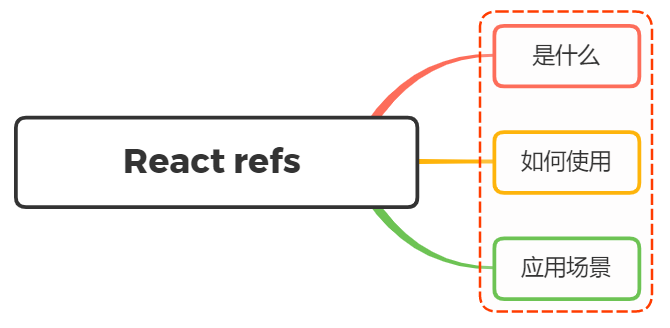
# 面试官：说说对React refs 的理解？应用场景？



## 一、是什么

Refs 在计算机中称为弹性文件系统（英语：Resilient File System，简称ReFS）

React 中的 Refs提供了一种方式，允许我们访问 DOM节点或在 render方法中创建的 React元素

本质为ReactDOM.render()返回的组件实例，如果是渲染组件则返回的是组件实例，如果渲染dom则返回的是具体的dom节点

## 二、如何使用

创建ref的形式有三种：

* 传入字符串，使用时通过 this.refs.传入的字符串的格式获取对应的元素
* 传入对象，对象是通过 React.createRef() 方式创建出来，使用时获取到创建的对象中存在 current 属性就是对应的元素
* 传入函数，该函数会在 DOM 被挂载时进行回调，这个函数会传入一个 元素对象，可以自己保存，使用时，直接拿到之前保存的元素对象即可
* 传入hook，hook是通过 useRef() 方式创建，使用时通过生成hook对象的 current 属性就是对应的元素

### 传入字符串

只需要在对应元素或组件中ref属性

class MyComponent extends React.Component {  
 constructor(props) {  
 super(props);  
 this.myRef = React.createRef();  
 }  
 render() {  
 return <div ref="myref" />;  
 }  
}

访问当前节点的方式如下：

this.refs.myref.innerHTML = "hello";

### 传入对象

refs通过React.createRef()创建，然后将ref属性添加到React元素中，如下：

class MyComponent extends React.Component {  
 constructor(props) {  
 super(props);  
 this.myRef = React.createRef();  
 }  
 render() {  
 return <div ref={this.myRef} />;  
 }  
}

当 ref 被传递给 render 中的元素时，对该节点的引用可以在 ref 的 current 属性中访问

const node = this.myRef.current;

### 传入函数

当ref传入为一个函数的时候，在渲染过程中，回调函数参数会传入一个元素对象，然后通过实例将对象进行保存

class MyComponent extends React.Component {  
 constructor(props) {  
 super(props);  
 this.myRef = React.createRef();  
 }  
 render() {  
 return <div ref={element => this.myref = element} />;  
 }  
}

获取ref对象只需要通过先前存储的对象即可

const node = this.myref

### 传入hook

通过useRef创建一个ref，整体使用方式与React.createRef一致

function App(props) {  
 const myref = useRef()  
 return (  
 <>  
 <div ref={myref}></div>  
 </>  
 )  
}

获取ref属性也是通过hook对象的current属性

const node = myref.current;

上述三种情况都是ref属性用于原生HTML元素上，如果ref设置的组件为一个类组件的时候，ref对象接收到的是组件的挂载实例

注意的是，不能在函数组件上使用ref属性，因为他们并没有实例

## 三、应用场景

在某些情况下，我们会通过使用refs来更新组件，但这种方式并不推荐，更多情况我们是通过props与state的方式进行去重新渲染子元素

过多使用refs，会使组件的实例或者是DOM结构暴露，违反组件封装的原则

例如，避免在 Dialog 组件里暴露 open() 和 close() 方法，最好传递 isOpen 属性

但下面的场景使用refs非常有用：

* 对Dom元素的焦点控制、内容选择、控制
* 对Dom元素的内容设置及媒体播放
* 对Dom元素的操作和对组件实例的操作
* 集成第三方 DOM 库

## 参考文献

* https://zh-hans.reactjs.org/docs/refs-and-the-dom.html
* https://segmentfault.com/a/1190000020842342
* https://vue3js.cn/interview