React 经典面试题

1. 什么时候使用状态管理器?

- 从项目的整体看
  - 目用户的使用方式复杂
  - 不同身份的用户有不同的使用方式（比如普通用户和管理员）
  - 多个用户之间可以协作
  - 与服务器大量交互，或者使用了 `WebSocket`
  - `View` 要从多个来源获取数据
- 从组件角度看
  - 某个组件的状态，需要共享
  - 某个状态需要在任何地方都可以拿到
  - 一个组件需要改变全局状态
  - 一个组件需要改变另一个组件的状态
- 组件有相当大量的，随时间变化的数据
- `state` 需要有一个单一可靠数据源
- 所有 `state` 放在顶层组件已经无法满足需求了

2. 区分 Real DOM 和 Virtual DOM?

Real DOM	Virtual DOM
1. 更新缓慢。	1. 更新更快。
2. 可以直接更新 HTML。	2. 无法直接更新 HTML。
3. 如果元素更新，则创建新DOM。	3. 如果元素更新，则更新 JSX 。
4. DOM操作代价很高。	4. DOM 操作非常简单。
5. 消耗的内存较多。	5. 很少的内存消耗。

3. 说说 React 有什么特点?

- 它使用**虚拟 DOM **而不是真正的 DOM。
- 它可以用服务器端渲染。
- 它遵循单向数据流或数据绑定

4. 列出 React 的一些主要优点？

- 它提高了应用的性能
- 可以方便地在客户端和服务器端使用
- 由于 JSX，代码的可读性很好
- React 很容易与 Meteor，Angular 等其他框架集成
- 使用 React，编写 UI 测试用例变得非常容易

5. 什么是 JSX?

- 它 JSX 是 J avaScript XML 的简写。是 React 使用的一种文件，它利用 JavaScript 的表现力和类似 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，并能够提高其性能
- JSX 的一个例子：

```
render(){
    return(
        <div>
            <h1> Hello World from Edureka!!</h1>
        </div>
    );
}
```

6. 你了解 Virtual DOM 吗？解释一下它的工作原理？

- Virtual DOM 是一个轻量级的 JavaScript 对象，它最初只是 real DOM 的副本。它是一个节点树，它将元素、它们的属性和内容作为对象及其属性。React 的渲染函数从 React 组件中创建一个节点树。然后它响应数据模型中的变化来更新该树，该变化是由用户或系统完成的各种动作引起的
- Virtual DOM 工作过程有三个简单的步骤：
  - 每当底层数据发生改变时，整个 UI 都将在 Virtual DOM 描述中重新渲染
  - 然后计算之前 DOM 表示与新表示的之间的差异
  - 完成计算后，将只用实际更改的内容更新 real DOM

7. 说说为什么浏览器无法读取 JSX?

- 浏览器只能处理 JavaScript 对象，而不能读取常规 JavaScript 对象中的 JSX。所以为了使浏览器能够读取 JSX，首先，需要用像 Babel 这样的 JSX 转换器将 JSX 文件转换为 JavaScript 对象，然后再将其传给浏览器

8. 与 ES5 相比，React 的 ES6 语法有何不同?

- require 与 import

```
// ES5
var React = require('react');
// ES6
import React from 'react';
```

- export 与 exports

```
// ES5
module.exports = Component;
// ES6
export default Component;
```

- component 和 function

```
// ES5
var MyComponent = React.createClass({
    render: function() {
        return
            <h3>Hello Edureka!</h3>;
    }
});
// ES6
class MyComponent extends React.Component {
    render() {
        return
            <h3>Hello Edureka!</h3>;
    }
}
```

- Props

```
var App = React.createClass({ // ES5
    propTypes: { name: React.PropTypes.string },
    render: function() {
        return <h3>Hello, {this.props.name}!</h3>;
    }
});
class App extends React.Component {  // ES6
    render() {
        return
            <h3>Hello, {this.props.name}!</h3>;
    }
}
```

- state

```
// ES5
var App = React.createClass({
    getInitialState: function() {
        return { name: 'world' };
    },
    render: function() {
        return
            <h3>Hello, {this.state.name}!</h3>;
    }
});
// ES6
class App extends React.Component {
    constructor() {
        super();
        this.state = { name: 'world' };
    }
    render() {
        return
            <h3>Hello, {this.state.name}!</h3>;
    }
}
```

9. React 与 Angular 有何不同?

主题	React	Angular
1. 体系结构	只有 MVC 中的 View	完整的 MVC
2. 渲染	可以在服务器端渲染	客户端渲染
3. DOM	使用 virtual DOM	使用 real DOM
4. 数据绑定	单向数据绑定	双向数据绑定
5. 调试	编译时调试	运行时调试
6. 作者	Facebook	Google

10. 你理解“在 React 中，一切都是组件”这句话?

- 组件是 `React` 应用 `UI` 的构建块。这些组件将整个 `UI` 分成小的独立并可重用的部分。每个组件彼此独立，而不会影响 `UI` 的其余部分

11. 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 <form>、<group>、 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

12. 如何将两个或多个组件嵌入到一个组件中?

