万达广场万达广场万达广场 永旺超市 颐提港3

重合客流 重合客群来源 ⓘ 重合客群属性

客流重合度 ⓘ



26%

本项目占比



10%

重合度

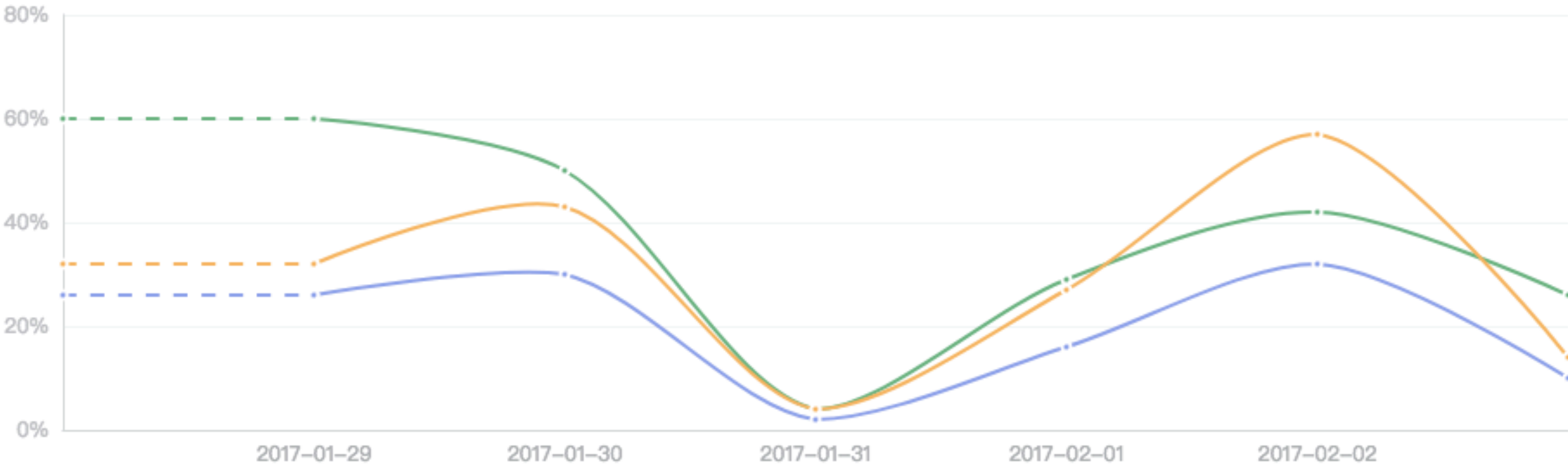


14%

竞品占比

历史趋势

● 本项目占比 ● 竞品占比 ● 重合度



会员分层

如果开通会员的话，会对会员的属性（兴趣，年龄等）进行分析
主要包括：位置分布，搜索关注，偏好分析，会员属性等分析

✓ 会员交易数据量（万条）

550.98

🕒 最后一次更新时间

2016/12/14

🎯 会员活跃度

低活跃

41.6%

150

一般活跃

30.5%

110

高活跃

28.0%

101



技术相关

framework

redux

react-redux

react



webpack

Babel

gulp

Mocha

项目采用了react 生态。整个网站的页面主要分为：首页、后台管理页。后台管理页以单页应用的形式实现，路由采用react-router。

主要介绍内容

一： 前后端交互数据的调用

二： 导航栏权限

前后端数据交互

1 initdata

2 ajax对api请求

前后端交互数据的调用

页面中的script标签

页面中指定的script标签的id为initData。

`<script id="pageinitData">var initData={};</script>`
php脚本需要将initData={}替换成对应的初始化数据。

```
<body>
  <div id="page_wrapper"><!-- html %></div>
  <script id="pageInitData">var initData={};</script>
  <!-- include('partial/ieCheck');%>
  <!-- include('partial/tongji');%>
  <!-- include('partial/userLog');%>
  <script src="/public/cas/api.js"></script>
  <script src="//api.map.baidu.com/api?v=2.0&ak=ZUbZ4H6pcayCtwsr0WQE1oGf&s=1"></script>
  <script src="/public/moment/moment-with-locales.min.js"></script>
  <script src="/public/heatmap/heatmap.min.js"></script>
  <script src="/public/echarts/echarts3.min.js"></script>
  <script src="/public/raphael/raphael.min.js"></script>
```

