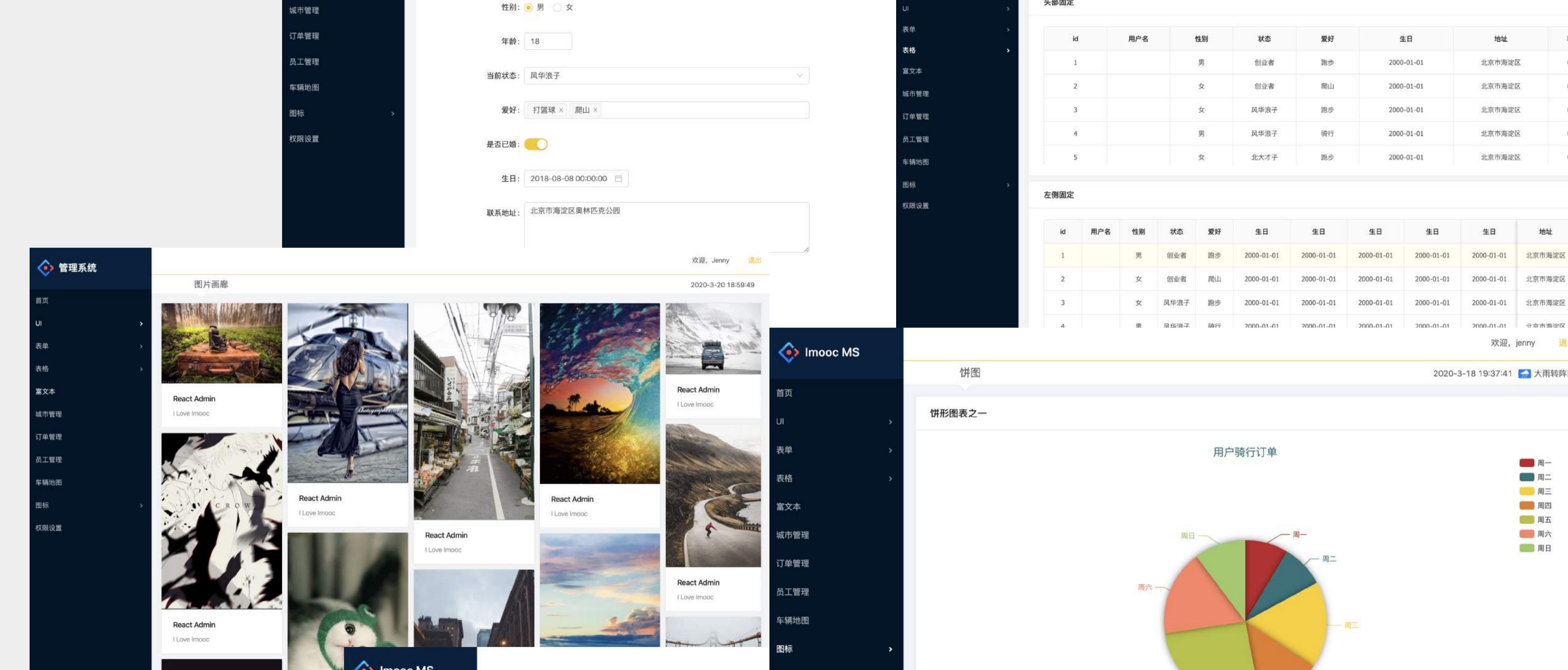


# 大家好，我是杨静怡

我是一名前端工程师，爱好coding和一切有逻辑的事物



## 共享单车后台管理系统

采用React+AntD框架开发的一套共享单车后台管理系统，功能特色有Echarts实现常用图表功能，百度Map SDK实现地图可视化，AntD实现轮播图、图片画廊、基础表单和表格等的使用，react-draft-wysiwyg插件实现富文本编辑器，React-Router4.0实现路由处理，axios、JsonP、天气调度、日期化、基础表单表格等公共机制的封装达到项目工程化的目的。

## 外卖APP

项目主要展示了外卖APP的商家模块，包括商品、评论、商家介绍3个部分，主要技术栈Vue.js，采用vue-cli脚手架搭建，并使用vue-resource进行前后端数据交互，vue-router实现路由，flex实现弹性布局，better-scroll插件解决滚动需求，html5 localStorage实现收藏功能等。

## 静态资源服务器

实现了一个简单的静态资源服务器，在终端通过anydoor 指令可以把当前文件夹作为静态资源服务器根目录。此项目使用了fs、buffer、http、path 等Node.js最核心 API，处理了压缩文件，range范围请求，缓存等，体验了一个项目从初始化到开发规范到发布上线的完整流程。

## 新冠HTML5动态数据报告

运用HTML5、CSS3、JS、jQuery流行技术，面向对象的编程思想，采用组件式开发模式，开发的Web App全站的HTML5动态数据报告。项目运用了FullPage.js实现全屏滚动，Canvas基本绘图API绘制了柱状图、饼状图、散状态、折线图、雷达图等。

## 微信小程序文影馆

从0开始构建了一个集阅读和电影资讯为一体的小程序。该项目运用了swiper轮播图组件、wx.setStorageSync数据缓存实现收藏功能、wx.showActionSheet实现底部分享、onShareAppMessage实现用户点击右上角转发、wx.playBackgroundAudio播放音乐、template嵌套模版、通过仿 lodash API请求电影数据、wx.onPullDownRefresh监听用户下拉刷新触发请求数据、scroll-view组件的bindscrolltolower事件监听滚动到底部时触发请求更多数据、声明类class Movie的属性和方法后new实例并调用实例的方法，达到组件化开发和模块化编程。

