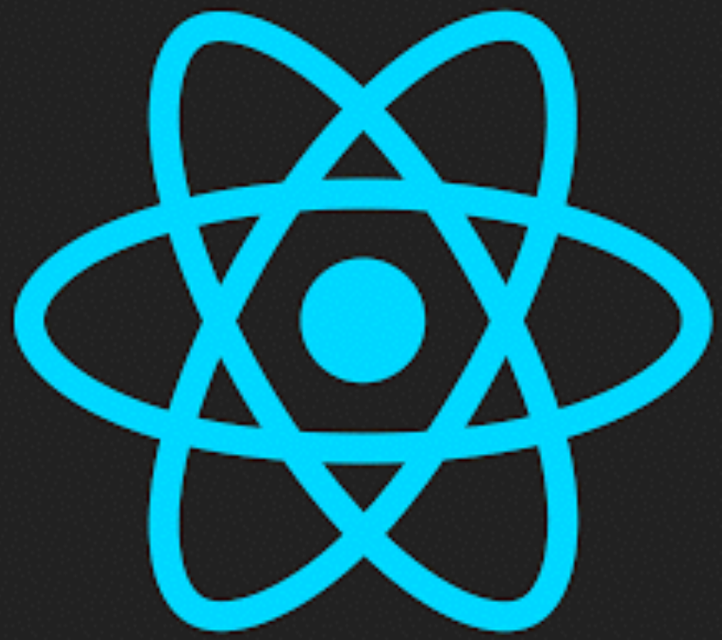
ReactJs 实践

onebox | 黄娇龙

- 是什么?
- 能干什么?
- 怎么用?
- 未来会怎样?
- 实践中的应用

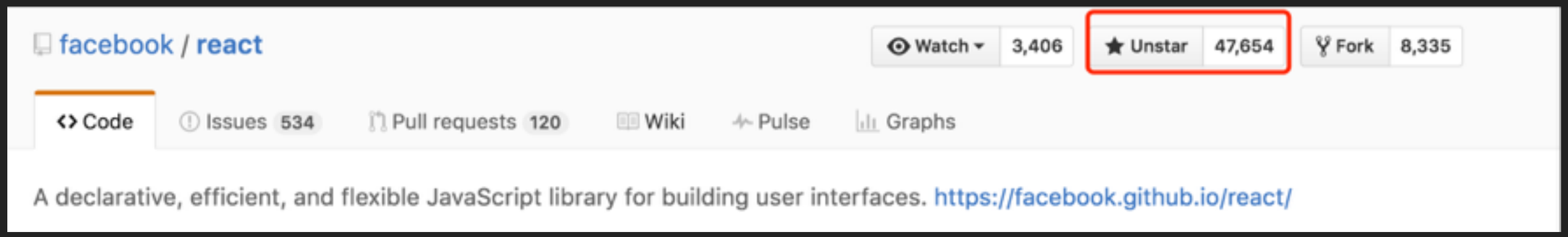


初见React

A JAVASCRIPT LIBRARY FOR BUILDING **USER INTERFACES**

起源于 Facebook 的内部项目，MVC

2013年5月开源



用**JSX语法**取代HTML模板，在JavaScript里声明式地描述UI

组件和基于组件的设计流程

虚拟DOM取代物理DOM作为操作对象

简单到**要命的**响应式更新

同时提出的Flux设计模式，**单向数据流**，用来管理**state**

IE8+

有哪些案例

一拍一

豆瓣市集

airbnb

Uber

淘宝

……

Hello World

JSX + inline style

```
class Hello extends React.Component {
  render() {
    let littleTitle = true
    return (
        <h1 style={{color: 'red'}}>{littleTitle? 'smallTitle': 'bigTittle'}</h1>
    )
  }
}

ReactDOM.render(<Hello />, document.getElementById('app'))
```

