**react总结**

# 一、基础知识

## 1.什么是React？它的优点是什么？

React 是 Facebook 开发的前端 JavaScript 库。它遵循基于组件的方法，构建可重用的 UI 组件。

优点：

1. 使用虚拟 DOM 而不是真实的 DOM，加快了页面渲染的速度

2.组件化，可以提高代码和组件的复用率

3.使用JSX，使得代码的可读性增加

## 2.什么是JSX

JSX全称为JavaScript XML 。本质上是React.createElement的语法糖。用来创建react元素

它结合了js和html。使得代码非常容易理解。

## 3.React必须使用JSX吗？

不是，JSX 是 React.createElement(component, props, ...children) 的语法糖。因此，使用 JSX 可以完成的任何事情都可以通过纯 JavaScript 完成。

## 4.React fiber

fiber 之前的 react存在的问题：

react的更新是同步的，不允许被外界打断， js 线程长时间占用主线程。在此期间，用户与页面的交互会得不到响应；渲染引擎也无法工作。

react fiber：

fiber 把react更新过程进行分片，每执行完一段更新过程，就将控制权交还主线程，看看是否有其他更高优先级的任务要做。有就去执行更高优先级任务，没有的话就继续更新

react把更新分为两个阶段

1.reconciliation(调解)

render之前，这个阶段的更新是可以被打断的，主要涉及的生命周期：

componentWillMount

componentWillReceiveProps

shouldComponentUpdate

componentWillUpdate

2.commit

render之后，这个阶段的更新是不能被打断的

componentDidUpdate

componentDidMount

componentWillUnmount

## React 合成事件

合成事件是react自定义的事件对象，与原生事件拥有相同的接口。

React会把所有的事件绑定到document上，当真实DOM触发事件之后，事件会冒泡到document上，react将事件封装并交给处理函数执行

好处：

1.跨平台，屏蔽各浏览器的底层差异

2.事件委托减少内存占用

3.合成事件采用了事件池，避免频繁地创建和销毁事件对象。

## 6.合成事件和原生事件的区别

1.命名方式不同

原生事件为全小写，react 事件采用小驼峰

2.调用时机不同

原生事件立即调用。合成事件要等到事件冒泡到document上才调用

3.绑定对象不同。

原生事件绑定在dom元素上，合成事件绑定在document上

4.阻止默认行为方式不同

原生事件返回false阻止默认行为，react必须显式调用preventDefault阻止

5.调用stopPropagation

合成事件调用事件身上的stopPropagation() 可阻止事件往上冒泡的，但由于事件绑定在document上，所以只能阻止 document 事件不能再冒泡到 window 上

但是原生事件调用stopPropagation() ，所有元素的事件将无法冒泡到document，所有react事件均无法执行

## 7.React.Component 和 React.PureComponent 的区别

PureComponent表示纯组件，用在纯展示组件上。减少render函数执行的次数

React.PureComponent会自动执行 shouldComponentUpdate并进行浅比较。

## 8.Element和Component之间有什么区别和联系？

一个组件中含有若干个元素

### 元素

元素是一个JavaScript对象，和界面上的一部分DOM对应，通过JSX语法创建

### 组件

组件分为类组件和函数组件。以props作为输入，以一棵元素树作为输出。

## 9.React.createClass和extends Component的区别有哪些？

二者都可以生成组件。区别主要在于：

### （1）语法区别

createClass是ES5的写法，本质上是一个工厂函数

extends的方式是最新的ES6规范写法。

### （2）声明状态

React.createClass：通过getInitialState()方法返回一个包含初始值的对象

React.Component：通过constructor设置初始状态

### （3）Mixins

React.createClass ：可以使用mixins中的代码

React.Component： React mixins不能被使用

## 10.Mixins

Mixins中定义的代码可以在不同的组件之间共享。包括函数和生命周期方法

## 11.函数组件和类组件的区别

1.类组件有state。函数组件无state

2.类组件有生命周期方法。函数组件无生命周期方法

3.类组件可以通过ref获取组件实例。函数组件无组件实例

## 12.对有状态组件和无状态组件的理解及使用场景

1.有状态组件使用state。无状态组件不使用state

2.有状态组件会频繁调用生命周期函数，影响性能。无状态组件不调用

3.有状态组件根据props和state进行渲染。无状态组件只根据props进行渲染

4.有状态组件一般是类组件。无状态组件函数组件类组件都可以

## 13.对React中Fragment的理解，它的使用场景是什么？

React组件返回的元素必须含有一个根元素。为了不添加多余的DOM节点，我们可以使用Fragment标签来包裹返回的元素

## 14.对React的插槽(Portals)的理解，如何使用，有哪些使用场景

Portals 使得组件可以脱离父组件层级而挂载在DOM树的任何位置。

Portals语法如下：

ReactDOM.createPortal(child, container);

第一个参数 child 是可渲染的 React 子项，比如元素，字符串或者片段等;