# 共享单车后台管理系统

## 项目介绍

在日常的开发中，很多应用都需要一套后台管理系统来管理数据。于是结合“共享单车”，开发了一套通用后台管理系统，涵盖了React/Redux/Router/Redux/antd/Baidu Map/Echarts/Webpack等技术。



## 技术细节

### 跨域请求封装

json函数封装了天气调用，返回promise对象，resolve时使用jsonp插件进行跨域请求天气api，reject时处理错误。慕课网原项目是请求百度天气api，但是百度天气api目前只能应用于后端，对浏览器端不开放，故我用的是聚合数据的api，缺点是只能显示苏州的天气。

ajax函数封装了所有假数据的请求，同样返回promise对象，resolve时axios插件进行跨域请求假数据，reject时处理错误。本项目所有模拟数据都是通过easymock网站随机生成的。



### React Router路由

首先`<switch>`表示只渲染出第一个与当前访问地址匹配的`<Route>`，其次，`<Route>`的书写顺序很重要，`path="/"`要写在最后，写在前面的话永远只会匹配到当前地址，之后的地址不会匹配到。

```
<HashRouter>
  <App>
    <Switch>
      <Route path="/login" component={Login}/>
      <Route path="/common" render={()=>
        <Common>
          <Route path="/common/order/detail/:orderId" component={OrderDetail} />
        </Common>
      }/>
      <Route path="/" render={()=>
        <Admin>
          <Switch>
            <Route path="/home" component={Home} />
            <Route path="/ui/buttons" component={Buttons} />
            <Route path="/ui/loading" component={Loadings} />
            <Route path="/ui/messages" component={Messages} />
            <Route path="/ui/tabs" component={Tabs} />
            <Route path="/ui/gallery" component={Gallery} />
            <Route path="/ui/carousel" component={Carousel} />
            <Route path="/form/login" component={FormLogin} />
            <Route path="/form/reg" component={FormRegister} />
            <Route path="/table/basic" component={BasicTable} />
            <Route path="/table/highlight" component={HighTable} />
            <Route path="/rich" component={Rich} />
            <Route path="/city" component={City} />
            <Route path="/order" component={Order} />
            <Route path="/bikeMap" component={BikeMap} />
            <Route path="/user" component={User} />
            <Route path="/charts/bar" component={Bar} />
            <Route path="/charts/pie" component={Pie} />
            <Route path="/charts/line" component={Line} />
            <Route path="/permission" component={Permission} />
            <Redirect to="/home" />
          </Switch>
        </Admin>
      }/>
    </Switch>
  </App>
</HashRouter>
```

```
// 涂染地图
renderMap = (res) => {
  let list = res.routes.list;
  this.map = new window.BMap.Map("container", {enableMapClick: false});
  let point1 = list[0].split(',');
  let startPoint = new window.BMap.Point(gps1[0], gps1[1]);
  let gps2 = list[list.length - 1].split(',');
  let endPoint = new window.BMap.Point(gps2[0], gps2[1]);

  this.map.centerAndZoom(endPoint, 11);
  // map.clearOverlays();
  // 清除图例
  let startIcon = new window.BMap.Icon("./assets/start_point.png", new window.BMap.Size(36, 42), {
    imageSize: new window.BMap.Size(36, 42),
    anchor: new window.BMap.Size(18, 42)
  });

  var bikeMarkerStart = new window.BMap.Marker(startPoint, { icon: startIcon });
  this.map.addOverlay(bikeMarkerStart);

  let endPointIcon = new window.BMap.Icon("./assets/end_point.png", new window.BMap.Size(36, 42), {
    imageSize: new window.BMap.Size(36, 42),
    anchor: new window.BMap.Size(18, 42)
  });
  var bikeMarkerEnd = new window.BMap.Marker(endPoint, { icon: endPointIcon });

  let routeList = [];
  list.forEach(item=>{
    let p = item.split(',');
    let point = new window.BMap.Point(p[0], p[1]);
    routeList.push(point);
  });
  // 行驶路线
  var polyline = new window.BMap.Polyline(routeList, {
    strokeColor: "#ef4136",
    strokeWeight: 3,
    strokeOpacity: 1
  });
  this.map.addOverlay(polyline);
  // 服务点路线
  let serviceList = res.service_list;
  let servicePointList = [];
  serviceList.forEach(item=> {
    let point = new window.BMap.Point(item.lon, item.lat);
    servicePointList.push(point);
  });
  // 画线
  var polyServiceLine = new window.BMap.Polyline(servicePointList, {
    strokeColor: "#ef4136",
    strokeWeight: 3,
    strokeOpacity: 1
  });
  this.map.addOverlay(polyServiceLine);
  // 添加地图中的自行车
  let bikeIcon = new window.BMap.Icon("./assets/bike.jpg", new window.BMap.Size(36, 42), {
    imageSize: new window.BMap.Size(36, 42),
    anchor: new window.BMap.Size(18, 42)
  });
  let bikeList = res.bike_list;
  let bikeIcon = new window.BMap.Icon("./assets/bike.jpg", new window.BMap.Size(36, 42), {
    imageSize: new window.BMap.Size(36, 42),
    anchor: new window.BMap.Size(18, 42)
  });
  bikeList.forEach(item=> {
    let p = item.split(',');
    let point = new window.BMap.Point(p[0], p[1]);
    var bikeMarker = new window.BMap.Marker(point, { icon: bikeIcon });
    this.map.addOverlay(bikeMarker);
  });
  // 添加地图控件
  this.addMapControl();
};

// 商用地图组件
addMapControl = () => {
  let map = this.map;
  // 左上角, 添加比例尺
  var top_right_control = new window.BMap.ScaleControl({anchor: window.BMAP_ANCHOR_TOP_RIGHT});
  var top_right_navigation = new window.BMap.NavigationControl({anchor: window.BMAP_ANCHOR_TOP_RIGHT});
  // 地图缩放
  map.addControl(top_right_control);
  map.addControl(top_right_navigation);
  map.enableScrollWheelZoom(true);
  // 地图添加图例
  legend.addLegend(map);
};
```

```
1 import React from 'react';
2 import { Card } from 'antd';
3 import ReactEcharts from 'echarts-for-react';
4 import echarts from './echartTheme'
5 // 引入echarts主题
6 // 引入Echarts主模块
7 import echarts from 'echarts/lib/echarts';
8 // 引入饼图折线图
9 import 'echarts/lib/chart/bar';
10 import 'echarts/lib/chart/pie';
11 import 'echarts/lib/chart/tooltip';
12 import 'echarts/lib/component/title';
13 import 'echarts/lib/component/legend';
14 import 'echarts/lib/component/markPoint';
15 import defaultClass from 'echarts';
16 export default class Bar extends React.Component {
17   state={
18     componentWillMount(){
19       echarts.registerTheme('imooc',echartTheme);
20     }
21   }
22   getOption(){
23     let option = {
24       title: {
25         text: '用户骑行订单'
26       },
27       tooltip : {
28         trigger: 'axis'
29       },
30       xAxis: {
31         data: [
32           '周一',
33           '周二',
34           '周三',
35           '周四',
36           '周五',
37           '周六',
38           '周日'
39         ]
40       },
41       yAxis: {
42         type: 'value'
43       },
44       series: [
45         {
46           name: '订单量',
47           type: 'bar',
48           data: [
49             1000,
50             2000,
51             1500,
52             3000,
53             2000,
54             1200,
55             800
56           ]
57         }
58       ]
59     }
60     return option;
61   }
62   getOption2(){
63   }
64   render(){
65     return (
66       <div>
67         <Card title="柱形图表之">
68           <ReactEcharts option={this.getOption()} theme="Imooc" notMerge={true} lazyUpdate={true} style={{ height: 500 }} />
69         </Card>
70         <Card title="柱形图表之二" style={{marginTop:10}}>
71           <ReactEcharts option={this.getOption2()} theme="Imooc" notMerge={true} lazyUpdate={true} style={{ height: 500 }} />
72         </Card>
73       </div>
74     );
75   }
76 }
77 export default Form.create()(FormLogin);
```