迭代快，之前都是0.13, 0.14, 之后的是15.xx

0.13 Beta 1 之后就可以用**ES6**

React/React Native 的ES5 ES6写法对照表

JSX本质上是为Virtual Dom准备的

JSX生成的真实DOM树的映射

\+

从服务端获取数据

Virtual Dom树 + 绑定事件

Virtual Dom

内存里的一种特殊的结构

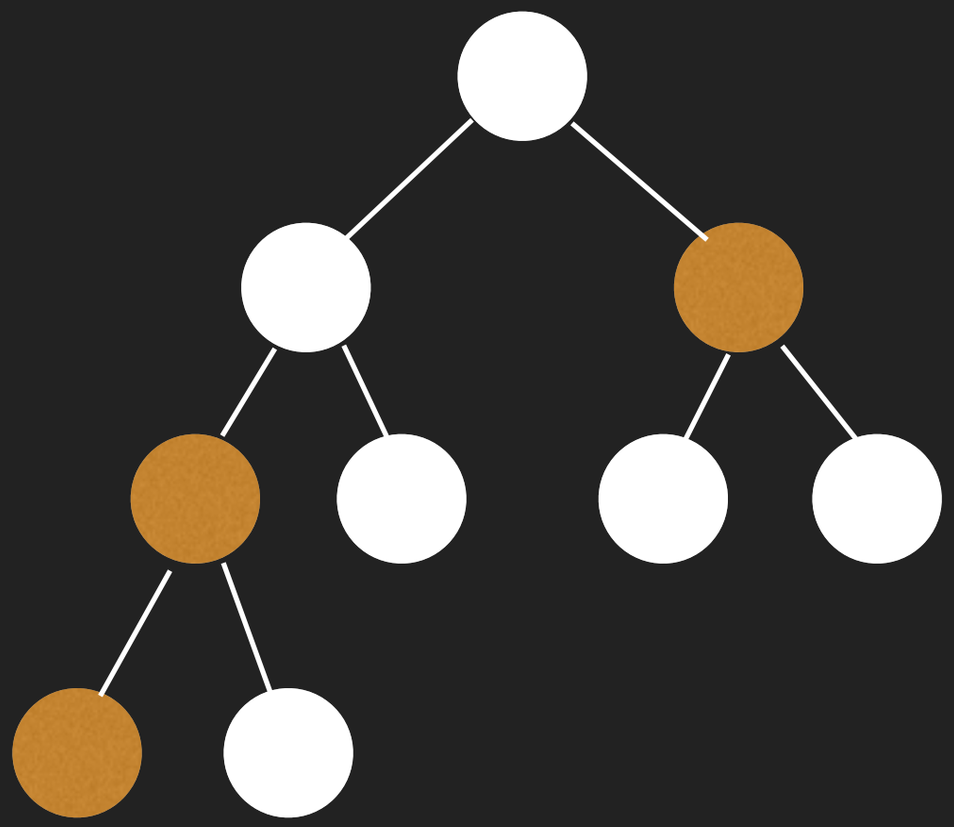
render()一次性全量更新真实 DOM

Diff 算法，更高性能的渲染，双刃剑

