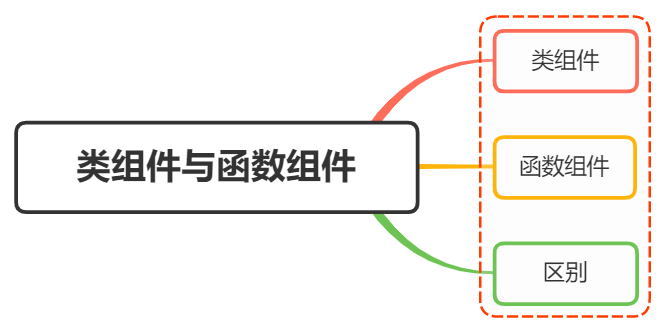
# 面试官：说说对React中类组件和函数组件的理解？有什么区别？



## 一、类组件

类组件，顾名思义，也就是通过使用ES6类的编写形式去编写组件，该类必须继承React.Component

如果想要访问父组件传递过来的参数，可通过this.props的方式去访问

在组件中必须实现render方法，在return中返回React对象，如下：

class Welcome extends React.Component {  
 constructor(props) {  
 super(props)  
 }  
 render() {  
 return <h1>Hello, {this.props.name}</h1>  
 }  
}

## 二、函数组件

函数组件，顾名思义，就是通过函数编写的形式去实现一个React组件，是React中定义组件最简单的方式

function Welcome(props) {  
 return <h1>Hello, {props.name}</h1>;  
}

函数第一个参数为props用于接收父组件传递过来的参数

## 三、区别

针对两种React组件，其区别主要分成以下几大方向：

* 编写形式
* 状态管理
* 生命周期
* 调用方式
* 获取渲染的值

### 编写形式

两者最明显的区别在于编写形式的不同，同一种功能的实现可以分别对应类组件和函数组件的编写形式

函数组件：

function Welcome(props) {  
 return <h1>Hello, {props.name}</h1>;  
}

类组件：

class Welcome extends React.Component {  
 constructor(props) {  
 super(props)  
 }  
 render() {  
 return <h1>Hello, {this.props.name}</h1>  
 }  
}

### 状态管理

在hooks出来之前，函数组件就是无状态组件，不能保管组件的状态，不像类组件中调用setState

如果想要管理state状态，可以使用useState，如下：

const FunctionalComponent = () => {  
 const [count, setCount] = React.useState(0);  
  
 return (  
 <div>  
 <p>count: {count}</p >  
 <button onClick={() => setCount(count + 1)}>Click</button>  
 </div>  
 );  
};

在使用hooks情况下，一般如果函数组件调用state，则需要创建一个类组件或者state提升到你的父组件中，然后通过props对象传递到子组件

### 生命周期

在函数组件中，并不存在生命周期，这是因为这些生命周期钩子都来自于继承的React.Component

所以，如果用到生命周期，就只能使用类组件

但是函数组件使用useEffect也能够完成替代生命周期的作用，这里给出一个简单的例子：

const FunctionalComponent = () => {  
 useEffect(() => {  
 console.log("Hello");  
 }, []);  
 return <h1>Hello, World</h1>;  
};

上述简单的例子对应类组件中的componentDidMount生命周期

如果在useEffect回调函数中return一个函数，则return函数会在组件卸载的时候执行，正如componentWillUnmount

const FunctionalComponent = () => {  
 React.useEffect(() => {  
 return () => {  
 console.log("Bye");  
 };  
 }, []);  
 return <h1>Bye, World</h1>;  
};

### 调用方式

如果是一个函数组件，调用则是执行函数即可：

// 你的代码   
function SayHi() {   
 return <p>Hello, React</p >   
}   
// React内部   
const result = SayHi(props) // » <p>Hello, React</p >

如果是一个类组件，则需要将组件进行实例化，然后调用实例对象的render方法：

// 你的代码   
class SayHi extends React.Component {   
 render() {   
 return <p>Hello, React</p >   
 }   
}   
// React内部   
const instance = new SayHi(props) // » SayHi {}   
const result = instance.render() // » <p>Hello, React</p >

### 获取渲染的值

首先给出一个示例

函数组件对应如下：

function ProfilePage(props) {  
 const showMessage = () => {  
 alert('Followed ' + props.user);  
 }  
  
 const handleClick = () => {  
 setTimeout(showMessage, 3000);  
 }  
  
 return (  
 <button onClick={handleClick}>Follow</button>  
 )  
}

类组件对应如下：

class ProfilePage extends React.Component {  
 showMessage() {  
 alert('Followed ' + this.props.user);  
 }  
  
 handleClick() {  
 setTimeout(this.showMessage.bind(this), 3000);  
 }  
  
 render() {  
 return <button onClick={this.handleClick.bind(this)}>Follow</button>  
 }  
}

两者看起来实现功能是一致的，但是在类组件中，输出this.props.user，Props在 React中是不可变的所以它永远不会改变，但是 this 总是可变的，以便您可以在 render 和生命周期函数中读取新版本

因此，如果我们的组件在请求运行时更新。this.props 将会改变。showMessage方法从“最新”的 props 中读取 user

而函数组件，本身就不存在this，props并不发生改变，因此同样是点击，alert的内容仍旧是之前的内容

### 小结

两种组件都有各自的优缺点

函数组件语法更短、更简单，这使得它更容易开发、理解和测试

而类组件也会因大量使用 this而让人感到困惑

## 参考文献

* https://zh-hans.reactjs.org/docs/components-and-props.html#function-and-class-components
* https://juejin.cn/post/6844903806140973069