React面试题

有关react的面试题

1.真实DOM和虚拟DOM的区别?

makefile 复制代码

```
真实DOM:
更新缓慢
可以直接更新html
如果元素更新，则创建新DOM

虚拟DOM
更新更快
无法直接更新html
如果元素更新，则更新JSX
```

简单介绍一下react

yaml 复制代码

```
React 是 Facebook 在 2011 年开发的前端 JavaScript 库。
它遵循基于组件的方法，有助于构建可重用的UI组件。
它用于开发复杂和交互式的 Web 和移动 UI。
尽管它仅在 2015 年开源，但有一个很大的支持社区。
```

react有什么特点?

复制代码

```
1、他使用虚拟DOM,而不是真实DOM
2、它可以使用服务器进行渲染
3、它遵循单向数据流或者是数据绑定
```

react有什么优点?

swift 复制代码

```
1、它提高了应用的性能
2、可以方便的在客户端和服务器使用
3、由于使用JSX，代码的可读性很好
```

```
4、react很容易与Metor，Angular等其他框架进行集成
5、使用react，编写UI测试用例变得非常容易
```

什么是JSX?

JSX是JavaScript XML的简写，是react使用的一种文件，它利用JavaScript的变现力和类似HTML的模板语法。这使得html文件非常容易理解。此文件能使应用非常可靠。

虚拟DOM的工作原理

```go
1、每当底层数据发生改变时，整个UI都将在虚拟DOM描述中重新渲染
2、然后计算之前DOM表示与新表示的之间的差异
3、完成计算后，将只用实际更改的内容进行更新real DOM
```

解释react中render的目的?

每个React组件强制要求必须有一个 render()。

它返回一个 React 元素，是原生 DOM 组件的表示。

如果需要渲染多个 HTML 元素，则必须将它们组合在一个封闭标记内，例如

、、
等。

此函数必须保持纯净，即必须每次调用时都返回相同的结果。

什么是props?

props是react中属性的简写。他们是只读组件，必须保持纯，即不可变。他们总是在整个应用中从父组件传递到子组件。子组件永远不能将props送回到父组件。这有助于维护数据的单向流，通常用于呈现动态生成的数据。

react中的状态是什么？它是如何使用的?

状态是 React 组件的核心，是数据的来源，必须尽可能简单。基本上状态是确定组件呈现和行为的对象。与props 不同，它们是可变的，并创建动态和交互式组件。可以通过 this.state() 访

问它们。

区分有状态组件和无状态组件?

有状态组件 1、在内存中存储有关组件状态变化的信息 2、有权改变状态 3、包含过去、现在、未来可能的状态变化情况 4、接受无状态组件状态变化要求的通知，然后将pros发送给他们

无状态组件 1、计算组件内部的状态 2、无权改变状态 3、不包含过去、现在和未来可能发生的状态变化情况 4、从有状态组件接收props并将其视为回调函数

react组件的生命周期的阶段是什么?

react组件的生命周期有三个不同的阶段

1、初始化渲染阶段：这是组件即将开始其声生命之旅并进入到DOM的阶段 2、更新阶段：一旦组件被添加到DOM,它只有在prop或者状态发生改变时才可能更新或者是重新渲染，这些只发生在这个阶段 3、卸载阶段：这是组件生命周期的最后阶段，组件被销毁时并从DOM中被删除

react中的事件是什么?

复制代码

在react中，事件是对鼠标悬停、鼠标单击、按键等特定操作的触发反应。
处理这些事件类似于处理DOM元素中的事件。但是有一些语法差异，

1、用驼峰命名法对事件命名而不是仅使用小写字母

2、事件作为函数而不是字符串进行传递

对react中的refs有什么理解?

scss 复制代码

refs是react中引用的简写。它是一个有助于存储对特定的React与秦愫或者组件的引用的属性，它将由组件渲染配置函数返用于对render()返回的特定元素或组件的引用。每当需要进行DOM测量或向组件添加方法时，他们会派上用场。

什么时候需要使用refs?

复制代码

```
需要管理焦点、选择文本或者媒体播放时
触发式动画
与第三方DOM库集成
```

受控组价和非受控组件

受控组件

1、没有维持自己的状态

2、数据由父组件控制

3、通过props来获取当前值，然后通过回调通知更改

非受控组件

1、保持着自己的状态

2、数据由DOM控制

3、refs来获取其当前值

MVC框架存在什么问题?