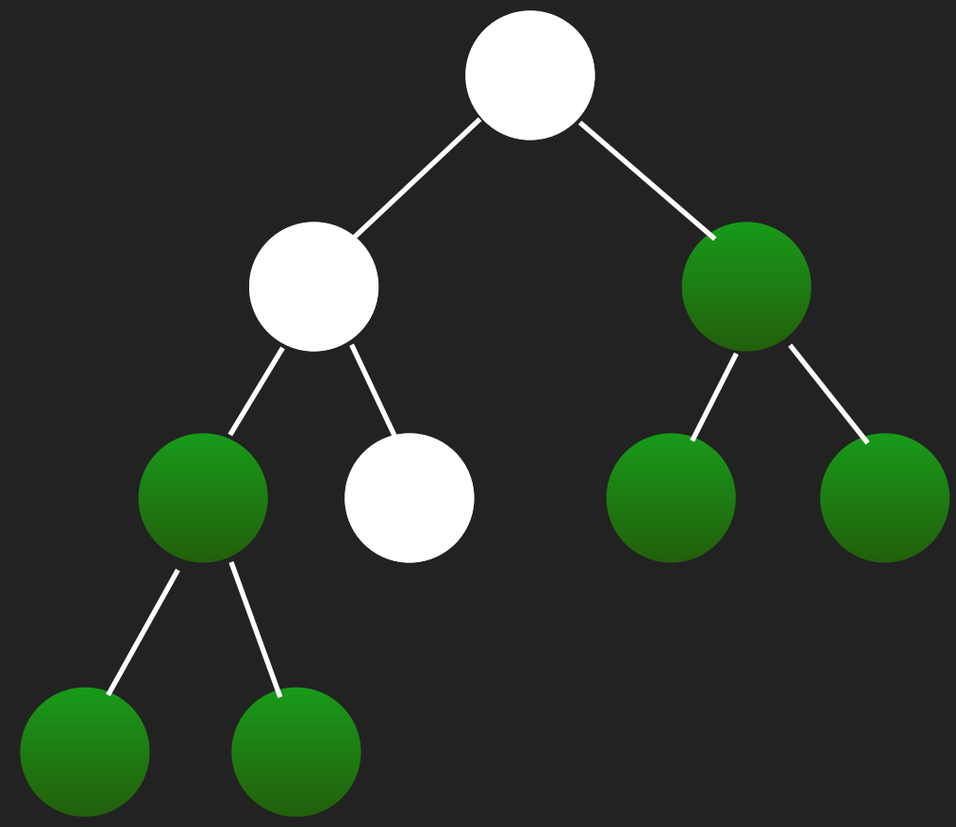
组件化开发的基础，解决代码臃肿、难以维护与测试等问题

详细的Diff算法

Render 流程



标记变化



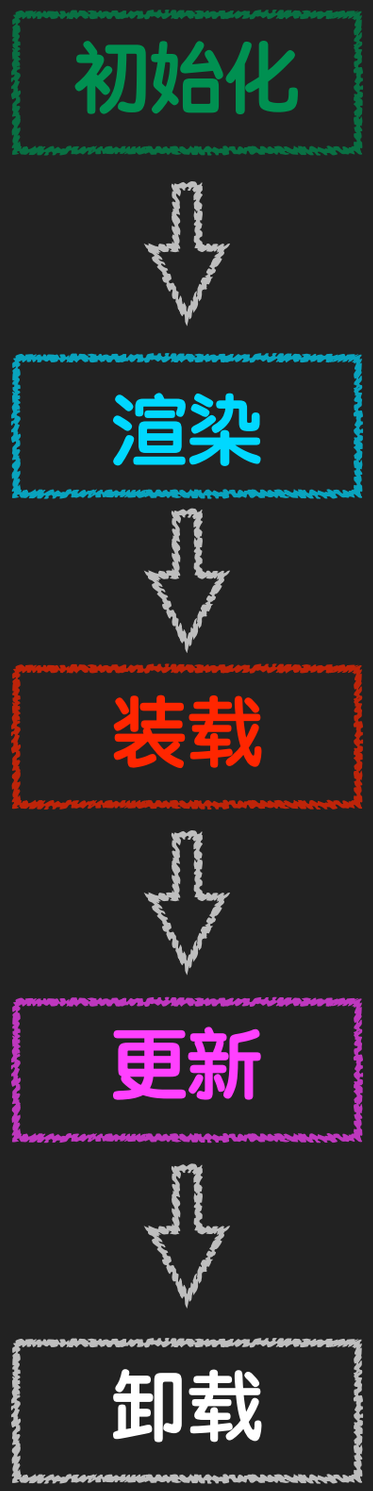
更新子树

支持服务端渲染

利于SEO

加速首屏渲染速度

生命周期



数据驱动，State+props

状态机，单向的

是一种思想，一种约束，一种标准

提出状态管理方法，Flux

目前比较好的实现，**Redux**