- 通过以下方式将组件嵌入到一个组件中：

```
class MyComponent extends React.Component{
    render(){
        return(
            <div>
                <h1>Hello</h1>
                <Header/>
            </div>
        );
    }
}
class Header extends React.Component{
    render(){
        return
            <h1>Header Component</h1>
    };
}
ReactDOM.render(
    <MyComponent/>, document.getElementById('content')
);
```

13. 什么是 Props?

- Props 是 React 中属性的简写。它们是只读组件，必须保持纯，即不可变。它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 prop 送回父组件。这有助于维护单向数据流，通常用于呈现动态生成的数据

14. 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。
-

15. 区分状态和 Props?

条件	State	Props
1. 从父组件中接收初始值	Yes	Yes
2. 父组件可以改变值	No	Yes
3. 在组件中设置默认值	Yes	Yes
4. 在组件的内部变化	Yes	No
5. 设置子组件的初始值	Yes	Yes
6. 在子组件的内部更改	No	Yes

16. 如何更新组件的状态?

- 使用 `this.setState()`更新组件的状态。

```
class MyComponent extends React.Component {
    constructor() {
        super();
        this.state = {
            name: 'Maxx',
            id: '101'
        }
    }
    render()
        {
            setTimeout(()=>{this.setState({name:'Jaeha', id:'222'})},2000)
            return (
                <div>
                    <h1>Hello {this.state.name}</h1>
                    <h2>Your Id is {this.state.id}</h2>
                </div>
            );
        }
    }
ReactDOM.render(
    <MyComponent/>, document.getElementById('content')
);
```

17. React 中的箭头函数是什么？怎么用？

- 箭头函数（=>）是用于编写函数表达式的简短语法。这些函数允许正确绑定组件的上下文，因为在 ES6 中默认下不能使用自动绑定。使用高阶函数时，箭头函数非常有用。

```
//General way
render() {
    return(
        <MyInput onChange = {this.handleChange.bind(this) } />
    );
}
//With Arrow Function
render() {
    return(
        <MyInput onChange = { (e)=>this.handleOnChange(e) } />
    );
}
```

18. React 组件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：
  - 初始渲染阶段：这是组件即将开始其生命之旅并进入 DOM 的阶段。
  - 更新阶段：一旦组件被添加到 DOM，它只有在 prop 或状态发生变化时才可能更新和重新渲染。这些只发生在这个阶段
  - 卸载阶段：这是组件生命周期的最后阶段，组件被销毁并从 DOM 中删除

19. React 中的事件是什么？

- 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，如：
  - 用驼峰命名法对事件命名而不是仅使用小写字母。
  - 事件作为函数而不是字符串传递
- 事件参数重包含一组特定于事件的属性。每个事件类型都包含自己的属性和行为，只能通过其事件处理程序访问

20. React 中的合成事件是什么？

- 合成事件是围绕浏览器原生事件充当跨浏览器包装器的对象。它们将不同浏览器的行为合并为一个 API。这样做是为了确保事件在不同浏览器中显示一致的属性

21. 如何在 React 中创建一个事件?

- 如下:

```
class Display extends React.Component({
    show(evt) {
        // code
    },
    render() {
        // Render the div with an onClick prop (value is a function)
        return (
            <div onClick={this.show}>Click Me!</div>
        );
    }
});
```

22. 你对 React 的 refs 有什么了解?

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

```
class ReferenceDemo extends React.Component{
     display() {
         const name = this.inputDemo.value;
         document.getElementById('disp').innerHTML = name;
     }
render() {
    return(
          <div>
            Name: <input type="text" ref={input => this.inputDemo = input} />
            <button name="Click" onClick={this.display}>Click</button>
            <h2>Hello <span id="disp"></span> !!!</h2>
          </div>
    );
   }
 }
```

23. 列出一些应该使用 refs 的情况?

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

24. 如何模块化 React 中的代码?

- 可以使用 `export` 和 `import` 属性来模块化代码。它们有助于在不同的文件中单独编写组件

```
//ChildComponent.jsx
export default class ChildComponent extends React.Component {
    render() {
        return(
              <div>
                <h1>This is a child component</h1>
              </div>
        );
    }
}

//ParentComponent.jsx
import ChildComponent from './childcomponent.js';
class ParentComponent extends React.Component {
    render() {
        return(
             <div>
                <App />
             </div>
        );
    }
}
```

25. 你对受控组件和非受控组件了解多少?

受控组件	非受控组件
1. 没有维持自己的状态	1. 保持着自己的状态
2.数据由父组件控制	2.数据由 DOM 控制
3. 通过 props 获取当前值，然后通过回调通知更改	3. Refs 用于获取其当前值

26. 什么是高阶组件 HOC?

- 高阶组件是重用组件逻辑的高级方法，是一种源于 `React` 的组件模式。 `HOC` 是自定义组件，在它之内包含另一个组件。它们可以接受子组件提供的任何动态，但不会修改或复制其输入组件中的任何行为。你可以认为 `HOC` 是“纯（Pure）”组件

27. 你能用 HOC 做什么?

