基于FLUX架构的项目开发

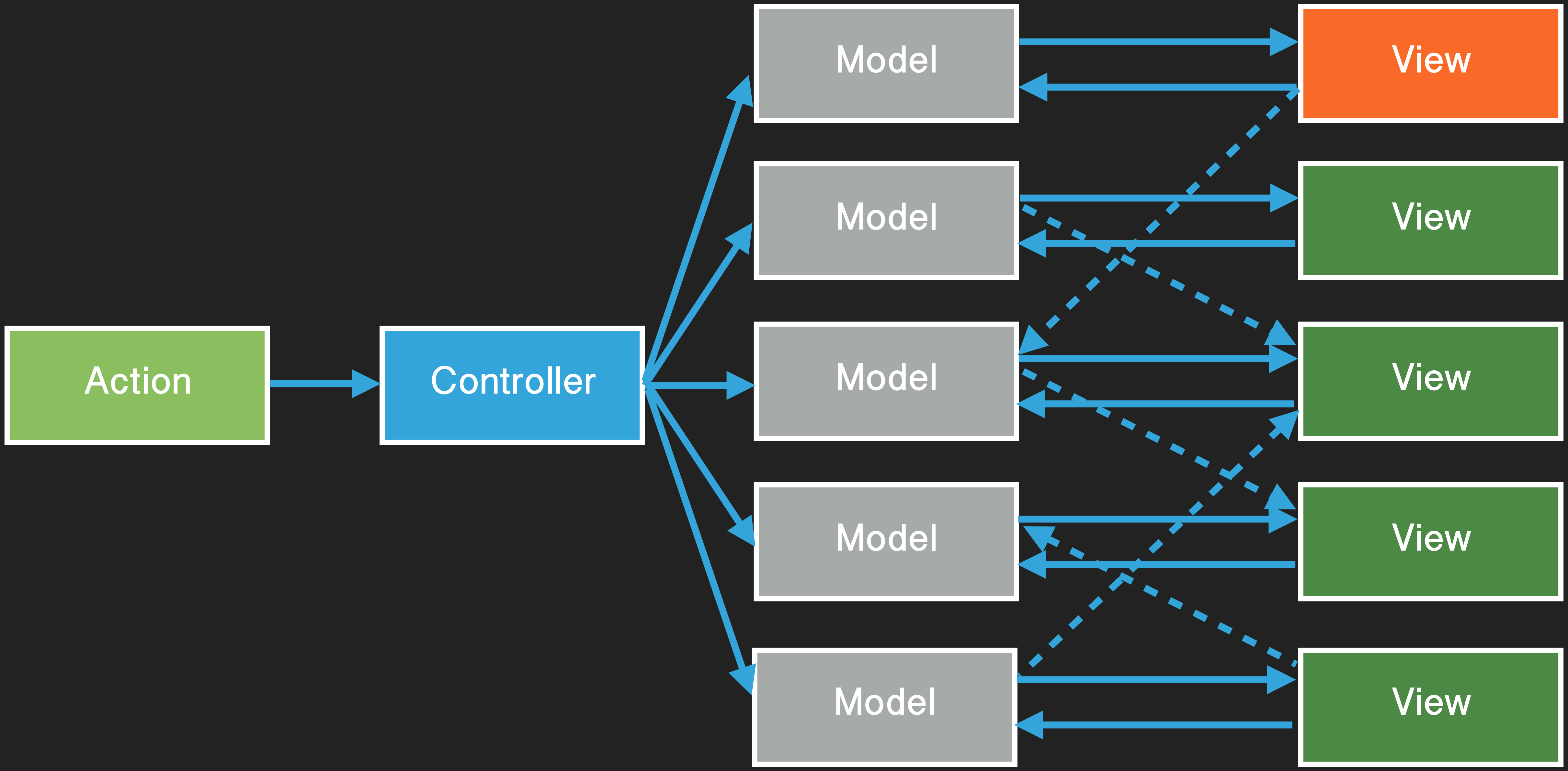
2016.09.25

悠鹤

目录

先顺道“黑”一下MVC(VM)