第二个参数 container 是一个 DOM 元素。

## 15.React中什么是受控组件和非控组件？

React中的组件根据是否受React控制可分为受控的和非受控的。任何时候都不需要修改组件value值时，使用非受控组件

### （1）受控组件

1.组件的value属性由React中的state控制

2.value值的修改通过在onChange函数中调用setState来完成

### （2）非受控组件

1.value不受react控制

2.通过ref来获取DOM节点，读取value。

## 16.React中refs

Refs 用于访问在 render 方法中创建的元素或节点。

1.字符串格式：<p ref="info">span</p>

2.函数格式：<p ref={ele => this.info = ele}></p>

3.createRef方法：React 16提供的一个API，使用React.createRef()来实现

## 17.React组件的构造函数有什么作用？它是必须的吗？

1.获取this。子类没有自己的this，而是通过调用super继承父类的this

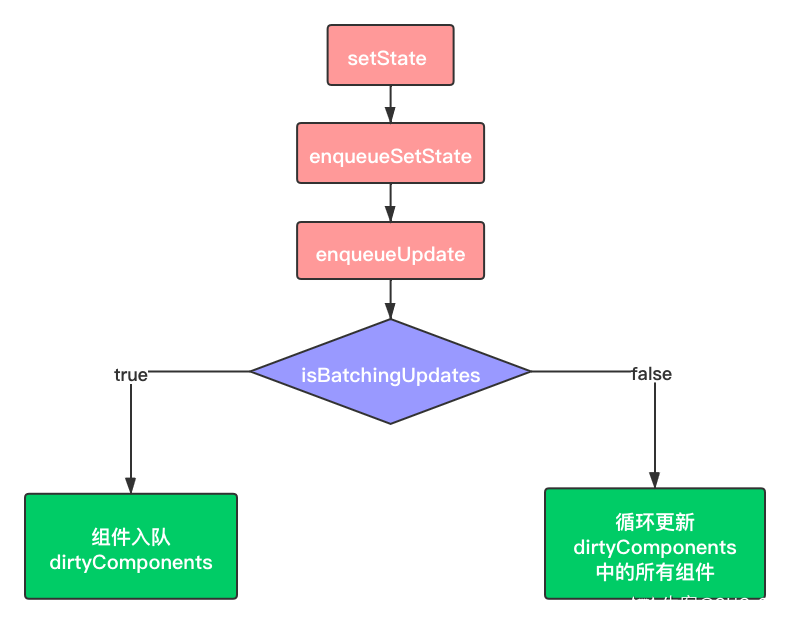
2.初始化状态

3.将事件处理函数绑定到实例上

# 数据管理

## 1.React setState 调用的原理

具体的执行过程如下（源码级解析）：



### 1.setState

将状态传递给enqueueSetState

### 2.enqueueSetState

方法将新的 state 放进组件的状态队列里

### 3.enqueueUpdate

该方法中有一个关键的对象——batchingStrategy，该对象的属性isBatchingUpdates 表明了react当前是否处于批量更新阶段，如果react处于批量更新阶段，那么该状态只能排队等待，等待下一次批量更新；如果react没有处于批量更新阶段，那么就进行批量更新。

## 2.setState是同步的还是异步的

某些情况下是同步的，某些情况下是异步的

1.setState 只在合成事件和钩子函数中是“异步”的，在原生事件和 setTimeout 中都是同步的。包括生命周期函数

2.setState的异步并不是真的异步，执行代码仍是同步。只是setState没法立马拿到更新后的值，形式了所谓的“异步”。

3.setState 异步本质上是为了批量更新。将多次对状态的修改合并为一次。减少操作真实DOM的次数，提升性能。

## 输出顺序为 0 0 2 3，click后输出3，但此时this.state.val实际值为4

## 4. setState之后发生了什么？

调用 setState 时，并不会立即更新state， react将要修改的 state 放入一个队列， 等待合适的时机进行更新，而且会将对同一状态的多次setState合并为一次