HOC 可用于许多任务:

- 用代码重用，逻辑和引导抽象
- 渲染劫持
- 状态抽象和控制
- Props 控制

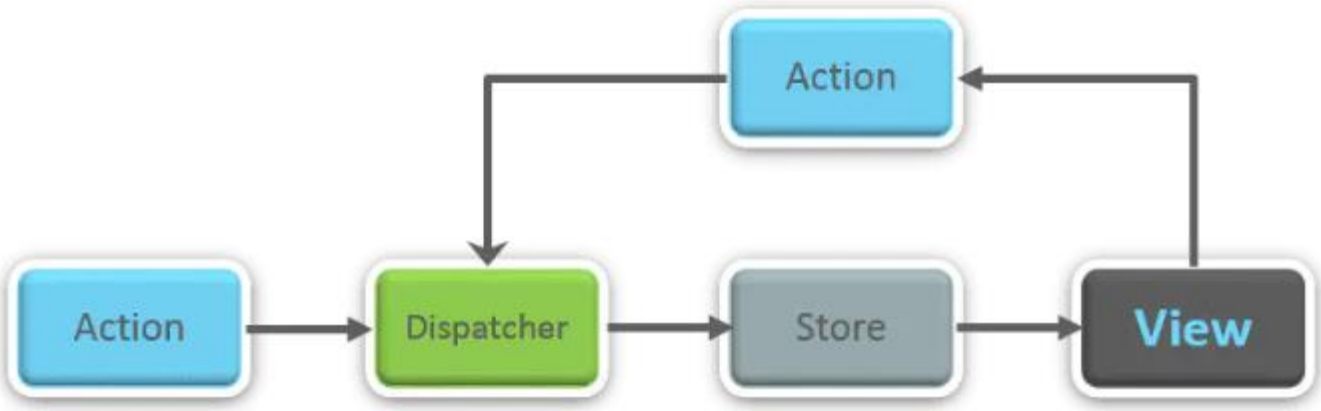
28. React 中 key 的重要性是什么?

- key 用于识别唯一的 Virtual DOM 元素及其驱动 UI 的相应数据。它们通过回收 DOM 中当前所有的元素来帮助 React 优化渲染。这些 key 必须是唯一的数字或字符串，React 只是重新排序元素而不是重新渲染它们。这可以提高应用程序的性能

29. MVC 框架的主要问题是什么?

- key 用对 DOM 操作的代价非常高
- 程序运行缓慢且效率低下
- 内存浪费严重
- 由于循环依赖性，组件模型需要围绕 models 和 views 进行创建

30. 请你解释一下 Flux?



- Flux 是一种强制单向数据流的架构模式。它控制派生数据，并使用具有所有数据权限的中心 store 实现多个组件之间的通信。整个应用中的数据更新必须只能在此处进行。 Flux 为应用提供稳定性并减少运行时的错误。

31. 什么是 Redux?

- Redux 是当今最热门的前端开发库之一。它是 JavaScript 程序的可预测状态容器，用于整个应用的状态管理。使用 Redux 开发的应用易于测试，可以在不同环境中运行，并显示一致的行为。

32. Redux 遵循的三个原则是什么?

- 单一事实来源：整个应用的状态存储在单个 store 中的对象/状态树里。单一状态树可以更容易地跟踪随时间的变化，并调试或检查应用程序。
- 状态是只读的：改变状态的唯一方法是去触发一个动作。动作是描述变化的普通 JS 对象。就像 state 是数据的最小表示一样，该操作是对数据更改的最小表示
- 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，你需要纯函数。纯函数是那些返回值仅取决于其参数值的函数

33. 你对“单一事实来源”有什么理解

- Redux 使用 “store” 将程序的整个状态存储在同一个地方。因此所有组件的状态都存储在 store 中，并且它们从 store 本身接收更新。单一状态树可以更容易地跟踪随时间的变化，并调试或检查程序。

34. 列出 Redux 的组件?

- Redux 由以下组件组成:
  - Action 这是一个用来描述发生了什么事情的对象
  - Reducer 这是一个确定状态将如何变化的地方
  - Store 整个程序的状态/对象树保存在 Store 中
  - View 查只显示 Store 提供的数据

35. 如何在 Redux 中定义 Action?

- React 中的 Action 必须具有 type 属性，该属性指示正在执行的 ACTION 的类型。必须将它们定义为字符串常量，并且还可以向其添加更多的属性。在 Redux 中，action 被名为 Action Creators 的函数所创建。以下是 Action 和 Action Creators 的示例:

```
function addTodo(text) {
    return {
        type: ADD_TODO,
        text
    }
}
```

36. 解释 Reducer 的作用?