MVC应用


Action
Controller
Model
Model
Model
Model
Model
View
View
View
View
View

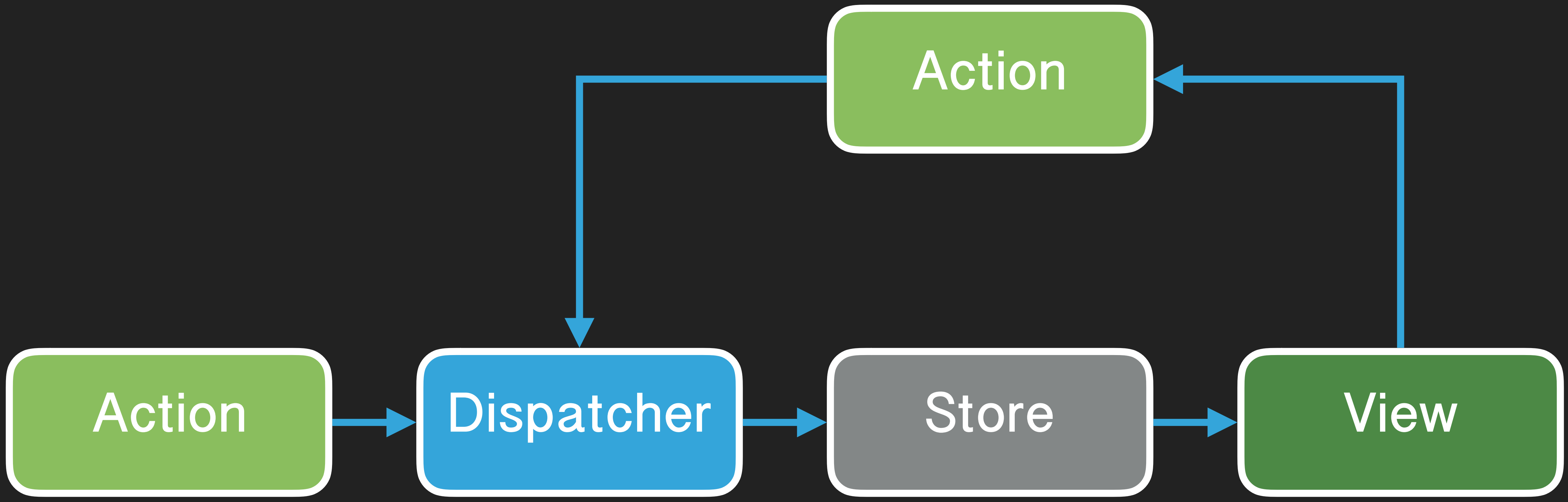
当MVC应用变得复杂和难以扩展

我们应该分离哪些关注点?

FLUX应用架构简介

FLUX的核心就是一个简单的约定

Flux



FLUX应用架构简介

```
class LoginComponent extends
React.Component {
  login(user) {
    userActions.login(user);
  }
}
```

react component

```
var UserActions = {
  login: function (user) {
    AppDispatcher.dispatch({
      actionType: 'LOGIN',
      user: user
    });
  },
};
```

action

store

```
export default Object.assign({
  user: {},
  updateUser: function (user) {
      this.user = user;
  },
  emitChange: function () {
    this.emit('change');
  }
});
```

mutation

```
AppDispatcher.register(function (action) {
  switch(action.actionType) {
    case 'LOGIN':
      Store.login(action.user);
      Store.emitChange();
      break;
    default:
  }
})
```