单向数据流 VS 双向数据流

单向

简单清晰，方向可追踪

更改麻烦，action->state

双向

方便！子->父，兄弟之间

数据相互依赖，源头难追踪

一个下拉菜单的两种状态

state: 收起

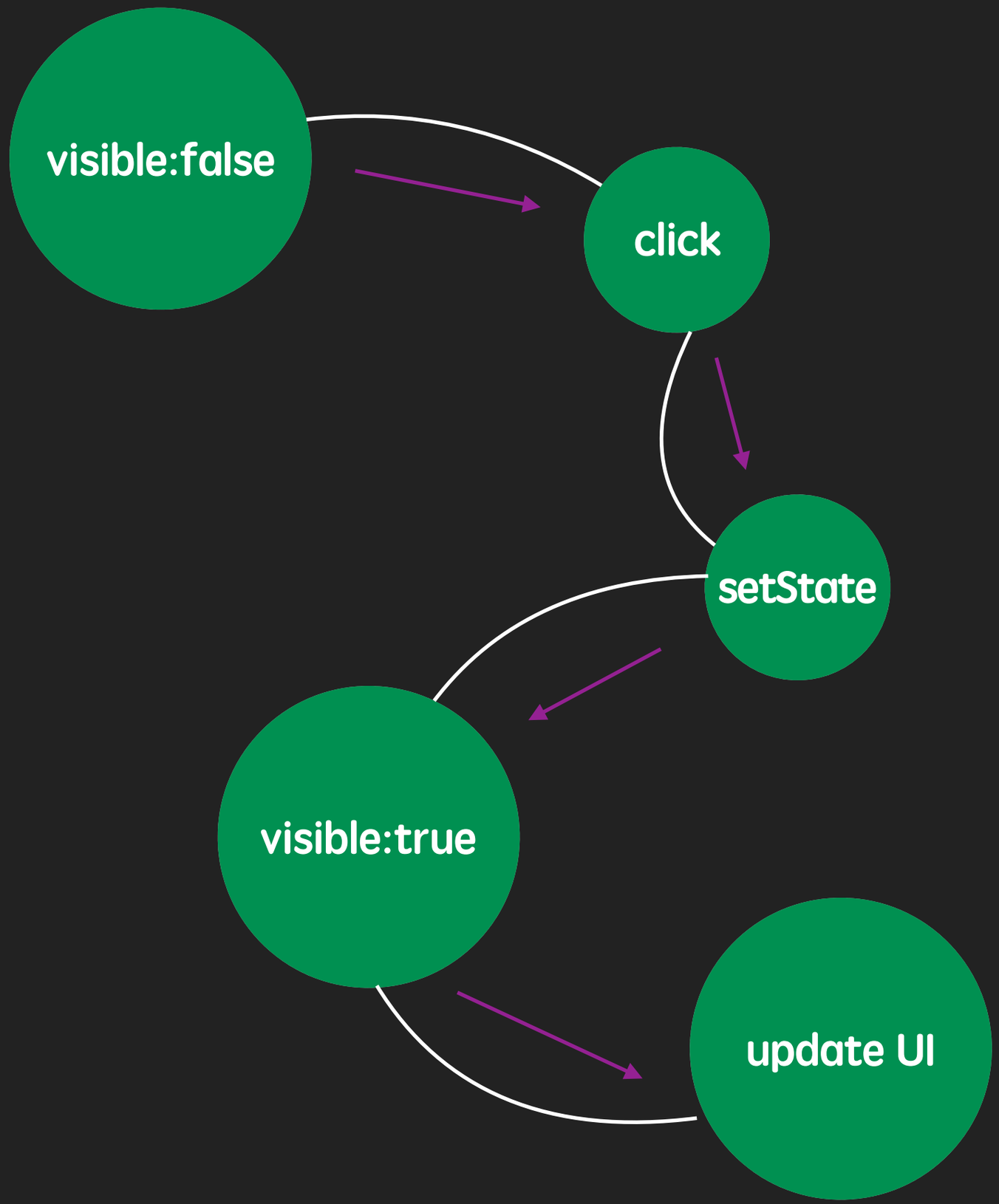
Dropdown ▼

state: 展开

Dropdown ▼

First link

Second link


visible:false
click
setState
visible:true
update UI

组件间通信

层级很深，怎么办?