- `Reducers` 是纯函数，它规定应用程序的状态怎样因响应 `ACTION` 而改变。`Reducers` 通过接受先前的状态和 `action` 来工作，然后它返回一个新的状态。它根据操作的类型确定需要执行哪种更新，然后返回新的值。如果不需要完成任务，它会返回原来的状态

37. Store 在 Redux 中的意义是什么?

- `store` 是一个 `JavaScript` 对象，它可以保存程序的状态，并提供一些方法来访问状态、调度操作和注册侦听器。应用程序的整个状态/对象树保存在单一存储中。因此，`Redux` 非常简单且是可预测的。我们可以将中间件传递到 `store` 来处理数据，并记录改变存储状态的各种操作。所有操作都通过 `Reducer` 返回一个新状态

38. Redux 与 Flux 有何不同?

Flux	Redux
1. Store 包含状态和更改逻辑	1. Store 和更改逻辑是分开的
2. 有多个 Store	2. 只有一个 Store
3. 所有 Store 都互不影响且是平级的	3. 带有分层 reducer 的单一 Store
4. 有单一调度器	4. 没有调度器的概念
5. React 组件订阅 store	5. 容器组件是有联系的
6. 状态是可变的	6. 状态是不可改变的

39. Redux 有哪些优点?

- 结果的可预测性 - 由于总是存在一个真实来源，即 `store`，因此不存在如何将当前状态与动作和应用的其他部分同步的问题。
- 可维护性 - 代码变得更容易维护，具有可预测的结果和严格的结构。
- 服务器端渲染 - 你只需将服务器上创建的 `store` 传到客户端即可。这对初始渲染非常有用，并且可以优化应用性能，从而提供更好的用户体验。
- 开发人员工具 - 从操作到状态更改，开发人员可以实时跟踪应用中发生的所有事情。
- 社区和生态系统 - `Redux` 背后有一个巨大的社区，这使得它更加迷人。一个由才华横溢的人组成的大型社区为库的改进做出了贡献，并开发了各种应用。
- 易于测试 - `Redux` 的代码主要是小巧、纯粹和独立的功能。这使代码可测试且独立。
- 组织 - `Redux` 准确地说明了代码的组织方式，这使得代码在团队使用时更加一致和简单。

40. 什么是 React 路由?

- React 路由是一个构建在 React 之上的强大的路由库，它有助于向应用程序添加新的屏幕和流。这使 URL 与网页上显示的数据保持同步。它负责维护标准化的结构和行为，并用于开发单页 Web 应用。 React 路由有一个简单的 API。

41. 为什么 React Router v4 中使用 switch 关键字?

- 虽然 ** 用于封装 Router 中的多个路由，当你想要仅显示要在多个定义的路线中呈现的单个路线时，可以使用 “switch” 关键字。使用时，< switch >** 标记会按顺序将已定义的 URL 与已定义的路由进行匹配。找到第一个匹配项后，它将渲染指定的路径。从而绕过其它路。

42. render 函数中 return 如果没有使用()会有什么问题?

- 我们在使用 JSX 语法书写 React 代码时，babel 会将 JSX 语法编译成 js，同时会在每行自动添加分号（；），如果 return 后换行了，那么就会变成 return； 一般情况下会报错：Nothing was returned from render. This usually means a return statement is missing. Or, to render nothing, return null.

- 正确的书写

```
const Nav = () => {
  return (
    <nav className="c_navbar">
      { some jsx magic here }
    </nav>
  )
}

const Nav = () => {
 return <nav className="c_navbar">
    { some jsx magic here }
  </nav>
}
```

- 错误的写法：

```
const Nav = () => {
  return
    <nav className="c_navbar">
      { some jsx magic here }
    </nav>
}
```

43. componentWillUpdate 可以直接修改 state 值吗?

- 首先根据 React 特性决定如果只是 this.state 来更改值的话，是不会起效的，并且这样的写法是会有警告的；
- 其次在 componentWillUpdate 中去使用 setState 来修改值的话，不一定就会产生死循环；具体要看你是怎么使用 setState 的，如果只是传入一个对象的话，那么是会产生死循环的；但是如果传入一个函数的话：(preState,props)=>{}，可以通过条件来避免死循环；也就是说在条件满足的情况下，才会进行修改来重渲染，否则就不修改，相应的就不会重渲染，也就不存在会一直触发 componentWillUpdate 函数

44. 说说你对 React 的渲染原理的理解?

- 单向数据流。React 是一个 MVVM 框架，简单来说是在 MVC 的模式下在前端部分拆分出数据层和视图层。单向数据流指的是只能由数据层的变化去影响视图层的变化，而不能反过来（除非双向绑定）
- 数据驱动视图。我们无需关注页面的 DOM，只需要关注数据即可
- 渲染过程，生命周期.....
- setState()大部分时候是异步执行的，提升性能。

45. React 中三种构建组件的方式?

- React.createClass()、ES6 class 和无状态函数。

46. 描述 React 事件处理?

