React细节知识点 - 实战3

1. 详情页面布局

1. 操作

```
//1.detail/index.js布局
import { DetailWrapper, Header, Content } from './style';

<DetailWrapper>
  <Header>当“小偷家族”遇上王多鱼</Header>
  <Content>
    <img src="图片地址" alt="" />
    <p>啦啦啦啦啦啦啦啦啦啦啦啦</p>
  </Content>
</DetailWrapper>

//2.在detail目录下新建style.js
import styled from 'styled-components';

export const DetailWrapper = styled.div`
  overflow: hidden;
  width: 620px;
  margin: 0 auto;
  padding-bottom: 100px;
`;

export const Header = styled.div`
  margin: 50px 0 20px 0;
  line-height: 44px;
  font-size: 34px;
  color: #333;
  font-weight: bold;
`;

export const Content = styled.div`
  color: #2f2f2f;
  img {
    width: 100%;
  }
  p {
    margin: 25px 0;
    font-size: 16px;
```

```
    line-height: 30px;
  }
`;
```

——

2. 使用redux管理详情页面数据

1. 操作

```
//1.在detail目录下新建store文件夹，在其中新建
index.js,reducer.js,actionCreators.js,actionTypes.js

/*detail/store/index.js*/
import reducer from './reducer';
import * as actionCreators from './actionCreators';
import * as constants from './actionTypes';

export { reducer, actionCreators, constants };

/*detail/store/reducer.js*/
import { fromJS } from 'immutable';
import * as constants from './actionTypes';

const defaultState = fromJS({
  title: '当“小偷家族”遇上王多鱼',
  content:
```

```
    '<img src="http://upload-images.jianshu.io/upload_images/18213612-23ed1bc2f0bd686f.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1000/format/webp"alt=""/><p><b>我总觉得，有两类电影最适合在电影院观看。</b>一类是喜剧片，“独乐乐不如众乐乐”，一群人笑声的相互传染更能让快乐加倍；另一类是节奏平淡如水的现实主义片子，毕竟在生活的快节奏和相伴的各种焦虑下，人们是很少能在两个小时的“流水账”中逼着自己坚持到最后。这个时候，黑乎乎的放映厅作为一个封闭的时空显现出特定的魔力（另一个我觉得很有意思并且常常在想的“封闭时空”是行驶中的火车车厢）。</p><p>对于喜剧片，观众涌入放映厅借以逃避现实的“不可乐”哪怕只有短暂的两小时也是莫大的精神放松，这是一个由“世俗”走向“世外”的转变过程（物质层面的由内到外）；而对于“白开水”似的现实性影片，观众自愿买票入场本身就是给自我下达了暗示，即“自己心里清楚，这两个小时只能老老实实坐在这里沉浸于电影，其他事务一概做不了”。</p><p>隔绝了世俗的事务和焦虑，黑暗而又封闭的放映厅时空让观众走回了内心，大银幕上跃动的不仅是艺术创作者完成的“别人的故事”，同时也是观众自身正在进行的“自己的人生”的另一重映射，现实主义往往能引起情感共鸣，也正是在于“透过别人在看自己”，从这个意义上讲，观众是在精神层面的“由外向内”。很有意思的是，很多电影里描绘角色的内心戏，也正是反过来用“自我”独自在影院看电影的形式进行表现。此外，也不得不承认，“现实主义”影片的“现实性”从先天上就决定了丧丧的情感基调，也就往往只能借助“封闭时空”来排除外界压力，让自己在相对轻松的环境下有勇气回味这种“日常丧”、“平淡丧”。</p><p>接连看完《西虹市首富》和《小偷家族》两部电影，这种差异对比会感觉的更明显。如果不是在电影院而是在电脑上，一个人看《西虹市首富》我会觉得不过瘾，而《小偷家族》很可能会直接被我囤在电脑硬盘角落里很久而没打开过。两种截然不同的类型片，没有任何可比性。</p><p>前者作为最迎合主流“合家欢”市场的商业喜剧片注定收获高票房，而不热衷讨好市场的现实主义文艺片的后者也已经赢得了高口碑，在戛纳金棕榈大奖的加持下，更让人印象深刻的是导演是枝裕和其后的回应和行动。看完《西虹市首富》我会继续期待开心麻花的下一部作品，在现实这么“不可乐”的社会环境下，能让观众发自内心大笑已经是一件很难得、也值得感谢的事；而看完《小偷家族》，我会想把是枝裕和之前的电影都翻一遍，在短暂的欢笑之后，回归生活必然还是要靠源于生活的平静而又充满力量的内省。</p>'
});

export default (state = defaultState, action) => {
  switch (action.type) {
    default:
      return state;
  }
};

//2.把detail的reducer加入到根目录的总reducer中
import { reducer as detailReducer } from '../pages/detail/store';

const reducer = combineReducers({
  header: headerReducer,
  home: homeReducer,
  detail: detailReducer       //新增
});

//3.组件使用store中的数据;在detail/index.js中
import { connect } from 'react-redux';
```

```
<DetailWrapper>
  <Header>{this.props.title}</Header>
  <Content dangerouslySetInnerHTML={{ __html: this.props.content }} />
</DetailWrapper>

const mapState = state => ({
  title: state.getIn(['detail', 'title']),
  content: state.getIn(['detail', 'content'])
});

export default connect(mapState, null)(Detail);
```