前后端交互数据的调用

(dev)启动服务 server.js

线上是php服务器端执行

express 会去处理对应get post 请求，并设置路由

Webpack 将page 下的所有react bundle到webroot

gulp会构建html 文件到webroot路径下

然后server.js 给script里面的initData变量的初始化

前后端交互数据的调用

```
// 替换initData, 并写回。
var fileData = fs.readFileSync(__dirname + '/webroot/' + filename + '.html', {encoding: 'utf8'});
var reg = new RegExp('initData=(.)*;', 'g');

var initData = requireUncached('./test/pageInitData/' + filename);

if (initData.getData) {
    req.session = session;
    initData = initData.getData(req);
}

// initData中的属性映射到实际的ajax接口
for (var key in initData) {
    if (fileExists('test/ajaxRequestData/' + key + '.js')) {
        var ajaxModule = requireUncached('./test/ajaxRequestData/' + key);
        if (ajaxModule.getData) {
            initData[key] = ajaxModule.getData(req);
        }
    }
}

// 将initData写到session中, 用于在ajax接口之间共享数据
session.initData = initData;
fileData = fileData.replace(reg, 'initData=' + JSON.stringify(initData));
res.send(fileData);
```

前后端交互数据的调用

替换initData

```
var fileData = fs.readFileSync(__dirname + '/webroot/' +  
    filename + '.html', {encoding: 'utf8'});
```

读取webroot里面的html文件
获取到script 标签下面的initdata变量

```
<script id="pageInitData">var initData={};</script>
```

前后端交互数据的调用

initData中的一些数据来自实际的ajax接口

例如"config/navConfig": [],
会通过server的ajaxMudule.getdata 请求到。
一并输入到变量initdata中

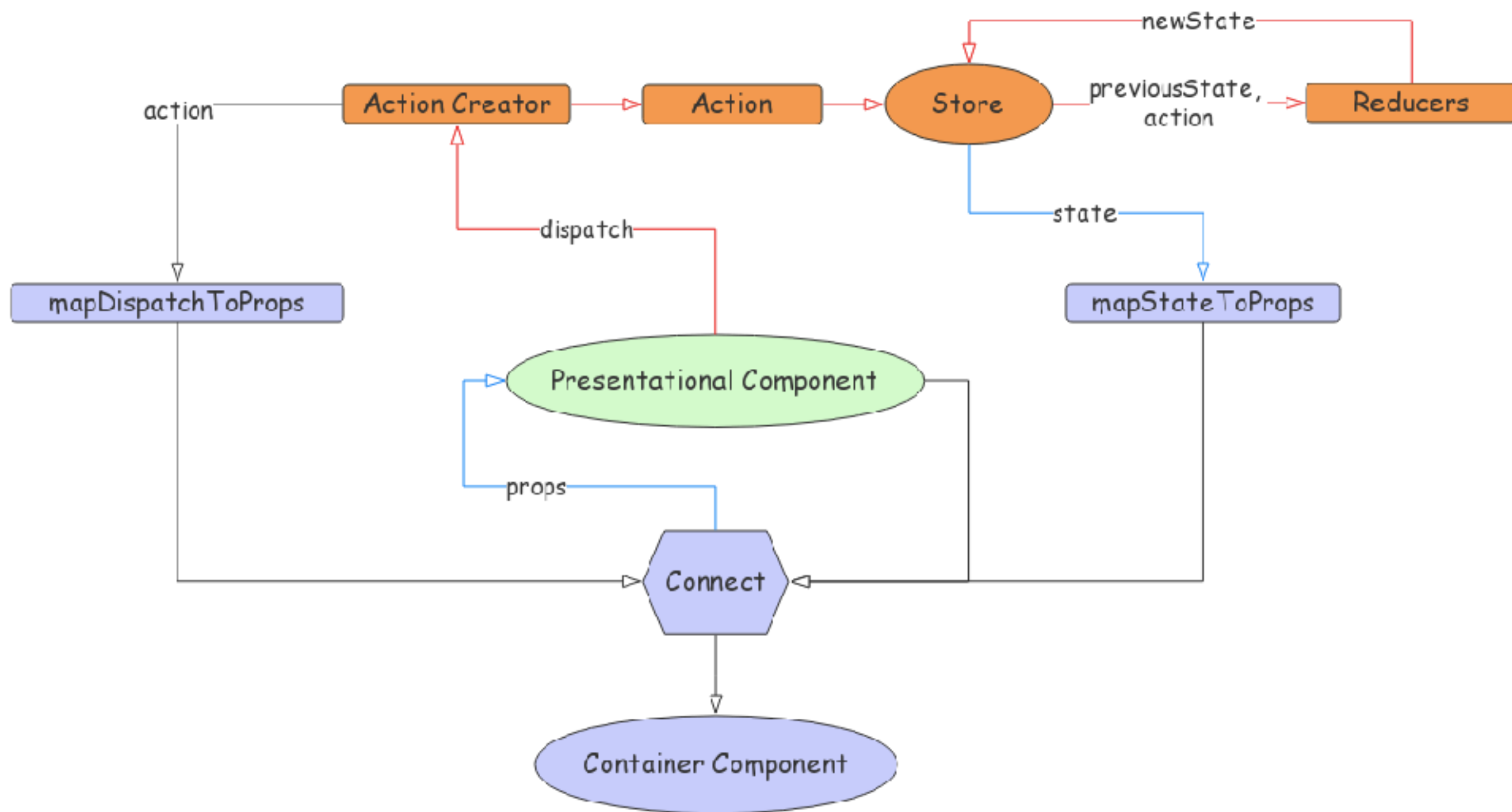
```
for (var key in initData) {  
    if (fileExists('test/ajaxRequestData/' + key + '.js')) {  
        var ajaxMudule = requireUncached('./test/ajaxRequestData/' + key);  
        if (ajaxMudule.getData) {  
            initData[key] = ajaxMudule.getData(req);  
        }  
    }  
}
```

