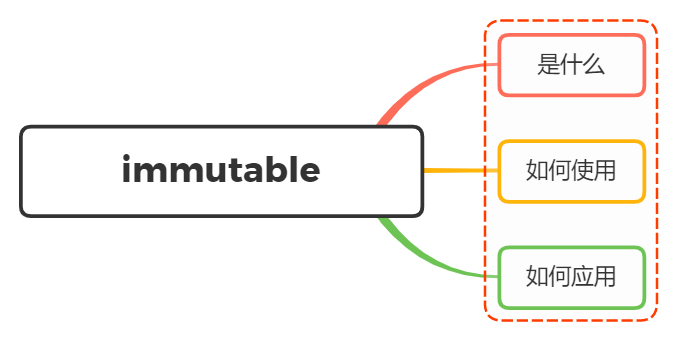
# 面试官：说说你对immutable的理解？如何应用在react项目中？



## 一、是什么

Immutable，不可改变的，在计算机中，即指一旦创建，就不能再被更改的数据

对 Immutable对象的任何修改或添加删除操作都会返回一个新的 Immutable对象

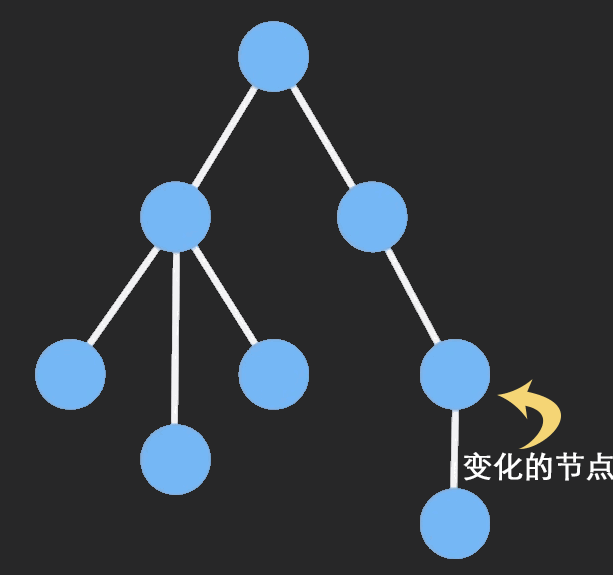
Immutable 实现的原理是 Persistent Data Structure（持久化数据结构）:

* 用一种数据结构来保存数据
* 当数据被修改时，会返回一个对象，但是新的对象会尽可能的利用之前的数据结构而不会对内存造成浪费

也就是使用旧数据创建新数据时，要保证旧数据同时可用且不变，同时为了避免 deepCopy把所有节点都复制一遍带来的性能损耗，Immutable 使用了 Structural Sharing（结构共享）

如果对象树中一个节点发生变化，只修改这个节点和受它影响的父节点，其它节点则进行共享

如下图所示：



## 二、如何使用

使用Immutable对象最主要的库是immutable.js

immutable.js 是一个完全独立的库，无论基于什么框架都可以用它

其出现场景在于弥补 Javascript 没有不可变数据结构的问题，通过 structural sharing来解决的性能问题

内部提供了一套完整的 Persistent Data Structure，还有很多易用的数据类型，如Collection、List、Map、Set、Record、Seq，其中：

* List: 有序索引集，类似 JavaScript 中的 Array
* Map: 无序索引集，类似 JavaScript 中的 Object
* Set: 没有重复值的集合

主要的方法如下：

* fromJS()：将一个js数据转换为Immutable类型的数据

const obj = Immutable.fromJS({a:'123',b:'234'})

* toJS()：将一个Immutable数据转换为JS类型的数据
* is()：对两个对象进行比较

import { Map, is } from 'immutable'  
const map1 = Map({ a: 1, b: 1, c: 1 })  
const map2 = Map({ a: 1, b: 1, c: 1 })  
map1 === map2 //false  
Object.is(map1, map2) // false  
is(map1, map2) // true

* get(key)：对数据或对象取值
* getIn([]) ：对嵌套对象或数组取值，传参为数组，表示位置

let abs = Immutable.fromJS({a: {b:2}});  
abs.getIn(['a', 'b']) // 2  
abs.getIn(['a', 'c']) // 子级没有值  
  
let arr = Immutable.fromJS([1 ,2, 3, {a: 5}]);  
arr.getIn([3, 'a']); // 5  
arr.getIn([3, 'c']); // 子级没有值

如下例子：使用方法如下：

import Immutable from 'immutable';  
foo = Immutable.fromJS({a: {b: 1}});  
bar = foo.setIn(['a', 'b'], 2); // 使用 setIn 赋值  
console.log(foo.getIn(['a', 'b'])); // 使用 getIn 取值，打印 1  
console.log(foo === bar); // 打印 false

如果换到原生的js，则对应如下：

let foo = {a: {b: 1}};  
let bar = foo;  
bar.a.b = 2;  
console.log(foo.a.b); // 打印 2  
console.log(foo === bar); // 打印 true

## 三、在React中应用

使用 Immutable可以给 React 应用带来性能的优化，主要体现在减少渲染的次数

在做react性能优化的时候，为了避免重复渲染，我们会在shouldComponentUpdate()中做对比，当返回true执行render方法

Immutable通过is方法则可以完成对比，而无需像一样通过深度比较的方式比较

在使用redux过程中也可以结合Immutable，不使用Immutable前修改一个数据需要做一个深拷贝

import '\_' from 'lodash';  
  
const Component = React.createClass({  
 getInitialState() {  
 return {  
 data: { times: 0 }  
 }  
 },  
 handleAdd() {  
 let data = \_.cloneDeep(this.state.data);  
 data.times = data.times + 1;  
 this.setState({ data: data });  
 }  
}

使用 Immutable 后：

getInitialState() {  
 return {  
 data: Map({ times: 0 })  
 }  
},  
 handleAdd() {  
 this.setState({ data: this.state.data.update('times', v => v + 1) });  
 // 这时的 times 并不会改变  
 console.log(this.state.data.get('times'));  
 }

同理，在redux中也可以将数据进行fromJS处理

import \* as constants from './constants'  
import {fromJS} from 'immutable'  
const defaultState = fromJS({ //将数据转化成immutable数据  
 home:true,  
 focused:false,  
 mouseIn:false,  
 list:[],  
 page:1,  
 totalPage:1  
})  
export default(state=defaultState,action)=>{  
 switch(action.type){  
 case constants.SEARCH\_FOCUS:  
 return state.set('focused',true) //更改immutable数据  
 case constants.CHANGE\_HOME\_ACTIVE:  
 return state.set('home',action.value)  
 case constants.SEARCH\_BLUR:  
 return state.set('focused',false)  
 case constants.CHANGE\_LIST:  
 // return state.set('list',action.data).set('totalPage',action.totalPage)  
 //merge效率更高，执行一次改变多个数据  
 return state.merge({  
 list:action.data,  
 totalPage:action.totalPage  
 })  
 case constants.MOUSE\_ENTER:  
 return state.set('mouseIn',true)  
 case constants.MOUSE\_LEAVE:  
 return state.set('mouseIn',false)  
 case constants.CHANGE\_PAGE:  
 return state.set('page',action.page)  
 default:  
 return state  
 }  
}

## 参考文献

* https://zhuanlan.zhihu.com/p/20295971?spm=a2c4e.11153940.blogcont69516.18.4f275a00EzBHjr&columnSlug=purerender
* https://www.jianshu.com/p/7bf04638e82a