- 为了解决跨浏览器兼容性问题，React 中的事件处理程序将传递 SyntheticEvent 实例，该实例是 React 跨浏览器本机事件的跨浏览器包装器。这些综合事件具有与您惯用的本机事件相同的界面，除了它们在所有浏览器中的工作方式相同
- React 实际上并未将事件附加到子节点本身。React 将使用单个事件侦听器在顶层侦听所有事件。这对性能有好处，也意味着 React 在更新 DOM 时无需担心跟踪事件监听器

47. state 和 Props 有什么区别?

- State 和 props 都是普通的 JavaScript 对象。尽管它们两者都具有影响渲染输出的信息，但它们在组件方面的功能不同
  - 但 props 是一个从外部传进组件的参数，主要作为就是从父组件向子组件传递数据，它具有可读性和不变性，只能通过外部组件主动传入新的 props 来重新渲染子组件，否则子组件的 props 以及展现形式不会改变。
  - State 的主要作用是用于组件保存、控制以及修改自己的状态，它只能在 constructor 中初始化，它算是组件的私有属性，不可通过外部访问和修改，只能通过组件内部的 this.setState 来修改，修改 State 属性会导致组件的重新渲染。

React

1. 什么时候使用状态管理器？
2. render 函数中 return 如果没有使用()会有什么问题？
3. componentWillUpdate 可以直接修改 state 的值吗？
4. 说说你对 React 的渲染原理的理解
5. 什么渲染劫持？
6. React Intl 是什么原理？
7. 你有使用过 React Intl 吗？
8. 怎么实现 React 组件的国际化呢？
9. 说说 Context 有哪些属性？
10. 怎么使用 Context 开发组件？
11. 为什么 React 并不推荐我们优先考虑使用 Context？
12. 除了实例的属性可以获取 Context 外哪些地方还能直接获取 Context 呢？
13. childContextTypes 是什么？它有什么用？
14. contextType 是什么？它有什么用？
15. Consumer 向上找不到 Provider 的时候怎么办？
16. 有使用过 Consumer 吗？
17. 在 React 怎么使用 Context？
18. React15 和 16 别支持 IE 几以上？
19. 说说你对 windowing 的了解
20. 举例说明 React 的插槽有哪些运用场景？
21. 你有用过 React 的插槽(Portals)吗？怎么用？
22. React 的严格模式有什么用处？
23. React 如何进行代码拆分？拆分的原则是什么？
24. React 组件的构造函数有什么作用？
25. React 组件的构造函数是必须的吗？
26. React 中在哪捕获错误？
27. React 怎样引入 svg 的文件？
28. 说说你对 Relay 的理解
29. 在 React 中你有经常使用常量吗？
30. 为什么说 React 中的 props 是只读的？
31. 你有使用过 formik 库吗？说说它的优缺点
32. 你有用过哪些 React 的表单库吗？说说它们的优缺点
33. 如果组件的属性没有传值，那么它的默认值是什么？
34. 可以使用 TypeScript 写 React 应用吗？怎么操作？
35. `super()`和 `super(props)`有什么区别？

36. 你有使用过 loadable 组件吗？它帮我们解决了什么问题？
37. 你有使用过 suspense 组件吗？它帮我们解决了什么问题？
38. 怎样动态导入组件？
39. 如何给非控组件设置默认的值？
40. 怎么在 React 中引入其它的 UI 库，例如 Bootstrap
41. 怎样将事件传递给子组件？
42. 怎样使用 Hooks 获取服务端数据？
43. 使用 Hooks 要遵守哪些原则？
44. render 方法的原理你有了解吗？它返回的数据类型是什么？
45. useEffect 和 useLayoutEffect 有什么区别？
46. 在 React 项目中你用过哪些动画的包？
47. React 必须使用 JSX 吗？
48. 自定义组件时 render 是可选的吗？为什么？
49. 需要把 keys 设置为全局唯一吗？
50. 怎么定时更新一个组件？
51. React 根据不同的环境打包不同的域名？
52. 使用 webpack 打包 React 项目，怎么减小生成的 js 大小？
53. 在 React 中怎么使用 async/await？
54. 你阅读了几遍 React 的源码？都有哪些收获？你是怎么阅读的？
55. 什么是 React.forwardRef？它有什么作用？
56. 写个例子说明什么是 JSX 的内联条件渲染
57. 在 React 中怎么将参数传递给事件？
58. React 的事件和普通的 HTML 事件有什么不同？
59. 在 React 中怎么阻止事件的默认行为？
60. 你最喜欢 React 的哪一个特性（说一个就好）？
61. 在 React 中什么时候使用箭头函数更方便呢？
62. 你最不喜欢 React 的哪一个特性（说一个就好）？
63. 说说你对 React 的 reconciliation（一致化算法）的理解
64. 使用 PropTypes 和 Flow 有什么区别？
65. 怎样有条件地渲染组件？
66. 在 JSX 中如何写注释？
67. constructor 和 getInitialState 有不同？
68. 写例子说明 React 如何在 JSX 中实现 for 循环
69. 为什么建议 Fragment 包裹元素？它的简写是什么？
70. 你有用过 React.Fragment 吗？说说它有什么用途？
71. 在 React 中你有遇到过安全问题吗？怎么解决？
72. React 中如何监听 state 的变化？