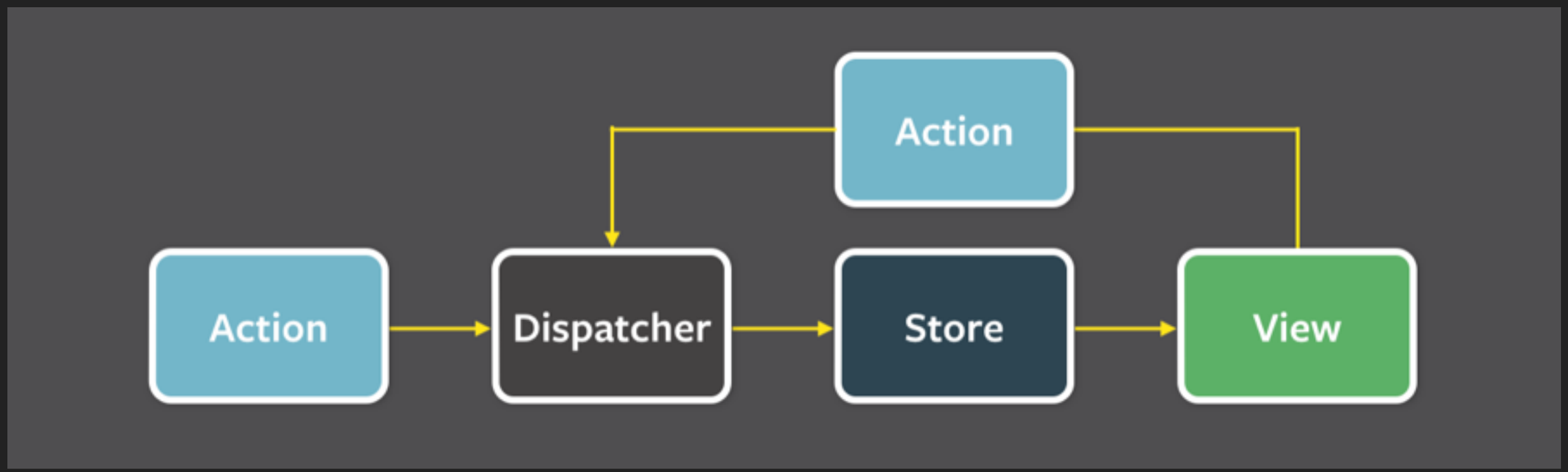
父子之间

父->子，this.props
子->父，回调

独立两个组件

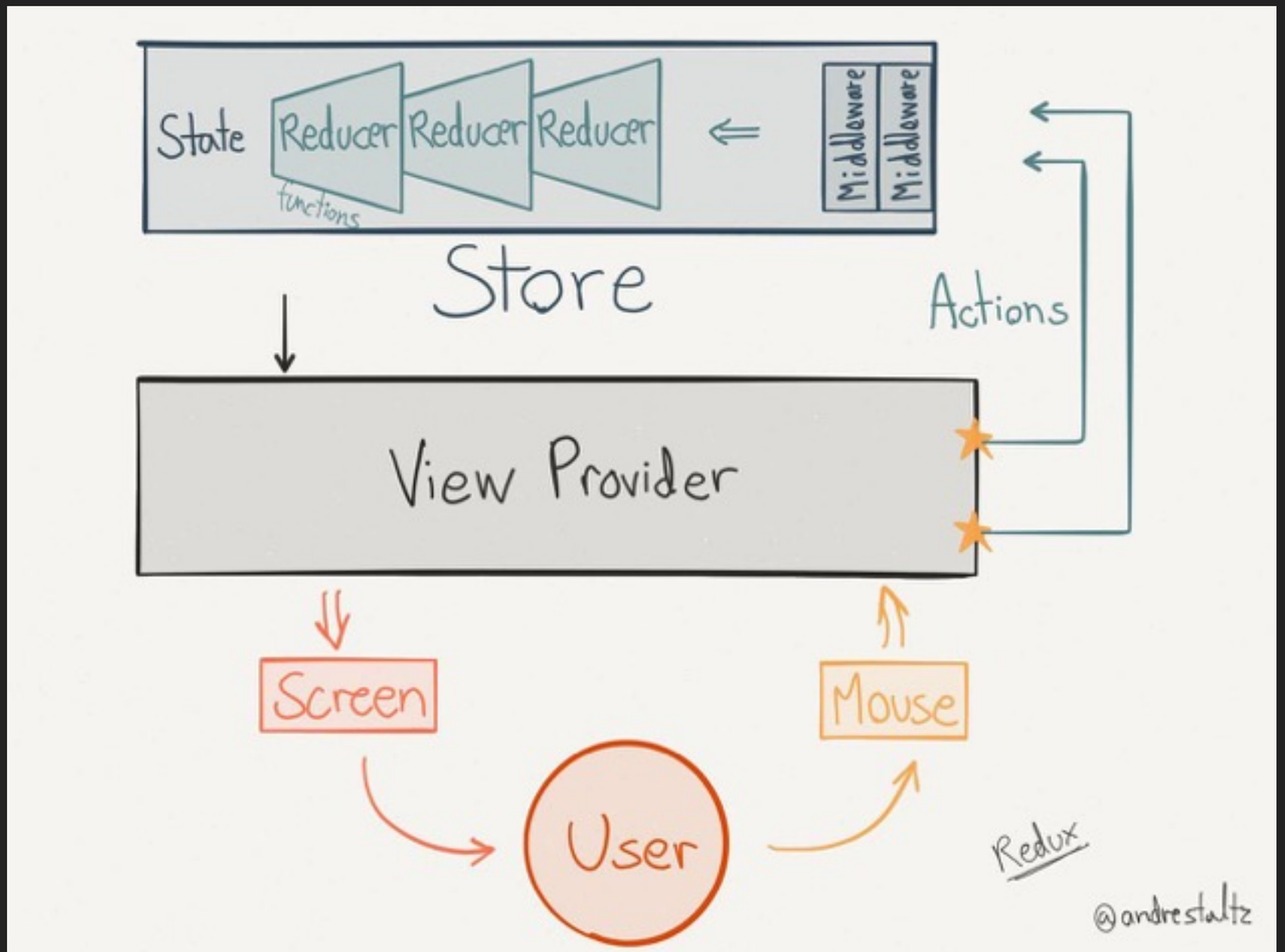
自行维护一个全局数据存储

Flux思维模型



Redux的工作流程

独立的，起源于React，但不止于
支持React、Angular、Ember...,甚至是jQuery



Redux中文文档

适用场景

Learn once, write anywhere.

计划让 React 成为通用平台的UI 开发工具

不止是UI, 跨平台/终端

得益于Virtual DOM，React自动将UI的改变渲染到真正的实现上

浏览器DOM、iOS UI、Android UI，终端文本UI。提高开发者体验