## 5.React中setState的第二个参数作用是什么？

setState 的第二个参数是一个可选的回调函数。这个回调函数将在组件重新渲染后执行，可以拿到更新后 state 的值。

## 6.React中的setState和replaceState的区别是什么？

setState 是修改其中的部分状态，状态的其余部分保持不变

replaceState 是完全替换原来的状态，将state 替换为一个新对象

## 7.React组件的state和props有什么区别？

（1）props

props是父组件传递过来的参数。具有可读性和不变性

（2）state

state是组件自身维护的状态。通过constructor进行初始化，通过setState修改

## 8.React中的props为什么是只读的？

React具有浓重的函数式编程的思想，函数式编程中有一个重要概念：纯函数。

纯函数有以下几个特点：

给定相同的输入，总是返回相同的输出。

过程没有副作用。（例如：修改变量、抛出异常）

不依赖外部状态。

## 9.React中怎么检验props？

根据PropTypes来检验Props，当Props类型不符合，控制台就会发出警告信息。

优点：

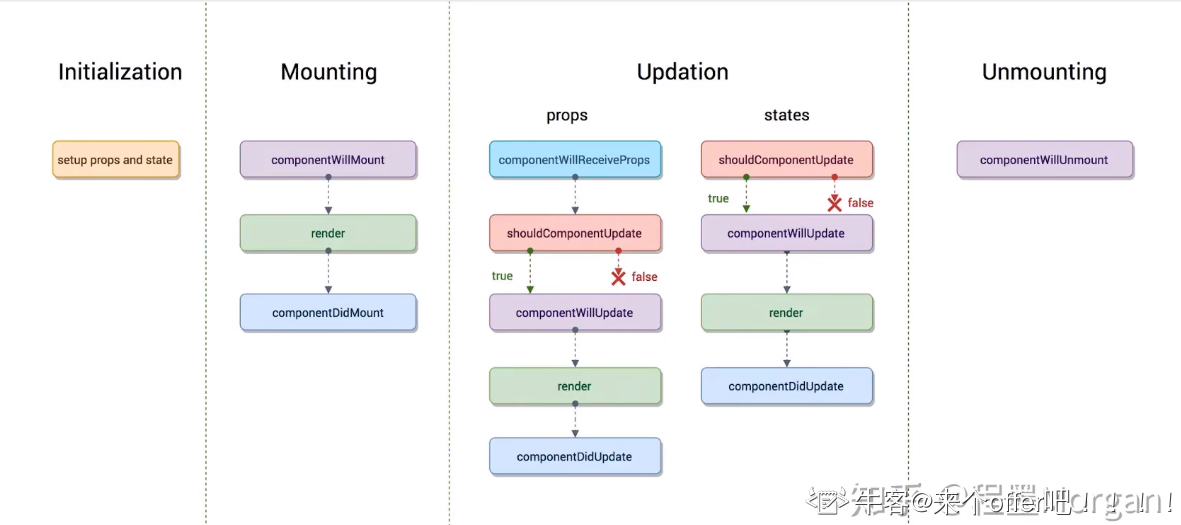
避免随着项目规模的扩大而出现的问题

可以让程序变得更易读

# 三、生命周期

## 1.React 生命周期

React 16.3 之前的生命周期



### 挂载生命周期

1.constructor(props)

初始化 props 和 State。

2.componentWillMount()：

componentWillMount是在页面第一次加载的时候做一些初始化的工作，在该函数中调用setState将不会产生效果，因此，不要在该函数中发送网络请求

3.render()

渲染

4.componentDidMount()

组件被渲染完毕之后触发，可以获取到 DOM。可以进行网络请求和setState

5.componentWillUnmount()

组件被卸载之前触发，在这里可以清除一些后台进程。

### 更新生命周期

更新生命周期是当组件 State 发生变化或者从父组件接收到新的属性时触发的一系列方法。

1.componentWillReceiveProps(nextProps)

当新的props被传递给了组件后才会调用。可以调用 setState 方法

2.shouldComponentUpdate(nextProps, nextState)

通过浅比较state和props，来判断是否进行更新操作

3.componentWillUpdate(nextProps, nextState)

组件更新之前触发。

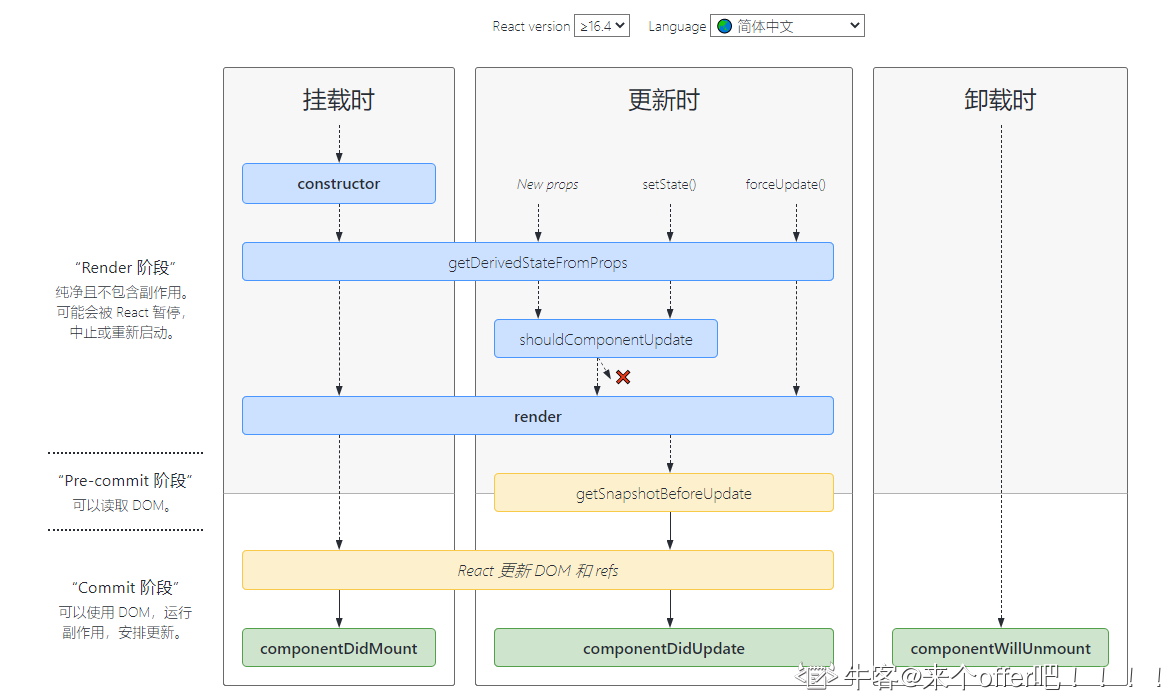
4.render（）

渲染

5.componentDidUpdate(prevProps, prevState)

只在更新操作发生后，调用 render 方法之后触发

React 大于等于 16.4 的生命周期图谱如下:



getDerivedStateFromProps，从props中获取state，是一个静态函数，不能使用this。通过参数提供的nextProps以及prevState来进行判断，是否根据新传入的props来映射state。（为了替代componentWillReceiveProps存在的）

