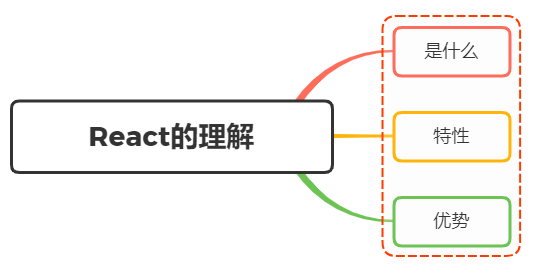
# 面试官：说说对 React 的理解？有哪些特性？



## 一、是什么

React，用于构建用户界面的 JavaScript 库，只提供了 UI 层面的解决方案

遵循组件设计模式、声明式编程范式和函数式编程概念，以使前端应用程序更高效

使用虚拟 DOM 来有效地操作 DOM，遵循从高阶组件到低阶组件的单向数据流

帮助我们将界面成了各个独立的小块，每一个块就是组件，这些组件之间可以组合、嵌套，构成整体页面

react 类组件使用一个名为 render() 的方法或者函数组件return，接收输入的数据并返回需要展示的内容

class HelloMessage extends React.Component {  
 render() {  
 return <div>Hello {this.props.name}</div>;  
 }  
}  
  
ReactDOM.render(  
 <HelloMessage name="Taylor" />,  
 document.getElementById("hello-example")  
);

上述这种类似 XML 形式就是 JSX，最终会被 babel 编译为合法的 JS 语句调用

被传入的数据可在组件中通过 this.props 在 render() 访问

## 二、特性

React 特性有很多，如：

* JSX 语法
* 单向数据绑定
* 虚拟 DOM
* 声明式编程
* Component

着重介绍下声明式编程及 Component

### 声明式编程

声明式编程是一种编程范式，它关注的是你要做什么，而不是如何做

它表达逻辑而不显式地定义步骤。这意味着我们需要根据逻辑的计算来声明要显示的组件

如实现一个标记的地图：

通过命令式创建地图、创建标记、以及在地图上添加的标记的步骤如下：

// 创建地图  
const map = new Map.map(document.getElementById("map"), {  
 zoom: 4,  
 center: { lat, lng },  
});  
  
// 创建标记  
const marker = new Map.marker({  
 position: { lat, lng },  
 title: "Hello Marker",  
});  
  
// 地图上添加标记  
marker.setMap(map);

而用 React 实现上述功能则如下：

<Map zoom={4} center={(lat, lng)}>  
 <Marker position={(lat, lng)} title={"Hello Marker"} />  
</Map>

声明式编程方式使得 React 组件很容易使用，最终的代码简单易于维护

### Component

在 React 中，一切皆为组件。通常将应用程序的整个逻辑分解为小的单个部分。 我们将每个单独的部分称为组件

组件可以是一个函数或者是一个类，接受数据输入，处理它并返回在 UI 中呈现的 React 元素

函数式组件如下：

const Header = () => {  
 return (  
 <Jumbotron style={{ backgroundColor: "orange" }}>  
 <h1>TODO App</h1>  
 </Jumbotron>  
 );  
};

类组件（有状态组件）如下：

class Dashboard extends React.Component {  
 constructor(props) {  
 super(props);  
  
 this.state = {};  
 }  
 render() {  
 return (  
 <div className="dashboard">  
 <ToDoForm />  
 <ToDolist />  
 </div>  
 );  
 }  
}

一个组件该有的特点如下：

* 可组合：每个组件易于和其它组件一起使用，或者嵌套在另一个组件内部
* 可重用：每个组件都是具有独立功能的，它可以被使用在多个 UI 场景
* 可维护：每个小的组件仅仅包含自身的逻辑，更容易被理解和维护

## 三、优势

通过上面的初步了解，可以感受到 React 存在的优势：

* 高效灵活
* 声明式的设计，简单使用
* 组件式开发，提高代码复用率
* 单向响应的数据流会比双向绑定的更安全，速度更快

## 参考文献

* <https://segmentfault.com/a/1190000015924762>
* <https://react.docschina.org/>