仅仅ReactJS

大型单页应用，复杂的大型应用

30+个页面100+个组件
伴随着大量的数据交互操作
多个页面的数据同步操作

用户交互比较多

Dom操作频繁的都可以用React

与其他框架的对比

jQuery

我们可以看看当我们尝试去修改一个功能或修复一个bug时的一般性做法：

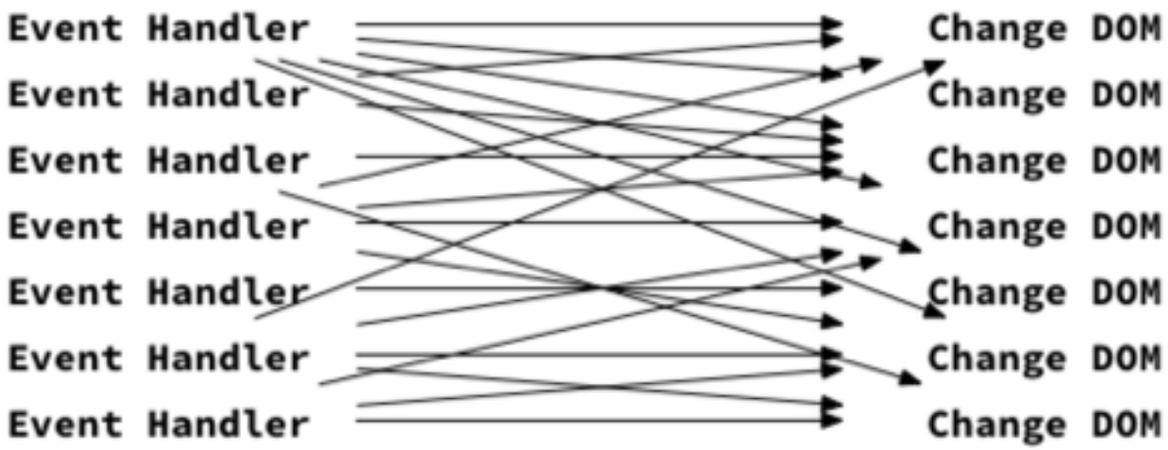
1. 寻找DOM（在浏览器审查元素）
2. 看一下这个DOM上比较像钩子的属性（`id`, `class`, `role`, `data-xxx`）
3. 搜索代码中用了这个钩子的地方（基本上通过文本搜索找`$()`）
4. 若命中则进入下一个地方，否则从1, 2, 3中的某一步重新开始，直到命中。