## 2.setState 函数在任何情况下都会导致组件重新渲染吗？

1.setState中传入null，不会触发重新渲染

2.shouldComponentUpdate 返回false

## 3.哪些方法触发 React 重新渲染？重新渲染 render 会做些什么？

### （1）哪些方法触发 react 重新渲染?

1.setState（）

通常情况下，执行 setState 会触发 render。

2.父组件重新渲染

只要父组件重新渲染，即使 props 未变化，子组件也会调用 render

3.forceupdate

强制更新

### （2）重新渲染 render 会做些什么?

会对比新旧虚拟DOM，也就是Diff算法。

## 4.React中可以在render访问refs吗？为什么？

不可以，render 阶段 DOM 还没有生成，无法获取 DOM。DOM 的获取需要在 pre-commit 阶段和 commit 阶段：

## 5.对componentWillReceiveProps 的理解

1.当props发生变化时被调用

2.可以调用setState()，不会触发额外的render调用。

## 6.React 废弃了哪些生命周期？为什么？

componentWillMount、componentWillReceiveProps、componentWillUpdate

1.fiber的出现，会导致render之前的生命周期函数被打断，从而执行多次的情况。

2.他们的功能可以被新的生命周期函数或者之前的生命周期函数替换

## 7.React中发起网络请求应该在哪个生命周期中进行？为什么？

在componentDidMount获取数据

### 1.componentWillMount并不会导致提早获取数据

componentWillMount比componentDidMount早不了多少微秒，网络上任何一点延迟，这一点差异都可忽略不计。

### 2.componentWillMount中setState无法触发render

componentWillMount方法在render之前，调用setState方法不会触发重新render，所以它一般不会用来作加载数据之用。

componentDidMount方法在render之后，调用setState方法会触发重新渲染。

### 3.fiber会导致componentWillMount被执行多次

fiber的出现会导致render之前的生命周期函数被打断，从而执行多次的情况

# 四、组件通信

## 1. 父子组件的通信方式？

父组件向子组件通信：props

子组件向父组件通信：: props+回调的方式

## 2. 一般的组件通信方式？

1.寻找公共父组件，利用props传递

2.context、redux等全局状态管理

# 五、路由

## 1. React-Router的种类和原理？

react基于单页面应用，因此要通过不同的路径来让该页面加载不同的组件

两种路由：

### 1.HashRouter

HashRouter使用的是url的hash值，路径中包含#，相当于锚点定位

路由改变时，通过hashchange事件，感知 hash 的变化，选择加载对应的组件

Onhashchange:当一个窗口的 hash（URL 中 # 后面的部分）改变时就会触发 hashchange 事件

### 2.BrowserRouter

使用的是HTML5的新特性history API，低版本浏览器可能不支持。history对象提供了访问浏览器历史记录的接口。

## 2.HashRouter和BrowserRouter的区别

1.底层原理不一样：

BrowserRouter使用的是H5的history API，不兼容IE9及以下版本。

HashRouter使用的是URL的哈希值。

2.path表现形式不一样

BrowserRouter的路径中没有#,例如：localhost:3000/demo/test

HashRouter的路径包含#,例如：localhost:3000/#/demo/test

3.刷新后对路由state参数的影响

BrowserRouter没有任何影响，因为state保存在history对象中

HashRouter刷新后会导致路由state参数的丢失

4.传递参数

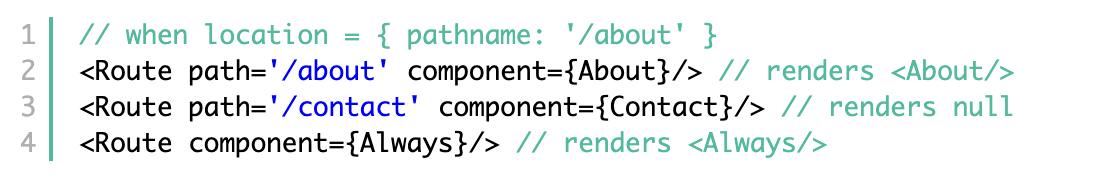
BrowserRouter进行组件跳转时，调用this.props.history.push，可以传递任意参数