——

3. 异步获取数据

1. **操作**

```
//1.在public/api目录下新建detail.json,把reducer.js中数据放到这里
/*detail/store/reducer.js中*/
const defaultState = fromJS({
  title: '',
  content: ''
});

/*public/api/detail.json中*/
{
  "success": true,
  "data": {
    "title": "当“小偷家族”遇上王多鱼",
```

```
    "content": "<img src=\"http://upload-
images.jianshu.io/upload_images/18213612-23ed1bc2f0bd686f.png?
imageMogr2/auto-orient/strip%7CimageView2/2/w/1000/format/webp\"alt=\"\"/><p>
<b>我总觉得，有两类电影最适合在电影院观看。</b>一类是喜剧片，“独乐乐不如众乐乐”，一群人笑声
的相互传染更能让快乐加倍；另一类是节奏平淡如水的现实主义片子，毕竟在生活的快节奏和相伴的各种
焦虑下，人们是很少能在两个小时的“流水账”中逼着自己坚持到最后。这个时候，黑乎乎的放映厅作为一
个封闭的时空显现出特定的魔力（另一个我觉得很有意思并且常常在想的“封闭时空”是行驶中的火车车
厢）。</p><p>对于喜剧片，观众涌入放映厅借以逃避现实的“不可乐”哪怕只有短暂的两小时也是莫大的
精神放松，这是一个由“世俗”走向“世外”的转变过程（物质层面的由内到外）；而对于“白开水”似的现实
性影片，观众自愿买票入场本身就是给自我下达了暗示，即“自己心里清楚，这两个小时只能老老实实坐在
这里沉浸于电影，其他事务一概做不了”。</p><p>隔绝了世俗的事务和焦虑，黑暗而又封闭的放映厅时
空让观众走回了内心，大银幕上跃动的不仅是艺术创作者完成的“别人的故事”，同时也是观众自身正在进
行的“自己的人生”的另一重映射，现实主义往往能引起情感共鸣，也正是在于“透过别人在看自己”，从这
个意义上讲，观众是在精神层面的“由外向内”。很有意思的是，很多电影里描绘角色的内心戏，也正是反
过来用“自我”独自在影院看电影的形式进行表现。此外，也不得不承认，“现实主义”影片的“现实性”从先
天上就决定了丧丧的情感基调，也就往往只能借助“封闭时空”来排除外界压力，让自己在相对轻松的环境
下有勇气回味这种“日常丧”、“平淡丧”。</p><p>接连看完《西虹市首富》和《小偷家族》两部电影，这
种差异对比会感觉的更明显。如果不是在电影院而是在电脑上，一个人看《西虹市首富》我会觉得不过
瘾，而《小偷家族》很可能会直接被我囤在电脑硬盘角落里很久而没打开过。两种截然不同的类型片，没
有任何可比性。</p><p>前者作为最迎合主流“合家欢”市场的商业喜剧片注定收获高票房，而不热衷讨好
市场的现实主义文艺片的后者也已经赢得了高口碑，在戛纳金棕榈大奖的加持下，更让人印象深刻的是导
演是枝裕和其后的回应和行动。看完《西虹市首富》我会继续期待开心麻花的下一部作品，在现实这么“不
可乐”的社会环境下，能让观众发自内心大笑已经是一件很难得、也值得感谢的事；而看完《小偷家族》，
我会想把是枝裕和之前的电影都翻一遍，在短暂的欢笑之后，回归生活必然还是要靠源于生活的平静而又
充满力量的内省。</p>"
  }
}

//2.发送ajax请求
  componentDidMount() {
    this.props.getDetail();
  }
}

const mapDispatch = dispatch => ({
  getDetail() {
    dispatch(actionCreators.getDetail());   //异步ajax请求
  }
});

/*在detail/store/actionCreators.js中*/
import axios from 'axios';
import * as constants from './actionTypes';

const changeDetail = (title, content) => ({
  type: constants.CHANGE_DETAIL,
```

```
  title,
  content
});

export const getDetail = () => {
  return dispatch => {
    axios.get('/api/detail.json').then(res => {
      const result = res.data.data;
      dispatch(changeDetail(result.title, result.content));
    });
  };
};

//3.在detail/store/reducer.js
export default (state = defaultState, action) => {
  switch (action.type) {
    case constants.CHANGE_DETAIL:
      return state.merge({
        title: action.title,
        content: action.content
      });
    default:
      return state;
  }
};
```