直接操作dom

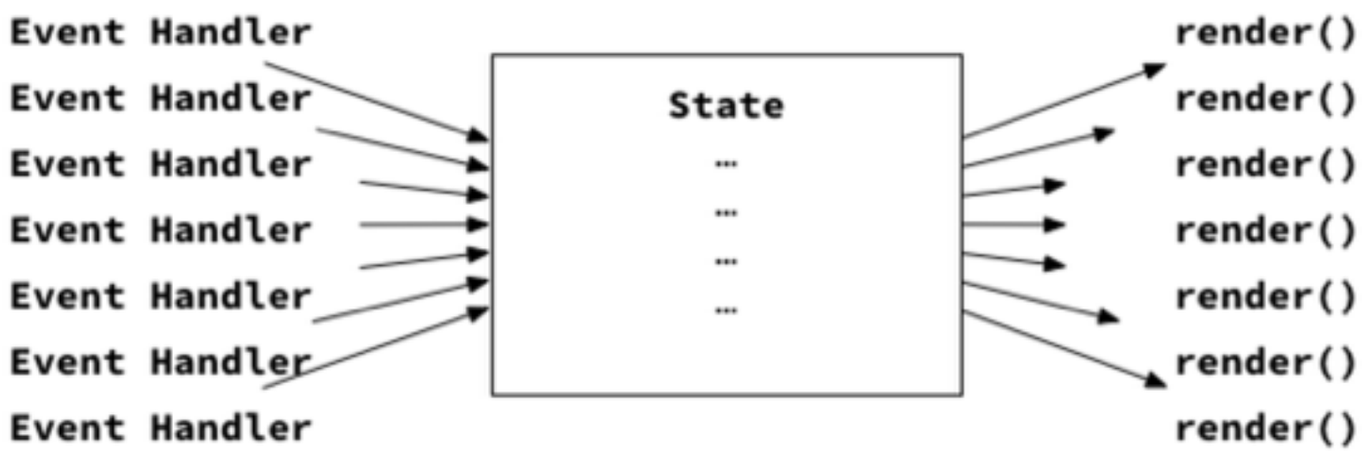
组件（插件）零散

大型应用，难预测，难维护，难扩展

jQuery Style



React.js Style



Vue

API 是以简单易上手为目标

html , js , css分离

1.x 使用真实 DOM 作为模板，数据绑定到真实节点

默认单向数据流

状态管理 Vuex，也可以与 Redux 一起用

2.0 **新增**Vitral Dom + 服务端渲染

IE8+

AngularJs

MVVM

双向数据流绑定

脏检查，性能问题

Polymer

面向未来，Web Component

兼容性差，Shadow Dom

框架并没有优劣之分，都是经验的聚合。

根据自身业务去选择

为大家的技术选型提供一些思路

React展望

组件化是趋势

强大的React 的社区/组件生态

越来越多的人参与

开发iOS/Android 应用不再是难事

2016React大会之React展望

知识储备

单向数据流管理，Redux

函数式编程

ES6

Webpack(一个模块加载打包工具)