FLUX的核心理念

- 单向数据流
- 统一调度
- 可预测
- 函数式编程

如果说REACT、VUE组件是对视图的抽象

那么，FLUX架构就是对用户行为的抽象

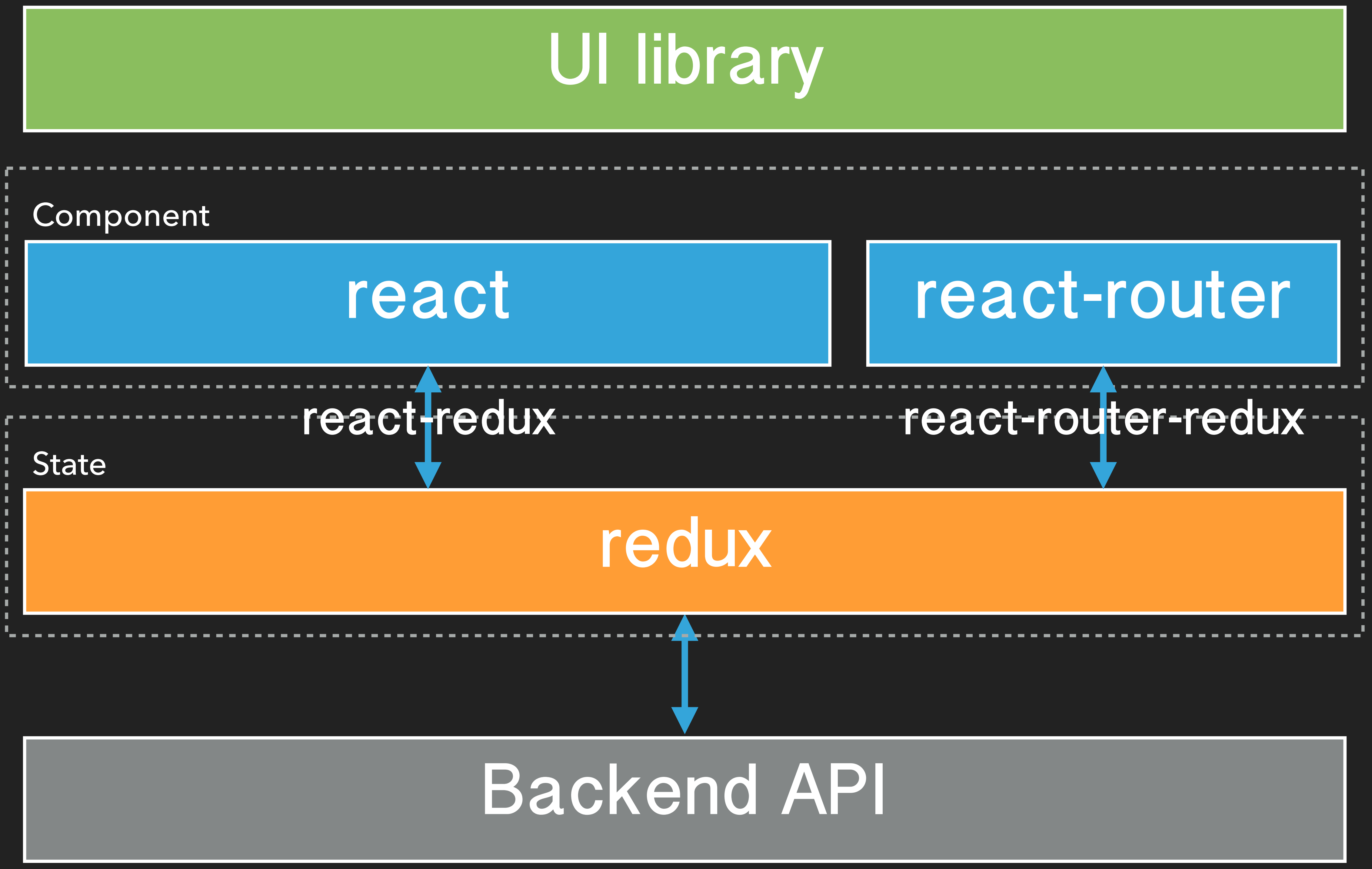
FLUX应用架构实现

$$UI = f(state)$$

FLUX应用架构实现

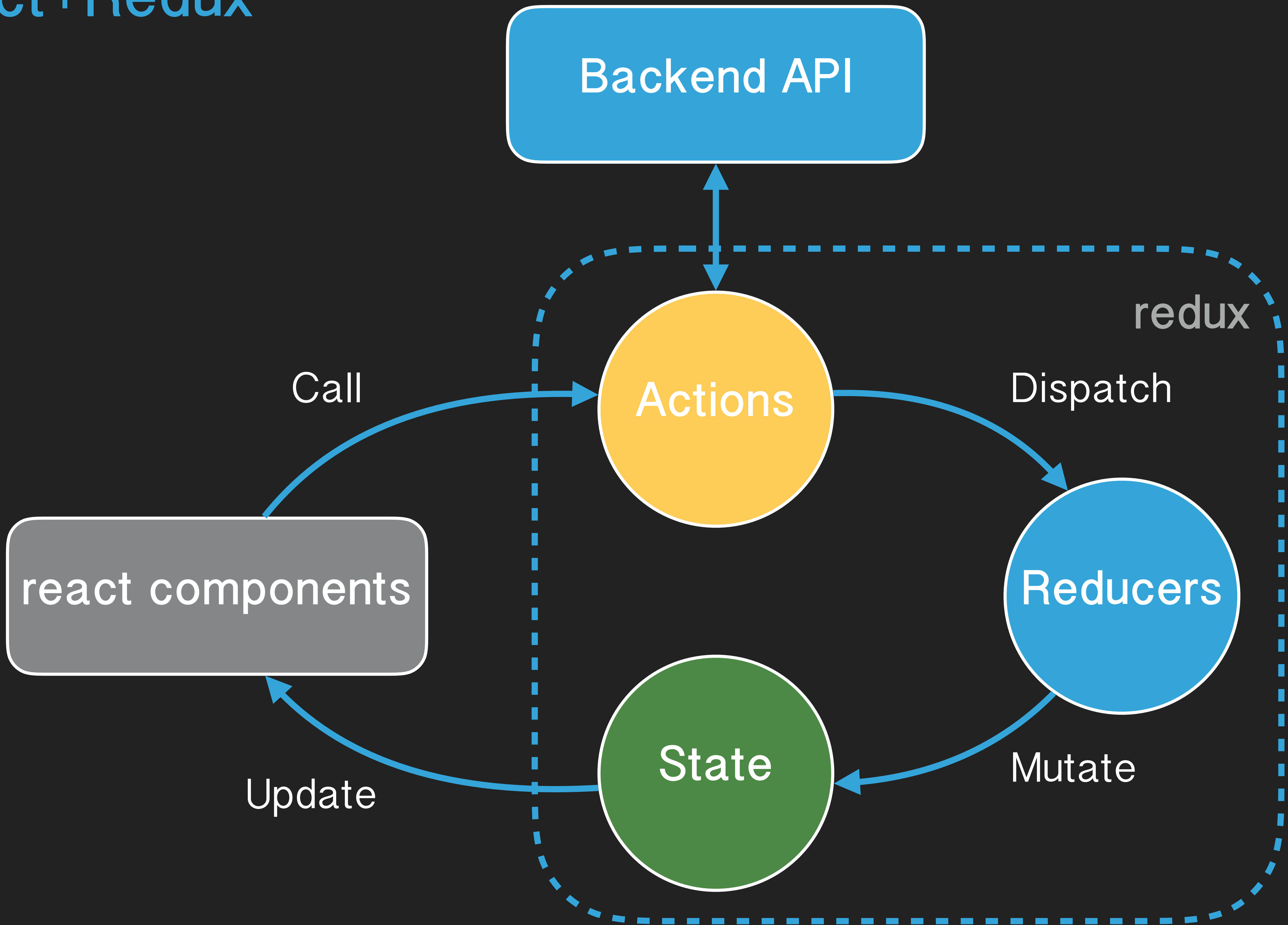
```
(state, action) => state
```

React+Redux


UI library
Component
react
react-router
react-redux
react-router-redux
State
redux
Backend API

Redux是一个
状态可预测的容器

React+Redux


Backend API
redux
Call
Actions
Dispatch
react components
Reducers
State
Mutate
Update

FLUX应用架构实现

```
class LoginComponent extends
React.Component {
  login() {
    userActions.login();
  }
}
```

react component

```
export function login() {
  return dispatch => fetch('/user/
login/')
    .then(res => dispatch({
      type: 'LOGIN',
      user: res.json()
  }));
}
```