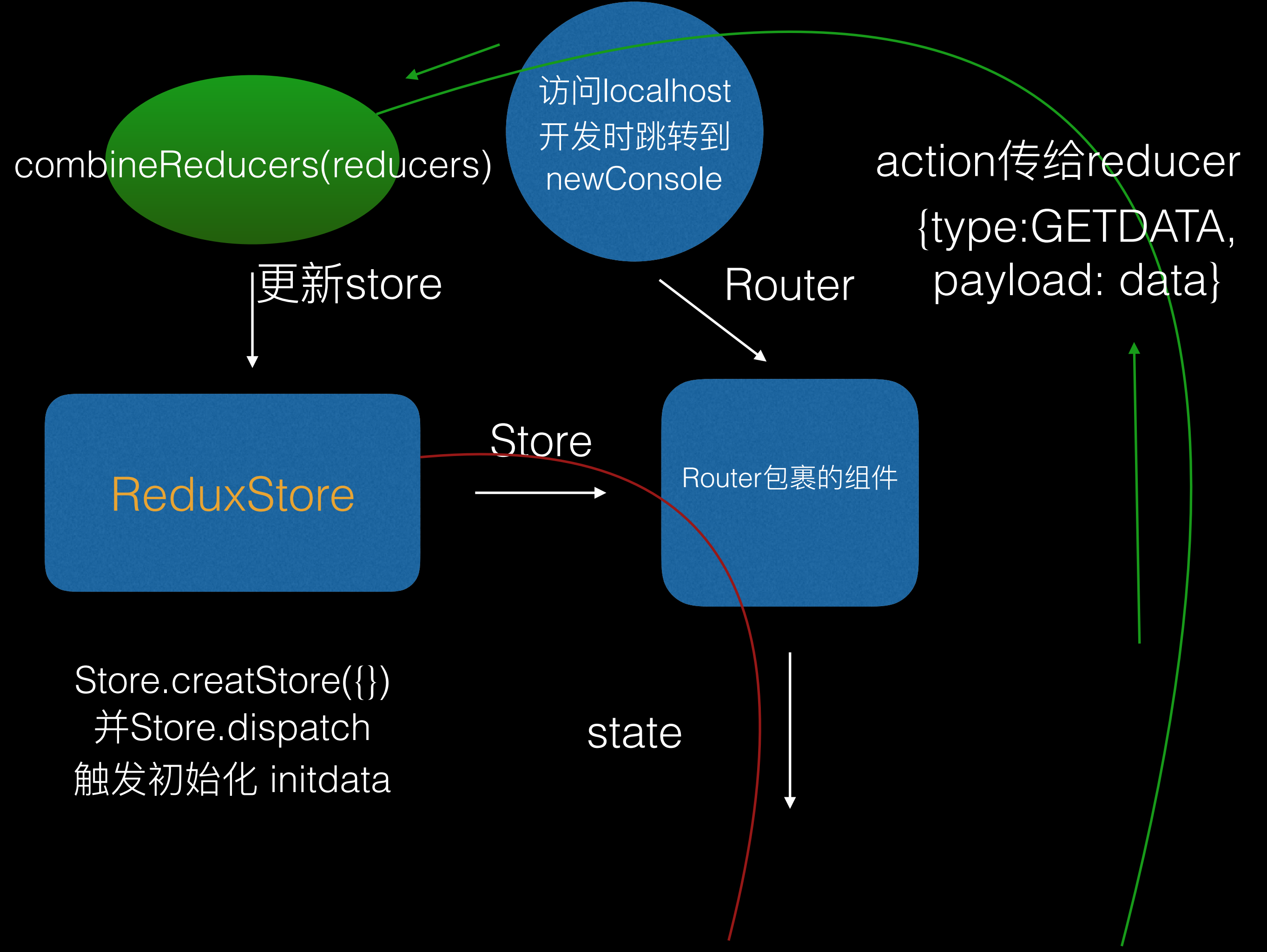
src/constants/InitData.js

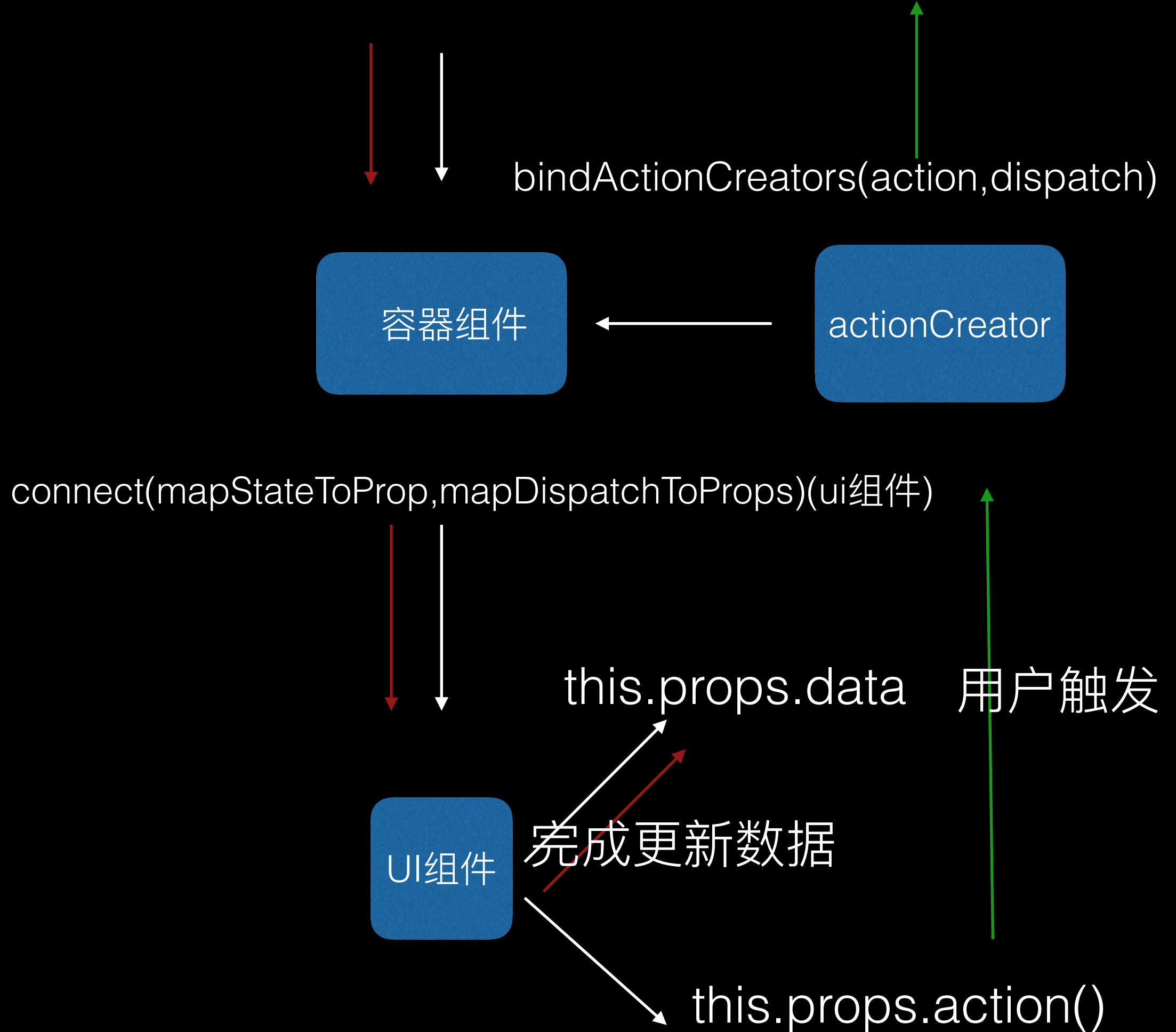
将全局变量initdata 获取到并输出到page下面的store

```
let data = {};  
if (typeof initData !== 'undefined') {  
  data = JSON.parse(JSON.stringify(initData));  
}  
  
function isPage(page) {  
  return location.href.indexOf(page) > -1;  
}  
  
if (process.env.TEST_API) {  
  let testApi = require('src/utis/testApi');  
  if (isPage('console')) {  
    if (typeof(data.accountInfo.extendAccountInfo) !== 'object') {  
      throw new Error('initData中需要提供extendAccountInfo对象');  
    }  
  }  
  for (let key in data) {  
    testApi.testData(key, data[key]);  
  }  
}
```