——

4. 页面路由参数的传递

1. 动态路由获取参数方式

```
//1.在List.js中
原本:
<Link key={index} to="/detail">
现在：这样当我们点击第一个文章时候，跳转过去url是.../detail/1这样的形式
<Link key={index} to={'/detail/' + item.get('id')}>

//2.但是我们发现页面空白，这是为什么呢？在App.js中
原本：
<Route path="/detail" exact component={Detail} />
现在:
<Route path="/detail/:id" exact component={Detail} />   //增加一个id
```

```
//3.然后在detail/index.js中我们在render里console.log(this.props)看下，发现文章id值可以通过this.props.match.params.id获得，我们把这个id传给发送ajax的函数，所以就知道去请求哪个id文章的数据了
  componentDidMount() {
    this.props.getDetail(this.props.match.params.id);
  }
}

const mapDispatch = dispatch => ({
  getDetail(id) {
    dispatch(actionCreators.getDetail(id));
  }
});

/*detail/store/actionCreators.js*/
export const getDetail = id => {       //发ajax请求时带上id
  return dispatch => {
    axios.get('/api/detail.json?id=' + id).then(res => {     //写法新增
      const result = res.data.data;
      dispatch(changeDetail(result.title, result.content));
    });
  };
};
```

2. **第二种方式**

```
//1.在List.js中
原本：
<Link key={index} to="/detail">
现在：这样当我们点击第一个文章时候，跳转过去url是.../detail?id=1这样的形式
<Link key={index} to={'/detail?id=' + item.get('id')}>

//2.App.js
原本：
<Route path="/detail" exact component={Detail} />
现在：
<Route path="/detail" exact component={Detail} />    //不变，动态路由写法需加/:id

//3.然后在detail/index.js中我们在render里console.log(this.props)看下，发现文章id值在location下search: "?id=1"，需要手动去解析出1，我们把这个id传给发送ajax的函数，所以就知道去请求哪个id文章的数据了
```

---

5. 登陆页面布局

1. **操作**

```
//1.在page目录下新建login文件夹，在其中新建index.js和style.js文件
//当我们想给除了导航栏部分一个背景颜色的时候，按照LoginWrapper的样式写，但是我们发现热门搜索框被挡住了，只需在LoginWrapper中写z-index: 0;在header的style.js中HeaderWrapper中写z-index: 1 即可

/*login/index.js中*/
import React, { PureComponent } from 'react';
import { connect } from 'react-redux';
import { LoginWrapper, LoginBox, Input, Button } from './style';

class Login extends PureComponent {
  render() {
    return (
      <LoginWrapper>
        <LoginBox>
          <Input placeholder="账号" />
          <Input placeholder="密码" />
          <Button>登陆</Button>
        </LoginBox>
      </LoginWrapper>
    );
  }
}

const mapState = state => ({});

const mapDispatch = dispatch => ({});

export default connect(
  mapState,
  mapDispatch
)(Login);

/*login/style.js中*/
import styled from 'styled-components';

export const LoginWrapper = styled.div`
  z-index: 0;
```

```
  position: absolute;
  left: 0;
  right: 0;
  bottom: 0;
  top: 56px;    //除去导航栏，导航栏刚好是56px高度
  background: #eee;
`;
export const LoginBox = styled.div`
  width: 400px;
  height: 220px;
  margin: 100px auto;
  padding-top: 40px;
  background: #fff;
  box-shadow: 0 0 8px rgba(0, 0, 0, 0.1);
`;
export const Input = styled.input`
  display: block;
  width: 200px;
  height: 30px;
  line-height: 30px;
  padding: 0 10px;
  margin: 15px auto;
  color: #777;
`;
export const Button = styled.div`
  width: 220px;
  height: 30px;
  line-height: 30px;
  color: #fff;
  background: #3194d0;
  border-radius: 15px;
  margin: 15px auto;
  text-align: center;
`;

//2.定义路由，在App.js中导入这个login组件
import Login from './pages/login';

<Route path="/login" exact component={Login} />
```