HashRouter一般配合redux使用，实现组件间的数据通信。

## 3. <route>、<switch>、<Link>

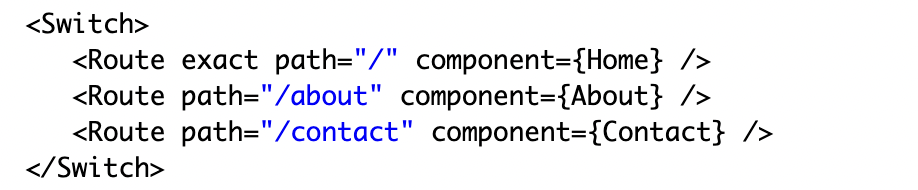
### （1）使用 <route>

用来进行路由匹配，当匹配成功时，它将渲染<route>中的组件。



### （2）结合使用 <switch> 和 <route>

<switch> 提高匹配效率，当匹配到第一个元素时就不再匹配



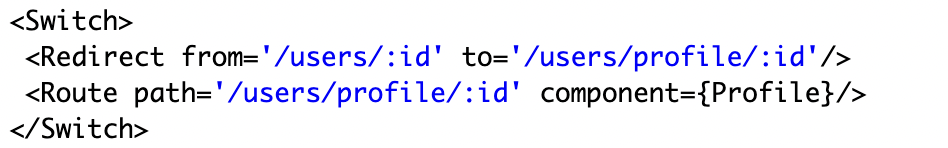
### （3）使用 <Link>、 <NavLink>

<Link> 是配合<route>使用的链接

<NavLink> 可以实现高亮效果

## 4. React-Router中的<redirect>

使用<redirect>实现路由的重定向，来将某个路径重定向到另一个路径



## 5. react-router 里的 Link 标签和 a 标签的异同

从最终渲染的 DOM 来看，这两者都是链接。

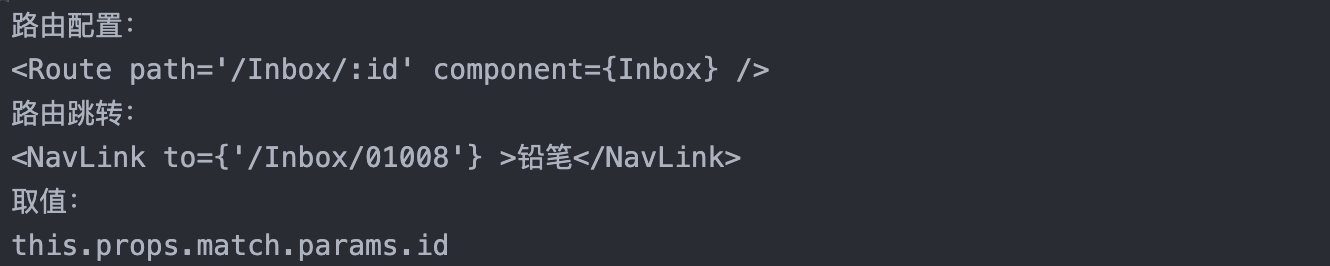
区别是∶

1.Link一般配合<route> 使用，实现路由的跳转。a标签单独使用

2.Link只会对<route>匹配的组件进行更新，a则会刷新整个页面

## 6. React-Router传递参数

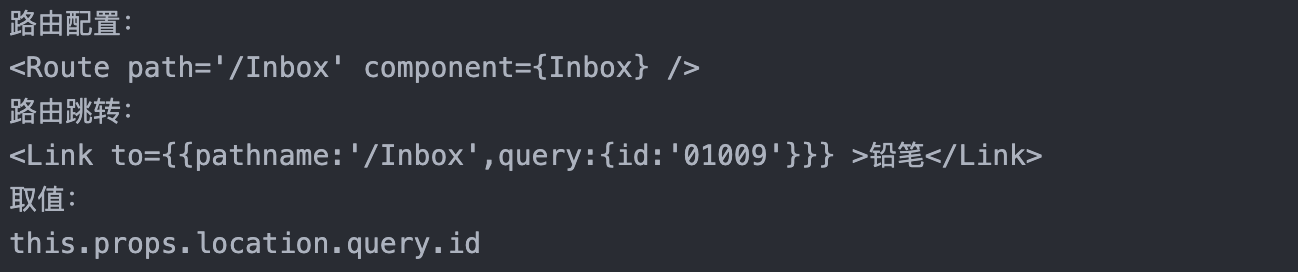
### 1.params



优势：刷新，参数依然存在

缺点：只能传字符串，传参过多url会变长

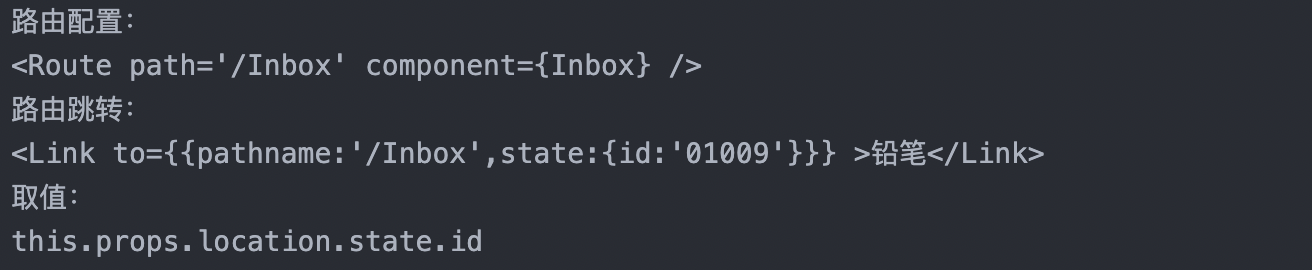
### 2.query



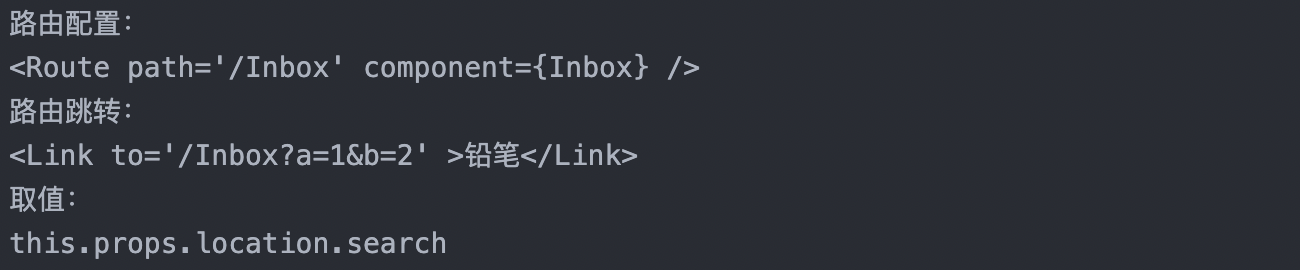
优势：可以传递对象

缺点：刷新地址栏，参数丢失（无论hash方式还是Browser模式）

### 3.state



优势：可以传递对象；state传的参数是加密的，query传的参数是公开的

缺点：刷新地址栏，hash方式会丢失参数，Browser模式不会丢失参数4.search

需要自己解析参数

## 7.路由组件和普通组件的区别

1.属性不同:

一般组件: <Demo/>

路由组件: <Route path="/demo" component={Demo}/>

2.存放位置不同:

一般组件: components

路由组件: pages

3.接收到的props不同:

一般组件: 写组件标签时传递了什么，就能收到什么

路由组件: 接收三个固定的属性

history:

go: ƒ go(n)

goBack: ƒ goBack()

goForward: ƒ goForward()

push: ƒ push(path, state)

replace: ƒ replace(path, state)

location:

pathname: "/home"

search: ""

state: undefined

match:

params: {}

path: "/home"

url: "/home"

# 六、Redux

## 1.Flux

Flux是一种架构理念，采用单项数据流的原则

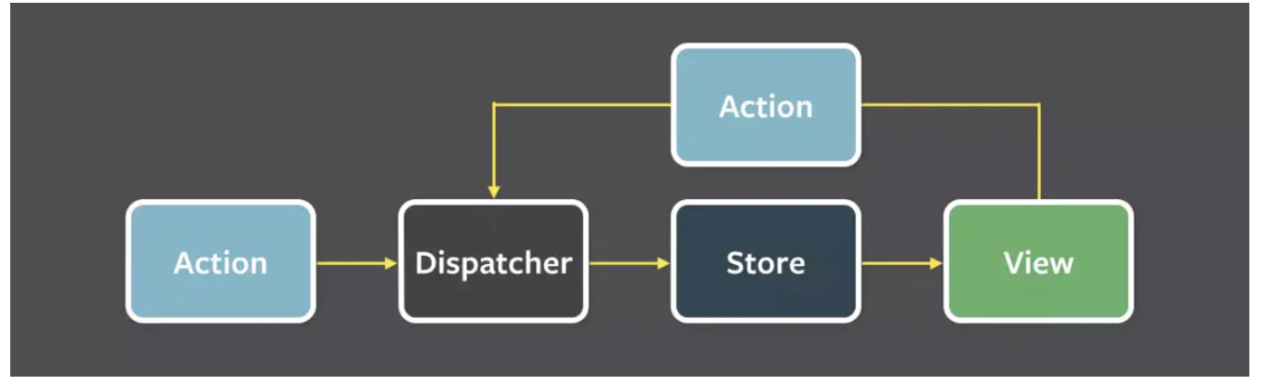
包括四个组成部分：

View：视图层

Action：视图发出的消息

Dispatcher：派发者，用来接收Action，执行回调函数

Store：数据层，存放状态



Flux 在进行数据更新时，会经历以下几步：

1.用户与 View 层交互，触发 Action

2.Action 通过dispatch 函数发送给dispatcher

3.dispatcher 通过Action传入的类型来触发对应的回调

4.Store 里进行相应的数据更新

5.View层收到新数据并进行更新

## 2.redux和flux的区别

flux是一个架构理念，redux是flux的一种实现

flux中只是提出了单向数据流，redux在此基础上提出三个原则：

### 1.唯一数据源

在redux的应用中只能有一个state数据源；

### 2.保持状态只读

在view层只能对state数据进行读取，修改状态只能通过action

### 3.状态改变只能通过纯函数完成

reducer函数只做状态的运算不做状态的存储

## 3. 对 Redux 的理解

Redux是一个状态管理工具。redux将管理状态的状态机和React的UI呈现绑定在一起，通过action改变state，并通知react更新页面。

为什么需要redux？

1.state 在大项目中相当复杂。

2. react 中组件间通信只能通过props完成

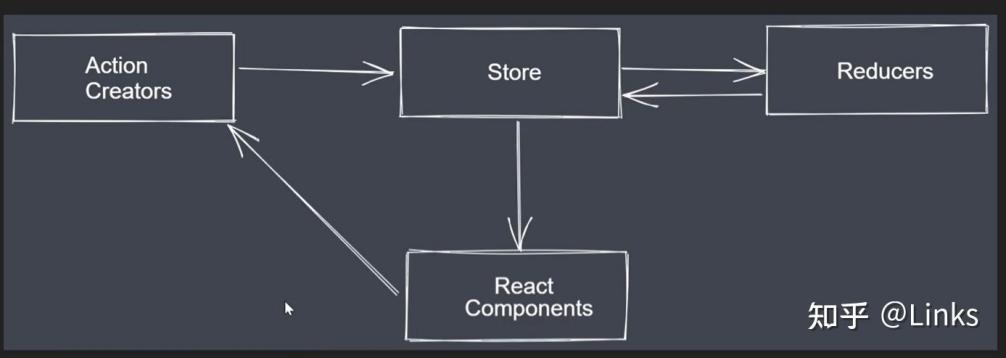
## 4.UI组件和容器组件

UI 组件负责呈现页面

容器组件负责管理数据和逻辑。

两者通过Redux 的connect方法联系起来

## 5. Redux工作流程



1. 在view层通过交互产生Action

2.action通过dispatch传递给store

3.store将接收到的action，和之前的state，一起传递给reducer

4.reducer进行计算，并向store返回新的state

5.store更新state，并通知视图层

6.视图层根据新的state进行渲染

## 6. Redux 中间件是什么？

Redux 的中间件是middleware ，可以对react的相关功能进行增强，位于 action 之后，到达 reducer 之前，在此期间可以做一些"副作用"的操作，如异步请求等