action

store

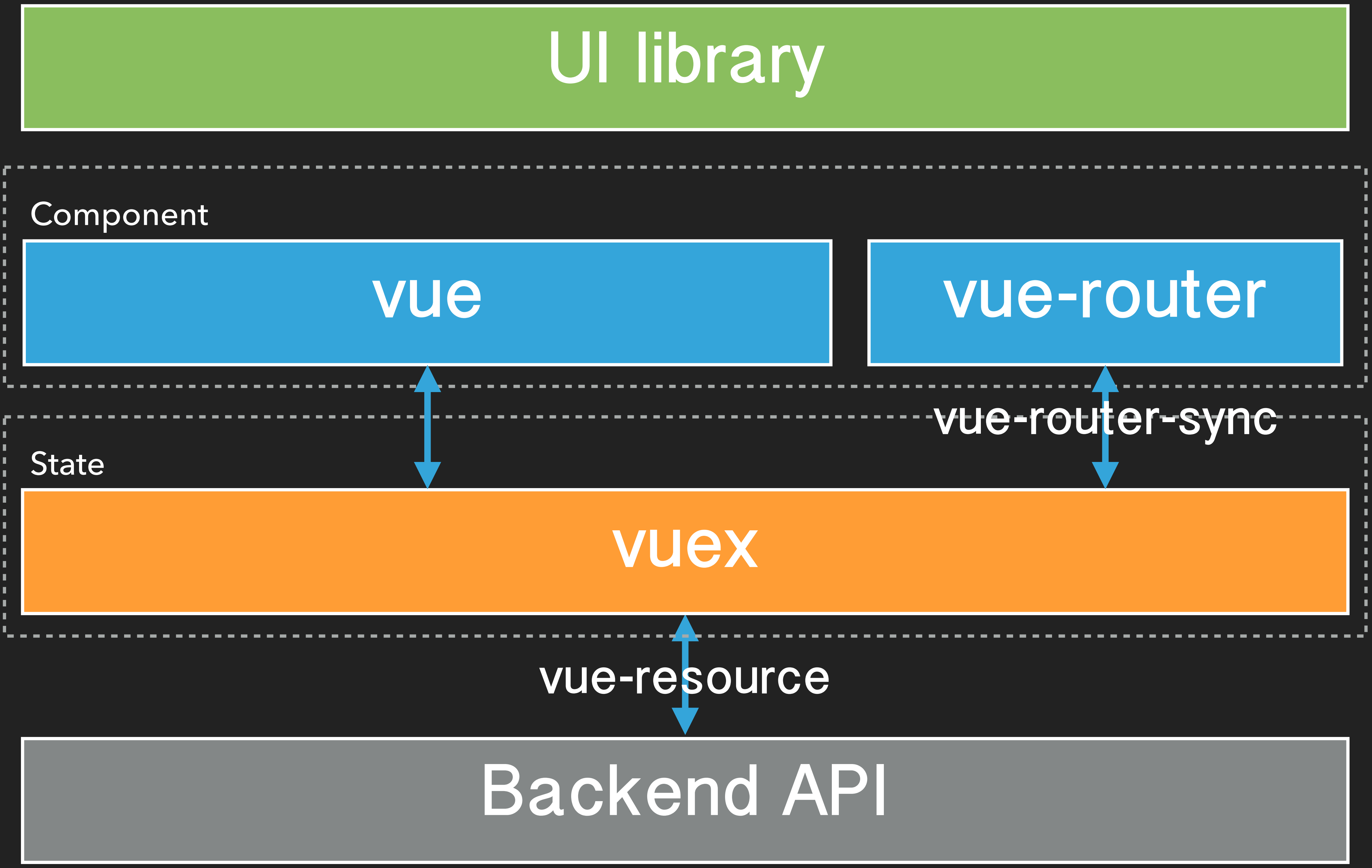
```
const reducers = combineReducers({
  account: userReducer
});
```

