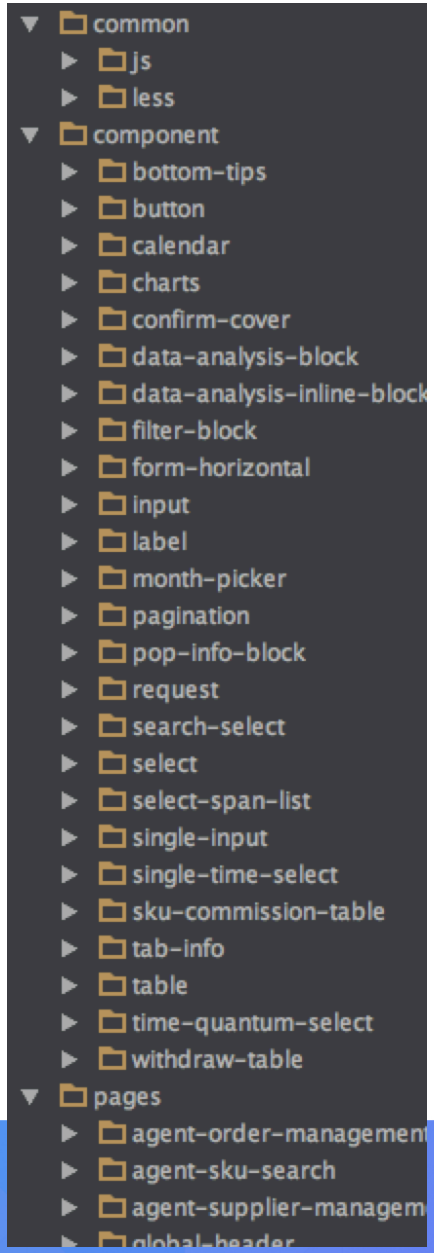
达人店后台技术分享

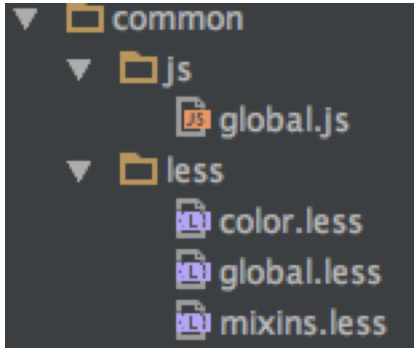
悟空

2016年10月21日

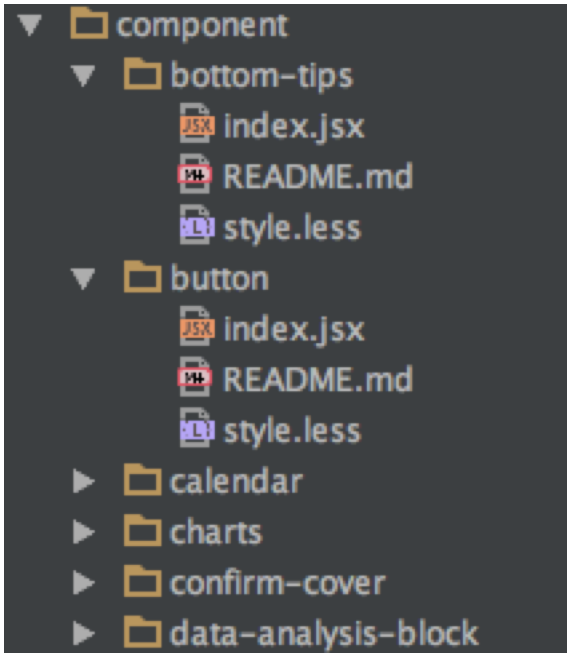
尚妆
SHOWJOY.COM
正 品 无 须 代 言

目录&文件说明

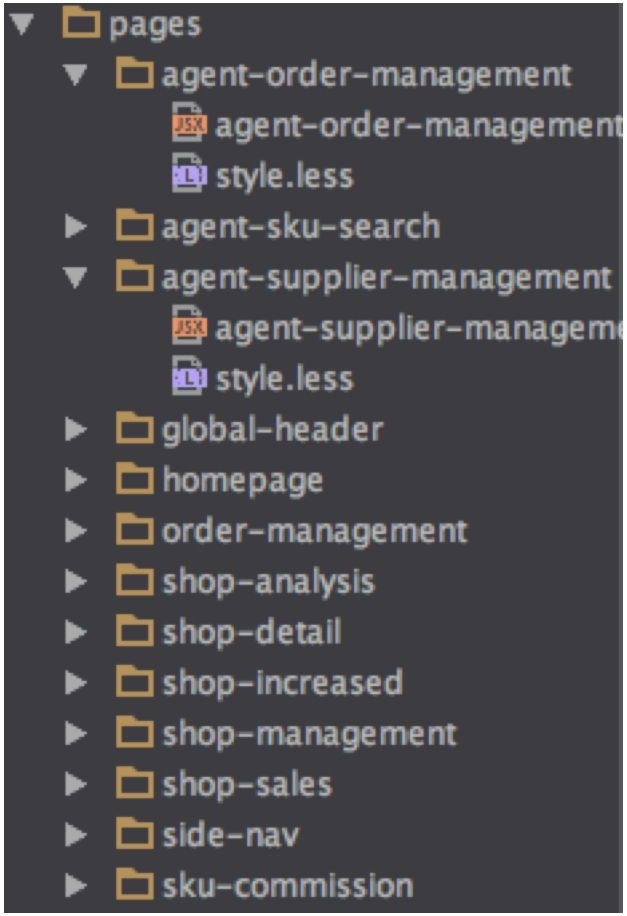




公共资源文件

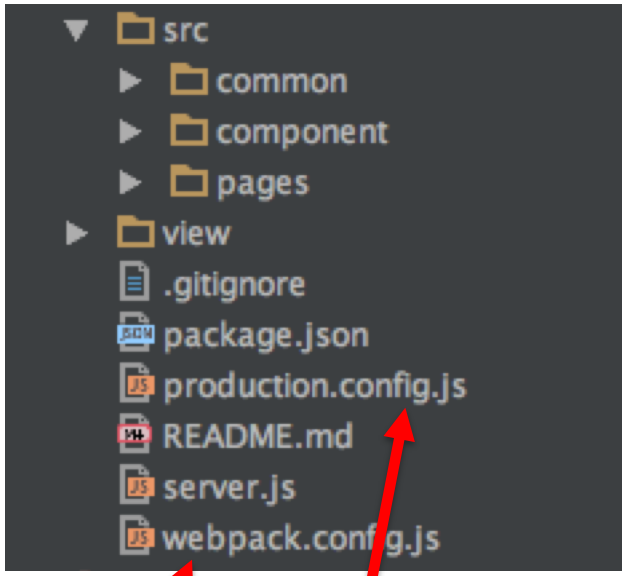


组件代码文件



页面代码文件

Webpack的应用