前后端交互数据的调用







这样基本的initdata就都进去啦

前后端交互数据的调用

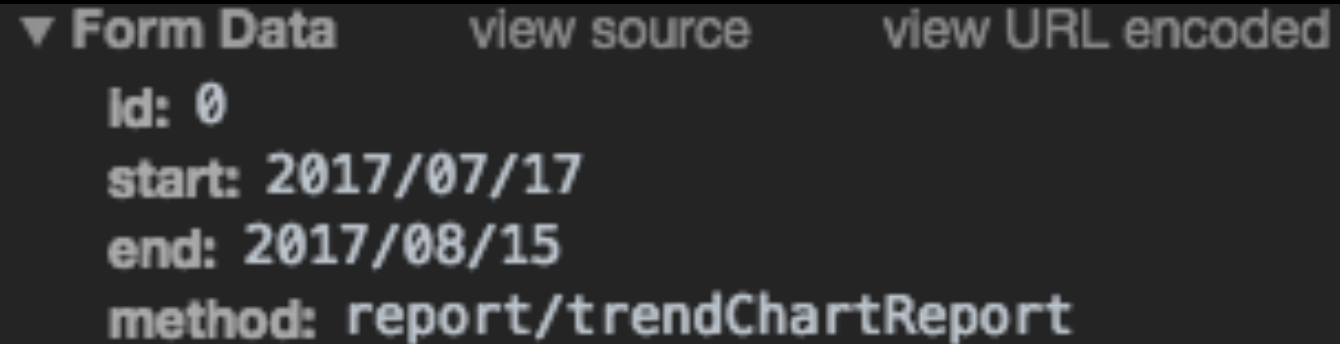
其他动态请求的数据（整场客流allstoreflow为例）

前后端交互数据的调用

Ajax请求api

格式:

```
//所有ajax请求指向stj / api, 该页面通过解析method来指向最终需要的方法
{
  method: string, // 具体的ajax方法, 如: "report / trendChartReport"
  param1: value1,
  param2: value2
}
```