73. React 什么是有状态组件？
74. React v15 中怎么处理错误边界？
75. React Fiber 它的目的是解决什么问题？
76. React 为什么不要直接修改 state？如果想修改怎么做？
77. create-react-app 有什么好处？
78. 装饰器(Decorator)在 React 中有什么应用？
79. 使用高阶组件(HOC)实现一个 loading 组件
80. 如何用 React 实现滚动动画？
81. 说出几点你认为的 React 最佳实践
82. 你是如何划分 React 组件的？
83. 举例说明如何在 React 创建一个事件
84. 如何更新组件的状态？
85. 怎样将多个组件嵌入到一个组件中？
86. React 的 render 中可以写{if else}这样的判断吗？
87. React 为什么要搞一个 Hooks？
88. React Hooks 帮我们解决了哪些问题？
89. 使用 React 的 memo 和 forwardRef 包装的组件为什么提示 children 类型不对？
90. 有在项目中使用过 Antd 吗？说说它的好处
91. 在 React 中如果去除生产环境上的 sourcemap？
92. 在 React 中怎么引用 sass 或 less？
93. 组件卸载前，加在 DOM 元素的监听事件和定时器要不要手动清除？为什么？
94. 为什么标签里的 for 要写成 htmlFor 呢？
95. 状态管理器解决了什么问题？什么时候用状态管理器？
96. 状态管理器它精髓是什么？
97. 函数式组件有没有生命周期？为什么？
98. 在 React 中怎么引用第三方插件？比如说 jQuery 等
99. React 的触摸事件有哪几种？
100. 路由切换时同一组件无法重新渲染的有什么方法可以解决？
101. React16 新特性有哪些？
102. 你有用过哪些 React 的 UI 库？它们的优缺点分别是什么？
103. `<div onClick={handlerClick}>`单击`</div>`和`<div onClick={handlerClick(1)}>`单击`</div>`有什么区别？
104. 在 React 中如何引入图片？哪种方式更好？
105. 在 React 中怎么使用字体图标？
106. React 的应用如何打包发布？它的步骤是什么？
107. ES6 的语法'...'在 React 中有哪些应用？
108. 如何封装一个 React 的全局公共组件？
109. 在 React 中组件的 props 改变时更新组件的有哪些方法？

110. immutable 的原理是什么？
111. 你对 immutable 有了解吗？它有什么作用？
112. 如何提高组件的渲染效率呢？
113. 在 React 中如何避免不必要的 render？
114. render 在什么时候会被触发？
115. 写出 React 动态改变 class 切换组件样式
116. React 中怎么操作虚拟 DOM 的 Class 属性？
117. 为什么属性使用 className 而不是 class 呢？
118. 请说下 react 组件更新的机制是什么？
119. 怎么在 JSX 里属性可以被覆盖吗？覆盖的原则是什么？
120. 怎么在 JSX 里使用自定义属性？
121. 怎么防止 HTML 被转义？
122. 经常用 React，你知道 React 的核心思想是什么吗？
123. 在 React 中我们怎么做静态类型检测？都有哪些方法可以做到？
124. 在 React 中组件的 state 和 setState 有什么区别？
125. React 怎样跳过重新渲染？
126. React 怎么判断什么时候重新渲染组件呢？
127. 什么是 React 的实例？函数式组件有没有实例？
128. 在 React 中如何判断点击元素属于哪一个组件？
129. 在 React 中组件和元素有什么区别？
130. 在 React 中声明组件时组件名的第一个字母必须是大写吗？为什么？
131. 举例说明什么是高阶组件(HOC)的反向继承？
132. 有用过 React Devtools 吗？说说它的优缺点分别是什么？
133. 举例说明什么是高阶组件(HOC)的属性代理？
134. React 的 isMounted 有什么作用？
135. React 组件命名推荐的方式是哪个？为什么不推荐使用 displayName？
136. React 的 displayName 有什么作用？
137. 说说你对 React 的组件命名规范的理解
138. 说说你对 React 的项目结构的理解
139. React16 废弃了哪些生命周期？为什么？
140. 怎样在 React 中开启生产模式？
141. React 中 getInitialState 方法的作用是什么？
142. React 中你知道 creatClass 的原理吗？
143. React 中验证 props 的目的是什么？
144. React 中你有使用过 getDefaultProps 吗？它有什么作用？
145. React 中你有使用过 propType 吗？它有什么作用？
146. React 中怎么检验 props？