reducer

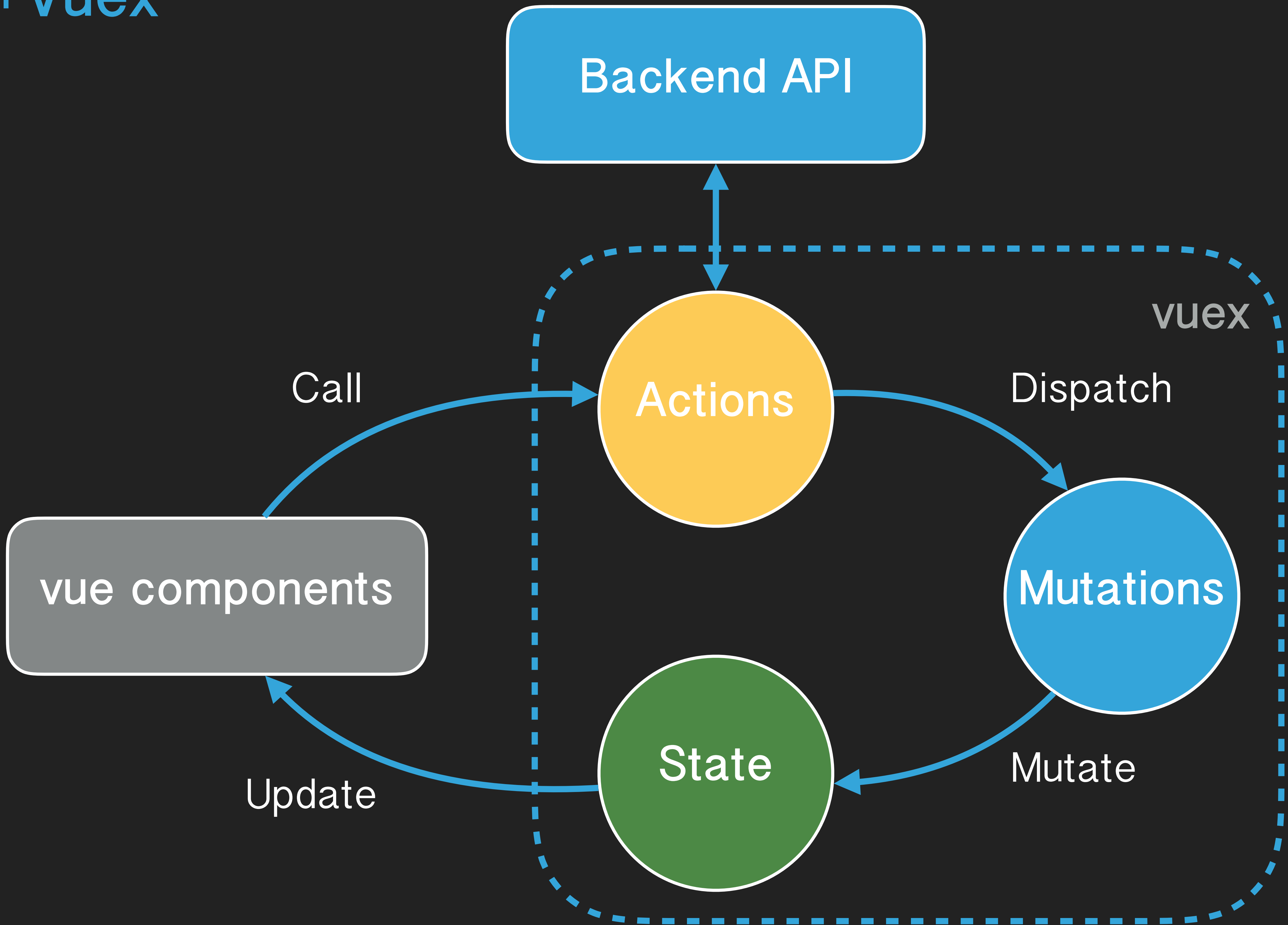
```
function reducer(state = {}, action) {
  switch(action.type) {
    case 'LOGIN': return
Object.assign({}, state, action.user);
    default: return state;
  }
}
```

FLUX应用架构实现

Vue+Vuex



Vue+Vuex


Backend API
vuex
Call
Actions
Dispatch
vue components
Mutations
State
Mutate
Update

FLUX应用架构实现

```
export default {
  vuex: {
    actions: {
      loginUser: userActions.login,
      getters: user
    }
  }
};
```

vue component

```
export function login({dispatch}) {
  return Vue.http.get('/user/login')
    .then(res => dispatch('LOGIN',
    res.json()));
}
```

action

store

```
export default {
     state: {
          user: {}
     }
     mutations
};
```

mutation

```
export default {
  LOGIN (state, { result }) {
    Vue.set(state, 'user', result);
  }
};
```

Redux和Vuex对比

React+Redux	Vue+Vuex
UI = *f*(state)	
connect、mapStateToProps	Vue.use(Vuex)、getters
`(action, state) => state`	
reducer	mutation
state tree	
combineReducers	modules
mutate state	
Object.assign	响应属性变化

项目搭建

脚手架

- 配置简化
- 一致性
- 自动化
- 关注分离

脚手架

generator-react-webpack-redux

```
$ npm install -g yeoman
$ npm install -g generator-react-webpack-redux
$ yo react-webpack-redux
```