复制代码

```
1、对DOM操作的代价非常高
2、程序运行缓慢且效率低下
3、内存浪费严重
4、由于循环的依赖性，组件模型需要围绕models和views进行创建
```

redux遵循的三个原则是什么?

复制代码

```
1、单一事件来源：整个应用的状态存储在单个store中的对象/状态树里
2、状态是只读的，改变状态的唯一方法时去触发一个动作。
3、使用纯函数进行更改，为了指定状态树如何通过操作进行转换。
```

列出redux的组件

1、Action--这是一个用来描述发生了什么事情的对象

2、Reducer--这是一个确定状态将如何变化的地方

3、Store--整个程序的状态/对象树保存在Store中

4、View---只显示Store提供的数据

redux有哪些优点?

结果的可预测性 可维护性 服务器端渲染 开发人员工具 社区和生态系统 易于测试 组织

关于react中的路由

Router用于定义多个路由，当用户定义特定的URL 时，如果此时URL与Router内定义的任何路由的路径匹配，则用户将重定向到该特定路由。所以基本上我们需要在自己的应用中添加一个Router库，允许创建多个路由，每个路由都会给我们提供一个独特的视图

使用switch关键字，可以仅显示第一个匹配成功的项，直接进行渲染，然后绕过其他路线

高阶组件

```swift
高阶组件就是一个函数，且该函数接收一个组件作为参数，并返回一个新的组件。

组件将props转换为UI，而高阶组件是将组件转换为另一个组件
```

当你调用setState的时候发生了什么事件?

当调用 `setState` 时，React会做的第一件事情是将传递给 `setState` 的对象合并到组件的当前状态。这将启动一个称为和解（reconciliation）的过程。和解（reconciliation）的最终目标是以最有效的方式，根据这个新的状态来更新UI。 为此，React将构建一个新的 `React` 元素树（您可以将其视为 UI 的对象表示）。

一旦有了这个树，为了弄清 UI 如何响应新的状态而改变，React 会将这个新树与上一个元素树相比较（ diff ）。

通过这样做， React 将会知道发生的确切变化，并且通过了解发生什么变化，只需在绝对必要的情况下进行更新即可最小化 UI 的占用空间。

什么是React中的refs，为什么他们很重要?

refs就像是一个逃生舱口，允许您直接访问DON元素或者组件实例。为了使用他们，您可以向组件添加一哥ref属性，该竖向的值是一个回调函数，它将接收底层的DOM元素或组件的已挂载实例，作为其第一个参数。

javascript 复制代码

```
class UnContorlledForm estends Component {
    render() {
        return (
            <from onSubmit={this.handleSubmit}>
                <input
                    type='text'
                    ref={(input) =>this.input = input}
                />
                <button type='submit'> Submit</button>
            </form>
        )
    }
}
```

以上注意到我们的输入字段有一个ref属性，其值是一个函数。该函数接收我们然后放在实例上的实际的DOM元素，一遍在handleSubmit函数内部访问他。经常误解的是，您需要使用类组件才能使用ref，其实不然，ref也可以通过利用JavaScript中的闭包与功能组件一起使用。

javascript 复制代码

```
function CusTomer ({handleSubmit}) {
    let inputElement
    return (
        <form onSubmit={() =>{handleSubmit(inputElement.value)}}>
            <input
            type='text'
            ref={input =>inputElement.value} />
            <button type='submit'>Submit</button>
        </form>
    )
}
```

React中的keys是什么，为什么它们很重要?

keys是帮助React跟踪哪些项目已更改、添加或从列表中删除

```
return (
    <ul>
        {this.state.todoItems.map(({task,uid}) =>{
            return <li key={uid}> {task} </li>
        })}
    </ul>
)
```

每个keys在兄弟元素之间是独一无二的。我们已经谈过几次关于和解的过程，而且这个和解过程中的一部分正在执行一个新的元素树与最前一个的差异。keys使处理列表时更加高效，因为React可以使用子元素上的keys快速知道元素是新的还是在比较树时才被移动。

而且keys不仅使这个过程更有效率，而且没有keys，React不知道哪个本地状态对应于移动中的哪个项目。所以当你map的时候，不要忽略了keys。