147. React.createClass 和 extends Component 的区别有哪些？
148. 高阶组件(HOC)有哪些优点和缺点？
149. 给组件设置很多属性时不想一个个去设置有什么办法可以解决这问题呢？
150. React16 跟之前的版本生命周期有哪些变化？
151. 怎样实现 React 组件的记忆？原理是什么？
152. 创建 React 动画有哪些方式？
153. 为什么建议不要过渡使用 Refs？
154. 在 React 使用高阶组件(HOC)有遇到过哪些问题？如何解决？
155. 在使用 React 过程中什么时候用高阶组件(HOC)？
156. 说说 React diff 的原理是什么？
157. React 怎么提高列表渲染的性能？
158. 使用 ES6 的 class 定义的组件不支持 mixins 了，那用什么可以替代呢？
159. 为何说虚拟 DOM 会提高性能？
160. React 的性能优化在哪个生命周期？它优化的原理是什么？
161. 你知道的 React 性能优化有哪些方法？
162. 举例说明在 React 中怎么使用样式？
163. React 有哪几种方法来处理表单输入？
164. 什么是浅层渲染？
165. 你有做过 React 的单元测试吗？如果有，用的是哪些工具？怎么做的？
166. 在 React 中什么是合成事件？有什么用？
167. 使用 React 写一个 todo 应用，说说你的思路
168. React16 的 reconciliation 和 commit 分别是什么？
169. React 的函数式组件有没有生命周期？
170. useState 和 this.state 的区别是什么？
171. 请说说什么是 useImperativeHandle？
172. 请说说什么是 useReducer？
173. 请说说什么是 useRef？
174. 请说说什么是 useEffect？
175. 举例说明 useState
176. 请说说什么是 useState？为什么要使用 useState？
177. 请描述下你对 React 的新特性 Hooks 的理解？它有哪些应用场景？
178. 说说你对 Error Boundaries 的理解
179. 说说你对 Fiber 架构的理解
180. 说说你是怎么理解 React 的业务组件和技术组件的？
181. 为什么建议 setState 的第一个参数是 callback 而不是一个对象呢？
182. 展示组件和容器组件有什么区别？
183. Mern 和 Yeoman 脚手架有什么区别？

184. 你有在项目中使用过 Yeoman 脚手架吗？
185. 你有在项目中使用过 Mern 脚手架吗？
186. shouldComponentUpdate 方法是做什么的？
187. 怎样在 React 中使用 innerHTML？
188. 你有写过 React 的中间件插件吗？
189. React 的中间件机制是怎么样的？这种机制有什么作用？
190. React 中你用过哪些第三方的中间件？
191. 不用脚手架，你会手动搭建 React 项目吗？
192. 请说说 React 中 Portal 是什么？
193. React 中修改 prop 引发的生命周期有哪几个？
194. React 多个 setState 调用的原理是什么？
195. React 中调用 setState 会更新的生命周期有哪几个？
196. React 中 setState 的第二个参数作用是什么呢？
197. React 中的 setState 是同步还是异步的呢？为什么 state 并不一定会同步更新？
198. React 中的 setState 批量更新的过程是什么？
199. React 中的 setState 执行机制是什么呢？
200. 在 React 中遍历的方法有哪些？它们有什么区别呢？
201. 请说说你对 React 的 render 方法的理解
202. props.children.map 和 js 的 map 有什么区别？为什么优先选择 React 的？
203. 有用过 React 的严格模式吗？
204. React 中的 setState 和 replaceState 的区别是什么？
205. React 中的 setState 缺点是什么呢？
206. 有用过 React 的 Fragment 吗？它的运用场景是什么？
207. React 组件间共享数据方法有哪些？
208. React 的状态提升是什么？使用场景有哪些？
209. 简单描述下你有做过哪些 React 项目？
210. 在构造函数中调用 super(props)的目的是什么？
211. 你是如何学习 React 的？
212. 从旧版本的 React 升级到新版本的 React 有做过吗？有遇到过什么坑？
213. 你用过 React 版本有哪些？
214. 有用过 React 的服务端渲染吗？怎么做的？
215. React 的 mixins 有什么作用？适用于什么场景？
216. React 怎么拿到组件对应的 DOM 元素？
217. 请描述下事件在 React 中的处理方式是什么？
218. JSX 和 HTML 有什么区别？
219. React 的书写规范有哪些？
220. create-react-app 创建新运用怎么解决卡的问题？

221. 使用 React 的方式有哪几种？
222. 说说你对 reader 的 context 的理解
223. 同时引用这三个库 React.js、React-dom.js 和 babel.js 它们都有什么作用？
224. 你知道 Virtual DOM 的工作原理吗？
225. 你阅读过 React 的源码吗？简要说下它的执行流程
226. React 中怎样阻止组件渲染？
227. React 非兄弟组件如何通信？
228. React 兄弟组件如何通信？
229. React 非父子组件如何通信？
230. React 父子组件如何通信？
231. React 组件间的通信有哪些？
232. 类组件和函数式组件有什么区别？
233. React 自定义组件你写过吗？说说看都写过哪些？
234. React 组件的 state 和 props 两者有什么区别？
235. React 有几种构建组件的方式？可以写出来吗？
236. React 中遍历时为什么不用索引作为唯一的 key 值？
237. React 中的 key 有什么作用？
238. React 中除了在构造函数中绑定 this,还有别的方式吗？
239. 在 React 中页面重新加载时怎样保留数据？
240. 请描述下 React 的事件机制
241. 怎样在 React 中创建一个事件？
242. 在 React 中无状态组件有什么运用场景？
243. 描述下在 React 中无状态组件和有状态组件的区别是什么？
244. 写一个 React 的高阶组件(HOC)并说明你对它的理解
245. React 中可以在 render 访问 refs 吗？为什么？
246. React 中 refs 的作用是什么？有哪些应用场景？
247. 请描述你对纯函数的理解？
248. 受控组件和非受控组件有什么区别？
249. React 中什么是非控组件？
250. React 中什么是受控组件？
251. React 中发起网络请求应该在哪个生命周期中进行？为什么？
252. 说说 React 的生命周期有哪些？
253. 说说你对“在 React 中，一切都是组件”的理解
254. 写 React 你是用 es6 还是 es5 的语法？有什么区别？
255. 浏览器为什么无法直接 JSX？怎么解决呢？
256. 在使用 React 过程中你都踩过哪些坑？你是怎么填坑的？
257. 说说你喜欢 React 的原因是什么？它有什么优缺点？