### 百度Map的调用

在百度地图开放平台申请应用AK后即可调用百度Map api。使用的时候先通过index.html中src引入，即可参考文档写代码，注意由于是通过html src来引入第三方库，而我们的项目是在模块中调用，因此api使用前都要加window。

### Echarts的使用

可以在Echarts官网自定义主题下载后再echarts.registerTheme来应用到项目来。

```
1 import React from 'react';
2 import { Card, Form, Input, Button, message, Icon, Checkbox } from "antd";
3 const FormItem = Form.Item;
4 export default class FormLogin extends React.Component{
5   formRef = React.createRef();
6   handleSubmit = ()=>{
7     let userInfo = this.form.current.getFieldsValue();
8     console.log(JSON.stringify(userInfo))
9     message.success(`$userInfo.userName` 恭喜你，您通过本次表单组件学习，当前密码为: ${userInfo.userPwd}`);
10   }
11   render(){
12     return (
13       <div>
14         <Card title="登录行内表单">
15           <Card title="登录水平表单" style={{marginTop:10}}>
16             <Form ref={this.formRef}>
17               <FormItem initialValue="" rules={[{ required:true, message:'用户名不能为空' }, { min:5,max:10, message:'长度不在范围内' }, { pattern:new RegExp(`^\\w+$`), message:'用户名必须为字母或者数字' } ]}>
18                 <Input prefix=<Icon type="user"/> placeholder="请输入用户名" />
19               </FormItem>
20             <FormItem>
21               <Input type="password" placeholder="请输入密码" />
22             </FormItem>
23             <Button type="primary" onClick={this.handleSubmit}>登录</Button>
24           </Form>
25         </Card>
26       </div>
27     );
28   }
29 }
```

# 外卖APP

## 项目介绍

项目主要展示了外卖APP的商家模块，包括商品、评论、商家介绍3个部分。主要技术栈Vue.js，采用vue-cli脚手架搭建，并使用vue-resource进行前后端数据交互，vue-router实现路由，flex实现弹性布局，better-scroll插件解决滚动需求，html5 localStorage实现收藏功能等。



## 技术细节

```
prod.server.js
```

```
1 var express = require('express');
2 var config = require('./config/index');
3
4 var port = process.env.PORT || config.build.port;
5
6 var app = express();
7
8 var router = express.Router();
9
10 router.get('/', function (req, res, next) {
11   req.url = '/index.html';
12   next();
13 });
14
15 app.use(router);
16
17 var appData = require('./data.json');
18 var seller = appData.seller;
19 var goods = appData.goods;
20 var ratings = appData.ratings;
21
22 var apiRoutes = express.Router();
23
24 apiRoutes.get('/seller', function (req, res) {
25   res.json({
26     errno: 0,
27     data: seller
28   });
29 });
30
31 apiRoutes.get('/goods', function (req, res) {
32   res.json({
33     errno: 0,
34     data: goods
35   });
36 });
37
38 apiRoutes.get('/ratings', function (req, res) {
39   res.json({
40     errno: 0,
41     data: ratings
42   });
43 });
44
45 app.use('/api', apiRoutes);
46
47 app.use(express.static('./dist'));
48
49 module.exports = app.listen(port, function (err) {
50   if (err) {
51     console.log(err);
52   }
53   console.log('Listening at http://localhost:' + port + '\n');
54 });