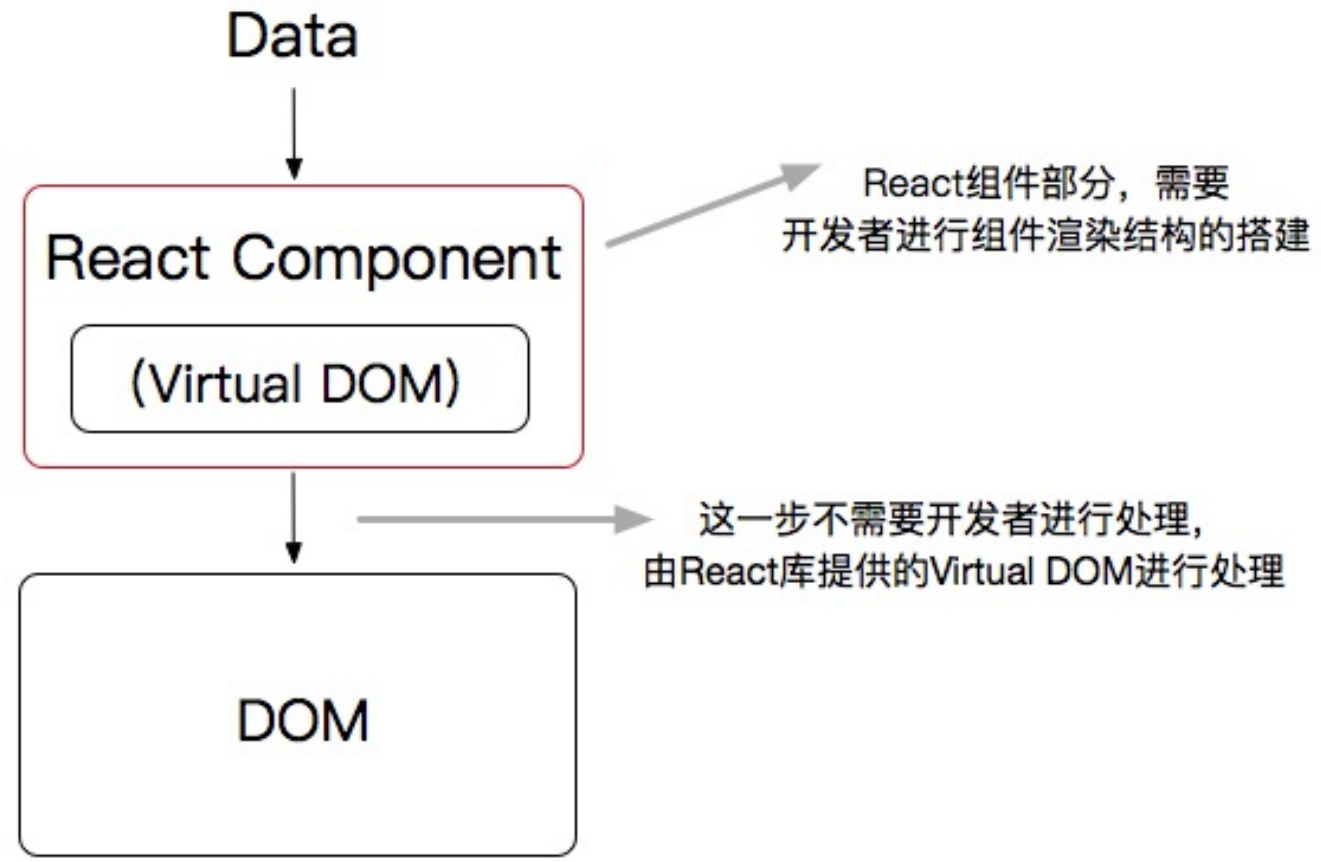
开发过程使用的配置文件

线上环境使用的配置文件

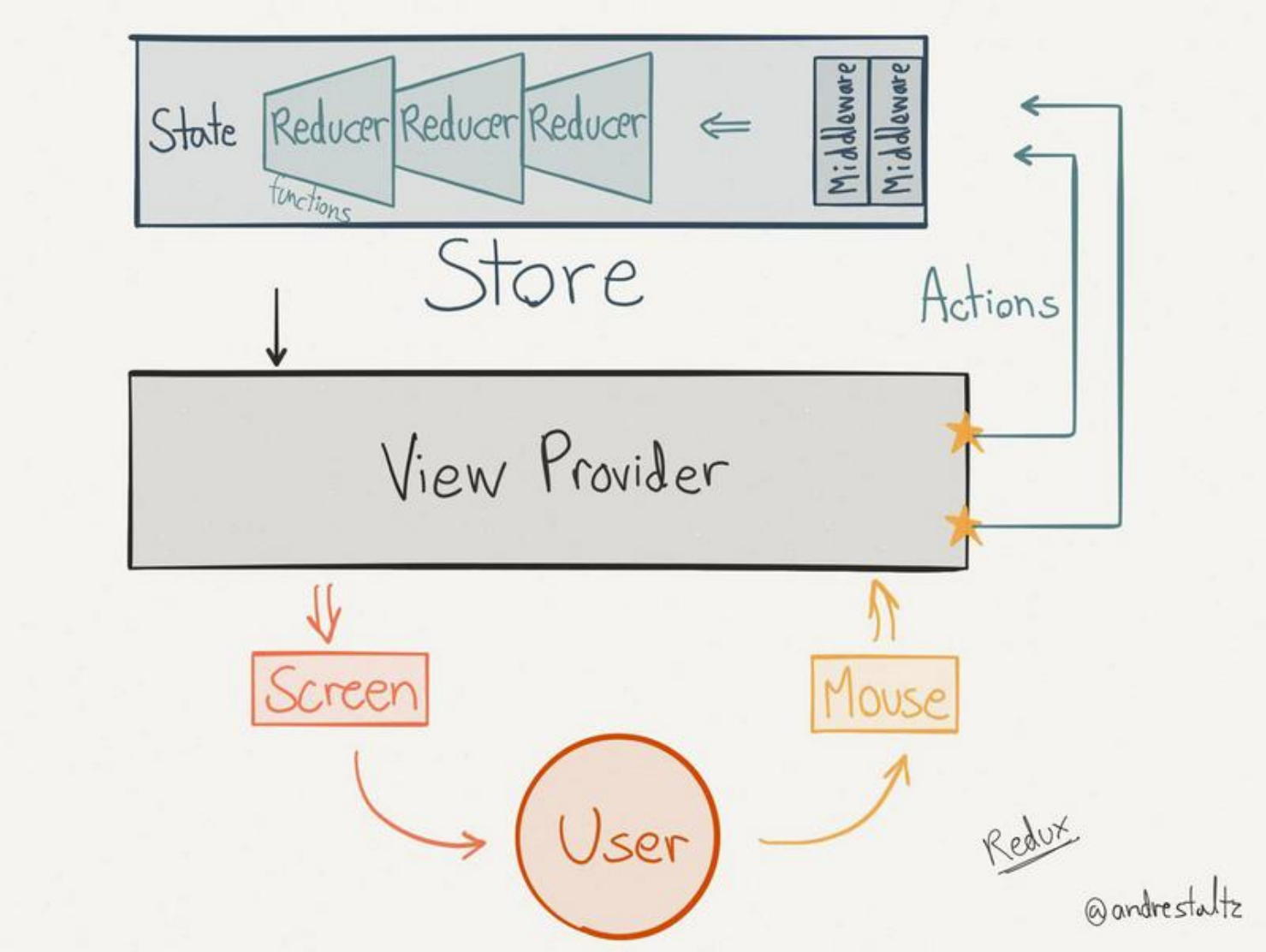
```
✗ webpack --config production.config.js
```

```
entry: entryFiles,
//输出文件配置
output: {
  path: path.resolve(__dirname, 'app'),
  publicPath: "http://localhost:3000/app/",
  filename: 'js/[name].js'
},
//引用依赖配置
resolve: {
  extensions: ['', '.js', '.jsx', '.json', '.less'],
  alias: {
    'react': pathToReact,
    'react-dom': pathToReactDOM,
    'redux': pathToRedux,
    'immutable': pathToImmutable
  }
},
//插件设置
plugins: [
  new webpack.DefinePlugin({
    'process.env.NODE_ENV': '"development"'
  }),
  new webpack.HotModuleReplacementPlugin(),
  //允许错误不打断程序
  new webpack.NoErrorsPlugin()
],
//加载器设置
module: {
```

React


Data
React Component
(Virtual DOM)
DOM
React组件部分，需要
开发者进行组件渲染结构的搭建
这一步不需要开发者进行处理，
由React库提供的Virtual DOM进行处理

Redux


State
Reducer
Reducer
Reducer
functions
Middleware
Middleware
Store
Actions
View Provider
Screen
Mouse
User
Redux
@andrestaltz