脚手架

vue-cli

```
$ npm install -g vue-cli
$ vue init webpack my-project
```

环境配置

- eslint
- babel
- npm scripts
- webpack

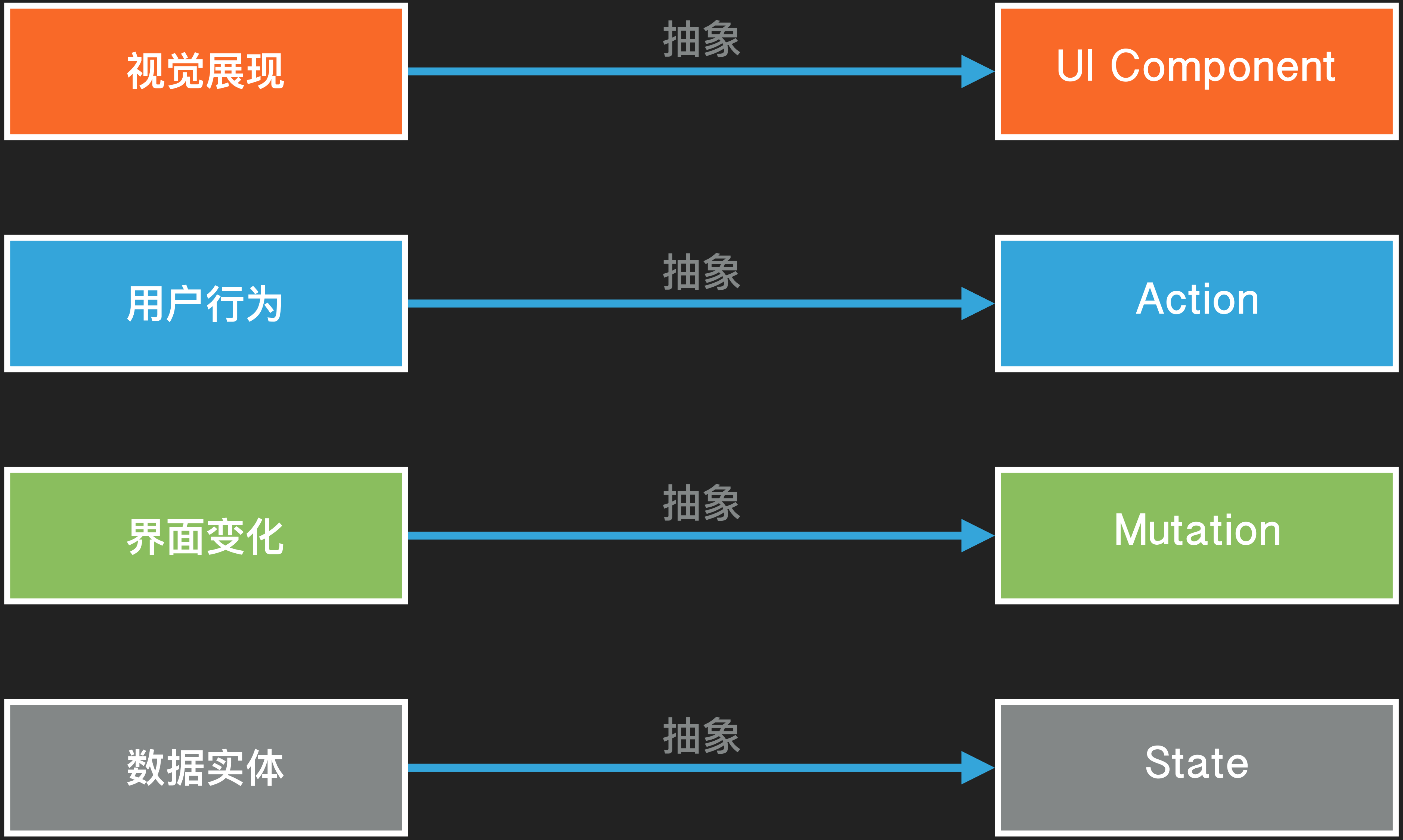
项目搭建

webpack

- 公共库提取
- 代码分块
- 热更新
- 各种loader

项目开发

数据流开发


视觉展现
抽象
UI Component
用户行为
抽象
Action
界面变化
抽象
Mutation
数据实体
抽象
State

遇到的一些坑

- form表单开发
- 集合数据更新
- 设置初始状态
- 非纯函数调用

调试工具

- React Developer Tools
- Vue.js devtools
- redux-logger middleware
- vuex/logger middleware

示例项目

- JSErrorMonitor
- houseSpider

学习资源

- https://facebook.github.io/flux/docs/overview.html
- https://www.zhihu.com/question/33864532
- https://www.infoq.com/news/2014/05/facebook-mvc-flux
- http://cn.redux.js.org/index.html
- http://vuex.vuejs.org/zh-cn/index.html
- https://github.com/icefox0801/JSErrorMonitor
- https://github.com/icefox0801/houseSpider

谢谢！

悠鹤
@支付宝北京团队
Github: icefox0801