```

```
devServer.js
```

```
13 before(app) {
  app.get('/api/seller', function(req, res) {
    res.json({
      errno: 0,
      data: seller
    })
  });
  app.get('/api/goods', function(req, res) {
    res.json({
      errno: 0,
      data: goods
    })
  });
  app.get('/api/ratings', function(req, res) {
    res.json({
      errno: 0,
      data: ratings
    })
  });
},
```

## mock数据

1. 使用express请求本地data.json文件获取到模拟数据。
2. 编写webpack本地服务器的路由及相应接口。

```
border-1px($color)
position: relative
&:after
  display: block
  position: absolute
  left: 0
  bottom: 0
  width: 100%
  border-top: 1px solid $color
  content: ''
```

```
@media (-webkit-min-device-pixel-ratio: 1.5), (min-device-pixel-ratio: 1.5)
.border-1px
&:after
  -webkit-transform: scaleY(0.7)
  transform: scaleY(0.7)

@media (-webkit-min-device-pixel-ratio: 2), (min-device-pixel-ratio: 2)
.border-1px
&:after
  -webkit-transform: scaleY(0.5)
  transform: scaleY(0.5)
```

## 移动端1px边框问题

问题背景：因为这是一个 webapp 应用，在手机端调试时会发现 1 像素边框有时候并不精确等于 1 像素，这是和手机的设备像素比 (dpr) 有关。

设计思路：通过“border-1px”的伪类 after，描绘一条下1px线（写在 mixin.styl 文件中），再根据手机 dpr 的不同对线进行缩放，从而达到1 像素边框的设置。

```
<transition name="fade">
  <div v-show="detailShow" class="detail">
    <div class="detail-wrapper clearfix"> ...
    </div>
    <div class="detail-close" @click="hideDetail">
      <i class="icon-close"></i>
    </div>
  </div>
</transition>
```

```
&.fade-enter-active, &.fade-leave-active
  transition: all 0.5s
&.fade-enter, &.fade-leave-active
```



## vue-resource处理请求

vue-resource 是 Vue.js 的一款插件，它可以通过 XMLHttpRequest 或 JSONP 发起请求并处理响应。也就是说，它可以用于处理前后端数据请求、数据交互，达到与 \$.ajax 同等功能。请求写在生命周期钩子函数 created 里。

## 进入/离开过渡

Vue 提供了 transition 的封装组件，在下列情形中，可以给任何元素和组件添加进入/离开过渡：

1. 条件渲染 (使用 v-if)
2. 条件展示 (使用 v-show)
3. 动态组件
4. 组件根节点

一般只需要定义以下4个类的CSS即可

1.v-enter-active进入的状态 transition

2.v-leave-active离开的状态 transition

3.v-enter + v-leave-to消失的状态

## css sticky footer

所谓“Sticky Footer”，指的就是一种网页效果：如果页面内容不足够长时，页脚固定在浏览器窗口的底部；如果内容足够长时，页脚固定在页面的最底部。但如果网页内容不够长，置底的页脚就会保持在浏览器窗口底部。

### CSS规则

- 1.detail-wrapper要min-height: 100%
- 2.detail-wrapper和detail-close是重叠的，设置detail-main的padding-bottom等于detail-close的高度，设置detail-close的margin-top等于本身高度的负数

## better-scroll插件解决滚动需求

better-scroll 是一款重点解决移动端（已支持PC）各种滚动场景需求的插件。它是基于原生JS 实现的，不依赖任何框架。它编译后的代码大小是 63kb，压缩后是 35kb，gzip 后仅有 9kb，是一款非常轻量的 JS lib。

使用方法 new 初始化，on 监听当前实例上的自定义事件，scrollToElement(el, time, offsetX,

offsetY, easing) 滚动到指定的目标元素。

vue 中的异步操作（比如 this.\$http.get，包括其中执行的 \_initScroll()）不能马上映射到DOM，所以要把 \_initScroll() 放在 \$nextTick() 里。