原本：view -→> action -> reducer -> store

加上中间件：view -> action -> middleware -> reducer -> store

两种常用的中间件：

（1）react-thunk中间件

（2）redux-saga中间件

## 7. Redux中的connect有什么作用

connect负责连接React和Redux。connect监听store，获取redux中的state和aciton，并通过props将state和action传递给组件。react使用Provider来包裹子组件，connect读取provider中的store，获取state和action传递给组件

## 8.对 React context 的理解

同redux一样，是一个状态管理工具，可以在组件间共享状态，原理是生产者和消费者模式

## 9.context和redux区别

1.Context适用于规模较小的项目。redux适用于规模较大的项目。

2.context没有action，直接修改state。redux通过action修改state。

3.context不能使用redux的中间件；redux可以使用中间件

## 10.redux有什么缺点

1.默认只支持同步处理

2.使用很繁琐、开发效率低

3.每次state改变会导致所有connect的组件重新render

# 七、Hooks

## 1. 对 React Hook 的理解

为函数组件补全相比于类组件缺失的state和生命周期函数等概念

## 2. 为什么 useState 要使用数组而不是对象

方便解构，使用数组可以直接按照顺序解构，而使用对象的话就必须跟useState内部的对象同名，多次使用需要取别名

## 3. React Hooks 解决了哪些问题？

（1）在组件之间复用状态逻辑很难

hook出现之前，使用 render props 和高阶组件。但是需要对原组件进行嵌套和包裹，会形成嵌套地狱。

Hook 可以在不修改组件结构的情况下复用状态逻辑。

（2）复杂组件变得难以理解

类组件中，每个生命周期常常包含一些不相关的逻辑，且不可拆分

Hook 将组件中关联的部分拆分成更小的函数，而并非强制按照生命周期划分

1. 难以理解的 class

## 4. React Hook 的使用限制有哪些？

1.不要在循环、条件或嵌套函数中调用 Hook；

React通过单链表管理Hooks ，每次调用useState，链表就会后移一位，如果判断不成立，某个useState没有执行，会导致后面的所有useState出现错位

2.只能函数组件中调用 Hook。

## 5. useEffect 与 useLayoutEffect 的区别

（1）共同点

运用效果：都用于处理副作用，包括改变 DOM、操作定时器等。

useEffect执行时机相当于componentDidMount

（2）不同点

调用方式：

useEffect 异步调用，在页面渲染后执行。如果useEffect 又改变了DOM，会重新渲染，引起页面的闪烁

LayoutEffect 同步调用，在DOM 建立之后、页面渲染之前执行，不会产生闪烁

执行顺序：

useLayoutEffect总是比useEffect先执行。

## 6. React Hooks 和生命周期的关系？

Hooks 让函数组件拥有了state，也有了生命周期的概念，主要是 useState、 useEffect() 和 useLayoutEffect() 。

例如：useEffect相当于componentDidmount和componentDidUpdate、useMemo相当于shouleComponentUpdate、useEffect中的return相当于componentWillUnmout