---

6. 登录功能实现

1. 操作

```
//1.在login目录下新建actionCreators.js,actionTypes.js,reducer.js,index.js
/*index.js中*/
import reducer from './reducer';
import * as actionCreators from './actionCreators';
import * as constants from './actionTypes';

export { reducer, actionCreators, constants };

/*reducer.js中*/
import { fromJS } from 'immutable';
import * as constants from './actionTypes';

const defaultState = fromJS({
  login: false
});

export default (state = defaultState, action) => {
  switch (action.type) {
    default:
      return state;
  }
};


//2.在根目录的reducer.js中引入login/store/reducer.js
import { reducer as loginReducer } from '../pages/login/store';

const reducer = combineReducers({
  header: headerReducer,
  home: homeReducer,
  detail: detailReducer,
  login: loginReducer          //新增
});


//3.实现用户已登录的话，导航栏显示退出，已退出的话显示登录，通过login这个变量的false / true来控制。在header/index.js中
const mapStateToProps = state => {
  return {
    focused: state.getIn(['header', 'focused']),
    list: state.getIn(['header', 'list']),
    page: state.getIn(['header', 'page']),
    totalPage: state.getIn(['header', 'totalPage']),
    mouseIn: state.getIn(['header', 'mouseIn']),
```

```
    login: state.getIn(['login', 'login'])      //新增，获取到这个变量值
  };
};

const { ..., login, ...} = this.props;
{
  login ? (<NavItem className="right">退出</NavItem>) :
  (<Link to="/login"><NavItem className="right">登陆</NavItem></Link>)
}


//4.当用户点击导航栏的登录时候,在login/index.js中
//通过ref获取到input框节点，ref写法：ref={这个dom => {this.变量名 = 这个dom}}
//把获取到的账号和密码输入框dom节点传给点击事件，在点击事件中派发异步action去验证下用户登录
import { actionCreators } from './store';

<Input placeholder="账号" ref={input => {this.account = input;}}/>
<Input placeholder="密码" type="password" ref={input => {this.password =
input;}}/>
<Button onClick={() => this.props.login(this.account, this.password)}>
 登陆
</Button>

const mapDispatch = dispatch => ({
  login(accountElem, passwordElem) {
    dispatch(actionCreators.login(accountElem.value, passwordElem.value));
  }
});


//5.在login/store/actionCreators.js中给后端请求
//需要我们先模拟下数据，在public/api下新建login.json
{
  "success": true,
  "data": true
}

/*login/store/actionCreators.js*/
import axios from 'axios';
import * as constants from './actionTypes';

const changeLogin = () => ({
  type: constants.CHANGE_LOGIN,
  value: true
});
```

```
export const login = (account, password) => {
  return dispatch => {
    axios
      .get('/api/login.json?account=' + account + '&password=' + password)
      .then(res => {
        const result = res.data.data;
        if (result) {
          dispatch(changeLogin());
        } else {
          alert('登陆失败');
        }
      });
  };
};

/*login/store/actionTypes.js*/
export const CHANGE_LOGIN = 'login/change_login';


//6.登录成功后应该跳转到首页，使用Redirect这个组件
import { Redirect } from 'react-router-dom';

const mapState = state => ({
  loginStatus: state.getIn(['login', 'login'])
});

const { loginStatus } = this.props;
if (!loginStatus) {
  return (
    <LoginWrapper>
      <LoginBox>
        <Input
          placeholder="账号"
          ref={input => {
            this.account = input;
          }}
          />
        <Input
          placeholder="密码"
          type="password"
          ref={input => {
            this.password = input;
          }}
          />
        <Button
          onClick={() => this.props.login(this.account, this.password)}
```

```
          >
            登陆
          </Button>
        </LoginBox>
      </LoginWrapper>
    );
  } else {
    return <Redirect to="/" />;    //已登录的情况下重定向至首页
  }


  //7.退出功能；绑定一个点击事件用于改变login值的状态，就实现了退出
  //注意：现在是在header的index.js中，此处的actionCreators是
  header/store/actionCreators.js的，而我们对login的操作应该放在
  login/store/actionCreators.js中，所以我们引入login/store下的actionCreators.js
  import { actionCreators as loginActionCreators } from
  '../../pages/login/store';

  const {..., logout} = this.props;
  <NavItem className="right" onClick={logout}>
    退出
  </NavItem>

  logout() {
    dispatch(loginActionCreators.logout());
  }


  //8.在login/store/actionCreators.js中
  export const logout = () => ({
    type: constants.LOGOUT,
    value: false
  });

  /*login/store/actionTypes.js中*/
  export const LOGOUT = 'login/logout';


  //9.在login/store/reducer.js中
  case constants.LOGOUT:
    return state.set('login', action.value);
```