## 组件通信

### 1.父组件向子组件传递数据通过 prop组件

### 2.子组件向父组件传递数据通过 events (\$emit)

### 3.ref 也可以访问组件实例

### 4.localStorage

```
<food @add="addFood" :food="selectedFood" ref="food"></food>
```

```
food.vue
```

```
export default {
  props: {
    food: {
      type: Object
    }
  },
},
```

```
cartcontrol.vue
```

```
<div class="cartcontrol">
  <div class="inner">
    <span class="decrease" v-show="food.count > 0" @click.stop.prevent="decreaseCart">-</span>
    <span class="count" v-show="food.count > 0"><{food.count}></span>
    <span class="add" @click.stop.prevent="addCart">+</span>
  </div>
</div>
```

```
<div class="cartcontrol-wrapper">
  <cartcontrol @add="addFood" :food="food"></cartcontrol>
</div>
addFood(target) {
  this._drop(target);
},
```

```
goods.vue
```

```
<food @add="addFood" :food="selectedFood" ref="food"></food>
```

```
food.vue
```

```
methods: {
  show() {
    this.showFlag = true;
    this.selectType = ALL;
    this.onlyContent = true;
    this.$nextTick(() => {
      if (!this.scroll) {
        this.scroll = new BScroll(this.$refs.food, {
          click: true
        });
      } else {
        this.scroll.refresh();
      }
    });
  }
},
```

```
export function saveToLocal(id, key, value) {
  let seller = window.localStorage.__seller__;
  if (!seller) {
    seller = {};
    seller[id] = {};
  } else {
    seller = JSON.parse(seller);
    if (!seller[id]) {
      seller[id] = {};
    }
  }
  seller[id][key] = value;
  window.localStorage.__seller__ = JSON.stringify(seller);
};

export function loadFromLocal(id, key, def) {
  let seller = window.localStorage.__seller__;
  if (!seller) {
    return def;
  }
  seller = JSON.parse(seller)[id];
  if (!seller) {
    return def;
  }
  let ret = seller[key];
  return ret || def;
};
```

# 静态资源服务器

## 项目介绍

实现了一个简单的静态资源服务器，在终端通过anydoor 指令可以把当前文件夹作为静态资源服务器根目录。此项目使用了 fs、buffer、http、path 等Node.js最核心 API，处理了压缩文件，range范围请求，缓存等，体验了一个项目从初始化到开发规范到发布上线的完整流程。



## 技术细节

```
compress.js
1 const {createGzip, createDeflate} = require('zlib');
2
3 module.exports = (rs, req, res) => {
4   const acceptEncoding = req.headers['accept-encoding'];
5   if (!acceptEncoding || !acceptEncoding.match(/\b(gzip|deflate)\b/)) {
6     return rs;
7   } else if (acceptEncoding.match(/\bgzip\b/)) {
8     res.setHeader('Content-Encoding', 'gzip');
9     return rs.pipe(createGzip());
10  } else if (acceptEncoding.match(/\bdeflate\b/)) {
11    res.setHeader('Content-Encoding', 'deflate');
12    return rs.pipe(createDeflate());
13  }
14};
15
```

## 压缩文件

采用了Node.js中的zlib模块封装了压缩文件的功能。根据请求首部Accept-Encoding的值，将http响应进行相应的压缩。

```
range.js
1 module.exports = (totalSize, req, res) => {
2   const range = req.headers['range'];
3   //Range 是一个请求首部，告知服务器返回文件的哪一部分, Range: bytes=200-1000, 2000-6576, 19000-
4   if (!range) {
5     return {code: 200};
6   }
7
8   const sizes = range.match(/bytes=(\d*)-(\d*)/);
9   const end = sizes[2] ? parseInt(sizes[2]) : totalSize - 1;
10  const start = sizes[1] ? parseInt(sizes[1]) : totalSize - end;
11
12  if (start > end || start < 0 || end > totalSize) {
13    return {code: 200};
14  }
15
16  res.setHeader('Accept-Ranges', 'bytes');
17  res.setHeader('Content-Range', `bytes ${start}-${end}/${totalSize}`);
18  res.setHeader('Content-Length', end - start);
19  return {
20    code: 206,
21    start: start,
22    end: end
23  };
24};
25
```