Babel

单元测试

推荐一些资源

开发工具

React Developer Tools, Redux DevTools

Sublime中显示JSX语法的插件Babel

库

UI框架：react-bootstrap, amaze UI, Antd, …

组件：redux-form, redux-dnd

Router: react-router，react-router-redux

HTTP: axios

测试框架: enzyme

自己的一点体会

思维模式的转换，数据驱动，服务端提供接口

多人协作开发

状态是可预测，代码条理清晰

组件划分的粒度：根据项目

后期更多是熟悉其他不同的框架的API

还有一些问题

性能优化：React, Webpack

小项目使用Redux，反而代码量变大

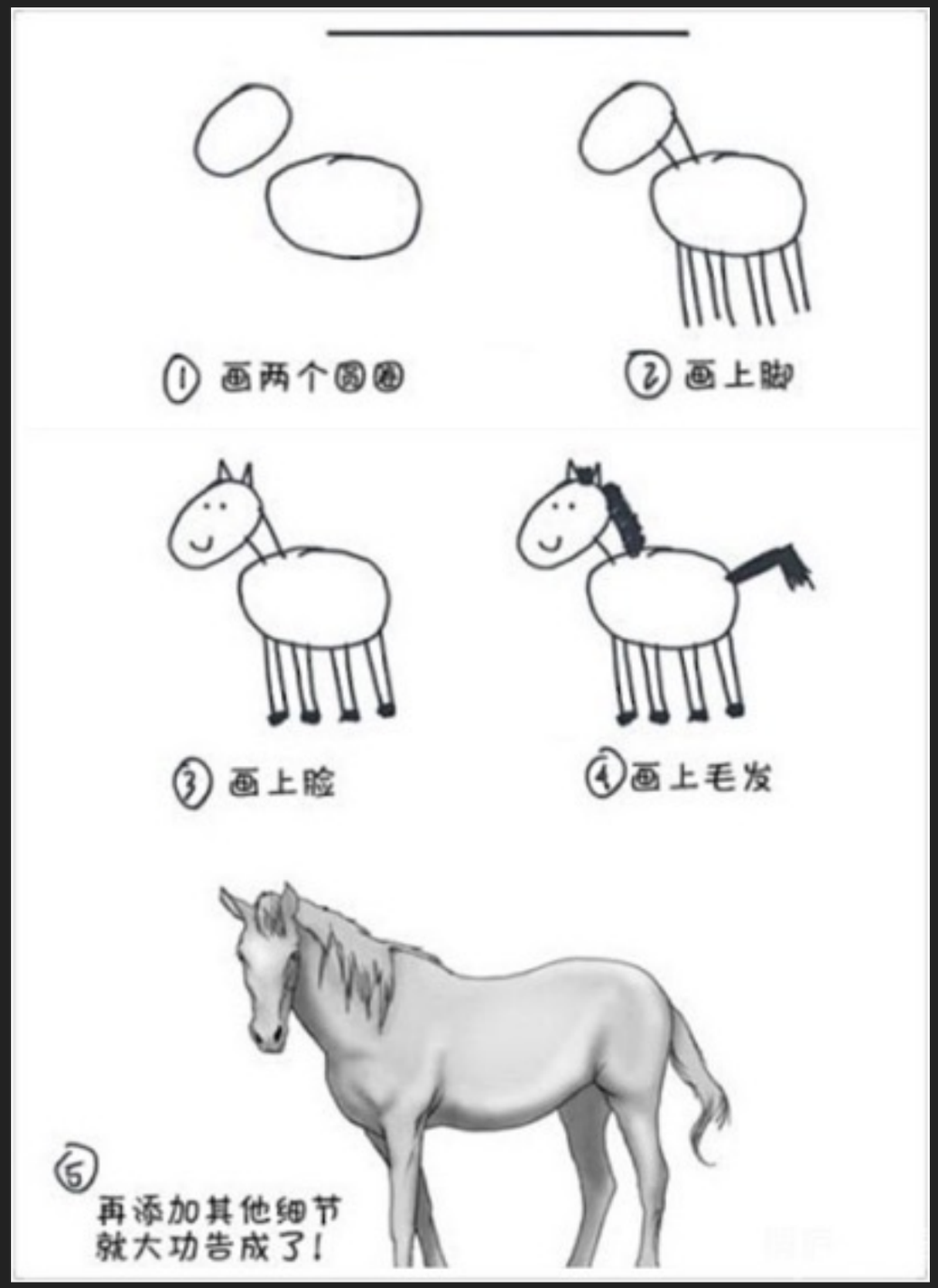
其他一些参考

React.js 2016最佳实践

React 入门实例教程

React China

为什么我选择了React来重构我们的前端



路漫漫其修远兮.....

（据说是接手新项目的程序员的表情）

愿我们无所“畏惧”……

Q&A

Thanks & End