258. 如何解决引用类型在 pureComponent 下修改值的时候，页面不渲染的问题？
259. createElement 与 cloneElement 两者有什么区别？
260. 解释下 React 中 Element 和 Component 两者的区别是什么？
261. 解释下 React 中 component 和 pureComponent 两者的区别是什么？
262. React 的虚拟 DOM 和 vue 的虚拟 DOM 有什么区别？
263. 你觉得 React 上手快不快？它有哪些限制？
264. 说说你对声明式编程的理解？
265. React 与 angular、vue 有什么区别？
266. React 是哪个公司开发的？
267. React 是什么？它的主要特点是什么？
268. 简要描述下你知道的 React 工作原理是什么？
269. 在 React 中怎样改变组件状态，以及状态改变的过程是什么？
270. 在 React 中你是怎么进行状态管理的？
271. React 声明组件有哪几种方法，各有什么不同？

ReactNative

1. 如何在 React Native 中设置环境变量？
2. 请描述下 Code Push 的原理是什么？
3. React Native 怎样查看日记？
4. React Native 怎样测试？
5. React Native 怎样调试？
6. React Native 和 React 有什么区别？
7. 有做过 React Native 项目吗？

React-Router

1. React-Router 怎么获取历史对象？
2. React-Router 怎么获取 URL 的参数？
3. 在 history 模式中 push 和 replace 有什么区别？
4. React-Router 怎么设置重定向？
5. React-Router 4 中`<Router>`组件有几种类型？
6. React-Router 3 和 React-Router 4 有什么变化？添加了什么好的特性？
7. React-Router 的实现原理是什么？
8. React-Router 4 的 switch 有什么用？
9. React-Router 的路由有几种模式？
10. React-Router 4 怎样在路由变化时重新渲染同一个组件？

11. React-Router 的<Link>标签和<a>标签有什么区别？
12. React 的路由和普通路由有什么区别？
13. 请你说说 React 的路由的优缺点？
14. 请你说说 React 的路由是什么？

Redux/Mobx

- 你有了解 Rxjs 是什么吗？它是做什么的？
- 在 Redux 中怎么发起网络请求？
- Redux 怎样重置状态？
- Redux 怎样设置初始状态？
- Context api 可以取代 Redux 吗？为什么？
- 推荐在 reducer 中触发 Action 吗？为什么？
- Redux 怎么添加新的中间件？
- redux-saga 和 redux-thunk 有什么本质的区别？
- 在 React 中你是怎么对异步方案进行选型的？
- 你知道 redux-saga 的原理吗？
- 你有使用过 redux-saga 中间件吗？它是干什么的？
- Redux 中异步 action 和同步 action 最大的区别是什么？
- Redux 和 vuex 有什么区别？
- Redux 的中间件是什么？你有用过哪些 Redux 的中间件？
- 说说 Redux 的实现流程
- Mobx 的设计思想是什么？
- Redux 由哪些组件构成？
- Mobx 和 Redux 有什么区别？
- 在 React 项目中你是如何选择 Redux 和 Mobx 的？说说你的理解
- 你有在 React 中使用过 Mobx 吗？它的运用场景有哪些？
- Redux 的 thunk 作用是什么？
- Redux 的数据存储和本地储存有什么区别？
- 在 Redux 中，什么是 reducer？它有什么作用？
- 举例说明怎么在 Redux 中定义 action？
- 在 Redux 中，什么是 action？
- 在 Redux 中，什么是 store？
- 为什么 Redux 能做到局部渲染呢？
- 说说 Redux 的优缺点分别是什么？
- Redux 和 Flux 的区别是什么？
- Redux 它的三个原则是什么？

- 什么是单一数据源？
- 什么是 Redux？说说你对 Redux 的理解？有哪些运用场景？

Flux

- 请说说点击按钮触发到状态更改，数据的流向？
- 请描述下 Flux 的思想
- 什么是 Flux？说说你对 Flux 的理解？有哪些运用场景？