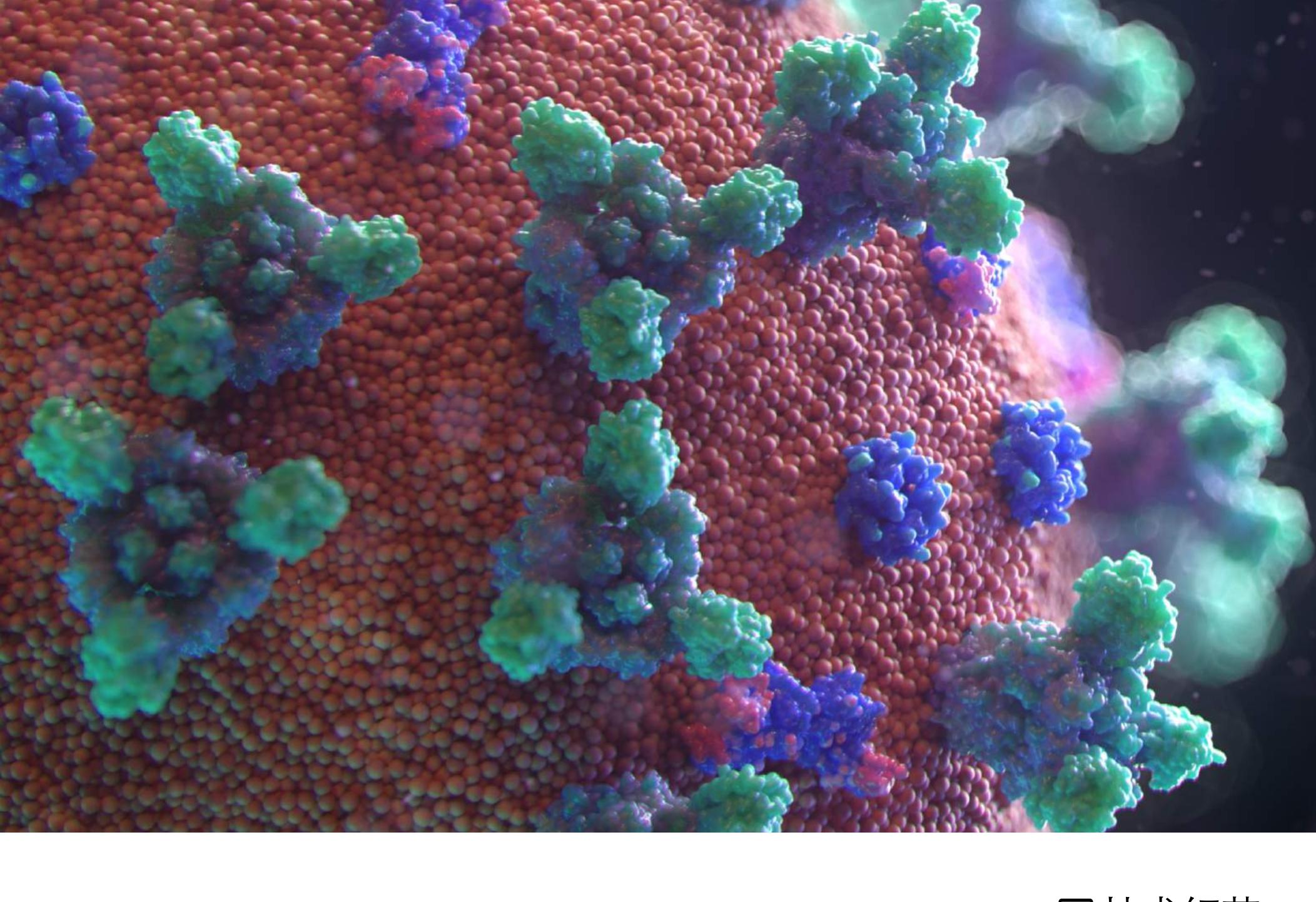
## range范围请求

根据请求Range的值，告知服务器返回文件的哪一部分。如果是部分返回，则状态码是206，其他状态码是200。

```
cache.js
1 const {cache} = require('../config/defaultConfig');
2
3 function refreshRes(stats, res) {
4   const {maxAge, expires, cacheControl, lastModified, etag} = cache;
5
6   if (expires) {
7     res.setHeader('Expires', (new Date(Date.now() + maxAge * 1000)).toUTCString());
8     //Expires: Wed, 21 Oct 2015 07:28:00 GMT
9   }
10
11   if (cacheControl) {
12     res.setHeader('Cache-Control', `public, max-age=${maxAge}`);
13   }
14
15   if (lastModified) {
16     res.setHeader('Last-Modified', stats.mtime.toUTCString());
17     //stats.mtime表明上次修改此文件的时间戳
18   }
19
20   if (etag) {
21     res.setHeader('ETag', `${stats.size}-${stats.mtime.toUTCString()}`);
22     // mtime 需要转成字符串，否则在 windows 环境下会报错
23   }
24 }
25
26 module.exports = function isFresh(stats, req, res) {
27   refreshRes(stats, res);
28
29   const lastModified = req.headers['if-modified-since'];
30   const etag = req.headers['if-none-match'];
31
32   if (!lastModified && !etag) {
33     return false;
34   }
35
36   if (lastModified && lastModified !== res.getHeader('Last-Modified')) {
37     return false;
38   }
39
40   if (etag && etag !== res.getHeader('ETag')) {
41     return false;
42   }
43
44   return true;
45};
```

## 缓存

用户发出请求后，首先看本地是否有缓存，如果有缓存，则通过Expires或者Cache-Control:max-age的响应头来看本地缓存是否失效，如果已失效，再通过If-Modified-Since请求头的值和Last-Modified响应头的值是否相等或者If-None-Match请求头的值和Etag响应头的值是否相等，如果相等说明文件未改变，状态码返回304，可以使用本地缓存文件。



## 项目介绍

运用HTML5、CSS3、JS、jQuery流行技术，面向对象的编程思想，采用组件式开发模式，开发的Web App全站的HTML5动态数据报告。项目运用了FullPage.js实现全屏滚动，Canvas基本绘图API绘制了柱状图、饼状图、散状态、折线图、雷达图等。

## 技术细节

```
this.loader = function(firstPage) {
  this.el.fullpage({
    onLeave: function(index, nextIndex, direction) {
      $(this).find('.h5_component').trigger('onLeave');
    },
    afterLoad: function(anchorLink, index) {
      $(this).find('.h5_component').trigger('onLoad');
    }
  });
  this.page[0].find('.h5_component').trigger('onLoad');
  this.el.show();
  if(firstPage){
    $.fn.fullpage.moveTo(firstPage);
    // $.fn.fullpage.moveTo(section, slide) 滚动到
  }
};
```

## FullPage.js实现全屏滚动

通过调用本FullPage.js插件可创建全屏滚动网站(也称为单页网站)。每一页需定义为包含section类的元素。

一旦用户离开section，过渡到新section，就会触发onLeave(origin, destination, direction)的回调。

滚动结束之后，一旦加载了section，就会触发afterLoad(origin, destination, direction)回调。

想要将页面滚动到目标section和滑动则使用moveTo(section, slide)。

```
H5.js
```

```
/* 内容管理对象 */
var H5 = function () {
  this.id = ('h5_' + Math.random()).replace('.', '_');
  this.el = $('<div class="h5" id="' + this.id + '">').hide();
  this.page = [];
  $('body').append(this.el);
  this.addPage = function (name, text) { ... };
  this.addComponent = function (name, cfg) { ... };
  this.loader = function (firstPage) { ... };
  return this;
};
```