▼ Form Data view source view URL encoded

Id: 0
start: 2017/07/17
end: 2017/08/15
method: report/trendChartReport

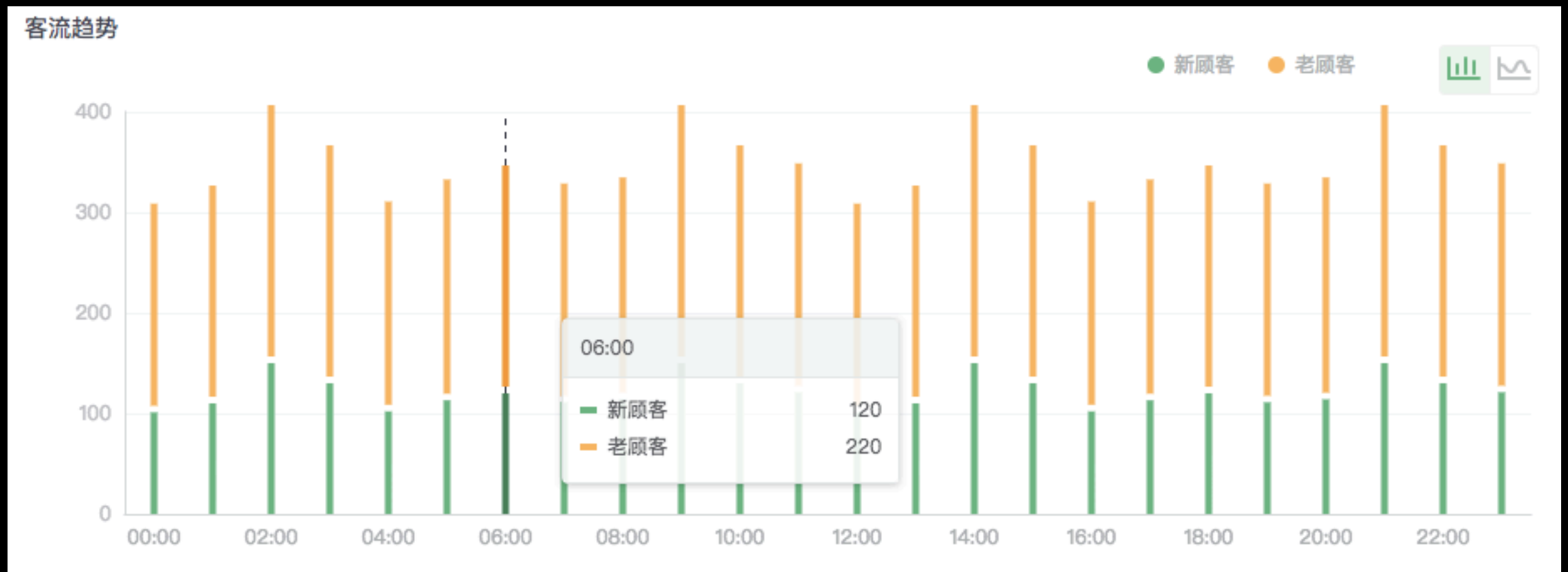
前后端交互数据的调用

```
this.props.onRefresh();
```

前后端交互数据的调用

客流趋势

<CustomerChart start={start} end={end} />



前后端交互数据的调用

Data

```
customerData: state.allStoreFlow.customerData
```

Action

```
onRefresh: bindActionCreators(actionCreators.getCustomerData, dispatch)
```

将refresh注入props

```
onRefresh: bindActionCreators(actionCreators.refresh, dispatch)
```

当组件挂载完成时候触发onfresh

```
componentDidMount() {  
  const {start, end, params} = this.props;  
  this.props.onRefresh({...params, start, end});  
}
```

action触发api 更新数据

```
export function getCustomerData(params = {}) {  
  return (dispatch, getState) => {  
    api.getCustomerData(params, function onSuccess(data) {  
      dispatch({  
        type: MALL.CUSTOMER_DATA_LOADED,  
        payload: data  
      });  
    });  
  });  
}
```