## 7.常见的hooks方法：

useState

返回 state及更新该 state 的函数。

useEffect

接收产生副作用的函数并执行

useRef

返回一个ref 对象，可以将该ref对象指向组件内的某个元素。

useContext

接收一个 context 对象并返回该 context 的当前值。

useMemo

类似于类组件中的shouldComponentUpdate，通过使用useMemo来减少相同props情况下的不必要渲染。

useCallback

传入回调函数及依赖项数组， useCallback将在某个依赖项改变时才会更新。

# 八、虚拟DOM

## 1. 对虚拟 DOM 的理解

虚拟 DOM 是一个 JavaScript 对象，它是对真实DOM的抽象。通过比较新旧虚拟dom，来减少对真实dom的操作。

优点：

（1）提高性能

（2）跨平台，解决兼容性问题

## 2.真实 DOM 和虚拟 DOM有什么区别？

真实DOM是DOM树，虚拟DOM是js对象

真实DOM可以更新HTML，虚拟DOM不行

真实DOM更新耗费大量资源，虚拟DOM更新很简单

## 3. React diff 算法的原理是什么？

### 1、只会对同一层次的节点进行比较

### 2、不同类型元素会产生不同的树形结构，相同类型元素会产生相同的树形结构

### 3、同一层级的节点可以通过唯一的key进行区分，提高效率

## 4. React key 是干嘛用的

Diff 算法借助元素的 Key 值来判断该元素是新创建的还是被移动而来的元素，从而减少不必要的更新。某个元素的 key 在其同级元素中具有唯一性

# 九、其他

## 1.react 最新版本增加了哪些东西

React 16三大新特性 Time Slicing、Suspense、 hooks

Time Slicing：fiber的进一步升级，划分时间片来执行不同的任务

Suspense：悬停，配合react.lazy可以实现懒加载

Hooks：为函数组件添加了state和生命周期的概念

## 2.**. React 数据持久化**

1.localStorage，数据永不消失，保存在硬盘中

2.sessionStorage，浏览器页面关闭，数据消失

3.redux，存储在内存中，页面刷新会导致redux中的状态重置

4.redux-persist，会将redux中的数据存储到localStorage或者sessionStorage中

## 3.TypeScript

Typescript语言是JavaScript的超集，并扩展了 JavaScript 的语法。Typescript支持强类型，提供编译时的静态类型检查

## 4.React 组件复用的方式

### Mixin

定义：Mixins允许多个React组件之间共享代码

缺点：

1.mixin之间会形成依赖链，当改动其中一个mixin的时候，会影响其他的mixin

2.不同mixin会产生命名冲突

### 高阶组件

定义：高阶组件（HOC）就是一个函数，且该函数接受一个组件作为参数，并返回一个新的组件，他会返回一个增强的React组件。

缺点：

HOC需要在原组件上进行包裹或者嵌套，大量嵌套会导致组件难以理解

### render props

render props中，子组件的逻辑和数据依赖于父组件，父组件通过设置子组件的render属性，来将逻辑和数据传递给子组件。

缺点：使用繁琐；嵌套过深

### hooks

将函数作为最小的代码复用单元

优点：

解决了HOC和render props的嵌套问题

## 5.React的状态提升是什么？

将多个组件需要共享的状态提升到它们最近的父组件上，通过props进行传递

## 6.React的严格模式如何使用，有什么用处？

StrictMode 可以为某些组件开启严格模式，显示程序中的潜在问题

优点：

检测不安全的生命周期

检测过时的ref

## 7.react.js、react-dom.js和babel.js都有什么作用？

react：包含react所必须的核心代码

react-dom：react渲染在不同平台所需要的核心代码

babel：将jsx转换成React代码的工具

## 8.对React SSR（服务端渲染）的理解

客户端渲染：服务端只返回一个html模板，页面内容通过js渲染

服务端渲染：服务端直接返回拼接好的html，浏览器只负责展示

SSR的优势：

1）更利于SEO

2）更利于首屏渲染

SSR的局限：  
服务端压力较大

## 10.如何理解“在React中，一切都是组件”？

1.组件是react中的核心概念，通过组件来提高代码的复用率

2.React中的state、props和生命周期函数都是基于组件实现的

## 11.createElement 与 cloneElement 的区别是什么

createElement 用来创建 React 元素

cloneElement 用于复制某个元素并传入新的 Props

## 12.React Intl是什么原理

react国际化的一个库，可以切换不同的语言文字

## 13.对webpack的理解：

webpack 是 JavaScript 打包工具。它根据项目所需的模块，并生成一个或多个 bundle

## 14.react 性能优化的手段

1.避免不必要的 render，主要包括 shouldComponentUpdate 和 PureComponent

2.使用 Suspense进行组件的懒加载

## 15.前端中的MV\*模式

所有MV\*的交互模式本质都是对DOM交互的抽象。

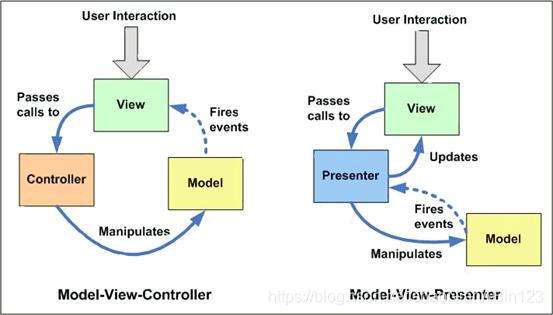
### 1.MVC模式

MVC包括即数据模型、视图、控制器三个部分。

M: 数据模型，负责存取数据结果；

V：视图，用来承载DOM的展示，包括更新渲染；

C：控制器，通过切换不同的model，来让view加载不同的数据



### 2.MVP模式

MVP 是从MVC演变而来，其中的presenter替代了controller

区别：

MVP中View并不直接使用Model，它们之间的通信是通过Presenter来进行的

MVC中View会从直接Model中读取数据

### 3.MVVM

MVVM可以认为是一个自动化的MVP，使用了ViewModel替代了Presenter。

ViewModel负责把Model的数据同步到View显示出来，还负责把View的修改同步回Model

## 16. spa和mpa区别

spa：单页面应用 mpa：多页面应用

1.刷新方式。SPA局部刷新或更改，MPA整页刷新

2.路由模式。SPA使用hash或history， MPA：普通链接跳转

3.用户体验。SPA页面切换快，用户体验良好，当初次加载文件过多时，需要调优。MPA页面切换加载缓慢，用户体验比较差，

5.数据传递。SPA传递数据可以通过路由传参，MPA依赖url传参，cookie，本地存储

6.搜索引擎优化（SEO）。SPA：实现较为困难，可以使用服务器端渲染（SSR） MPA：实现方法容易

## 17.redux和react-redux区别

redux：redux是react中进行state状态管理的JS库

react-redux:React-Redux是Redux的官方React绑定库

区别：

1.获取state的方式不一样

redux获取state是直接通过store.getState()。

react-redux获取state是通过mapStateToProps函数，只要state数据变化就能获取最新数据

2.触发action的方式不一样

redux是使用dispatch直接触发，来操作store的数据

react-redux是使用mapDispathToProps函数然后在调用dispatch进行触发

## 18.react中传递数据的方式

1.props

2.路由传参

3.状态提升

4.context

5.redux

## 19.context的原理

创建一个装上下文的容器，在该容器的子树中，使用生产者/消费者模式来传递数据，生产者provider提供数据，消费者consumer读取数据使用

## 20.观察者模式

当对象间存在一对多关系时，则使用观察者模式（Observer Pattern）。比如，当一个对象被修改时，则会自动通知依赖它的对象。观察者模式属于行为型模式。

应用：消息订阅与发布

观察者模式和消息订阅的一些区别：

观察者模式知道观察者是谁，一旦发生改变，可以通知观察者

消息订阅与发布里面，发布者和接收者不知道对方的存在

## 21.如何减少首页等待时间

1.DNS预解析。在访问链接之前提前解析域名。

使用方法：<link rel="dns-prefetch" href="xxx.com">

2.异步加载JS

3.服务端渲染

4.懒加载

5.预加载