## 面向对象编程

典型的面向对象编程语言(比如C++和Java)，存在“类”(class)这个概念。所谓“类”就是对象的模板，对象就是“类”的实例。但是，在JavaScript语言的对象体系，不是基于“类”的，而是基于构造函数(constructor)和原型链(prototype)。“面向对象编程”(Object Oriented Programming，缩写为OOP)是目前主流的编程范式。它的核心思想是将真实世界中各种复杂的关系，抽象为一个个对象，然后由对象之间的分工与合作，完成对真实世界的模拟。这里创建了构造函数H5，并定义了一系列的属性和方法。之后通过new调用这个构造函数，返回一个对象实例。

注意return this是为了可以链式调用。

```
DERS
```

```
H5-webapp-master
  - css
  - imgs
  - js
  - lib
  /* H5.js
  /* H5ComponentBar.js
  /* H5ComponentBar_v.js
  /* H5ComponentBase.js
  /* H5ComponentPie.js
  /* H5ComponentPoint.js
  /* H5ComponentPolyline.js
  /* H5ComponentRadar.js
  /* H5ComponentRing.js
  <> index.html
  <> README.md
```

```
H5ComponentBar.js
```

```
/* 柱图组件对象 */
var H5ComponentBar = function (name, cfg) {
  var component = new H5ComponentBase(name, cfg);
  $.each(cfg.data, function (idx, item) {
    var line = $('<div class="line">');
    var name = $('<div class="name">');
    var rate = $('<div class="rate">');
    var per = $('<div class="per">');
    var width = item[1]*100 + '%';
    var bgStyle = '';
    if(item[2]){
      bgStyle = 'style="background-color:' + item[2] + '"';
    }
    rate.html('<div class="bg" ' + bgStyle + '></div>');
    rate.css('width', width);
    name.text(item[0]);
    per.text(width);
    line.append(name).append(rate).append(per);
    component.append(line);
  });
  return component;
};
```

## 组件化开发

有时候页面代码量太大，逻辑太多或者同一个功能组件在许多页面均有使用，维护起来相当复杂，这个时候，就需要组件化开发来进行功能拆分、组件封装，已达到组件通用性，增强代码可读性，维护成本也能大大降低。

此项目封装了柱状图、饼状图、散状态、折线图、雷达图等组了，方便复用。

```
//绘制网格线
var w = cfg.width/2;
var h = cfg.height/2;

//加入一个画布(网格线背景)
var cns = document.createElement('canvas');//画布
var ctx = cns.getContext('2d');//画布对象
cns.width = ctx.width = w;
cns.height = ctx.height = h;
component.append(cns);

var r = w/2;
var step = cfg.data.length;

ctx.beginPath();
ctx.arc(r,r-5,0,2*Math.PI);
ctx.stroke();

//绘制网格背景(分面绘制，分成10份)
for(var s=10;s>0:s--){
  ctx.beginPath();
  for(var i=0;i<step;i++){
    var rad = 2*Math.PI / step * i;
    x = r + Math.sin(rad) * (r-5)*s*0.1;
    y = r + Math.cos(rad) * (r-5)*s*0.1;

    //ctx.moveTo(r,r);
    //ctx.lineTo(x,y);
    //ctx.arc(x,y,5,0,2*Math.PI);
    ctx.lineTo(x,y);
  }
  ctx.closePath();
  ctx.fillStyle = !(s%2) ? "#99c0ff" : "#f1f9ff";
  ctx.fill();
}
```

## Canvas绘图

本项目使用了Canvas基本绘图API绘制了柱状图、饼状图、散状态、折线图、雷达图等。

计算一个圆周上的坐标

已知圆心坐标(a,b)半径r角度deg

$$x = a + \text{Math.sin}(\text{rad}) * r;$$

$$y = b + \text{Math.cos}(\text{rad}) * r;$$

# 微信小程序文影馆



## 项目介绍

从0开始构建了一个集阅读和电影资讯为一体的微信小程序。此项目运用了小程序的一些核心技术，例如：swiper轮播图组件、wx.setStorageSync数据缓存实现收藏功能、wx.showActionSheet实现底部分享、wx.playBackgroundAudio播放音乐、template嵌套模版、通过仿豆瓣API请求电影数据、scroll-view组件的bindscrolltolower监听滚动到底部时触发请求更多数据、声明类class Movie的属性和方法后new实例并调用实例的方法，达到组件化开发和模块化编程。

## 技术细节

```
app.js
1 //app.js
2 App({
3   onLaunch: function () {
4     // 展示本地存储能力
5     var logs = wx.getStorageSync('logs') || []
6     logs.unshift(Date.now())
7     wx.setStorageSync('logs', logs)
8
9   // 登录
10  wx.login({
11    success: res => {
12      // 发送 res.code 到后台换取 openId, sessionKey, unionId
13    }
14  })
15 // 获取用户信息
16  wx.getSetting({
17    success: res => {
18      if (res.authSetting['scope.userInfo']) {
19        // 已经授权，可以直接调用 getUserInfo 获取头像昵称，不会弹框
20        wx.getUserInfo({
21          success: res => {
22            // 可以将 res 发送给后台解码出 unionId
23            this.globalData.userInfo = res.userInfo
24
25            // 由于 getUserInfo 是网络请求，可能会在 Page.onLoad 之后才返回
26            // 所以此处加入 callback 以防止这种情况
27            if (this.userInfoReadyCallback) {
28              this.userInfoReadyCallback(res)
29            }
30          }
31        })
32      }
33    }
34  })
35  },
36  globalData: {
37    g_isPlayingMusic: false,
38    g_currentMusicPostId: null,
39    doubanBase: "https://douban.uieee.com/"
40  }
41})