二： 导航栏权限功能

config/navconfig

```
{
  label: '整场',
  id: 'mall',
  children: [
    {label: '整体概览', id: 'mallOverview'},
    {
      id: 'mallFlow',
      label: '客流管理',
      isOpen: true,
      children: [
        {label: '整场客流', id: 'allStoreFlow'},
        {label: '楼层客流', id: 'floorFlow'},
        {label: '店铺客流', id: 'storeFlow'}
      ]
    },
    {
      id: 'mallCustomerAnalysis',
      label: '客群洞察',
      isOpen: true,
      children: [
        {label: '顾客属性', id: 'mallProperty'},
        {
```



整场



品牌



门店



会员



设置

根据navconfig 的 id加载nav

```
<Route key={config.id} path={path} children={props => {  
  let match = props.match;  
  if (match) {  
    return (  
      <SubPage url={path} navConfig={config.children} />  
    );  
  }  
  else {  
    return null;  
  }  
}
```



```
function loadPage(id) {  
  return {  
    mall: Mall,  
    brand: Brand,  
    store: Store,  
    vip: Vip,  
    settings: Settings  
  }[id];  
}
```

递归更新节点的label

有些节点的Label是由当前选中门店的行业决定的

```
export default createSelector(
  [getBaseNavConfig, getSelectedStore],
  (baseNavConfig = [], selectedStore = {}) => {
    let industry = selectedStore.industry;
    let clonedNavConfig = cloneDeep(baseNavConfig);

    clonedNavConfig.forEach(node => {
      setLabel(node, industry);
    });

    return clonedNavConfig;
  }
);
```

```
{
  label: {
    default: '到访时段',
    [CAN_YIN]: '到店时段',
    [MOVIE]: '到店时段',
    [LIN_SHOU]: '到店时段'
  },
  id: 'arrivePeriod'
},
```

创建一个privilegeMap [authority] = true

```
// 计算一个privilegeMap, 其中既包括后端返回的authorities, 也包括前端控制的一些权限, 如: vip
// 后端返回的authorities是由配置文件生成的
// 前端认为: 在用户的一次session间, authorities的数据不会发生改变
function getPrivilegeMap(authorities = []) {
    let privilegeMap = {};

    authorities.forEach(authority => {
        privilegeMap[authority] = true;
    });

    // // 如果有vipOverview权限, 则vip这个权限也应该有
    // if (privilegeMap.vipOverview) {
    //     privilegeMap.vip = true;
    // }

    return privilegeMap;
}
```

根据条件修改privilegeMap [authority]

```
// 根据当前的应用状态, 判断指定privilege是否可以访问
function isAuthorized(privilege, storeInfo = {}, initData = {}) {
  let {storeMap = {}, selectedStoreId} = storeInfo;
  let selectedStore = storeMap[selectedStoreId];

  // 非餐饮、电影、零售行业, 不能访问"门店>兴趣偏好"
  // 非餐饮、电影、零售行业的品牌, 不能访问"品牌>兴趣偏好", 品牌的行业也直接拿选中门店的行业即可
  if (privilege === 'interest' || privilege === 'brandInterest') {
    let industry = selectedStore && selectedStore.industry;
    if ([CAN_YIN, MOVIE, LIN_SHOU].indexOf(industry) < 0) {
      return false;
    }
  }
}
```

```
// 没有门店时，不能访问门店相关的报表
if (Object.keys(storeMap).length < 1) {
  const storeReports = [
    'overview', 'realtimeFlow', 'flowTrend', 'interest',
    'arriveFrequency', 'arrivePeriod', 'stayDuration',
    'property', 'searchReport', 'district', 'correlation'
  ];
  if (storeReports.indexOf(privilege) > -1) {
    return false;
  }
}

// 没有会员报告，不能访问设置页等子页面
if (initData['account/businessInfo'] && !initData['account/businessInfo'].hasVipReport) {
  if (privilege === 'vipSetting') {
    return false;
  }
}
}
```


根据privileges[navconfig.id] == true
添加hasprivileges属性 = true

```
function markHasPrivilege(nodeList, privileges) {  
  nodeList.forEach(node => {  
    if (privileges[node.id]) {  
      node.hasPrivilege = true;  
    }  
    node.children && markHasPrivilege(node.children, privileges);  
  });  
}
```