——

7. 登录后才能使用写文章功能

1. 操作

```
//1.pages目录下新建write文件夹，其中新建index.js
//用户登录了才能进入写文章页面 localhost/write
//用户没登录则重定向到登陆界面/login
import React, { PureComponent } from 'react';
import { Redirect } from 'react-router-dom';
import { connect } from 'react-redux';

class Write extends PureComponent {
  render() {
    const { loginStatus } = this.props;
    if (loginStatus) {
      return <div>写文章页面</div>;
    } else {
      return <Redirect to="/login" />;
    }
  }
}

const mapState = state => ({
  loginStatus: state.getIn(['login', 'login'])
});

export default connect(
  mapState,
  null
)(Write);


//2.在App.js中加入这个/write的路由
import Write from './pages/write';

<Route path="/write" exact component={Write} />


//3.在header/index.js中,当点击写文章按钮后跳转到/write路由界面
<Link to="/write">
  <Button className="writting">
    <i className="iconfont">&#xe615;</i>
    写文章
  </Button>
</Link>
```

——

8. 异步组件及withRouter路由方法的使用

1. 操作

```
//1.我们的项目不管如何切换页面，都只有最开始加载的那个bundle.js，这就说明一个问题，我们所有组件对应的代码都在这个bundle.js里。现在我只访问了首页，但它顺便把详情页面和登录页面一起都加载了，这样的话，首页加载的速度是比较慢的。如何解决这个问题呢？我们希望当访问首页的时候只加载首页的代码，当访问详情页面的时候再加载详情页面的代码，这就需要我们使用异步组件来实现我们的想法，我们使用一个第三方模块react-loadable


//2.安装
yarn add react-loadable


//3.我们想当我们点击详情页时再加载详情页;在detail目录下新建loadable.js
//这样写完Detail就变成了一个异步组件了
import React from 'react';
import Loadable from 'react-loadable';

const LoadableComponent = Loadable({
  loader: () => import('./'),       //加载当前目录下的index.js
  loading() {
    return <div>正在加载...</div>;
  }
});

export default () => <LoadableComponent />;


//4.App.js中
原本：
import Detail from './pages/detail';
现在：
import Detail from './pages/detail/loadable.js';

//接着页面报错了 TypeError: Cannot read property 'params' of undefined

原因在这里：detail/index.js中：this.props.getDetail(this.props.match.params.id);
报错了
```

以前可以获得路由的参数，以前App.js是加载./pages/detail现在是加载./pages/detail/loadable.js，<LoadableComponent />这个异步组件可以获得路由的参数，但是它下面的index就获取不到路由的参数了，因为它不是<Router />路由项直接对应的那个组件，那么怎么办呢?

```
//在detail/index.js中
import { withRouter } from 'react-router-dom';

//意思是让Detail有能力获取到路由里的参数和内容
export default connect(mapState, mapDispatch)(withRouter(Detail));


//5.验证
//我们刷新首页加载了bundle.js，当我们点击进入详情页的时候，这时候控制台network / js中，加载了一个新的chunk.js
```