```

```
wx.navigateTo({
  url: "post-detail/post-detail?id=" + postId
})
onLoad: function (option) {
  var postId = option.id;
  this.data.currentPostId = postId;
  var postData = postsData.postList[postId];
  this.setData({
    postData: postData
  })
}
// 展示本地存储能力
var logs = wx.getStorageSync('logs') || []
logs.unshift(Date.now())
wx.setStorageSync('logs', logs)
```

```
globalData: {
  g_isPlayingMusic: false,
  g_currentMusicPostId: null,
  doubanBase: "https://douban.uieee.com/"
}
var inTheatersUrl = app.globalData.doubanBase +
  "/v2/movie/in_theaters" + "?start=0&count=3";
var comingSoonUrl = app.globalData.doubanBase +
  "/v2/movie/coming_soon" + "?start=0&count=3";
var top250Url = app.globalData.doubanBase +
  "/v2/movie/top250" + "?start=0&count=3";
```

```
movie-list-template.wxml
1 <import src="../../movie/movie-template.wxml" />
2 <template name="movieListTemplate">
3   <view class="movie-list-container">
4     <view class="inner-container">
5       <view class="movie-head">
6         <text>{{categoryTitle}}</text>
7         <view catchtap="onMoreTap" class="more" data-category="{{categoryTitle}}"
8           <text>更多</text>
9           <image class="more-img" src="/images/icon/arrow-right.png"/>
10      </view>
11    </view>
12    <view class="movies-container">
13      <block wx:for="{{movies}}" wx:for-item="movie">
14        <template is="movieTemplate" data="{{...movie}}"/>
15      </block>
16      <!--<template is="movieTemplate" />
17      <template is="movieTemplate" />-->
18    </view>
19  </view>
20 </template>
```

## 小程序登陆

小程序通过wx.login返回res.code，再通过wx.login发送这个code到开发者后端服务器，开发者后端服务器再发送code,appid,appsecret到腾讯微信服务器返回session\_key,openid等。现在有云开发后，也可以直接用云函数返回session\_key,openid等。

通过wx.getSetting和wx.getUserInfo可获取用户信息。

## 页面传参

微信小程序页面与页面间传参可以利用  
1.url  
2.事件triggerEvent  
3.全局变量  
4.缓存

## template

WXML提供模板（template），可以在模板中定义代码片段，然后在不同的地方调用。使用is属性，声明需要使用的模板，然后将模板所需要的data传入。

## 组件化开发

声明类class Movie的属性和方法后new实例并调用实例的方法，达到组件化开发和模块化编程。

```
class Movie {
  constructor(url) {
    this.url = url;
  }

  getMovieData(cb) {
    this.cb = cb;
    util.http(this.url, this.processDoubanData.bind(this));
  }

  processDoubanData(data) {
    if (!data) {
      return;
    }
    var director = {
      avatar: '',
      name: '',
      id: ''
    }
    if (data.directors[0] != null) {
      if (data.directors[0].avatars != null) {
        director.avatar = data.directors[0].avatars.large
      }
      director.name = data.directors[0].name;
      director.id = data.directors[0].id;
    }
    var movie = {
      movieImg: data.images ? data.images.large : '',
      country: data.countries[0],
      title: data.title,
      originalTitle: data.original_title,
      wishCount: data.wish_count,
      commentCount: data.comments_count,
      year: data.year,
      genres: data.genres.join("、"),
      stars: util.convertToStarsArray(data.rating.stars),
      score: data.rating.average,
      director: director,
      casts: util.convertToCastString(data.casts),
      castsInfo: util.convertToCastInfos(data.casts),
      summary: data.summary
    }
    this.cb(movie);
  }
}

export {Movie}

var movie = new Movie(url);
movie.getMovieData((movie) => {
  this.setData({
    movie: movie
  })
})
```

## 可优化

1.Component自定义组件代替template，template只有wxml和wxss，没有js，实现了模块化，没有实现模块化。  
2.小程序提高性能的方法：

1) 无用文件、函数、样式剔除，小程序里的所有依赖模块都是有迹可循的，我们只需要利用这些关键字信息递归查找，遍历出文件依赖树，然后把没用的模块剔除掉。

2) 减少代码包中的静态资源文件，图片裁剪&降质，图片懒加载、雪碧图（CSS Sprite）优化，或把图片、视频等静态资源都放在CDN上。

3) 分包加载，小程序启动时只会下载主包/独立分包，启用分包可以有效减少下载时间。

4) 数据预拉取，就是在小程序启动时，微信服务器代理小程序客户端发起一个HTTP请求到第三方服务器来获取数据，并且把响应数据存储在本地客户端供小程序前端调取。当小程序加载完成后，只需调用微信提供的API

wx.getBackgroundFetchData从本地缓存获取数据即可。这种做法可以充分利用小程序启动和初始化阶段的等待时间，使更快地完成页面渲染。

5) 尽可能地把多次setData调用合并成一次，并且只把与界面渲染相关的数据放在data中，提升渲染性能。

6) 请求合并，在小程序中，wx.request（HTTP连接）的最大并发限制是10个；wx.connectSocket（WebSocket连接）的最大并发限制是5个；超出并发限制数目的HTTP请求将会被阻塞，需要在队列中等待前面的请求完成，从而一定程度上增加了请求时延。因此，对于职责类似的网络请求，最好采用节流的方式，先在一定时间间隔内收集数据，再合并到一个请求体中发送给服务端。