组件类型

React组件

React组件内部状态由组件的state进行控制，通过props进行方法的暴露。

优：封装好的组件内部实现不可见，**可复用性强**。

缺：外部组件或页面**不可以对内部组件状态进行修改**，是一个独立的组件个体存在。

Redux组件

Redux组件状态统一通过redux的store进行管理，通过dispatch aciton进状态的修改。

优：action全局可调用，**页面组件间可通信**，方式统一。

缺：使用方法繁琐，构建成本高于react组件，**可复用性相对较差，需要了解一定程度上的内部实现方式。**

React组件搭建

```
//import node modules
import React from 'react';

//import style
import './style';

//component body
export class FilterBlock extends React.Component {
  constructor(props) {
    super(props);

    this.state = {
      currentKey: this.props.init ? this.props.init : this.props.dataSource[0].key
    };
  }

  listClick = (key) => {
    let tempState = Object({}, this.state);
    tempState.currentKey = key;
    this.setState(tempState);
    this.props.onFilterClick(key);
  };

  render() {
    let onFilterClick = this.props.dataSource.map(data => {
      let className = '';
      if (data.key === this.state.currentKey) {
        className = 'active';
      }

      return (
        <li className={className} onClick={(e) => this.listClick(data.key)}>{data.value}</li>
      )
    });

    return (
      <div className="filter-block">
        <ul>
          {onFilterClick}
        </ul>
      </div>
    );
  }
}
```

```
//import node modules
import React from 'react';

//import style
import './style';
```

```
constructor(props) {
  super(props);

  this.state = {
    currentKey: this.props.init ? this.props.init : this.props.dataSource[0].key
  };
}
```

Constructor:
组件初始化 -> state初始化

---

this.props.onFilterClick(key);

暴露点击方法

React组件使用

FilterBlock为例子

```
let filterProps = [{
  key: 0,
  value: '店铺详情'
}, {
  key: 1,
  value: '家族详情'
}];
```

配置信息 -> 页面引用

```
<FilterBlock dataSource={filterProps} init={0}
             onFilterClick={(key) => this.changeCurrentFilter(key)}/>
```

优：封装好的组件内部实现不可见，**可复用性强**。

缺：外部组件或页面**不可以对内部组件状态进行修改**，是一个独立的组件个体存在。

Redux组件搭建

```
//define action type
const _componentAction = {
  REQUEST_START: 'REQUEST_START',
  REQUEST_FINISHED: 'REQUEST_FINISHED'
};
```

_componentAction -> 可触发的action
Property -> 数据结构
Reducer -> action触发后的数据改变

```
export const RequestProperty = {
  request: false
};

export function RequestReducer(state = RequestProperty, action) {
  if (state.id && state.id !== action.id) {
    return state;
  } else {
    let tempState = Object.assign({}, state);
    switch (action.type) {
      case _componentAction.REQUEST_START:
        tempState.request = true;
        return tempState;
      case _componentAction.REQUEST_FINISHED:
        tempState.request = false;
        return tempState;
      default:
        return tempState;
    }
  }
}
```

可以内部dispatchAction进行方法的调用，或者采用和React组件相同的方式通过props暴露接口

Redux组件使用

Request为例子

```
//init page state -> server data init
let store = createStoreWithMiddleware(combineReducers({
  shopSearch: InputReducer,
  inviteShopSearch: InputReducer,
  pagination: PaginationReducer,
  getShop: RequestReducer
}), {
  shopSearch: Object.assign({}, InputProperty, {id: 'shop-search', placeholder: '输入搜索店铺名'}),
  inviteShopSearch: Object.assign({}, InputProperty, {id: 'invite-shop-search', placeholder: '输入搜索邀请店铺名'}),
  getShop: Object.assign({}, RequestProperty, {id: 'get-shop'})
});
```

获取组件reducer，修改property

```
<Request id="get-shop" dataSource={getShop} init={true} url={shopSalesApi}
         requestData={getShopRequestData} action={(id, type, e) => this.dispatchActions(id, type, e)}
         response={(result, init) => dispatch(this.renderTable(result, init))}/>
```

```
setPage = (pageNum) => {
  let tempState = Object.assign({}, this.state);
  tempState.page = pageNum;
  tempState.tipsShow = true;
  this.setState(tempState);
  return (dispatch) => {
    dispatch({type: 'PAGINATION_SET_CURRENT_PAGE', value: pageNum});
    dispatch({id: 'get-shop', type: 'REQUEST_START', value: true});
  };
};
```

在其他地方可以对组件的状态进行修改。

THANKS FOR YOUR ATTENTION

尚妆

SHOWJOY.COM

正 品 无 须 代 言