递归过滤掉节点

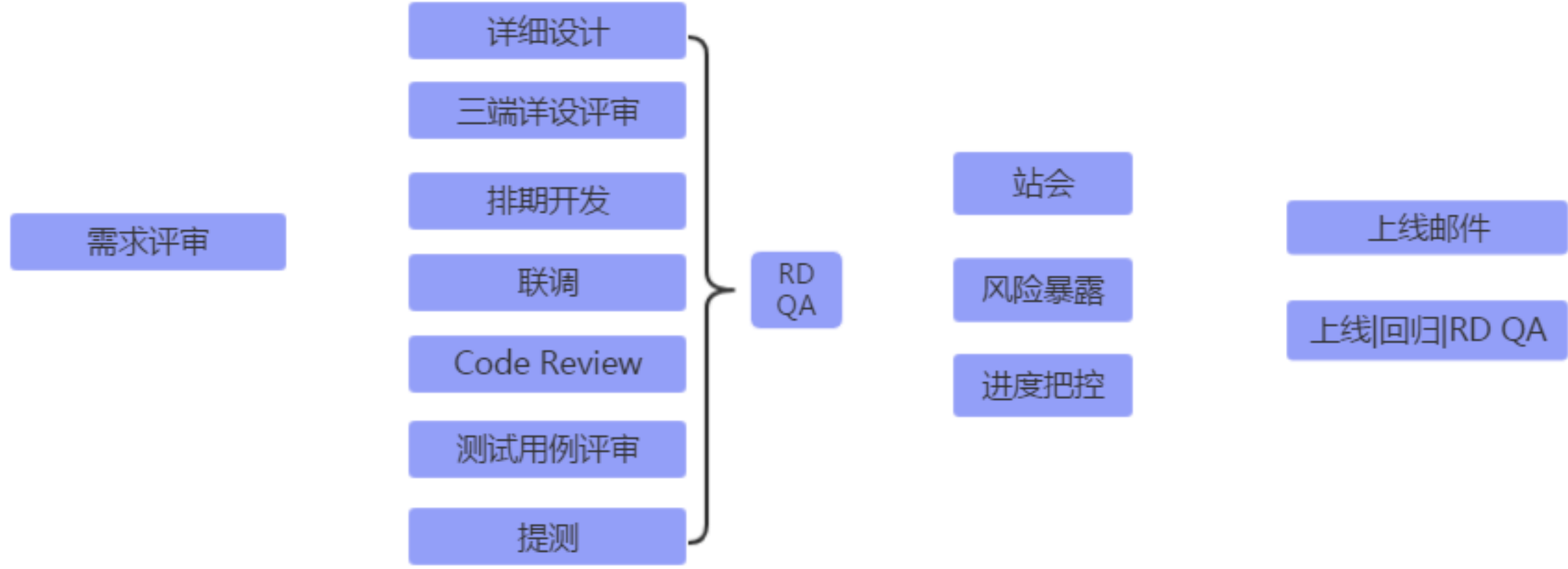
```
// 递归地修改配置，删掉没有权限的节点
function filterPrivilegedNodes(nodeList) {
  return nodeList.filter(node => {
    if (node.children) {
      node.children = filterPrivilegedNodes(node.children);
    }
    return shouldKeep(node);
  });
}

// 是否保留节点
// 1. 节点本身有权限
// 2. 节点的某个子节点有权限
function shouldKeep(node) {
  if (node.hasPrivilege) {
    return true;
  }

  if (node.children) {
    return node.children.some(childNode => shouldKeep(childNode));
  }

  return false;
}
```

开发流程



评审 对pm的需求，功能进行讨论明确对方目的，和设计逻辑

详设 一个story中具体的功能，根据功能设计相应接口，对应api请求格式，接口测试数据，设计功能相关的组件，主要是前后端对接口的统一，以及明确要实现的功能以及思路。。

排期开发 根据详设 排期开发 及时反馈 风险

提测 沙盒PM验收 功能验证，自动化回归测试，集体评审上线步骤

上线 QA验证 + RD帐号类型验证，出现问题回滚修改完上线PM验收

Thanks

Q&A

欢迎批评指正